# CareerCraft: ATS-Optimized Resume Analyzer Using Gemini Model

## Intoduction:

### Project description:

CareerCraft is a state-of-the-art ATS-Optimized Resume Analyzer designed to enhance the job application process for both job seekers and employers. Leveraging advanced Applicant Tracking System (ATS) technology, CareerCraft ensures that resumes are not only eye-catching to recruiters but also easily readable and rankable by ATS software, which is commonly used by companies to filter and rank job applications.

### Scenario 1: Resume Optimization

CareerCraft assists job seekers in optimizing their resumes for specific job openings. By analyzing job descriptions and resumes, CareerCraft identifies the percentage match between the two, suggests missing keywords, and offers recommendations to improve resume alignment with the desired job roles. This feature streamlines the application process and increases the chances of securing interviews.

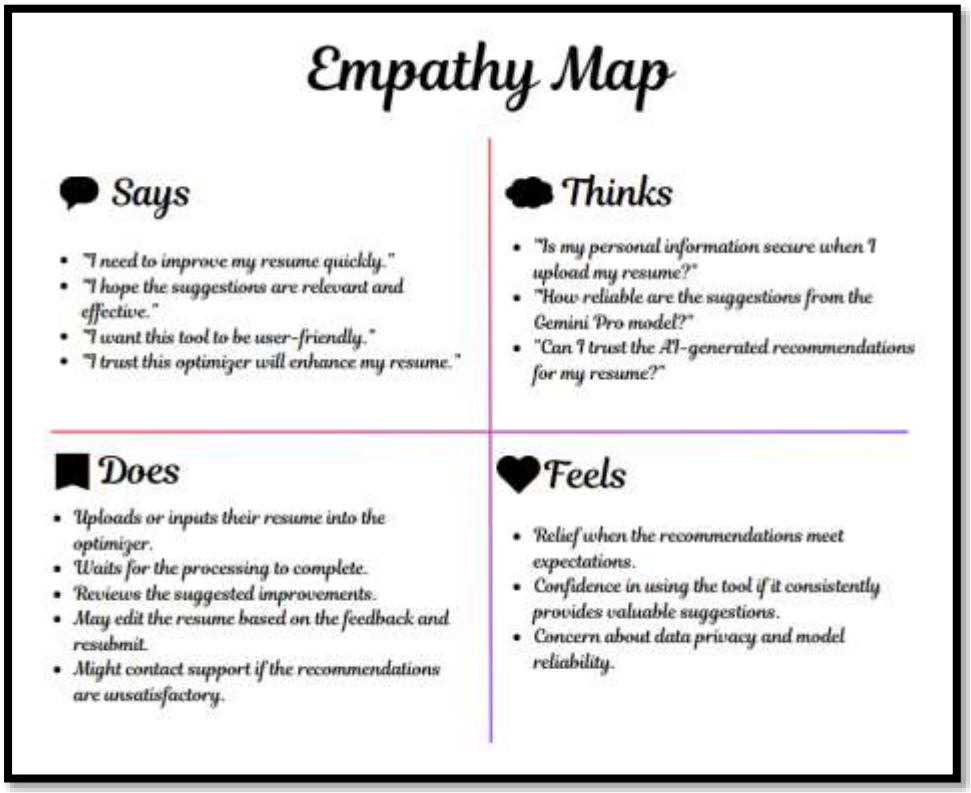
### Scenario 2: Skill Enhancement

For individuals looking to enhance their skills and qualifications, CareerCraft serves as a valuable tool for identifying areas of improvement. By comparing resumes to industry-standard job descriptions, CareerCraft identifies skill gaps and provides personalized suggestions for skill development and enhancement. This feature empowers users to tailor their professional profiles to meet the demands of their desired career paths.

### Scenario 3: Career Progression Guidance

Professionals seeking career advancement opportunities can rely on CareerCraft for strategic guidance. By analyzing resumes and job descriptions, CareerCraft offers insights into potential career trajectories, identifies relevant skills and experiences, and provides personalized recommendations for achieving career goals. This feature helps users navigate their career paths effectively and capitalize on growth opportunities.

