

OPEN SOURCE JOB WINNING RESUME USING MACHINE LEARNING

Submitted in partial fulfillment of the requirement for the award of the Bachelor of
Engineering degree in Electronics and Communication Engineering

By

D.SAHITI (37130099)

D.LAKSHMI MEGHANA (37130118)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTROINCS ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL – 2021



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **D. SRI JAYA HARIKA SAHITI (37130099)** and **D. LAKSHMI MEGHANA (37130118)** who carried out the project entitled “**OPEN SOURCE JOB WINNING RESUME USING MACHINE LEARNING** ” under my supervision from September 2020 to April 2021.

.

Internal Guide

Dr. V. VIJAYAKUMAR, MTech, Ph.D

Head of Department

Dr. T. RAVI M.E, Ph.D

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We , **D. SAHITI (37130099)** and **D. LAKSHMI MEGHANA(37130118)** hereby declare that the Project Report entitled “**OPEN SOURCE JOB WINNING RESUME USING MACHINE LEARNING**” done by us under the guidance of Dr. V. Vijayakumar submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE:

PLACE: SATHYABAMA UNIVERSITY,CHENNAI

1. D. SAHITI (37130099)

2. D. LAKSHMI MEGHANA (37130118)

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. Nandhitha, M.E, PhD**, Dean, School of Electrical and Electronics and **Dr. T Ravi, M.E. PhD**, Head of Department, Dept. of Electronics and Communication Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr. V. Vijayakumar, MTech, Ph.D** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The objective of the project is to form a Profile Leveraging System using natural language processing. Approximately 80% of companies in India (and 90% of companies within the world) hunt for qualified candidates from job applications. The algorithm will parse resumes one by one and can create a Candidate Profile supporting the abilities mentioned within the resume. Resumes are filtered based on work description. This can be done by assigning an accuracy score to every resume by intelligently comparing them against the corresponding job description. The “Profile Leveraging System” is a web application hosted on the Heroku cloud platform which offers service to the users. Users can login into the system and check their resume scores respect to their job descriptions. Companies can use this system to shortlist the potential candidates easily in no time. Resumes play an important role in shortlisting candidates. Profile Leveraging System helps to leverage the resumes and to get their desired job easily for the job seekers. This project comes up with an open source software which helps companies/students to find the score of resumes of file extensions(pdf/docx/txt/rtf) scans with respective job description (excel sheet) for getting shortlisted in the company interviews easily.

TABLE OF CONTENTS

CHAPT ER NO.	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	(iv)
	ABSTRACT	(v)
	LIST OF FIGURES AND TABLE	(vii)
	LIST OF PROGRAMMING LANGUAGES	(viii)
	LIST OF TECHNOLOGIES	(viii)
1	INTRODUCTION	1
1.1	Objective	2
1.2	Background & Literature Survey	2
1.3	Organization of the Report	3
2	PROFILE LEVERAGING SYSTEM	5
2.1	Proposed System	5
2.2	Working Methodology	6

CHAPT ER NO.	TITLE	PAGE NO.
2.3	Standards	7
2.4	System Details	7
2.4.1	Software Details	7
2.4.2	Hardware	16
3	COST ANALYSIS	17
4	RESULTS AND DISCUSSION	18
5	CONCLUSION & FUTUREWORKS	24
6	APPENDIX	25
7	REFERENCES	85

LIST OF FIGURES

FIGURE NO.	FIGURE TITLE	PAGE NO.
1.1	Screening Candidates	1
2.1	System Block Diagram	5
2.2	Home Page	8
2.3	Login Page	9
2.4	Main Page	10
2.5	Upload Resume page	11
2.6	Upload Job	12
2.7	Description page	
2.7	Heroku Github	15
2.8	Connection	
2.8	Deploy Branch on	16
	Heroku	
4.1	Running the Python	19
	application	
4.2	Uploaded Resumes	20
4.3	Uploaded Job	20
	Description	
4.4	Sample Job	21
	Description	
4.5	Score of the Resumes	21
4.6	Report of the Resumes	22
4.7	Online Hosting	22

LIST OF PROGRAMMING LANGUAGE

Python

HTML

CSS

JAVASCRIPT

LIST OF TECHNOLOGIES

Web Technology

Machine Learning

Cloud Computing

CHAPTER 1 INTRODUCTION

Resume is the first impression to get on the shortlist for our dream job interview. Figure 1 shows how recent job applications become hard for the Talent Acquisition team of companies. Majority of the companies use the ATS system to shortlist potential candidates. But in many cases people don't take it seriously about their resumes. Finding your dream job or employer as a software professional is often tied to your resume. Crafting a curriculum vitae that swiftly and simply translate why you're better than other applicants is critical. Your education and certification are crucial areas that you need to make sure are easy to find on the resume. Employers value these credentials in applicants.



Figure 1.1 Screening Candidates

Your breadth of expertise in different programming languages and software programs also is main to list in your expertise section. And because communications and collaboration play a pivotal role in the interactions between you and other members

of the enlargement team, your interpersonal and communications skills are important to articulate-in one and the other your highlights as well as in the experiences section.

So this project comes up with an open source software which helps companies/students to find the score of resumes of file extensions(pdf/docx/text/doc) scans with respective job description (excel sheet) for getting shortlisted in the company interviews easily.

1.1 Objective

- design an efficient system which can scan the resume of the user with file extensions(Pdf/docx/txt/rtf) with the job description (xlsx).
- Profile leveraging system provides a login system and then the user can upload his files to check his resume matching score.
- When the user uploads his resume, text in the resume is extracted and then applied to predictive analytics algorithms to find the matching score.
- Profile leveraging system frontend is developed using Python Flask whereas score is predicted using natural language toolkit.
- Deploy this software online using Heroku cloud system.

1.2 Background and Literature Survey

Resume parser which extracts the data from the resume is available as “resume-parser” in python library. This is my base paper; the title of our base paper is “Resume Parser with Natural Language Processing”. It discussed a case, in which information from the resumes are extracted and converted into JSON data. It used the concept of JSON to showcase to the company employees based on the job description.

Developing an application was also a learning process for me, I referred to a paper titled This paper discusses web application development using Python and Flask.Flask uses Jinja Template Engine.

1.3 Organization of the Report

- Chapter 2 proposed system, methodology, hardware and software details.
- Chapter 3 gives the cost involved in the implementation of the project.
- Chapter 4 discusses results and implemented.
- Chapter 5 concludes.
- Chapter 6 consists of codes.
- Chapter 7 gives references.

CHAPTER 2

PROFILE LEVERAGING SYSTEM

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

The following block diagram (figure 2.1) shows the system architecture of this project.

SYSTEM DIAGRAM

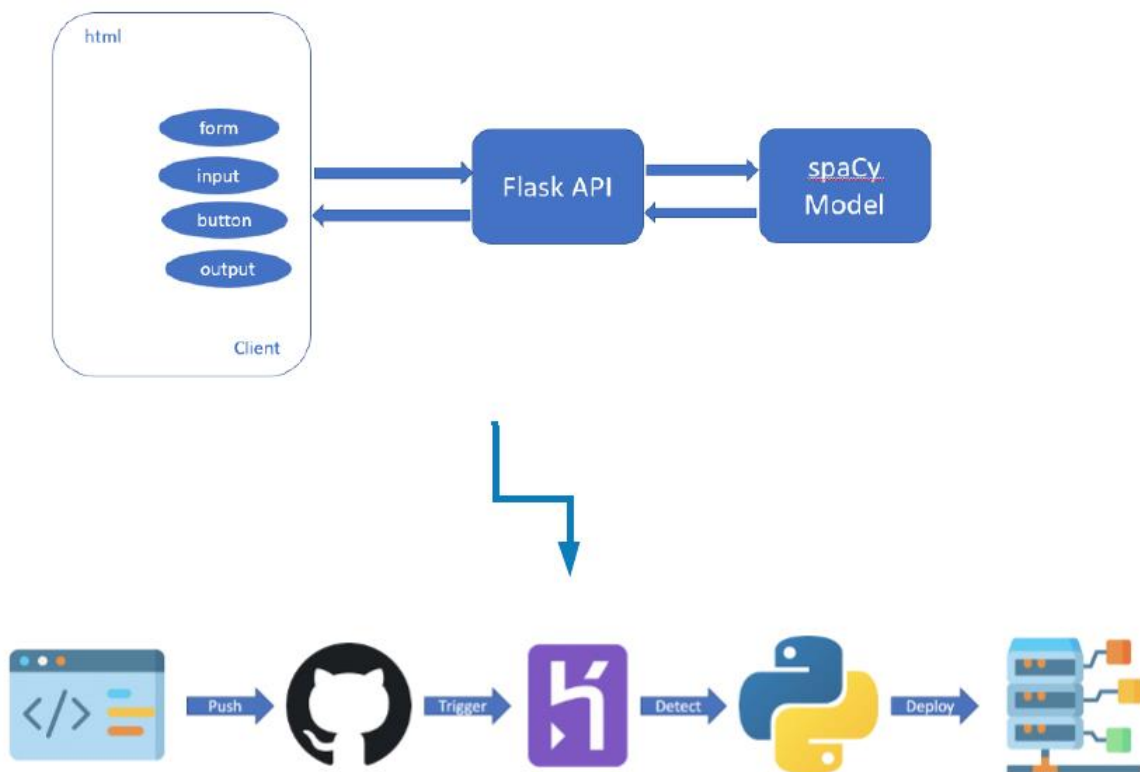


Figure 2.1 Block Diagram

2.2 Working Methodology

The system has two sections, hardware and software. Users with any mobile/tablet/computer with internet connection can use this system. After users logged into the system and then the user uploads the resume (file extension txt/rtf/docx/pdf) after that uploads the job description and gets the score of the resume that is the matching percentage of the resume with the job description.

