IBM HACK CHALLENGE

2020

Project Report

on

AI Resume Selector(Resume Matcher)

by

Sayed Mohd Kazim Mehdi

kazimsayed954@gmail.com

and

SALMAN RIZVI

rizvisalman48@gmail.com

PROJECT ID- SPS_PRO_1538

APPLICATION ID- SPS_CH_APL_20200004120

APPLICATION ID- SPS_CH_APL_20200004121

1. Introduction

1.1 Overview We will build a web app that calculate the similarity between Company Specific Resume with Given Candidate given Resume

- Project Requirements : Python 3.X,Flask framework,Web Browser
- Functional Requirements : SK Learn ,Numpy ,Pandas
- Technical Requirements : python with required library
- Software Requirements :Web Browser
- Project Deliverables : AI Resume Selector(Resume Matcher)
- Project Team : Rizvi’s
- Project Id: SPS_PRO_1538

1.2 Purpose the HR will get hectic to select a resume of a candidate with company job profile so our project will help HR to select a candidate with the given job description

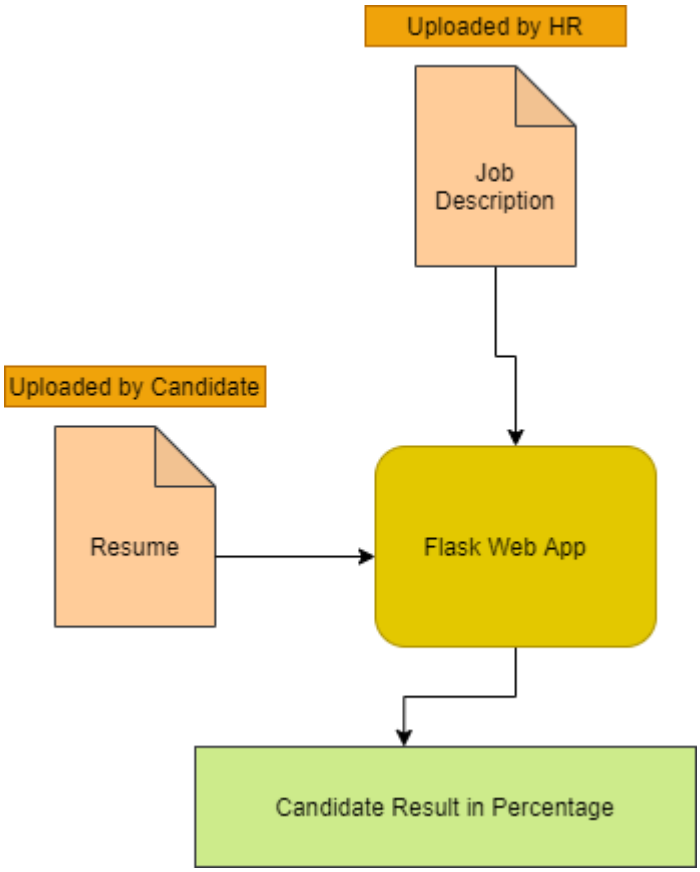
2. Literature Survey

2.1 Existing Problem HR will get hectic to select a resume with a candidate with a company job profile so our project will help HR to select a candidate with the given job description and it’s a manual process so definitely in takes time.

2.2 Proposed Solution In this project, we built a web app that calculates the similarity between two resumes using Cosine Similarity and it will automate the task

3. Theoretical Analysis

Block / Flow Diagram



Hardware / Software Designing

1. Created a UI

2.Used a SQL Alchemy Database to store a data.