**Empathy Map:**




**Project Flow:**

- User interacts with the UI to enter the input.
- User input is collected from the UI and transmitted to the backend using the Google API key.
- The input is then forwarded to the Gemini Pro pre-trained model via an API call.
- The Gemini Pro pre-trained model processes the input and generates the output.
- The results are returned to the frontend for formatting and display.

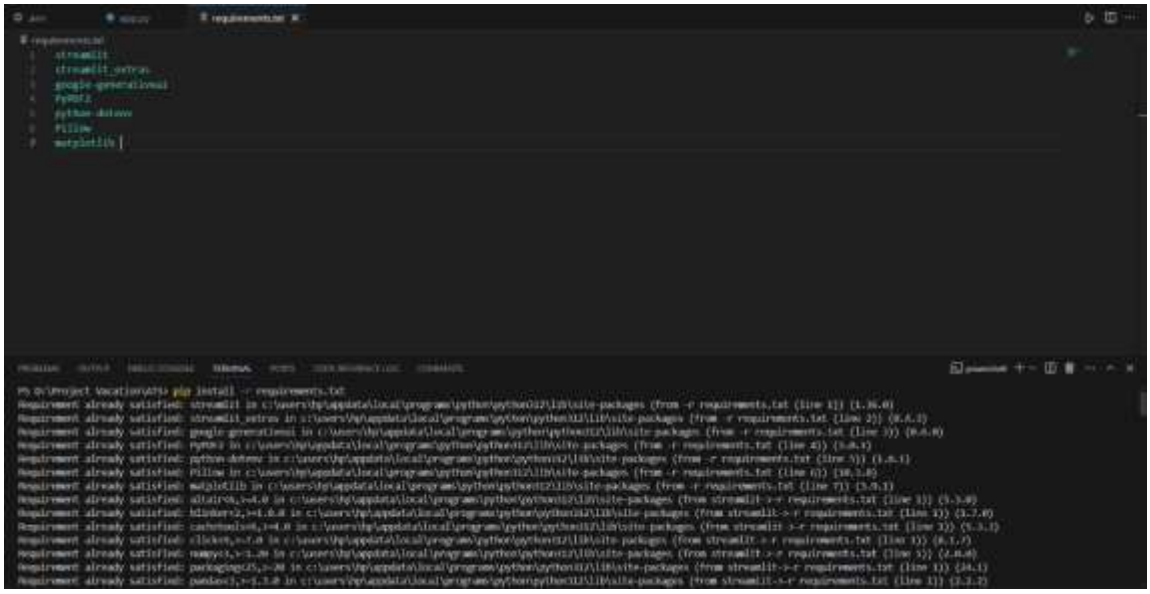
**Requirement Specification:**

- Create a Requirements.Txt File to list the required libraries:

```
requirements.txt
streamlit
streamlit_extras
google-generativeai
PyPDF2
python-dotenv
Pillow
matplotlib
```

  - **streamlit**: Streamlit is a powerful framework for building interactive web applications with Python.
  - **streamlit_extras**: Additional utilities and enhancements for Streamlit applications.
  - **google-generativeai**: Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
  - **python-dotenv**: Python-dotenv allows you to manage environment variables stored in a .env file for your Python projects.
  - **PyPDF2:** It is a Python library for extracting text and manipulating PDF documents.
  - **Pillow:** Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different image file formats.