2.3 Standards

Various standards used in this project are:

- **Heroku (CLI)**

Command Line Interface construct it easily to create & control your apps straightly from end .

- **Github**

Web Application is deployed on Github using GitBash. Flask uses Jinja Template Engine.

- **Flask**

Flask uses Jinja Template Engine.

2.4 System Details

This section describes the software and hardware details of the system:

2.4.1 Software Details

Python, HTML, CSS, JavaScript languages are used. Bootstrap and Codepen website are used for frontend design. Github and Heroku applications are used.

i) Web Application

The web application is built on the Python Flask platform with HTML, CSS and JavaScript programming languages. Bootstrap is used for the frontend part.

Developing Web Application

- Download Python IDE (3.3X version)
- Install Flask==1.1.2
- Install gunicorn==20.0.4
- Install Werkzeug==1.0.1
- Install Spacy, nltk and pandas.
- Install sklearn, gensim

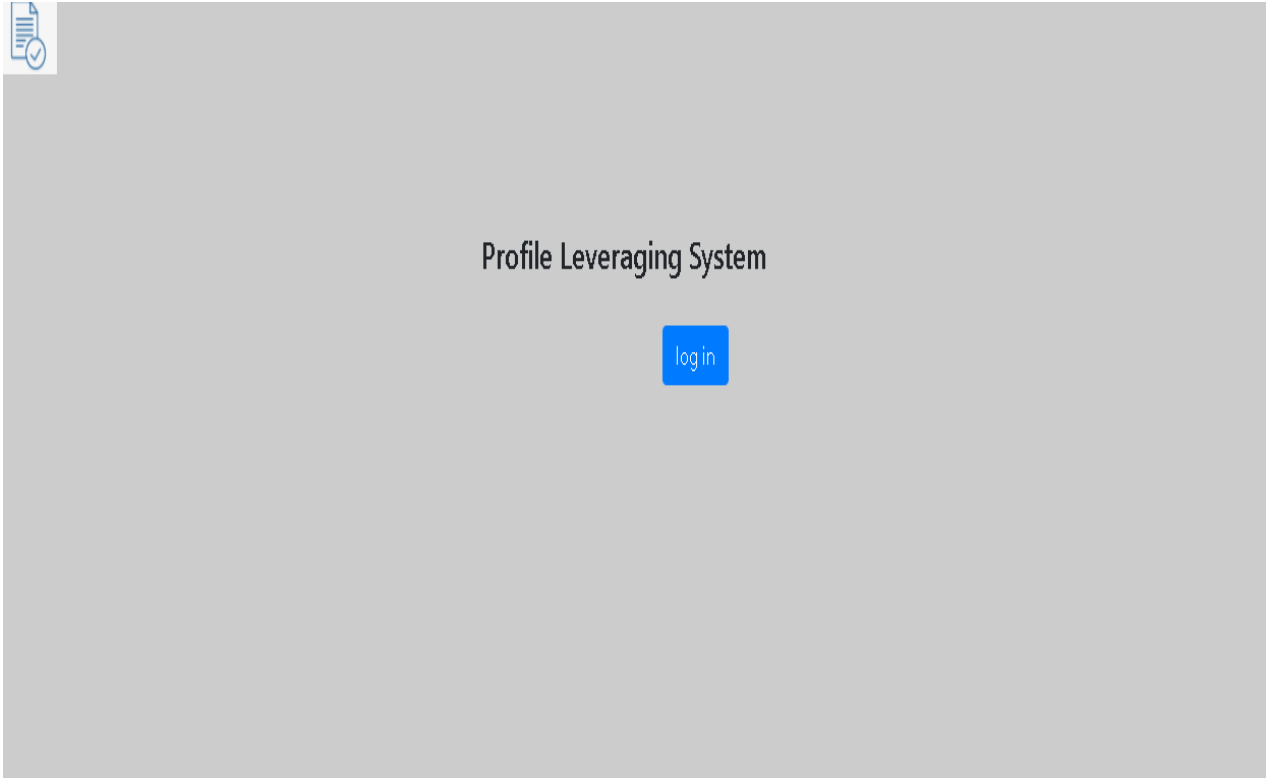
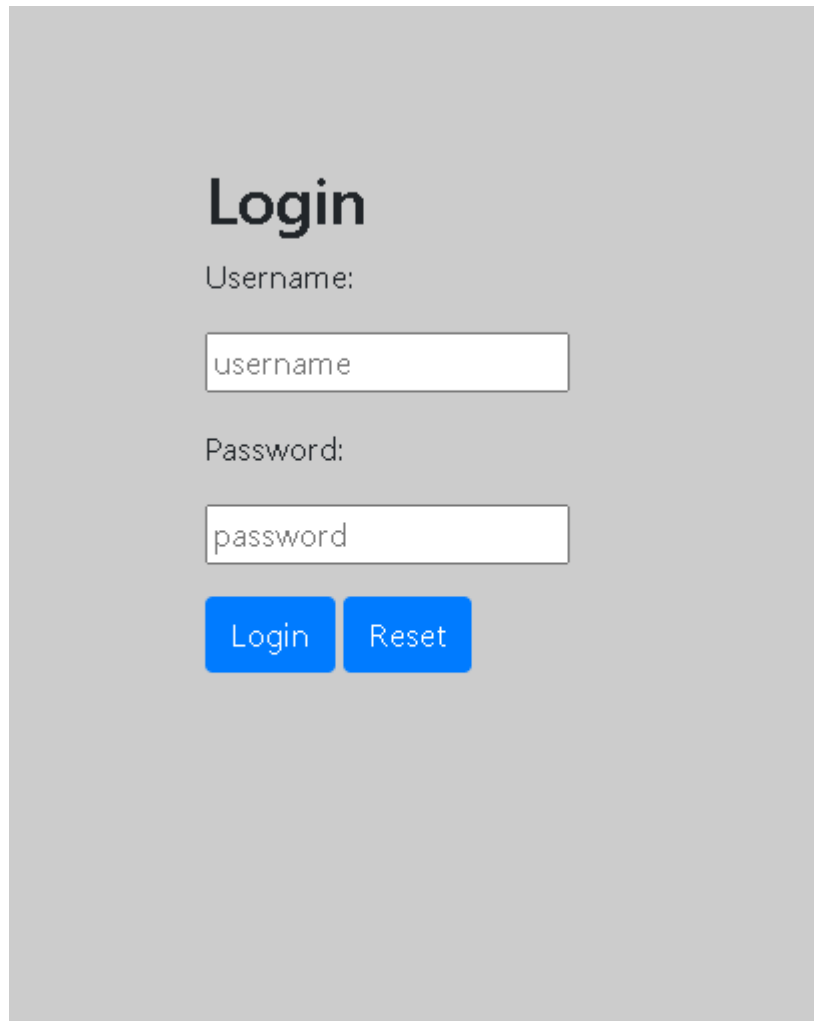


Figure 2.2 Home Page

Figure 2.2 home page is created, users first go to this page to login. And then come to the login page.

A login page with a light gray background. At the top, the word "Login" is displayed in a large, bold, black font. Below it, the text "Username:" is followed by a white rectangular input field containing the placeholder text "username". Underneath, the text "Password:" is followed by a white rectangular input field containing the placeholder text "password". At the bottom, there are two blue rectangular buttons with white text: "Login" on the left and "Reset" on the right.

Login

Username:

Password:

Login Reset

Figure 2.3 Login Page

In this page users login into the profile leveraging system figure 2.3

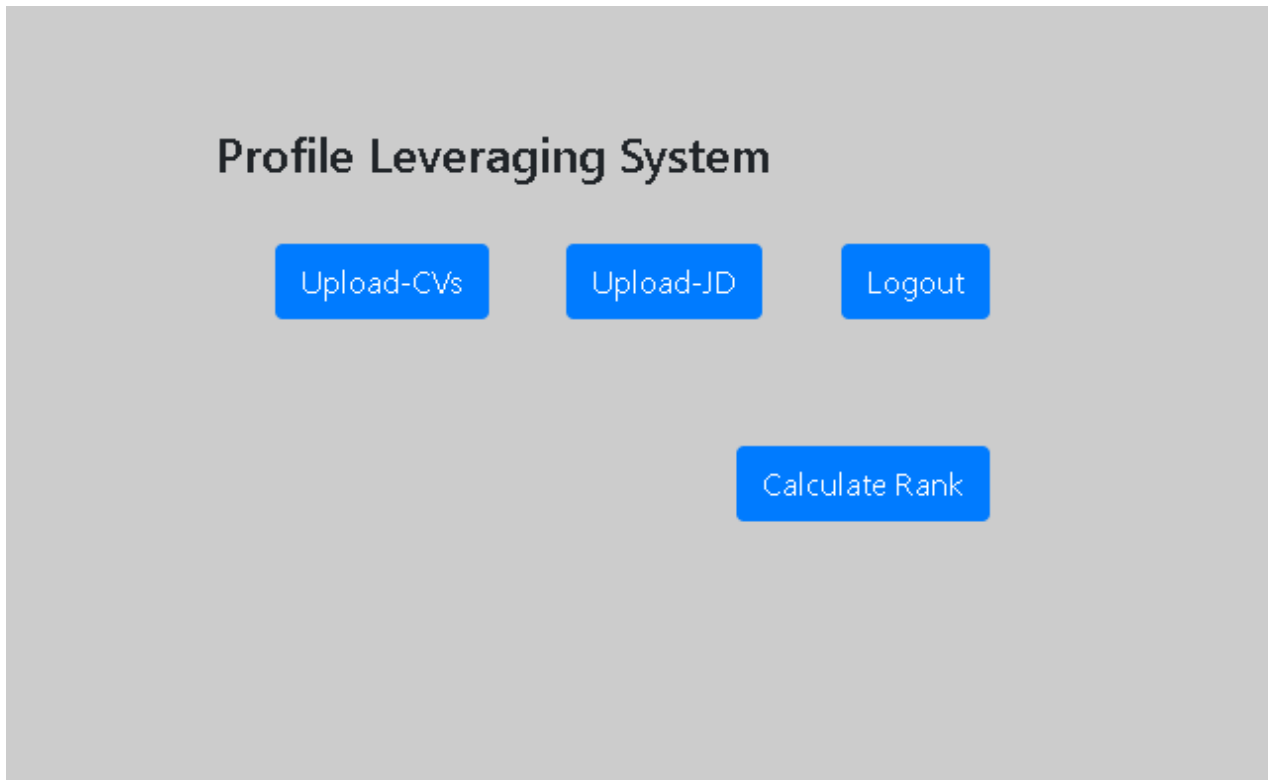


Figure 2.4 Main Page

This is the main page after users logged in. This main section is visible to the users. They can upload resumes using the Upload-CVs button. They can upload the job description using the Upload-JD button. They can calculate their rank using the Calculate Rank button.