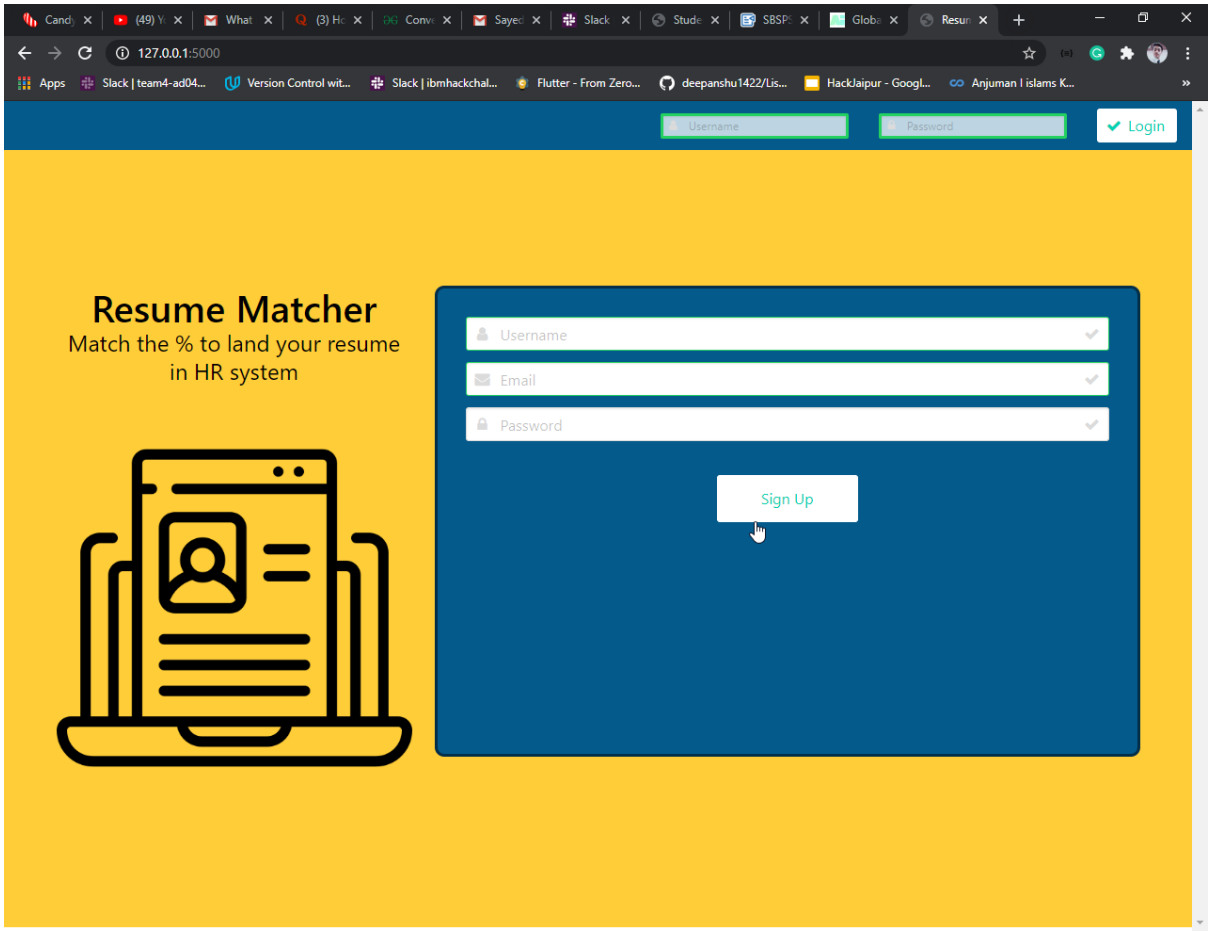
3.Make a app.py and put the 5.