**Installing required libraries:**



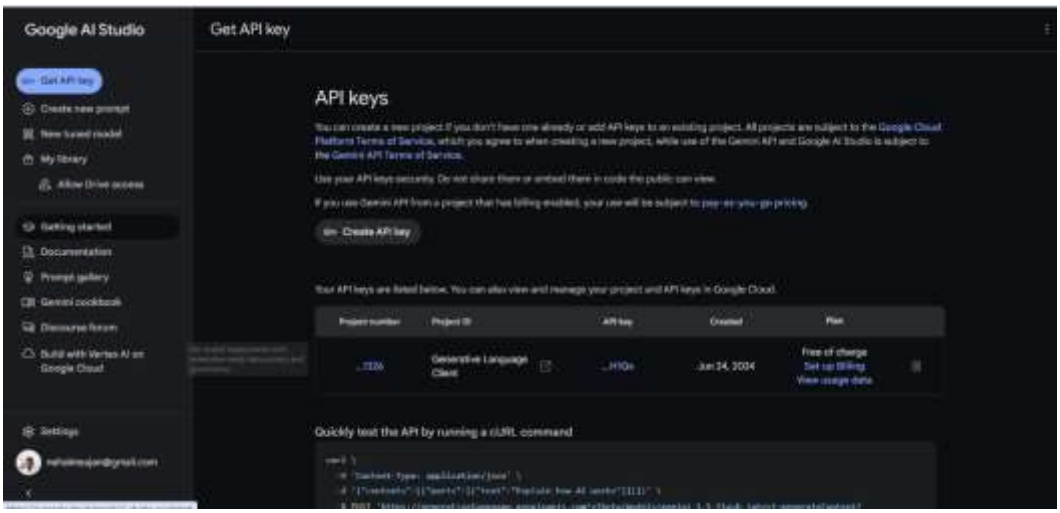
Steps:

- Open the terminal.
- Run the command: pip install -r requirements.txt
- This command installs all the libraries listed in the requirements.txt file

## Initialization of Google API key

- **What is a Google API key**:

  The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API requests, ensuring that only authorized users can access and utilize Google's services.




- **Initializing the API key:**

## Load The Gemini Pro Pre-Trained Model:

- **Loading the Gemini Pro pretrained model and implementing a function to get Gemini response:**

```python
import streamlit as st
from streamlit_extras import add_vertical_space as avs
import google.generativeai as genai
import os
import PyPDF2 as pdf
from dotenv import load_dotenv
import json
from PIL import Image
import matplotlib.pyplot as plt

load_dotenv()  # Load all our environment variables

genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

def get_gemini_response(input):
    try:
        model = genai.GenerativeModel('gemini-pro')
        response = model.generate_content(input)
        return response.text
    except Exception as e:
        st.error(f"Error generating response: {e}")
        return None
```

**Steps:**

- The code begins by importing necessary libraries and modules, including dotenv, Streamlit, os, PyPDF2, GenerativeAIfrom Google, PIL (Python Imaging Library), and a custom module for adding vertical space in Streamlit.It loads environment variables from the .env file using the load_dotenv() function.
- The GenerativeAI module is configured with the Google API key stored in the environment variable GOOGLE_API_KEY.
- A GenerativeModel object named "model" is created using the Gemini Pro pre-trained model from Google.
- The code is essentially setting up the environment, configuring the GenerativeAI module with the API key,and loading the Gemini Pro model for generating responses to user inputs in the Streamlit app.
- The function get_gemini_response takes an input text as a parameter.

- It calls the generate_content method of the model object to generate a response.
- The generated response is returned as text.

## ➢ Implement a function to read PDF content:

```python
#Convert PDF content to Text format
def input_pdf_text(uploaded_file):
    try:
        reader = pdf.PdfReader(uploaded_file)
        text = ""
        for page in range(len(reader.pages)):
            page = reader.pages[page]
            text += str(page.extract_text())
        return text
    except Exception as e:
        st.error(f"Error reading PDF file: {e}")
        return None

```

- The function input_pdf_text takes an uploaded PDF file as input.
- It creates a PdfReader object from the PyPDF2 library to read the uploaded PDF file.
- It initializes an empty string variable text to store the extracted text from the PDF.
- It iterates over each page of the PDF using a loop.
- For each page, it extracts the text using the extract_text() method and appends it to the text variable.
- Finally, it returns the concatenated text extracted from all pages of the PDF

## ➢ Implementing a function to represent data graphically:

```python
# Function to create a doughnut chart
def create_doughnut_chart(percentage):
    fig, ax = plt.subplots(figsize=(6, 6))
    sizes = [percentage, 100 - percentage]
    colors = ['#4CAF50', '#E0E0E0']
    explode = (0.1, 0)
    ax.pie(sizes, colors=colors, startangle=90, explode=explode, wedgeprops=dict(width=0.3))
    ax.text(0, 0, f'{percentage}%', ha='center', va='center', fontsize=24, fontweight='bold')
    ax.set_title("Match Percentage", fontsize=16)
    ax.axis('equal')
    return fig

