**RESUME SCREENING ASSISTANT**

USING IBM WATSONX.AI FOUNDATION MODEL

Submitted by:

Name : Mohammad Thabassum Sulthana

Roll Number : 23BCE9337

College Name : VIT-AP UNIVERSITY AMARAVATHI

Branch : AIML

Date of Submission: 26 June 2025

|  |  |  |
| --- | --- | --- |
| **Page No.** | **Section Title** | **Description** |
| 1 | **Title Page** | Project title, your name, roll number, college name, branch, date of submission, and logos of your college and IBM. |
| 2 | **Table of Contents** | A clear and organized list of all the sections in the report with short descriptions. |
| 3 | **Introduction** | Overview of the Resume Screening Assistant project using GenAI to automate resume classification. |
| 3 | **Objective** | Defines the project's main aim: classify resumes into job categories using IBM foundation models. |
| 4 | **Tools, Technologies & Platforms Used** | Details of tools and libraries used: IBM watsonx.ai, Google Colab, Streamlit, Tesseract OCR, OpenCV, etc. |
| 5 | **Methodology – Project Workflow with Flowchart** | Step-by-step working of the system with a flowchart: from uploading a resume to getting prediction output. |
| 6–7 | **Code Implementation – Google Colab Workflow** | Code snippets from Google Colab including: file input, OCR, IBM model API calls, and results display. |
| 8–13 | **Screenshots of Output – Google Colab & Streamlit** | Visual proof of execution and results from both Colab and Streamlit app. Includes labeled screenshots. |
| 14 | **Project Links** | GitHub repository, Google Colab notebook, and Streamlit app URL for project access and testing. |
| 14 | **Conclusion** | Summarizes the outcomes, strengths of the project, and areas for future enhancement. |
| 14 | **References** | External sources, official docs, and libraries (IBM, Streamlit, Tesseract) referred during the project. |

**Introduction**

In today’s competitive job market, recruiters receive hundreds to thousands of resumes for a single job opening. Manually reviewing and categorizing these resumes is time-consuming and prone to bias or error. To address this challenge, the “**Resume Screening Assistant**” project leverages **Generative AI** and **IBM watsonx.ai foundation models** to automatically classify resumes based on their content into suitable job roles such as **Data Scientist**, **Web Developer**, or **Marketing Specialist**.

This project provides a smart, automated solution to reduce recruitment time and increase accuracy in shortlisting candidates. It supports both **PDF and image-based resumes** by integrating **Tesseract OCR** for text extraction from screenshots or scanned documents.

The application was first developed and tested using **Google Colab**, where all resume files were processed, predictions were generated using IBM models, and results were analyzed. Later, the solution was deployed in a user-friendly web app format using **Streamlit**, allowing users to upload resumes and get instant job role predictions.

This project showcases how modern AI technologies can simplify real-world tasks like resume screening, making the hiring process faster, smarter, and more efficient.

**Objective**

The main objective of this project is to develop an intelligent and fully automated **Resume Screening Assistant** that uses **IBM watsonx.ai Foundation Models** to classify resumes into suitable job categories such as **Data Science**, **Software Development**, **Marketing**, and more.

This project solves a real-world recruitment challenge by minimizing manual effort and increasing efficiency in candidate shortlisting. It supports both **PDF and image-based resumes** through integrated **text extraction techniques** using **PyMuPDF** for PDFs and **Tesseract OCR** for images.

**Specifically, the project aims to:**

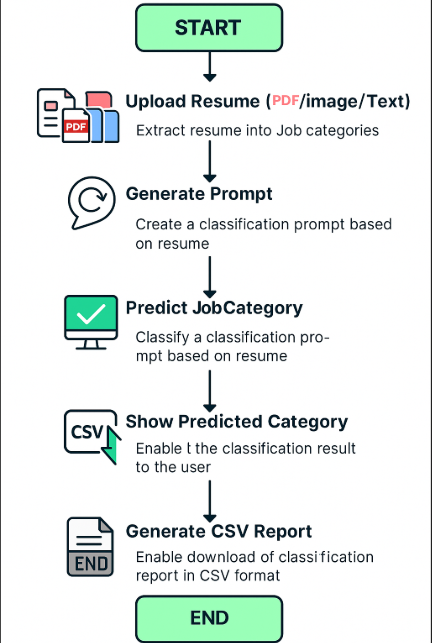
* Build a system that supports multiple resume input formats: **PDF, image (JPG/PNG), and plain text**.
* Extract meaningful text from resumes using **OCR (Tesseract)** and **PyMuPDF**.
* Use the **IBM Granite model (ibm/granite-3-3-8b-instruct)** hosted on **watsonx.ai** to accurately predict the most relevant job category.
* Leverage **Google Colab** as a development and testing environment for fast iteration and cloud-based execution.
* Deploy the final application through a **Streamlit interface**, allowing users to upload resumes and get real-time classification results.
* Showcase the potential of **Generative AI in HR automation**, reducing screening time and improving decision-making for recruiters.