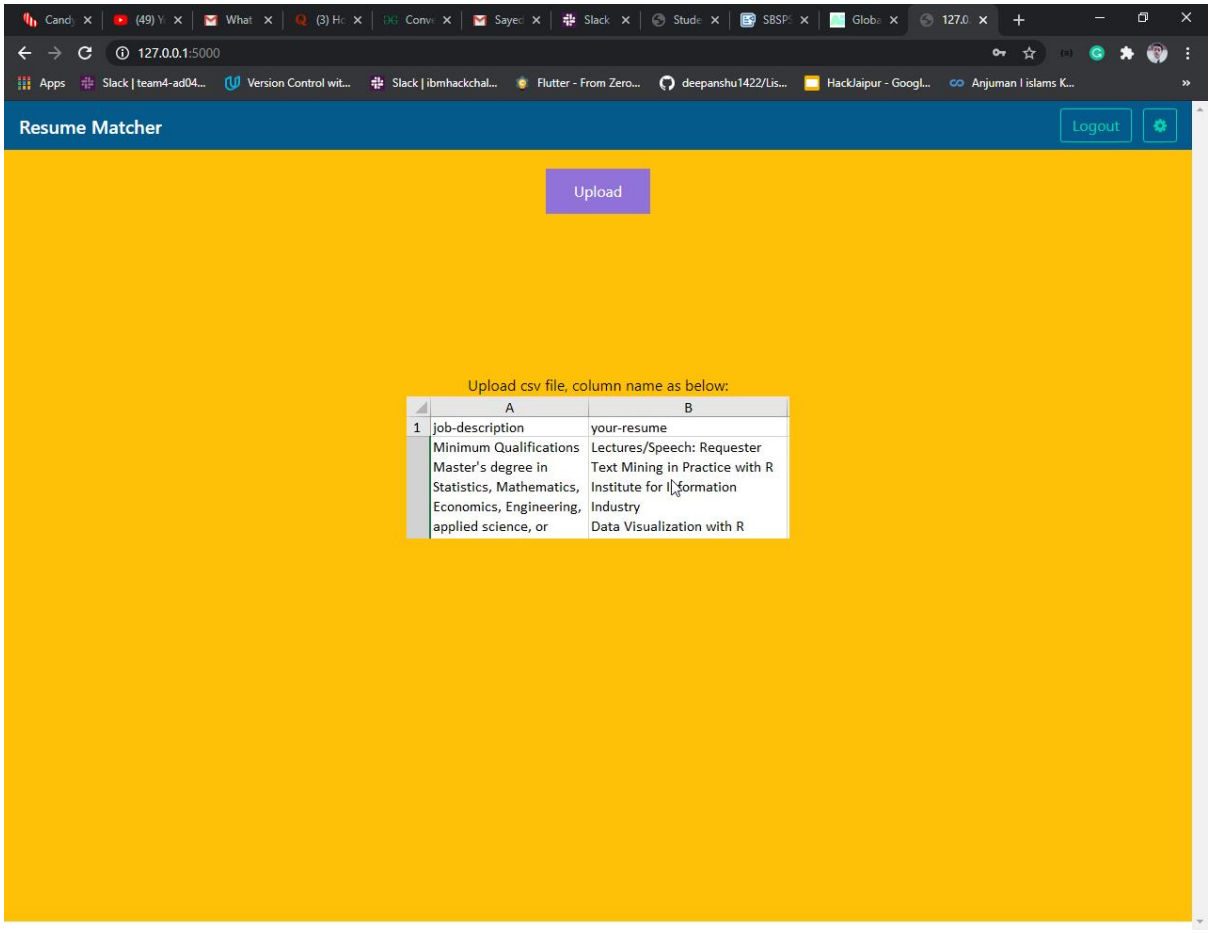
4. Configure Watson Assistant.