```

- The function create_doughnut_chart(percentage) is defined to take a single argument percentage.
- fig, ax = plt.subplots(figsize=(6, 6)): Creates a figure and a set of subplots with a size of 6x6 inches.
- sizes = [percentage, 100 - percentage]: Defines the sizes of the slices in the pie chart. One slice represents the given percentage, and the other slice represents the remainder to make up 100%.
- ax.pie(sizes, colors=colors, startangle=90, explode=explode, wedgeprops=dict(width=0.3)): Creates the pie chart with the given sizes, colors, starting angle, and explode settings. The wedgeprops parameter sets the width of the wedges to 0.3 to create a doughnut shape.
- ax.axis('equal'): Ensures that the pie chart is drawn as a circle.

## ➢ Writing a prompt for Gemini model

```
col1, col2 = st.columns([3, 2])
with col1:
    st.markdown("<h1 class='header-title'>Embark on Your Career Adventures/h1>", unsafe_allow_html=True)
    jd = st.text_area("Paste the Job Description")
    uploaded_file = st.file_uploader("Upload Your Resume", type="pdf", help="Please upload the PDF")
    submit = st.button("Submit")
    if submit:
        if uploaded_file is not None and jd:
            text = input_pdf_text(uploaded_file)
            input_prompt = f"""
            As an experienced ATS (Applicant Tracking System), proficient in the technical domain encompassing Software Engineering,
            Data Science, Data Analysis, Big Data Engineering, Web Developer, Mobile App Developer,
            DevOps Engineer, Machine Learning Engineer, Cybersecurity Analyst, Cloud Solutions Architect,
            Database Administrator, Network Engineer, AI Engineer, Systems Analyst, Full Stack Developer, UI/UX Designer,
            IT Project Manager, and additional specialized areas, your objective is to meticulously assess resumes against provided job descriptions.
            In a fiercely competitive job market, your expertise is crucial in offering top-notch guidance for resume enhancement.
            Assign precise matching percentages based on the JD (Job Description) and meticulously identify any missing keywords with utmost accuracy.
            resume: {text}
            description: {jd}
            I want the response in the following structure:
            The first line indicates the percentage match with the job description (JD).
            The second line presents a list of missing keywords.
            The third section provides a profile summary.
            Mention the title for all the three sections.
            While generating the response put some space to separate all the three sections.
            """
            response = get_gemini_response(input_prompt)
```

- The input_prompt is a multiline string containing instructions for an ATS (Applicant Tracking System).
- It describes the expertise required for the ATS, including proficiency in various technical domains such as Software Engineering, Data Science, etc.
- The objective of the ATS is to assess resumes against provided job descriptions in a competitive job market.
- It requests the response to be structured into three sections: percentage match with the job description, a list of missing keywords, and a profile summary.

## Model Deployment:

- **Integrate with web framework**

```
# Streamlit UI
st.set_page_config(page_title="Resume ATS Tracker", layout="wide")


st.markdown(
    """
    <style>
    @import url('https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap');

    .stApp {
        background-color: #34C0BC;
        color: black;
        font-family: 'Roboto', sans-serif;
    }
    h1, h2, h3, h4, h5, h6 {
        color: black;
        font-family: 'Roboto', sans-serif;
    }
    p, div, span {
        color: black;
        font-family: 'Roboto', sans-serif;
    }
    .stImage {
        margin: auto;
        display: block;
        margin: 12px 12px;
        border-radius: 15px;
    }
    .stButton>button {
        background-color: #006400;
        color: #ffffff !important;
```

```
    .stButton>button:hover {
        background-color: #004d00;
    }
    .header-title {
        text-align: center;
        font-weight: 700;
        font-size: 2.5em;
    }
    .sub-header {
        text-align: center;
        font-weight: 400;
        font-size: 1.5em;
    }
    .description {
        text-align: justify;
        font-weight: 300;
        font-size: 1.2em;
    }
    .offerings {
        font-weight: 300;
        font-size: 1.2em;
    }
    .faq-container {
        background-color: white;
        border-radius: 15px;
        padding: 20px;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        margin-bottom: 20px;
    }
    .faq-question {
        font-weight: 600;
```