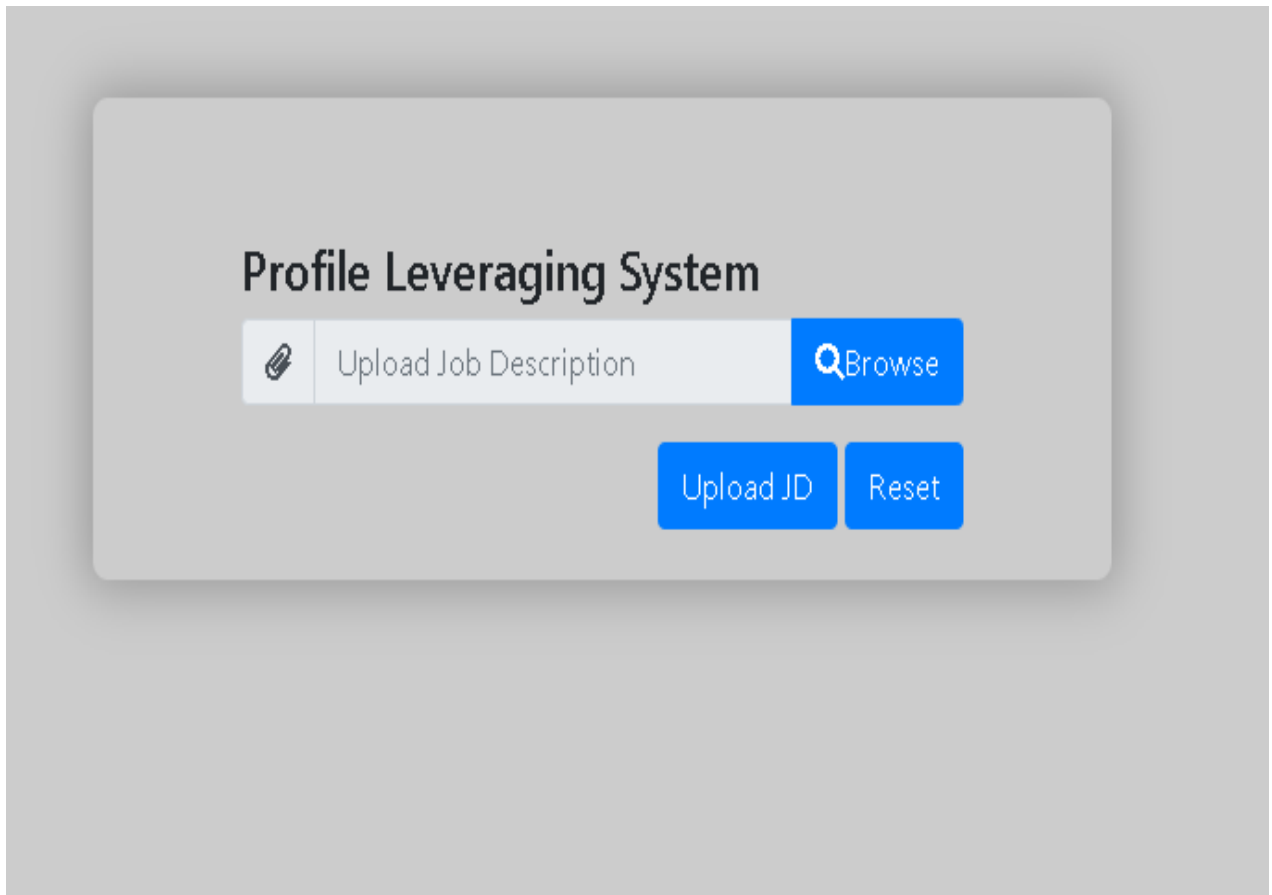


Figure 2.5 Upload Resume Page

users can use the Browse button to upload their resumes (txt/pdf/docx/rtf file extension).

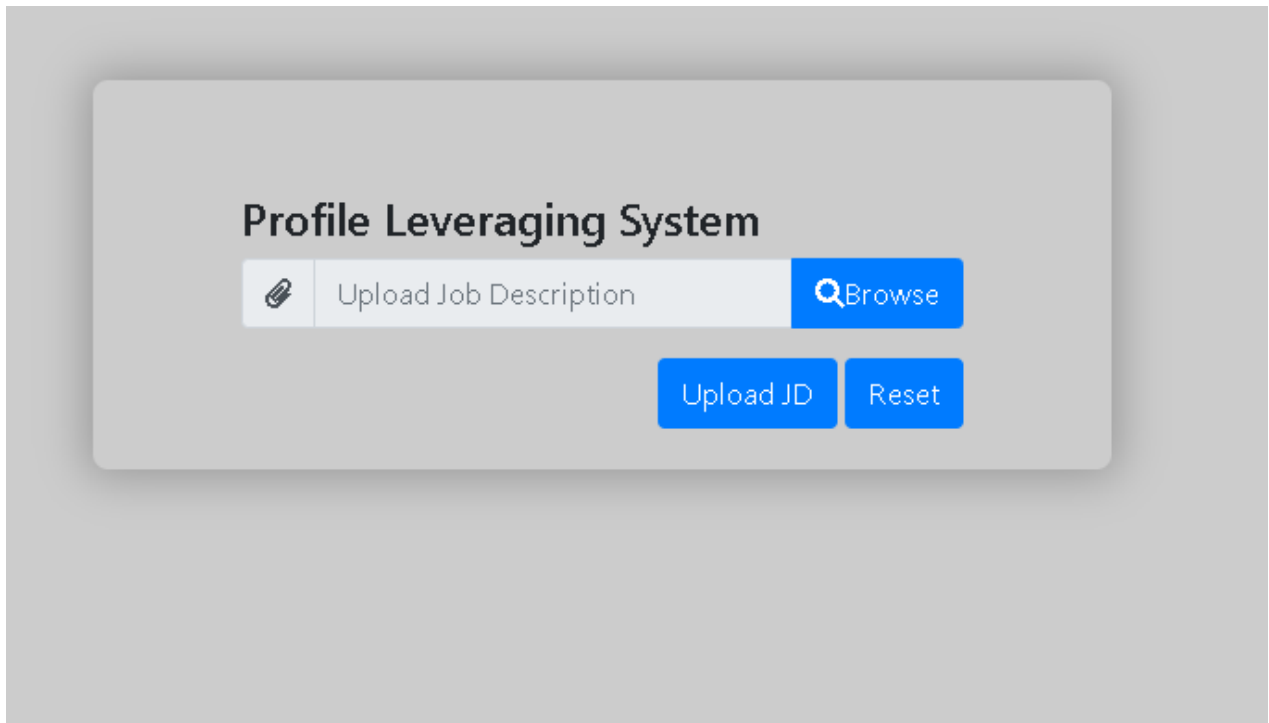


Figure 2.6 Upload Job Description page

users can use the Browse button to upload their job description (.xlsx file extension).

ii) Connecting Python Flask with web application

a. Login Page Connection

We can connect Python flask to the websites using below python code.

```
@apps.route
```

```
def login ():  
    error = none  
    return redirected(url__for('home'))  
    return render__templates('__login.html______',error=error)
```

b. Logout Page Connection

We can connect Python flask to the websites using below python code.

@ app.route

```
def logout ():  
    session.poped()  
    flash('logged out')  
    return redirect(url____for('home'))
```

c.Results Page Connection

We can connect Python flask to the websites using below python code.

@ app.route

```
define res():  
    glob.glob('*.*xlsx', recursive=False)  
    data_set = pd.read_excel(file[0])  
    search_st = data_set['High Level Job Description'][0]  
    skill_text = data_set['Technology'][0] + data_set['Primary Skill'][0]  
    jd_exp = data_set['Yrs Of Exp '][0]  
    title = data_set['Job Title'][0]  
    min_qual=data_set['Qualification'][0]  
    flask_return = core.res(search_st,skill_text,jd_exp,min_qual)  
    df = pd.DataFrame(columns=['Title','Experience','Primary Skill','Technology'])  
    df = df.append({'Title': title,'Experience':jd_exp,'Primary Skill':data_set['Primary  
Skill'][0],'Technology':data_set['Technology'][0]}, ignore_index=True)  
    return render_template('result.html', results = flask_return,jd = df)
```

d.UploadResume Page Connection

We can connect Python flask to the websites using below python code.

@ app.route

```
def uploadResume():
```

```
    return render____templates('uploadresume.html')
```

e.UploadJobDescription Page Connection

We can connect Python flask to the websites using below python code.

@ app.route

```
def uploadjdDesc():
```

```
    return render__templates('uploadjd.html')
```

iii) Deploying Web Application

- Make an account on <https://www.github.com>(Github)
- Upload the Web Application code on the github using GitBash.
- Make an account on <https://www.heroku.com/>(Heroku App)
- In Heroku, connect your github account.