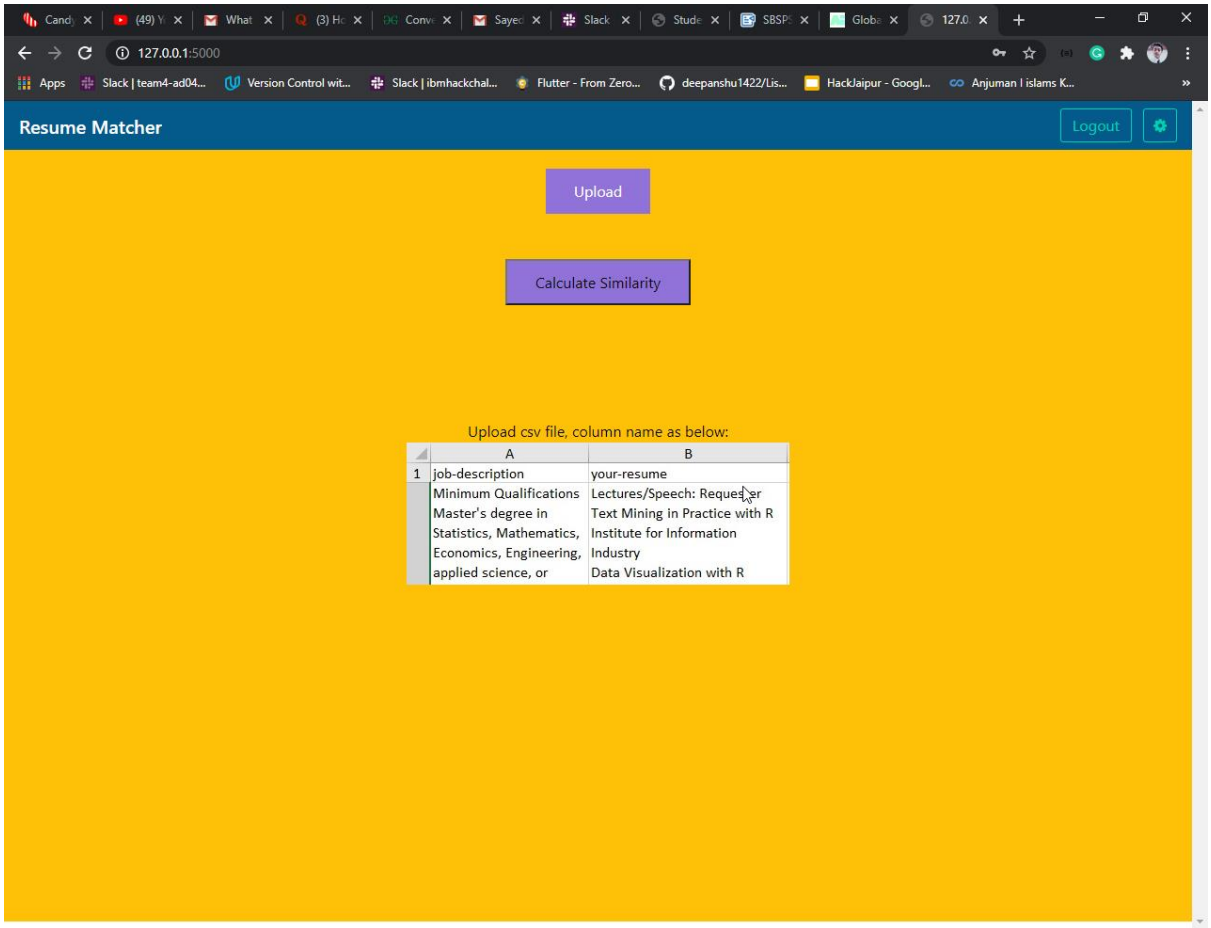
5. Integrate Watson Discovery with Watson Assistant using webhook.