**Tools & Technologies Used**

|  |  |  |
| --- | --- | --- |
| **Category** | **Tool / Technology** | **Purpose / Description** |
| Programming Language | Python | Core language used to build, test, and deploy the entire application. |
| Development Environment | Google Colab | Used for writing, running, and testing the core model integration and logic. |
| AI/ML Model Platform | IBM watsonx.ai | IBM’s GenAI platform used to access and run foundation models via API. |
| Foundation Model | ibm/granite-3-3-8b-instruct | Instruction-tuned LLM used to classify resumes into relevant job roles. |
| Web Framework | Streamlit | Used to build and deploy a simple, interactive user interface for the app. |
| PDF Parsing | PyMuPDF (fitz) | Library used to extract text from PDF resumes. |
| OCR Engine | Tesseract OCR (pytesseract) | Used to extract text from image-based resumes (JPG, PNG). |
| Image Processing | OpenCV, PIL | Helps preprocess uploaded images before OCR is applied. |
| API Integration | ibm\_watsonx\_ai Python SDK | Used to authenticate and send prompts to IBM watsonx.ai models. |
| Temporary File Handling | tempfile | Saves uploaded resumes temporarily for text extraction. |
| Version Control | GitHub | Stores and shares the source code of the project. |
| Deployment Platform | Streamlit Community Cloud | Hosts the final web app online for public use and demonstration. |

**Methodology**

The Resume Screening Assistant follows a **step-by-step pipeline** to classify resumes into job categories using IBM watsonx.ai. The system supports multiple input formats, processes the content, sends it to the IBM foundation model, and returns a prediction—all seamlessly handled via a web interface or Google Colab..

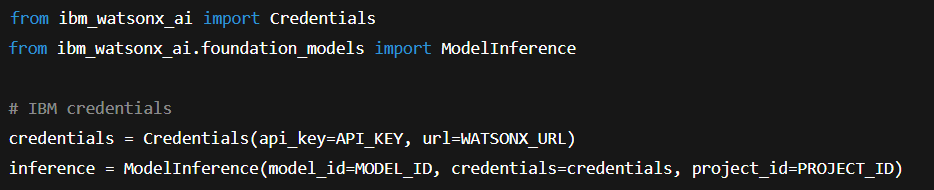


**Step-by-Step Workflow:**

1. **Input Upload**  
   Users can upload resumes in **PDF**, **Image (PNG/JPG)** format, or paste text directly.
2. **Text Extraction**
   * For **PDFs**, text is extracted using **PyMuPDF (fitz)**.
   * For **images**, OCR is performed using **Tesseract** (via pytesseract).
   * For pasted text, it directly proceeds to prediction.
3. **Prompt Generation**  
   A custom prompt is dynamically created using the extracted resume text.
4. **Model Inference (Prediction)**  
   The prompt is sent to the **IBM Foundation Model**:  
   ✅ ibm/granite-3-3-8b-instruct  
   The model returns a **job category** (e.g., Data Science).
5. **Result Display**  
   The predicted category is shown in the **Streamlit app** or in **Google Colab** output cell.