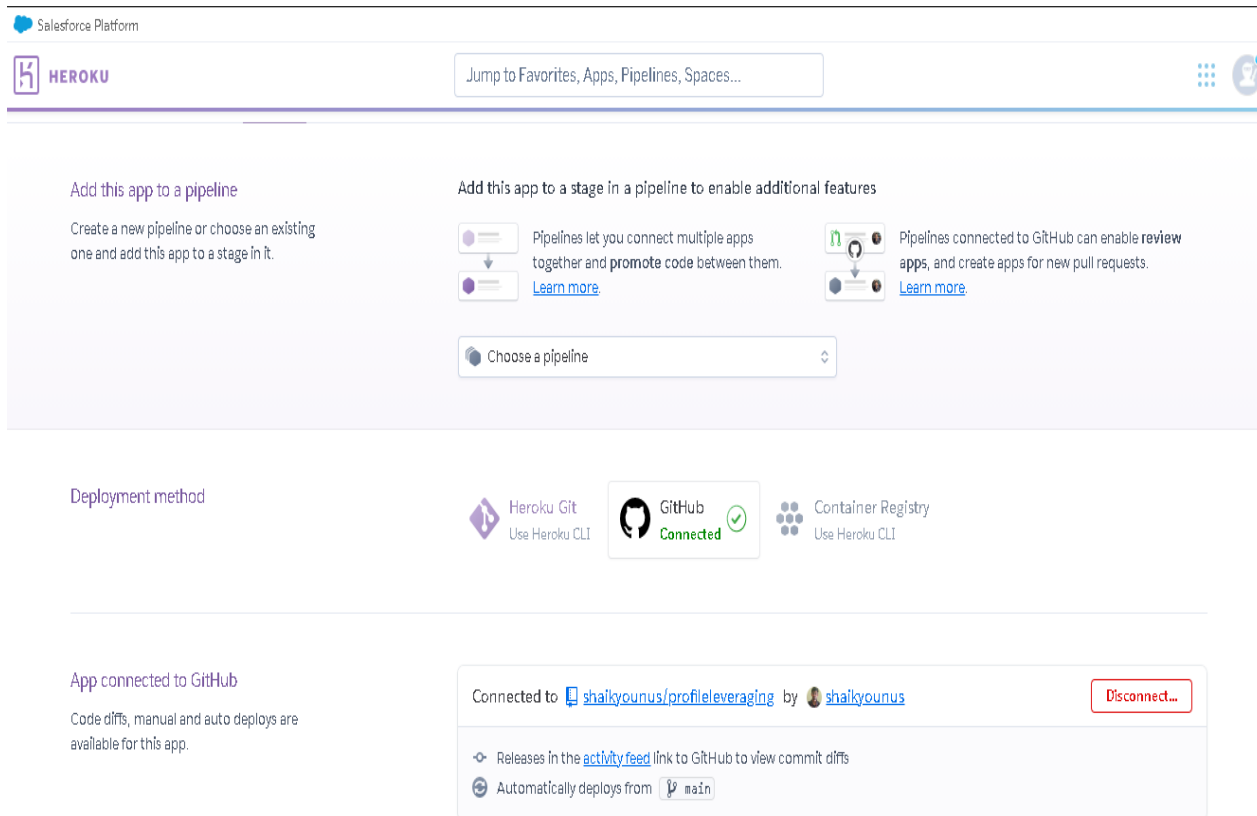


Figure 2.7 Heroku Github Connection

- And then Enable Automatic Deploys figure 9
- Next, click on deploy branch figure 9
- Heroku installs all the python modules present in requirements.txt file
- And then Web application is deployed successfully
- We can use the system at **<https://profileleverage.herokuapp.com/>**

✔ Automatic deploys from `main` are enabled

Every push to `main` will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. [Learn more](#).

☒ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Disable Automatic Deploys

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

 main

⌵

Deploy Branch

Figure 2.8 Deploy Branch on Heroku

2.4.2 Hardware Details

- Any Mobile phone/ Tablet/ Laptop with internet connection can use this Profile Leveraging System web application efficiently.
- To run this application in your local host
 - A. 1.6GHz processor or more
 - B. 1GB RAM or more
 - C. ROM memory

CHAPTER 3

COST ANALYSIS

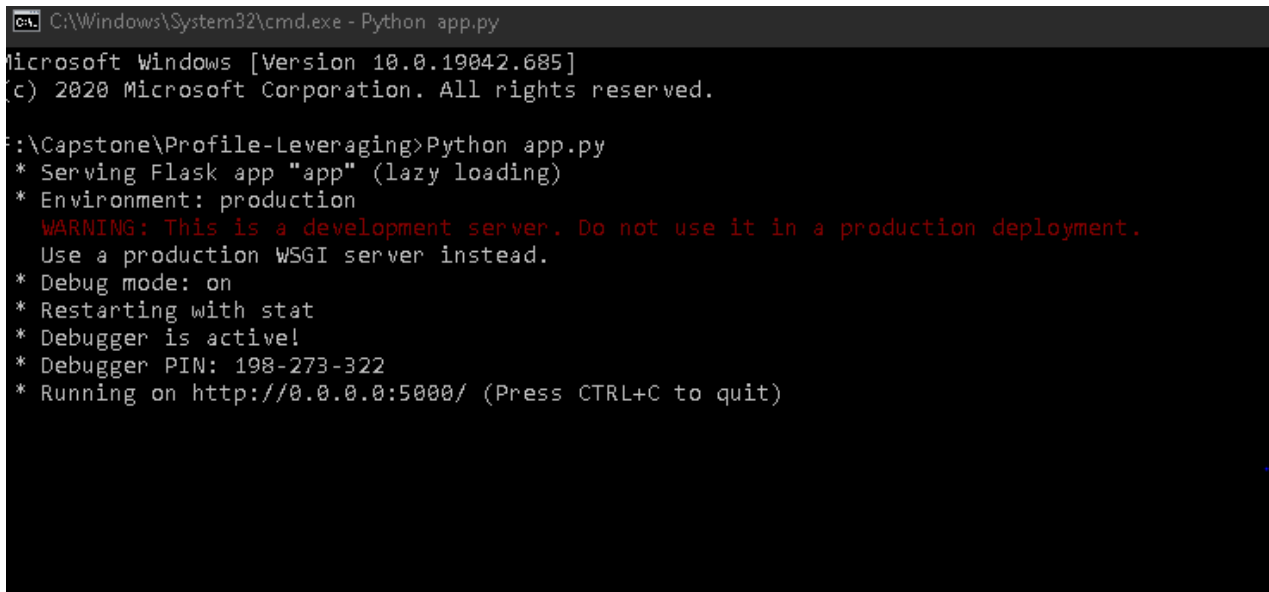
Since there is no hardware involved in this application it is cost-free application. Heroku offers 650 hours free hosting. After that we need to pay the money for cloud usage on demand.

CHAPTER 4

RESULTS AND DISCUSSIONS

a. Score of resumes(txt/rtf/docx/pdf)

After uploading the resumes on the system and uploading the job description figure 10 we get the output figure 11. Which resume has more score is considered as the best resume for the given job description. Score is calculated as based on Technical skills, non-Technical skills and Job description matching with the resume.