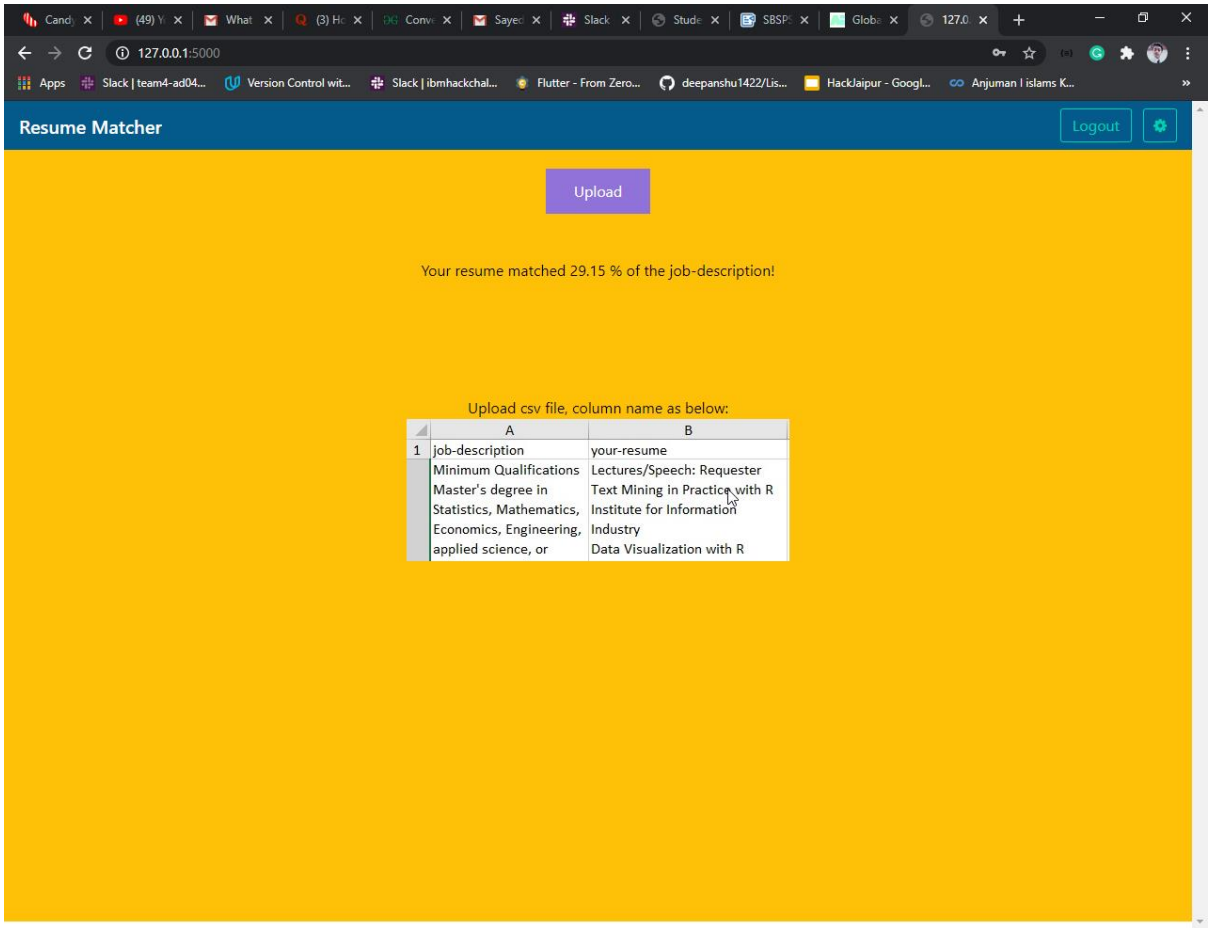
6. Build Node-RED flow to integrate Watson Assistant and Web Dashboard.