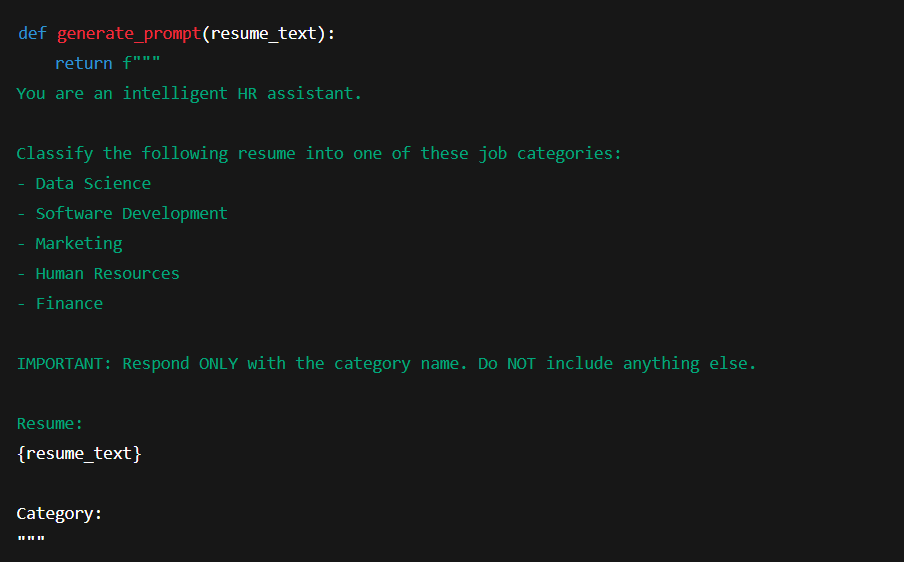
**Code Snippets with Explanation**

1. **Connecting to IBM watsonx.ai**

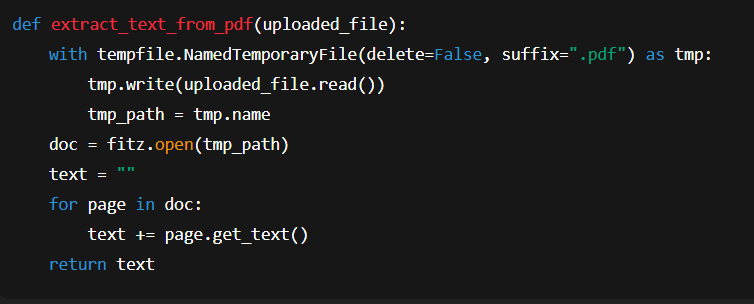
****

**Explanation:**  
This code connects your app or notebook to the IBM watsonx.ai platform using your unique API\_KEY, PROJECT\_ID, and the model ibm/granite-3-3-8b-instruct. It allows secure interaction with IBM's foundation model.

1. **Prompt Generation Function**

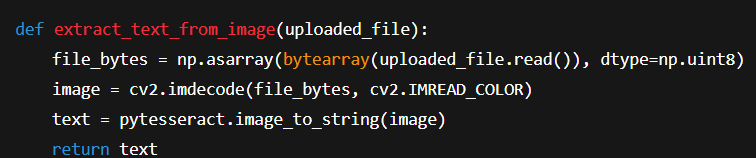
**Explanation:**  
This function builds a natural language prompt to be sent to the model. It tells the model exactly what it should do — **classify the resume into a job category only**, with no extra explanation.

3. **Extracting Text from PDF Resumes**



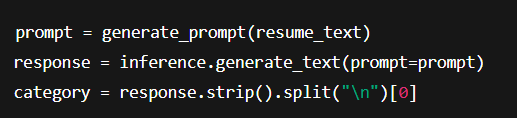
. **Explanation:**  
This function uses **PyMuPDF (fitz)** to open and read a PDF file. It loops through all pages and extracts plain text, which is then passed to the model.

1. **Extracting Text from Image Resumes (OCR)**

****

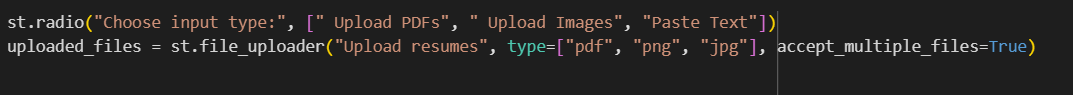
**Explanation:**  
This function handles image-based resumes. It uses **OpenCV** to convert the uploaded image into a format Tesseract can read. Then, **Tesseract OCR** extracts the visible text from the image.

1. **Running the Prediction**

****

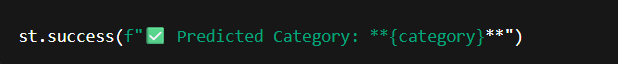
**Explanation:**  
Here, your generated prompt is sent to IBM's foundation model, and the returned response is trimmed and filtered to extract just the job category.

1. **Streamlit Interface Logic (Upload + Display)**

****

**Explanation:**  
This part sets up the Streamlit UI. It allows the user to choose how they want to submit a resume — as a PDF, an image, or plain text.