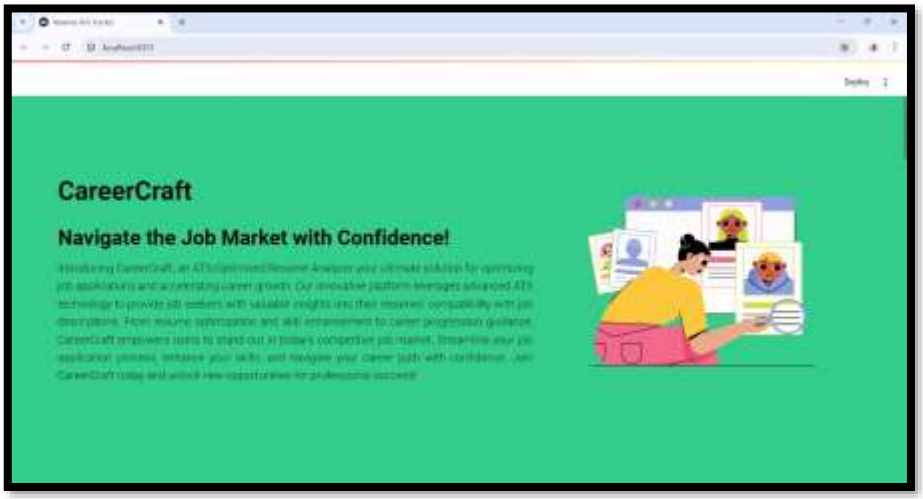
```
.stButton>button:hover {
    background-color: #004d00;
}
.header-title {
    text-align: center;
    font-weight: 700;
    font-size: 2.5em;
}
.sub-header {
    text-align: center;
    font-weight: 400;
    font-size: 1.5em;
}
.description {
    text-align: justify;
    font-weight: 300;
    font-size: 1.2em;
}
.offerings {
    font-weight: 300;
    font-size: 1.2em;
}
.faq-container {
    background-color: white;
    border-radius: 15px;
    padding: 20px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
}
.faq-question {
    font-weight: 400;
```