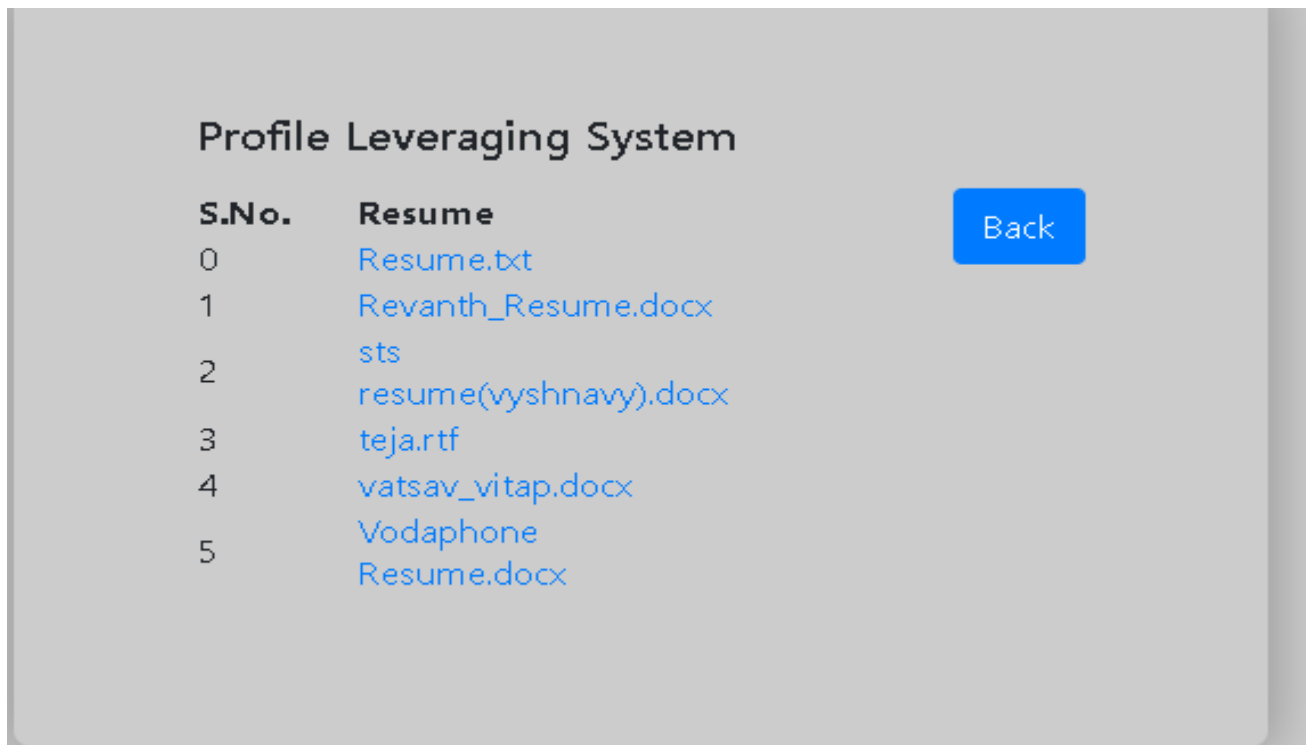
A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\System32\cmd.exe - Python app.py". The window content shows the following text:

```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Capstone\Profile-Leveraging>Python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 198-273-322
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Figure 4.1 Running the Python application

b. Uploading resumes



The screenshot shows a web interface titled "Profile Leveraging System". It contains a table with two columns: "S.No." and "Resume". The table lists six entries. To the right of the table is a blue button labeled "Back".

S.No.	Resume
0	Resume.txt
1	Revanth_Resume.docx
2	sts resume(vyshnavy).docx
3	teja.rtf
4	vatsav_vitap.docx
5	Vodaphone Resume.docx

Figure 4.2 UPLOADED RESUMES

After uploading the resumes the Upload-Resume page is seen figure 4.2.

c. Upload Job Description

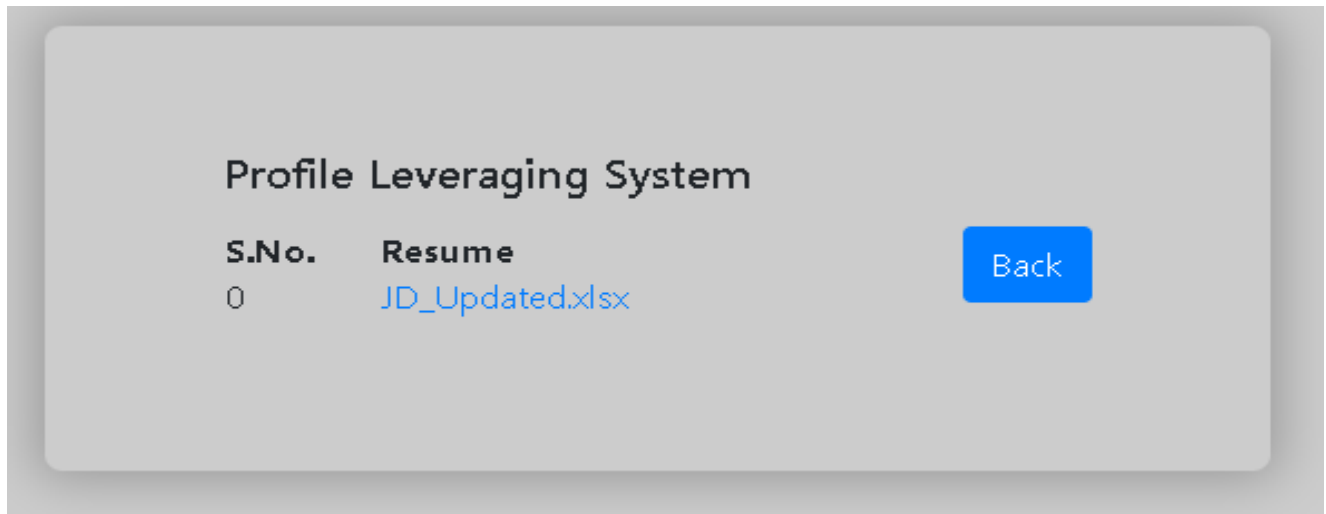


Figure 4.3 UPLOADED JOB DESCRIPTION

After uploading the resumes the Upload-Resume page is seen figure 4.3.

d. Sample Job Description

A	B	C	D	E	F	G
Job Title	Yrs Of Exp	Relevant Yrs Of Exp	Qualification	High Level Job Description	Technology	Primary Skill
Software Engineer	0	0	BE/BTECH	Currently enrolled in a Bachelor's or Master's Degree in Computer Science, Computer Engineering, or related field at time of application. Knowledge of the syntax of languages such as Java, C/C++ or Python. Knowledge of Computer Science fundamentals such as object-oriented design, algorithm design, data structures, problem solving, and complexity analysis. Database Architecture Product Enhancement Problem Solving Skills Linux/Unix Perl Shell Optimization Strategic Planning Design Reviews Agile Scrum Team Computer Architecture Operating Systems Data Structures	Engineering , Information Technology	JavaScript, Java , Python , CSS , HTML , C, C++ , SQL

Figure 4.4 SAMPLE JOB DESCRIPTION

Users need to upload their Job description figure 4.4.

e. Score of the Resumes

Figure 14 phone, email entities are extracted from the resumes. And the weightage is given 50 percent to the Job description matching with the resume another 50 percent is given to the Technical s and Non-Technical skills matching Job description. First Job description is summarised using Gensim and made a Job description vector. Second resumes are converted into text files and then information extracted and made a resumes vector. Then the cosine similarity is applied between resumes vector and Job description vector. Value given by cosine similarity function helps to found out the Job description matching with the resumes and also the skills(Technical + Non-Technical skills) matching with the job description skills.

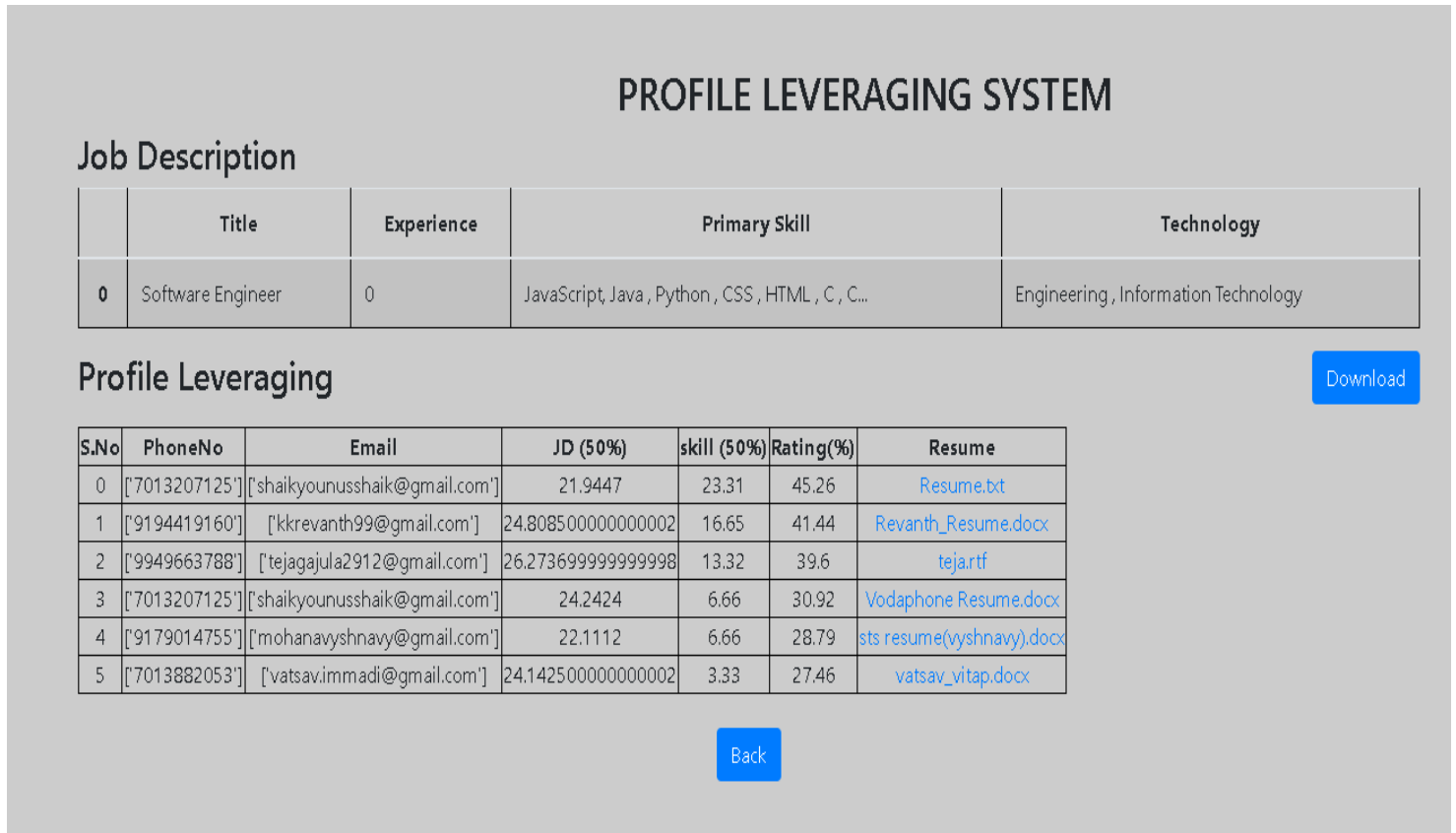


Figure 4.5 Score of the Resumes

f. Report in CSV

A	B	C	D	E	F	G
S.No.	PhoneNo	Email	JD (15%)	skill (40%)	Rating(%)	Resume
0	['7013207125']	['shaikyounusshaik@gmail.c	21.9447	23.31	45.26	Resume.txt
1	['9194419160']	['kkrevanth99@gmail.com']	24.8085	16.65	41.44	Revanth_Resume.docx
2	['9949663788']	['tejagajula2912@gmail.com	26.2737	13.32	39.6	teja.rtf
3	['7013207125']	['shaikyounusshaik@gmail.c	24.2424	6.66	30.92	Vodafone Resume.docx
4	['9179014755']	['mohanavyshnavy@gmail.c	22.1112	6.66	28.79	sts resume(vyshnavy).docx
5	['7013882053']	['vatsav.immadi@gmail.com	24.1425	3.33	27.46	vatsav_vitap.docx

Figure 4.6 Report of the Resumes

Users can download the report of the resumes score in the CSV file figure 4.6.

g. Online hosting in Heroku Application

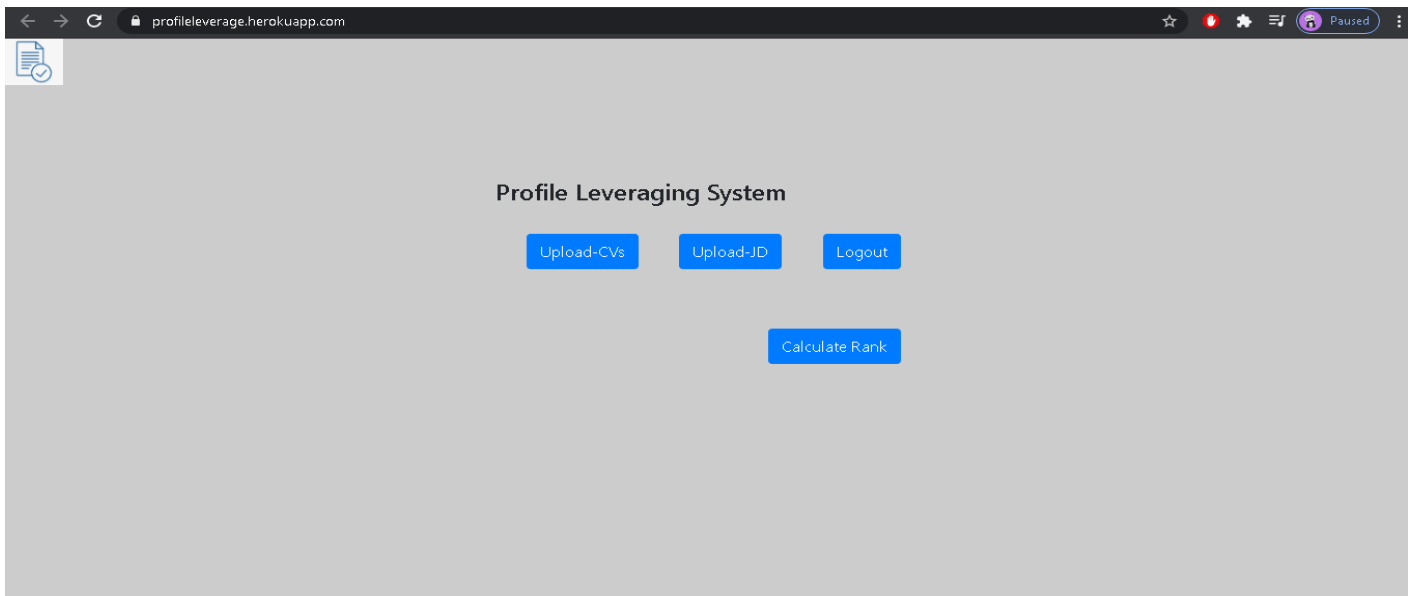


Figure 4.7 Online Hosting

Figure 16 this system is hosted online using heroku application. Users can use this application from the given link .

CHAPTER 5

CONCLUSION AND FUTURE WORK

Hiring potential candidates is a difficult task for many companies. Tech Giants uses their ATS system to shortlist candidates for interviews. Whereas middle order companies pay money to resume matching softwares to shortlist. Without paying any money “Profile Leveraging System” comes in handy to recruiters to shortlist candidates easily and also it is useful for job seeking candidates to know how much percentage they are eligible to the role they are going to apply beforehand.

Future work is to search for the company’s all job profiles and automatically the system will predict which role is best suitable for the user whey they upload their resume. And to develop an open source resume builder with modern layouts and templates to avoid the old convention of writing resumes.

CHAPTER 6

APPENDIX

App Code