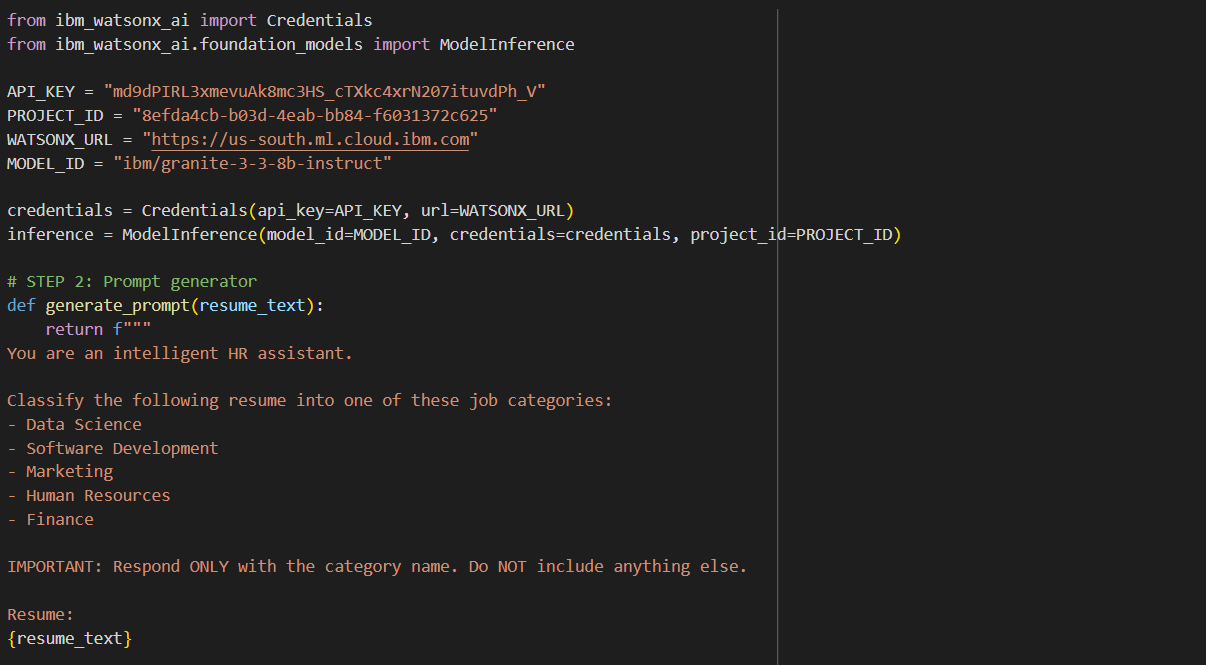
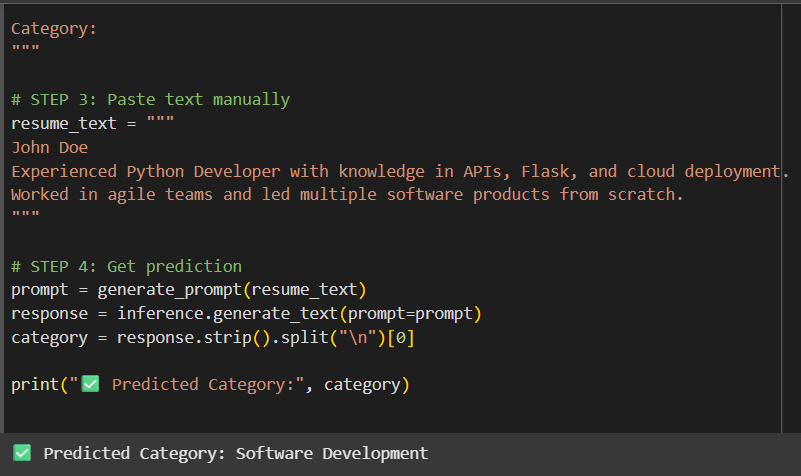
### ****Displaying Prediction Results****

****

**Explanation:**  
Once the IBM model responds, the app shows the predicted job role to the user in a clear, highlighted success message.

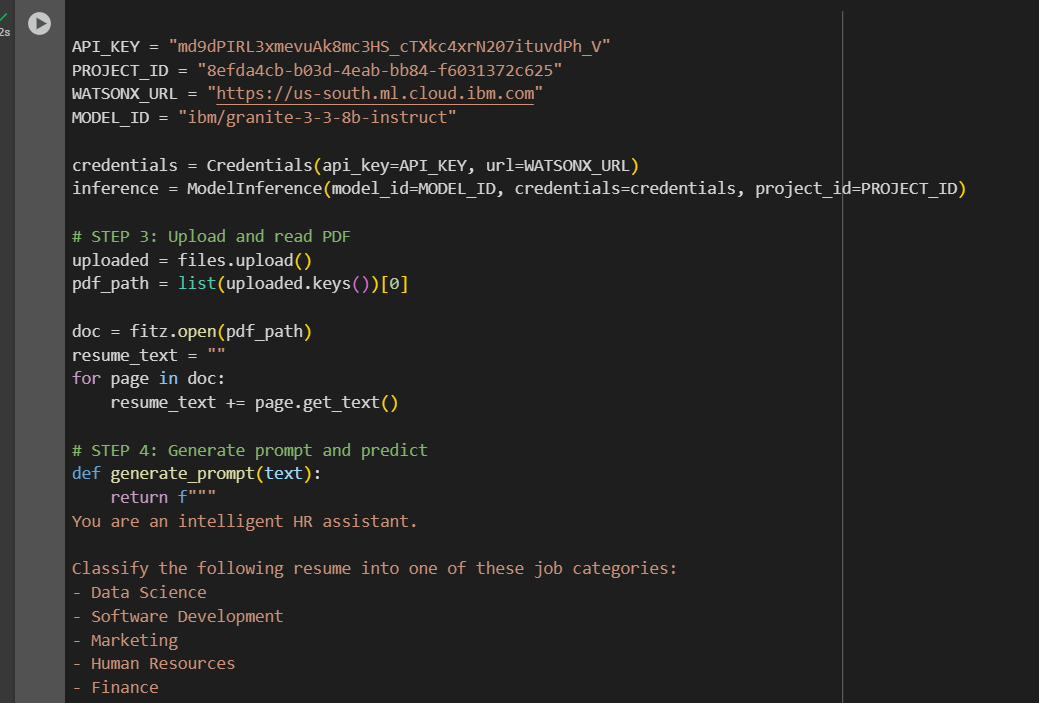
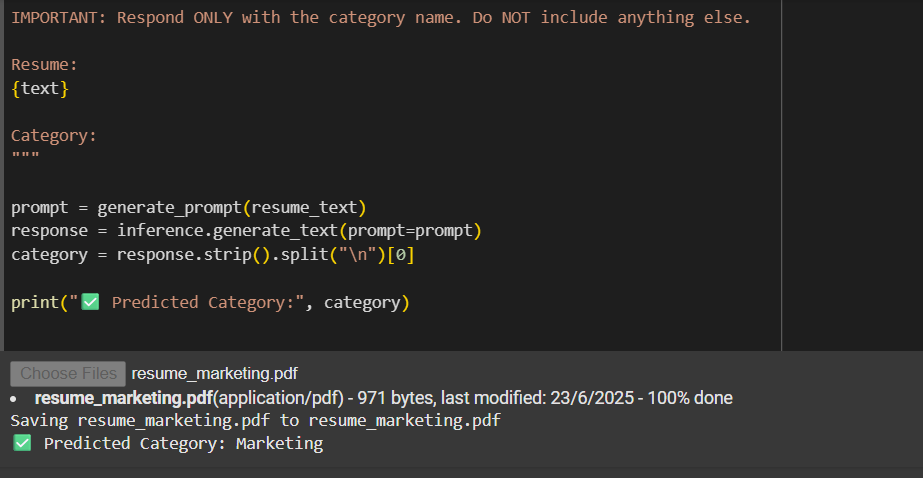
**Screenshots / Output Results with Explanation**