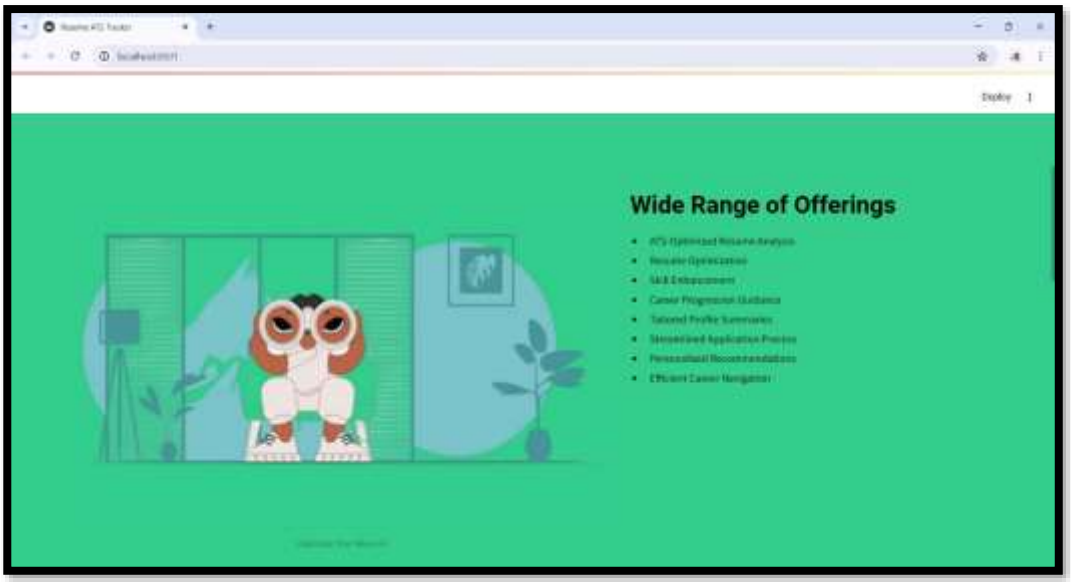
- **Hosting the application**

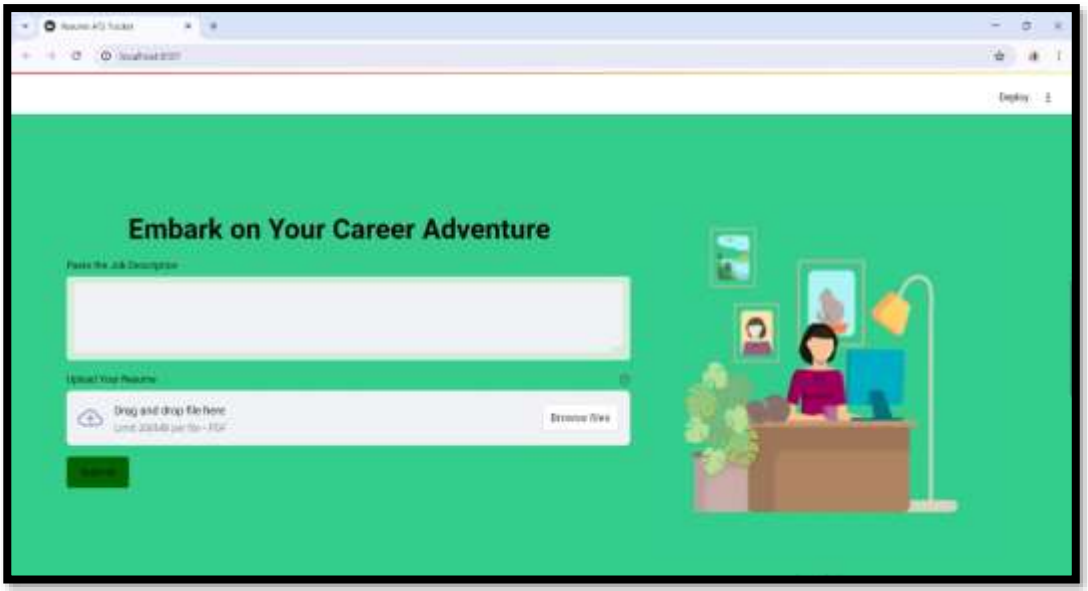
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CODE REFERENCE LOG   COMMENTS