```
import warnings
import core
import search
import pandas as pd
import nltk
#nltk.download('maxent_ne_chunker')

    USERNAME='admin',
    PASSWORD='admin',
    SECRET_KEY='development key',
))
app.config['UPLOAD_FOLDER'] = r'F:\Capstone\Profile-Leveraging\Upload-Resume'
app.config['UPLOAD_JD_FOLDER'] = r'F:\Capstone\Profile-Leveraging\Upload-
Jobdescription'
app.config['ALLOWED_EXTENSIONS'] = set(['txt', 'pdf', 'doc', 'docx'])
class jd:
    getfilepath(loc):
        temp = str(loc).split('\\')
        return temp[-1]
    @ app.route
    define login ():
```

```

os.chdir(app.config['UPLOAD_JD_FOLDER'])
file = glob.glob('*.*xlsx', recursive=False)
data_set = pd.read_excel(file[0])
search_st = data_set['High Level Job Description'][0]
skill_text = data_set['Technology'][0] + data_set['Primary Skill'][0]
jd_exp = data_set['Yrs Of Exp '][0]
title = data_set['Job Title'][0]
min_qual=data_set['Qualification'][0]
flask_return = core.res(search_st,skill_text,jd_exp,min_qual)
df = pd.DataFrame(columns=['Title','Experience','Primary Skill','Technology'])
df = df.append({'Title': title,'Experience':jd_exp,'Primary Skill':data_set['Primary Skill'][0],'Technology':data_set['Technology'][0]}, ignore_index=True)
return render_template('result.html', results = flask_return,jd = df)

```

```

@ app.route ('/uploadResume')
def uploadResume():
    return render_templates('uploadresume ')

    x = os.listdir(app.config['UPLOAD_FOLDER'])
    return render_templates("resultlist ", name=x)

```

```

@ app.route
def uploadjdDesc():
    return render__templates('uploadjd')

```

```

("resume-jd",os.getcwd())

```

```

os.chdir(app.config['UPLOAD_JD_FOLDER'])
file = glob.glob('*.*xlsx', recursive=False)
data_set = pd.read_excel(file[0])

```

```

search_st = data_set['High Level Job Description']][0]
result = search.res(search_st)
# return result
return render_template('result.html', results = result)

@app.route('/Upload-Resume/<path:filename>')
def custom_static(filename):
    return send_from_directory('./Upload-Resume', filename)

if __name__ == '__main__':
    # app.run(debug = True)
    # app.run('127.0.0.1', 5000, debug=True)
    app.run('0.0.0.0', 5000, debug=True, threaded=True)

```

Core.py Code

```

self.finalRank = finalRank
self.skillList = skillList
self.nonTechskillList = nonTechskillList
self.min_qual = min_qual
self.is_min_qual = is_min_qual

def getfilepath(loc):
    temp = str(loc)
    temp = temp.replace('\\', '/')
    return temp

```

```

def res(jobfile,skillset,jd_exp,min_qual):
    Resume_Vector = []
    Resume_skill_vector = []
    min_qual_vector = []
    is_min_qual = []
    Resume_skill_list = []
    Resume_non_skill_list = []
    Resume_email_vector = []
    Resume_phoneNo_vector = []
    Resume_ApplicantName_vector = []
    Resume_name_vector = []
    Resume_nonTechSkills_vector = []
    Resume_exp_vector = []
    Ordered_list_Resume = []
    _FILES = []
    _FILES_PDF = []
    _FILES_DOC = []
    _FILES_DOCX = []
    Resumes = []
    Temp_pdf = []
    Resume_title = []
    jd_weightage = 15
    not_found = 'Not Found'
    extract_exp = ExtractExp()

    resumePath = r'F:\Capstone\Profile-Leveraging\Upload-Resume'

    for count,i in enumerate(LIST_OF_FILES):

        Temp = i.rsplit('.', 1)

```

```

Resume_title.extend(str(i))
if Temp[1] == "pdf" or Temp[1] == "Pdf" or Temp[1] == "PDF":
    try:
        print(count," This is PDF" , i)

        with open(i,'rb') as pdf_file:
            for page in p.extract_text_by_page(i):

                page = page.replace('\n', ' ')
                Temp_pdf = str(page)
                Resumes.extend([Temp_pdf])
                Temp_pdf = "
                Ordered_list_Resume.append(i)

    except Exception as e:
        print(e)
        print(traceback.format_exc())
elif Temp[1] == "doc" or Temp[1] == "Doc" or Temp[1] == "DOC":
    print(count," This is DOC" , i)

    #parse_docfile(i)

elif Temp[1] == "rtf" or Temp[1] == "Rtf" or Temp[1] == "RTF":
    print(count," This is Rtf" , i)

    try:

        rtf_path = Path(i)
        with rtf_path.open() as source:
            docText = rtf_to_text(source.read())

```

```

        c = [docText]
        Resumes.extend(c)
        Ordered_list_Resume.append(i)
    exceptException as e: print(e)

elif Temp[1] == "docx" or Temp[1] == "Docx" or Temp[1] == "DOCX":
    print(count," This is DOCX" , i)
    try:

        c = [str(a)]
        Resumes.extend(c)
        Ordered_list_Resume.append(i)
        f.close()
    exceptException as e: print(e)

elif Temp[1] == "ex" or Temp[1] == "Exe" or Temp[1] == "EXE":
    print("This is EXE" , i)
    pass

print("Done Parsing.")
print("Please wait we are preparing ranking.")