## **Text Input Resume Classification Code**

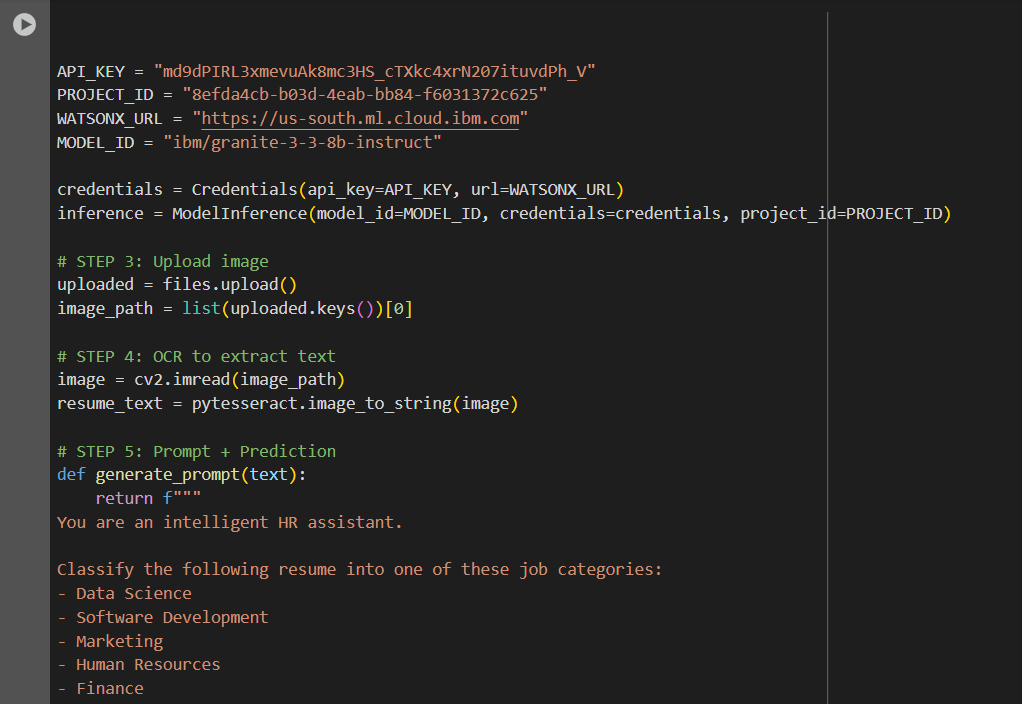
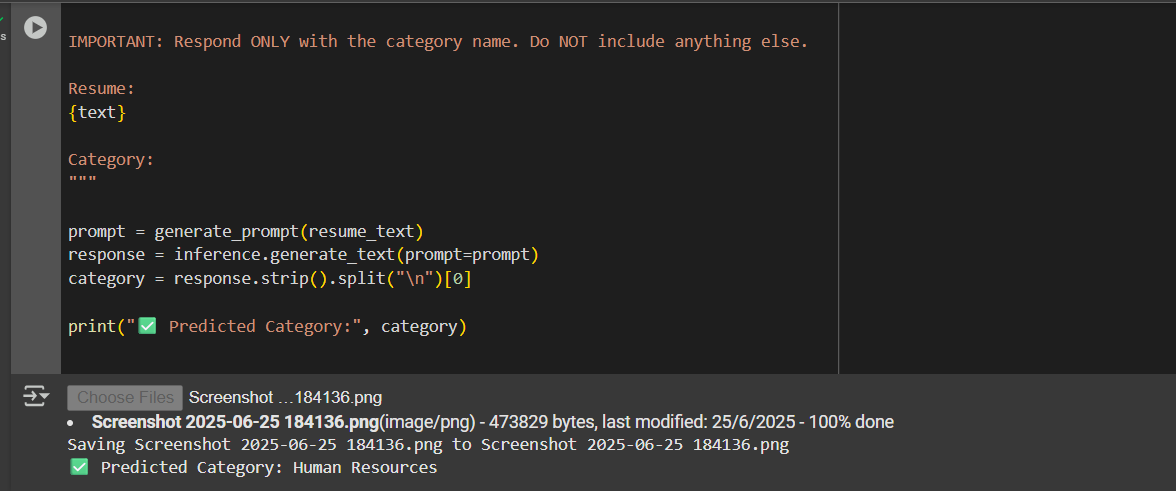
**** ****