PS D:\Project Vacation\ATS> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.20.17.70:8501
```

  - To host the application, go to the terminal, type - streamlit run app.py
  - Here app.py refers to a python script.

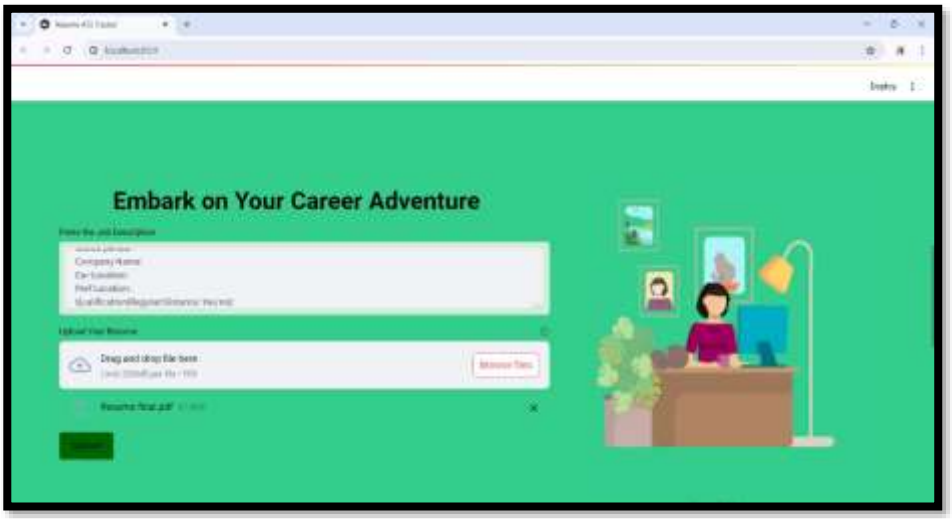
- **Website output:**



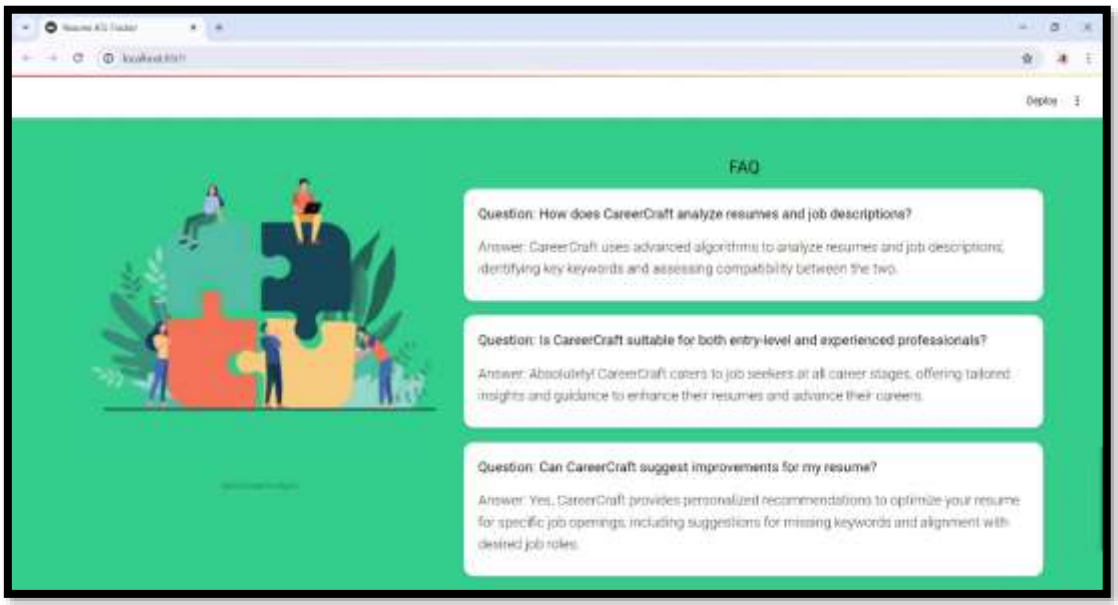

Wide Range of Offerings


Embark on Your Career Adventure
Drag and drop file here

- **Input :**


Embark on Your Career Adventure


Deploy
Analysis Result
Job Description (JD) Matching Percentage: 70%
Missing Keywords:
Profile Summary:
Match Percentage
70%


Deploy
FAQ
Question: How does CareerCraft analyze resumes and job descriptions?
Answer: CareerCraft uses advanced algorithms to analyze resumes and job descriptions, identifying key keywords and assessing compatibility between the two.
Question: Is CareerCraft suitable for both entry-level and experienced professionals?
Answer: Absolutely! CareerCraft caters to job seekers at all career stages, offering tailored insights and guidance to enhance their resumes and advance their careers.
Question: Can CareerCraft suggest improvements for my resume?
Answer: Yes, CareerCraft provides personalized recommendations to optimize your resume for specific job openings, including suggestions for missing keywords and alignment with desired job roles.

## Conclusion:

CareerCraft stands as a revolutionary tool in the realm of job applications, leveraging the power of the Gemini Model to optimize resumes for ATS systems, enhance skillsets, and guide career progression. By ensuring that resumes are both appealing to human recruiters and compatible with ATS software, CareerCraft significantly boosts job seekers' chances of securing interviews. The platform's ability to identify skill gaps and provide targeted recommendations enables users to continuously improve their qualifications and stay competitive in the job market. Furthermore, CareerCraft's strategic career guidance empowers professionals to make informed decisions and successfully navigate their career journeys. Overall, CareerCraft is an indispensable asset for job seekers aiming to maximize their potential and achieve their career aspirations.