Job_Desc = 0

try:
    tttt = str(jobfile)
    tttt = summarize(tttt, word_count=100)
    text = [tttt]
except:
    text = 'None'

```

```

#vectorizer = TfidfVectorizer(stop_words='english')
vectorizer = TfidfVectorizer()

# print(text)

print("This is jobbbbbbb")
print(Job_Desc)
# print("\n\n")
# print("This is job desc : " , Job_Desc)
tempList = Ordered_list_Resume
#os.chdir('../')
flask_return = []

for index,i in enumerate(Resumes):
    text = i
    temptext = str(text)
    tttt = str(text)
    print("This is resume text")

    try:
        #tttt = summarize(tttt, word_count=1000)
        text = [tttt]

        Resume_Vector.append(vector.toarray())
        #min_qual_score = skills.minQualificationScore(temptext,min_qual)
        min_qual_score = skills.programmingScore(temptext,min_qual,progWords =
None)

```

```

print(min_qual_score)

min_qual_vector.append(min_qual_score)
confidence = {}
#score = int((min_qual_score/globals.min_qual_weightage)*100)
score = int((min_qual_score/10)*100)

confidence['confidence'] = score
if score >= 60:
    confidence['min qual'] = 'Yes'
elif score < 60 and score > 0:
    confidence['min qual'] = 'May Be'
else:
    confidence['min qual'] = 'Yes'
is_min_qual.append(confidence)

```

```

Resume_skill_vector.append(skills.programmingScore(temptext,jobfile+skillset,progW
ords = None))

```

```

Resume_skill_list.append(skills.NonTechnicalSkillScore(temptext,jobfile+skillset,prog
Words = None))

```

```

Resume_non_skill_list.append(skills.NonTechnicalSkillScore(temptext,jobfile+skillset,
progWords = None))
    experience = extract_exp.get_features(temptext)
    Resume_name_vector.append(experience)
    temp_applicantName = entity.extractPersonName(temptext,
Resume_title[index])
    Resume_ApplicantName_vector.append(temp_applicantName)
    temp_phone = entity.extract_phone_numbers(temptext)
    if(len(temp_phone) == 0):

```

```

        Resume_phoneNo_vector.append(not_found)
    else:
        Resume_phoneNo_vector.append(temp_phone)
    temp_email = entity.extract_email_addresses(temptext)
    if(len(temp_email) == 0):
        Resume_email_vector.append(not_found)
    else:
        Resume_email_vector.append(temp_email)

Resume_exp_vector.append(extract_exp.get_exp_weightage(str(jd_exp),experience)
)

Resume_nonTechSkills_vector.append(skills.NonTechnicalSkillScore(temptext,jobfile
+skillset))

    print("Rank prepared for ",Ordered_list_Resume.__getitem__(index))
exceptException:
    print(traceback.format_exc())
    tempList.__delitem__(index)

for index,i in enumerate(Resume_Vector):
    samples = i
    print("This is sampleees")
    print(samples)
    similarity = cosine_similarity(samples,Job_Desc)[0][0]
    final_rating =
round((similarity*jd_weightage)*3.33,2)+(Resume_skill_vector.__getitem__(index))*3.
33

```

```

        res = ResultElement(round(similarity*jd_weightage,2)*3.33,
os.path.basename(tempList.__getitem__(index)),round(Resume_skill_vector.__getitem__(index)*3.33,2),
        Resume_name_vector.__getitem__(index),
Resume_phoneNo_vector.__getitem__(index),Resume_email_vector.__getitem__(index),

Resume_nonTechSkills_vector.__getitem__(index),Resume_exp_vector.__getitem__(index),round(final_rating,2),Resume_skill_list.__getitem__(index),

Resume_non_skill_list.__getitem__(index),min_qual_vector.__getitem__(index),is_min_qual.__getitem__(index))
        flask_return.append(res)
    flask_return.sort(key=lambda x: x.finalRank, reverse=True)
    return flask_return

```

ExtractEntities.py

```

import spacy
import re

#Function to extract names from the string using spacy
def extract_name(resume):

    nlp = en_core_web_sm.load()
    (ent.label_ == 'PER'):

```

```

        #print(ent.text)
        break
    return ent.text
def extract_phone_numbers(string):

extractPersonName(tttt, resumeTitle):
    #a = 'Cv_saurabh+keshari_1234_Resume'
    #a = "1234"
    titleSplit = re.split(r'[-~!@#$%^&*()_+\\[\]{};\\\:|<./<>?]', resumeTitle)
    title_isNotDigit = []
    for word in titleSplit:
        if not word.isdigit():
            title_isNotDigit.append(word)
    strr = " ".join(title_isNotDigit)
    strr_list = strr.split(" ")
    #titleSplit = re.findall(r'[w]'+",", strr)

    Names = []
    for Nouns in strr_list:
        if not wordnet.synsets(Nouns):
            #Not an English word
            Names.append(Nouns)
    titleSplit_lower = to_lowercase(Names)
    titleSplit_cleaned = remove_stopwords(titleSplit_lower)
    personName = "ABC"
    if len(titleSplit_cleaned) > 0:
        if len(titleSplit_cleaned) == 1:
            firstName = titleSplit_cleaned[0]
            personName = firstName
        if len(titleSplit_cleaned) == 2:

```

```

    firstName = titleSplit_cleaned[0]
    secondName = titleSplit_cleaned[1]
    personName = firstName + " " + secondName
if len(titleSplit_cleaned) >= 3:
    firstName = titleSplit_cleaned[0]
    secondName = titleSplit_cleaned[1]
    thirdName = titleSplit_cleaned[2]
    personName = firstName + " " + secondName + " " + thirdName

else:
    print("within else")
    TITLE = r"(?:[A-Z][a-z]*\.\s*)?"
    NAME1 = r"[A-Z][a-z]+,\?\s+"
    MIDDLE_I = r"(?:[A-Z][a-z]*\.\?\s*)?"
    NAME2 = r"[A-Z][a-z]+"
    name = []
    name = re.findall(TITLE + NAME1 + MIDDLE_I + NAME2, tttt)
    if len(name) > 0:
        rep1 = {"Career": "", "career": "", "Objective": "", "objective": "", "Email": "",
"email": "", "Experience Summary": "", "ph": "", "Ph": "", "Professional": "", "Curriculum
Vitae": "", "Resume": "", "Profile": "", "Professional": "", "Recruiter": "", "Lead": "",
"Summary": "", "HR": "", "new": "", "New": ""} # define desired replacements here

name[0])
    else:
        personName = "No name found"

stopwords = ['HR', 'hr', 'pdf', 'xls', 'docx', 'doc', 'rtf', 'txt', "assembly", "bash", " c "
"c++", "c#", "coffeescript", "emacs lisp",
"go!", "groovy", "haskell", "java", "javascript", "matlab", "max MSP", "objective c",
"qlikview", "crm", "CRM", "dynamics",

```

```

    "perl", "php", "html", "xml", "css", "processing", "python", "ruby", "sml", "swift",
    "resume", "Resume",
    "latex", "unity", "unix", "visual basic", "wolfram language", "xquery", "sql",
    "node.js", "finance", "liferay",
    "scala", "kdb", "jquery", "mongodb", "CMMI", "ISO", "finance", "Banking",
    "Finacle", "Oracle Flexcube", "Fiserv", "successfactor", "sf", "success factor",
    "Success Factor",
    "TCS BaNcs", "FIS Profile", "Self-directed learning", "Collaboration",
    "Communication", "Resilience", "Big-picture mindset", "Prioritization ", "Creativity ",
    "creative", "Insight", "curiosity", "curious", "Openness", "Teamwork", "Time
    management", "Emotional intelligence",
    "quick learner", "problem solver", "Customer-service skills", "Planning and
    skills", "Sociability", "Self-management", "Integrity", "Honesty", "Human resources",
    "Participates as a team member", "Works with diversity", "Exercises leadership",
    "leadership", "Exercises leadership", "Monitors and corrects performance",
    "Understands systems", 'experience', 'exp', 'exp.']
    querywords = personName

```

getCategory.py

```

list1 = jdSkillMatched
list2 = resumeCorpus
results = {}
for i in list1:
    results[i] = list2.count(i)

print("Dictionary is ", results)

#end of code

constantValue def programmingScore(resume, jdTxt, progWords = None):
    skill_weightage = 40

```

```

skill_threshold = 5
fout = open("results.tex", "a")
fout.write("\\textbf{Programming Languages:} \\\\n")

if(progWords == None):
    programming = ["assembly", "bash", "c", "c++", "c#", "coffeescript", "emacs lisp",
        "go!", "groovy", "haskell", "java", "javascript", "matlab", "max MSP", "objective c",
        "perl", "php", "html", "xml", "css", "processing", "python", "ruby", "sml", "swift",
        "latex", "unity", "unix", "visual basic", "wolfram language", "xquery", "sql",
        "node.js",
        "scala", "kdb", "jquery", "mongodb", "CMMI", "ISO", "finance", "Banking",
        "Finacle", "Oracle Flexcube", "Fiserv",
        "TCS BaNcs", "FIS Profile"]
    else:
        programming = progWords
    programmingTotal = 0

jdSkillCount = 0
jdSkillMatched = []

for i in range(len(programming)):
    if programming[i].lower() in jdTxt.lower() != -1:
        jdSkillCount += 1
        jdSkillMatched.append(programming[i].lower())
print("jdSkillCount", jdSkillCount)

for x in range(len(jdSkillMatched)):

    print("jd Skills matched are ",jdSkillMatched)
#END

```

```

individualSkillWeightage = 0

if( jdSkillCount > 0):
    individualSkillWeightage = skill_weightage/jdSkillCount

ResumeProgrammingSkillsMatchedWithJD = []
for i in range(len(jdSkillMatched)):
    if jdSkillMatched[i].lower() in resume.lower() != -1:
        programmingTotal += 1
        ResumeProgrammingSkillsMatchedWithJD.append(jdSkillMatched[i].lower())
    if not("#" in jdSkillMatched[i]):
        fout.write(jdSkillMatched[i]+" ", " ")
print("Resume skills matched with JD are ",
ResumeProgrammingSkillsMatchedWithJD)
print("programming total is ", programmingTotal)

# My Code
resumeCorpus = resume.split()
resumeCorpus = []
jdSkillMatched = []
= (individualSkillWeightage/skill_threshold)
# Updating Dictionary
results.update({n: constantValue * results[n] for n in results.keys()})
print("updated dict is ", results)