This code initializes and runs a resume classification pipeline using IBM watsonx.ai in a simple and structured way. In **Step 1**, it sets up authentication with IBM watsonx by using your API key, project ID, and endpoint URL, and loads the ibm/granite-3-3-8b-instruct foundation model designed for natural language tasks. In **Step 2**, it defines a prompt that clearly instructs the model to classify the given resume text into one of five predefined job categories, returning only the category name. **Step 3** simulates a resume by manually pasting job-related text, and in **Step 4**, the model is queried with the prompt and the resume text. The output is parsed and the predicted category (e.g., "Software Development") is printed, demonstrating how IBM watsonx.ai can be used to intelligently classify resume content using prompt-based inference.

1. **Classify PDF Resume**

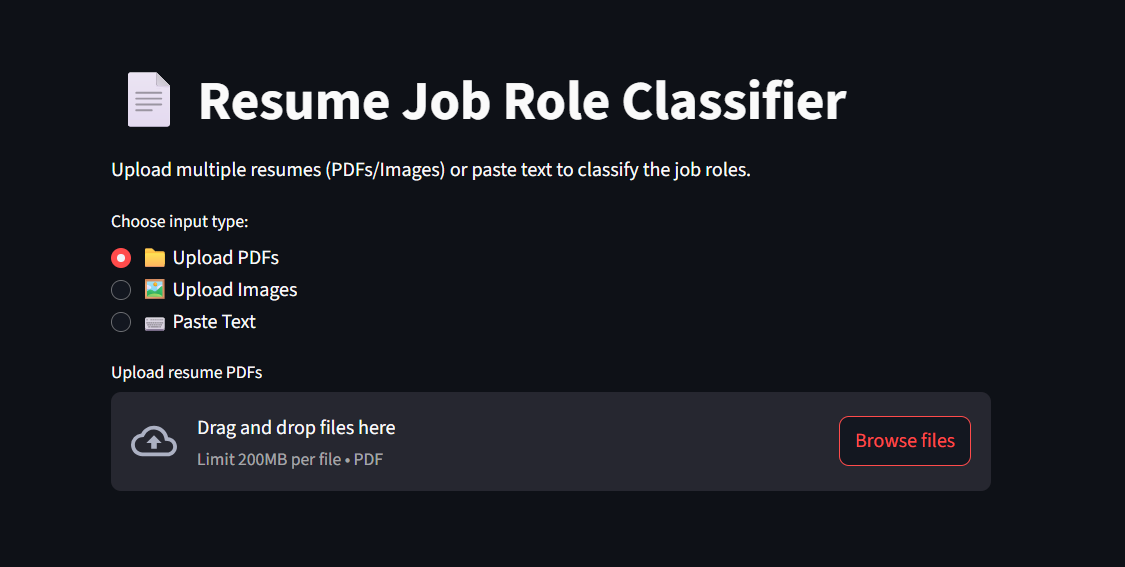
  This code enables resume classification directly from a **PDF file** using IBM watsonx.ai in Google Colab. First, it installs required libraries like PyMuPDF (for PDF parsing) and the IBM watsonx SDK. It then authenticates with IBM watsonx.ai using your API key and project ID, and loads the ibm/granite-3-3-8b-instruct model. After that, the user is prompted to upload a PDF resume, which is processed using PyMuPDF to extract text content from all pages. A prompt is then generated that instructs the model to classify the resume into one of five job roles (e.g., "Data Science", "Marketing"), and this prompt is sent to the model. The model's plain-text response is parsed to retrieve and display the predicted job category, making this a seamless way to classify resumes from PDF format.

1. **Classify Image Resume (JPG/PNG)**

****

This code allows classification of **image-based resumes** (e.g., scanned JPG or PNG files) using **IBM watsonx.ai** in Google Colab. It begins by installing the required dependencies such as Tesseract OCR, OpenCV, and the IBM watsonx SDK. Then, the script imports necessary libraries and sets up authentication using your API key and project ID, initializing the ibm/granite-3-3-8b-instruct model for inference. After a user uploads an image of a resume, the code uses OpenCV to load the image and **pytesseract** to extract the printed text from it using Optical Character Recognition (OCR). This extracted text is formatted into a prompt that instructs the AI to classify the resume into one of five categories (e.g., "Data Science", "Marketing"). Finally, the prompt is sent to the IBM model, and the returned prediction is displayed. This approach enables intelligent classification of resumes even when they are provided as images instead of documents or raw text.

**Streamlit App Output**

****

### 📄 ****Streamlit App Interface (Home Page) – Explanation****

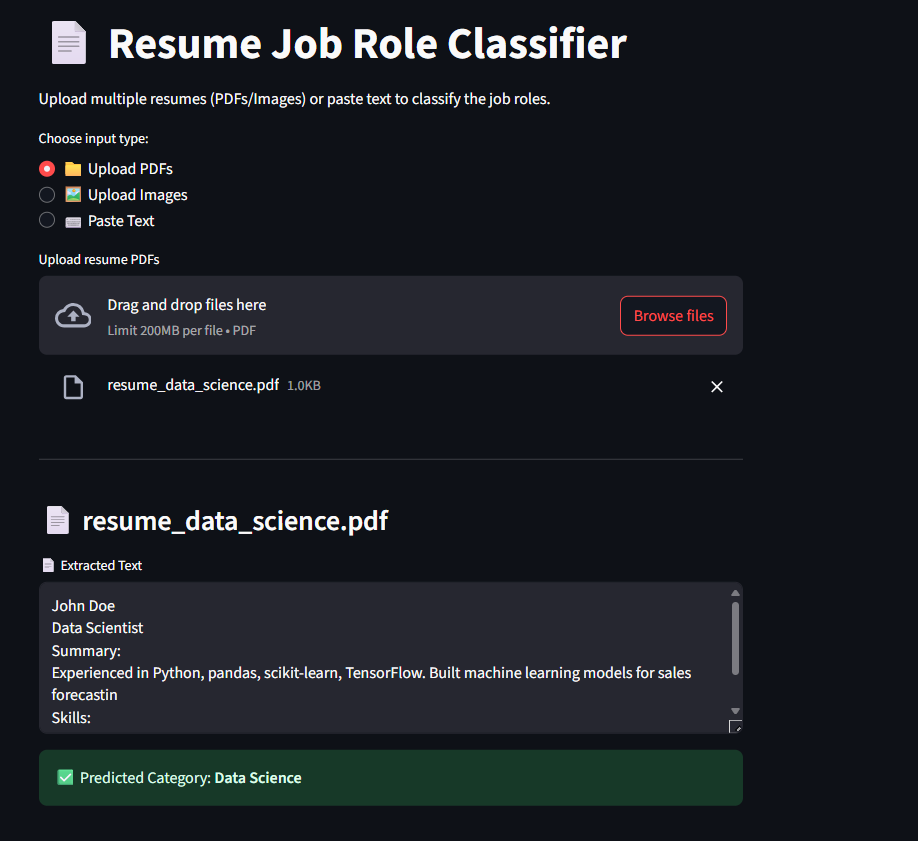
This is the **homepage interface** of your web application titled **“Resume Job Role Classifier.”** The app is designed to help HR professionals or recruiters quickly determine the job category a resume fits into using AI. It uses the **IBM watsonx.ai Foundation Model** (ibm/granite-3-3-8b-instruct) to classify resumes into one of the following categories:

* Data Science
* Software Development
* Marketing
* Human Resources
* Finance

On this screen, the user is presented with **three input options**:

1. **Upload PDFs** – for uploading resume documents in PDF format.
2. **Upload Images** – for submitting screenshots or scans of resumes (e.g., JPG or PNG).
3. **Paste Text** – allows direct text input if the user has copied resume content.

The interface is minimal and user-friendly, with drag-and-drop file upload support, making it easy for users to test or use the application in real-world HR or resume-screening scenarios.