TotalScore = sum(results.values())
print("Score is ", TotalScore)

fout.close()

progScore = min(programmingTotal/10.0, 1) * 5.0

```

```

return TotalScore

def NonTechnicalSkillScore(resume, jd_txt, progWords = None):
    skill_weightage = 5
    skill_threshold = 5
    fout = open("results.tex", "a")
    fout.write("\\textbf{Programming Languages:} \\\\n")

    if(progWords == None):
        NonTechnicalSkill = ["Self-directed learning", "Collaboration", "Communication",
"Resilience", "Big-picture mindset", "Prioritization ", "Creativity ",
        "creative", "Insight", "curiosity", "curious", "Openness", "Teamwork", "Time
management", "Emotional intelligence",
        "quick learner", "problem solver", "Customer-service skills", "Planning and
organizing", "innovative",
        skills", "Sociability", "Self-management", "Integrity", "Honesty", "Human
resources",
        "Participates as a team member", "Works with diversity", "Exercises leadership",
"leadership", "Exercises leadership", "Monitors and corrects performance",
"Understands systems"]
    else:
        NonTechnicalSkill = progWords
        programmingTotal = 0

    jdSkillCount = 0
    jdSkillMatched = []
    for i in range(len(NonTechnicalSkill)):
        if NonTechnicalSkill[i].lower() in jd_txt.lower() != -1:

```

```

        jdSkillCount += 1
        jdSkillMatched.append(NonTechnicalSkill[i].lower())
print("jdSkillCount", jdSkillCount)
for x in range(len(jdSkillMatched)):
    print("jd Skills matched are ",jdSkillMatched)
#END
if (jdSkillCount > 0):
    individualSkillWeightage = skill_weightage/jdSkillCount
else :
    individualSkillWeightage = 0

ResumeProgrammingSkillsMatchedWithJD = []
fori in range(len(jdSkillMatched)):
    if jdSkillMatched[i].lower() in resume.lower() != -1:
        programmingTotal += 1
        ResumeProgrammingSkillsMatchedWithJD.append(jdSkillMatched[i].lower())
    if not("#" in jdSkillMatched[i]):
        fout.write(jdSkillMatched[i]+"", ")
print("Resume skills matched with JD are ",
ResumeProgrammingSkillsMatchedWithJD)
print("Non Technical skill total is ", programmingTotal)

# My Code
resumeCorpus = resume.split()
""" Modify below """
resumeCorpus = resumeCorpus + ResumeProgrammingSkillsMatchedWithJD
resumeCorpus = []
jdSkillMatched = []
print(type(resumeCorpus))
print("jd skills matched in lower case",jdSkillMatched)
list1 = jdSkillMatched

```

```

list2 = resumeCorpus
results = {}
for i in list1:
    if list2.count(i) > skill_threshold:
        results[i] = skill_threshold
    else:
        results[i] = list2.count(i)

print("Relevant non-technical skills and their count in resume as per the JD are
below")
print("Dictionary from resume is ",results)
print(type(results))
#end of code

constantValue = (individualSkillWeightage/skill_threshold)
# Updating Dictionary
results.update({n: constantValue * results[n] for n in results.keys()})
print("updated dict is ", results)

TotalScore = sum(results.values())
print("Score is ", TotalScore)

fout.close()

progScore = min(programmingTotal/10.0, 1) * 5.0

return TotalScore

```

Search.py

```
import glob
```

```
class ResultElement:
```

```
    def __init__(self, rank):
```

```
        self.rank = rank
```

```
        self.filename = filename
```

```
def allowed_file(filename):
```

```
def getfilepath(loc):
```

```
    temp = str(loc)
```

```
    temp = temp.replace('\\', '/')
```

```
    return temp
```

```
def res(jobfile):
```

```
    Final_Array = []
```

```
def lcs(X, Y):
```

```
    try:
```

```
def semanticSearch(searchString, searchSentencesList):
```

```
    result = None
```

```
    searchString = spellCorrect(searchString)
```

```
    bestScore = 0
```

```
    for i in searchSentencesList:
```

```
        score = lcs(searchString, i)
```

```
        print(score , i[0:100])
```

```

print("")
temp = [score]
Final_Array.extend(temp)
if score > bestScore:
    bestScore = score
    result = i
return result

```

```

Resume_Vector = []
Ordered_list_Resume = []
Ordered_list_Resume_Score = []
_FILES = []
_FILES_PDF = []
_FILES_DOC = []
_FILES_DOCX = []
Resumes_File_Names = []
Resumes = []
Temp_pdf = "

```

```

# LIST_OF_FILES.remove("antiword.exe")
print("This is LIST OF FILES")
print(LIST_OF_FILES)

```

```

# print("Total Files to Parse\t" , len(LIST_OF_PDF_FILES))
print("##### PARSING #####")
for nooo,i in enumerate(LIST_OF_FILES):
    Ordered_list_Resume.append(i)

```

```

Temp = i.split(".")
if Temp[1] == "pdf" or Temp[1] == "Pdf" or Temp[1] == "PDF":
    try:
        print("This is PDF" , nooo)
        with open(i,'rb') as pdf_file:

            page_content = page.extractText()
            page_content = page_content.replace('\n', ' ')
            # page_content.replace("\r", "")
            Temp_pdf = Temp_pdf + str(page_content)
            # Temp_pdf.append(page_content)
            # print(Temp_pdf)
            Resumes.extend([Temp_pdf])
            Temp_pdf = "
            Resumes_File_Names.append(i)
    exceptException as e: print(e)
if Temp[1] == "doc" or Temp[1] == "Doc" or Temp[1] == "DOC":
    print("This is DOC" , i)

    Resumes.extend(c)
    Resumes_File_Names.append(i)
exceptException as e: print(e)

if Temp[1] == "docx" or Temp[1] == "Docx" or Temp[1] == "DOCX":
    print("This is DOCX" , i)

    Resumes.extend(c)
    Resumes_File_Names.append(i)
exceptException as e: print(e)

```

```

# Resumes.extend(texttract.process(i))
if Temp[1] == "ex" or Temp[1] == "Exe" or Temp[1] == "EXE":
    # print("This is EXE" , i)
    pass

for m,i in enumerate(Resumes):
    Resumes[m] = nltk.word_tokenize(Resumes[m])
    Resumes[m] = normalize(Resumes[m])
    Resumes[m] = ' '.join(map(str, Resumes[m]))

jobfile = nltk.word_tokenize(jobfile)
jobfile = normalize(jobfile)
jobfile = ' '.join(map(str, jobfile))

print("This is len Resumes : " , len(Resumes))
os.chdir('../')

result = semanticSearch(jobfile, Resumes)

Ordered_list_Resume = Ordered_list_Resume[1:]

Z = [x for _,x in sorted(zip(Final_Array,Resumes_File_Names) , reverse=True)]
flask_return = []

for n,i in enumerate(Z):

```

```

name = getfilepath(i)

rank = n
res = ResultElement(rank, name)
flask_return.append(res)

return flask_return

```

skillsDB.txt

DevOps,assembly,bash,c,c++,c#,coffeescript,emacs
 lisp,go!,groov,askell,java,javascript,matlab,max MSP,objective
 c,perl,php,html,xml,css,processing,python,ruby,sml,swift,sic,wolfram
 language,xquery,sql,node.js,scala,kdb,jquery,mongodb,CMMI,ISO,MySQL,MS
 Access, General Ledge,Finacle,Oracle Flexcube,Fiserv,TCS BaNcs,FIS
 Profile,agile,C++-Microsoft Technologies--Expert,scala,ASP.Net--Master,dot
 net,AJAX, jQuery,Angular JS 2.0,JavaScript,MVC,Silverlight,Jquery,ASP.Net,SQL
 Server 2005,Java,Liferay,JBoss ESB,Eclipse,Spring,hibernate,J2EE,Core
 Java,JUnit,Log4j,Version Control,RESTFull,REST,JSP,MS SharePoint Cloud,
 sharepoint,Share Point 2007,ISMS,selenium,codedui,coded-ui,automation,vb.net,vb

Index.html

```
<!DOCTYPE html>
```

```
<html class="gr__getpapercss_com">
```

```
“AI based Resume screening”
```

```
<meta name="author" content="EVRY">
```

```
<style>
```

```
body
```

```
{
```

```
}
```

```
.mainDiv
```

```
{
```

```
}
```

```
</style>
```

```
<script>
```

```
$("#formid").submit(function(e) {
```

```
    $("#formdiv").hide();
```

```
    $("#resultdiv").show();
```

```
    resetResult();
```

```
console.log("submit");
console.log($("#formid").serialize());
```

```
    e.preventDefault();
});
```

```
$("#resetResult").click(function(){
    $("#formdiv").show();
    $("#resultdiv").hide();
});
```

```
$( document ).ready(function() {
    $("#resultdiv").hide();
});
```

```
function resetResult(){
    $("#result").innerHTML = "Processing....";
}
```

```

```

```
{% else %}
```

Login

```
<style>
body
{

}
.mainDiv
{

}
</style>
</head>
<body translate="no">

</body>
</html>
```

REFERENCES

- [1] Resume parser
- [2] Resume Parser with Natural Language Processing
- [3] Custom Name Entity Recognition <https://medium.com/analytics-vidhya/resume-parser-name-entity-recognition-using-spacy-86663d4683fa>
- [4] Gensim open source Incubator.
- [5] Gensim source code on Github .
- [6] Software Companies why structuring your data is crucial for your business .
- [7] Salesforce Blog , "the terrible trouble with NLP, 17 August 2000.
- [8] Automated Screening, Resume Hacking, 2 Jan. 2016.
- [9] Techniques for Extracting Information.
- [10] Opening the flask .Archived from the original (PDF) on 2016-12-17.

[11] Job Seekers Can't Stand the Application Process." *Business Insider*, Business Insider, 16 June 2016.

[12] <https://medium.com/@blondiebytes/heroku-hosting-put-your-website-on-the-cloud-cdc8729a7e09>

[13] Daxtra technologies Limited ,18 July 2016.

[14] Fortune time hiring ,8 june 2016

[15] Heroku hosting put your website on the cloud.