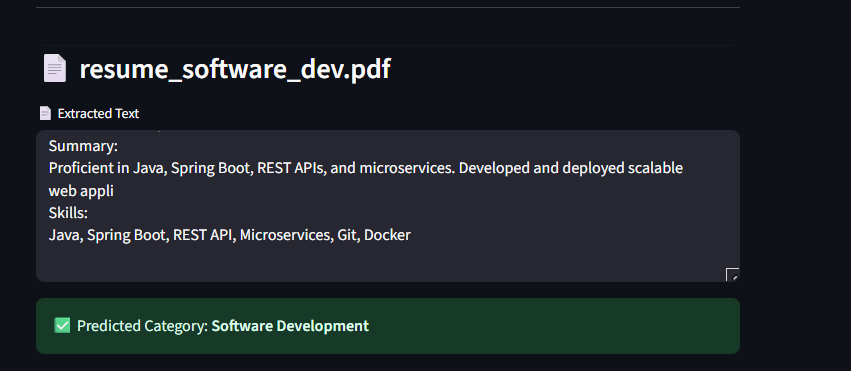
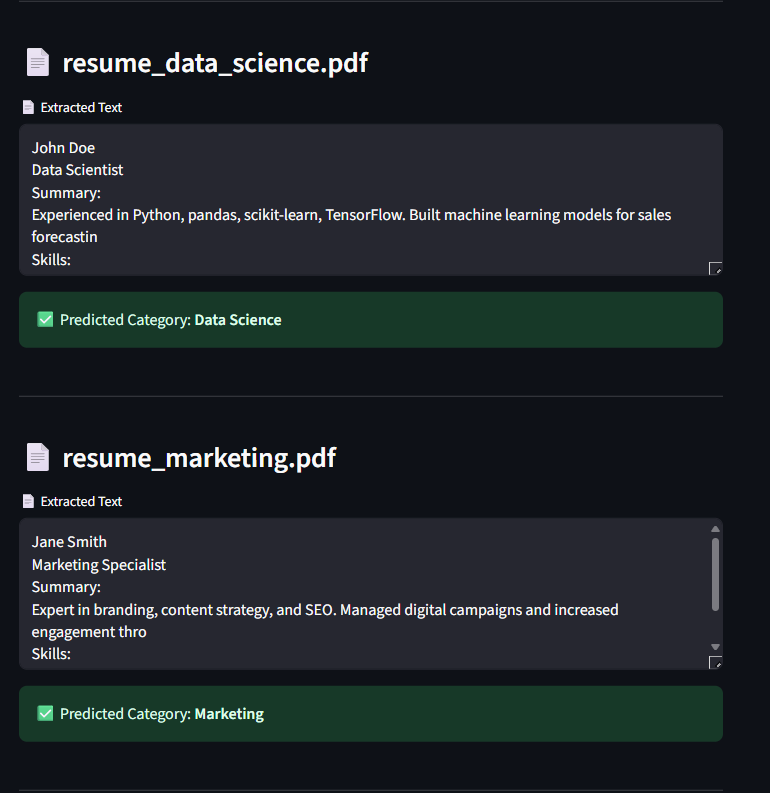


#### **Uploaded File:** resume\_data\_science.pdf

* The user selected the **“Upload PDFs”** option and uploaded a sample resume named resume\_data\_science.pdf.
* The application then used **PyMuPDF** to extract the text content from the uploaded PDF.

**Classification Output:**

* The text is passed to the IBM watsonx.ai **ibm/granite-3-3-8b-instruct** model.
* The model classifies the resume as:  
  ✅ **Predicted Category: Data Science**

**Links for Your Project**

### ****GitHub Repository****

**Project Source Code:**  
<https://github.com/Thabassum-sulthana/resume-screening->

This repository contains the complete codebase of the Resume Screening Assistant, including:

* Streamlit app source (app.py)
* IBM watsonx.ai model integration
* PDF and image resume processing logic
* Prompt templates and utility functions

### ****Deployed Streamlit Web App****

**Live Application:**  
<https://3jthkypqqhlwukg36ybjd8.streamlit.app/>

This is the publicly accessible version of the Resume Classifier. Users can:

* Upload PDF or image resumes
* Paste raw text from resumes
* Instantly receive job category predictions powered by IBM watsonx.ai

**Conclusion**

The Resume Screening Assistant project successfully demonstrates how generative AI can be applied to real-world HR tasks using IBM watsonx.ai. By integrating the ibm/granite-3-3-8b-instruct foundation model, the system intelligently classifies resumes into job categories such as Data Science, Software Development, Marketing, Human Resources, and Finance.

The project supports multiple input formats including PDFs, images (JPG/PNG), and plain text, making it highly flexible and user-friendly. Using tools like PyMuPDF for PDF extraction and Tesseract OCR for image processing, the app ensures accurate text retrieval before classification. The final solution is deployed using Streamlit, providing an interactive web-based interface that can be accessed and tested by anyone.

Overall, this project showcases the power of prompt engineering, foundation models, and modern deployment techniques to build a smart, efficient, and accessible AI solution for resume analysis. It serves as a practical example of how AI can automate and enhance recruitment workflows in real-time.