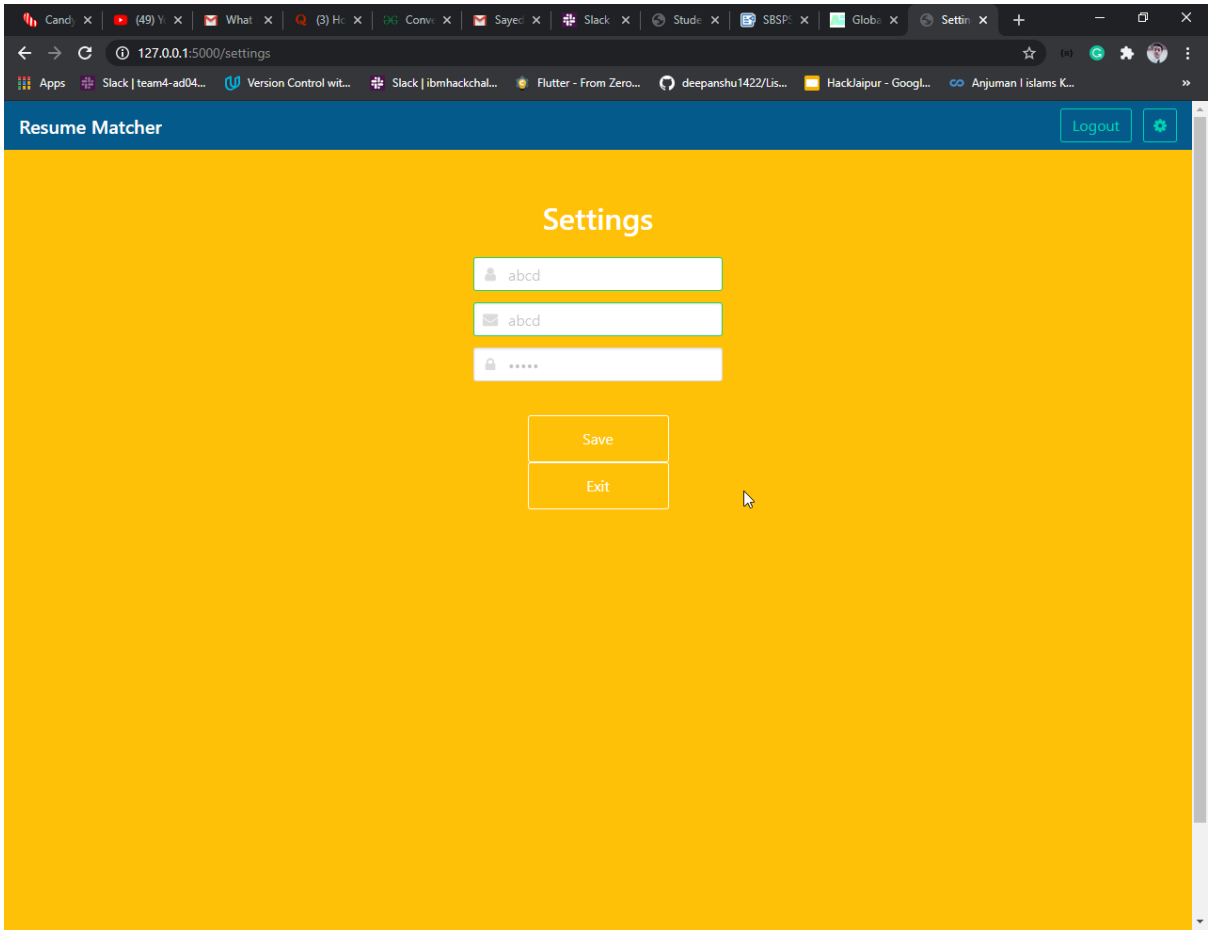
4. Experimental Investigation




127.0.0.1:5000
Apps
Slack | team4-ad04...
Version Control wit...
Slack | ibmhackchal...
Flutter - From Zero...
deepanshu1422/Lis...
HackJaipur - Googl...
Anjuman I islams K...
Resume Matcher
Logout
Upload
Upload csv file, column name as below:
A
B
1
job-description
your-resume
Minimum Qualifications
Master's degree in
Statistics, Mathematics,
Economics, Engineering,
applied science, or
Lectures/Speech: Requester
Text Mining in Practice with R
Institute for Information
Industry
Data Visualization with R


127.0.0.1:5000
Apps
Slack | team4-ad04...
Version Control wit...
Slack | ibmhackchal...
Flutter - From Zero...
deepanshu1422/Lis...
HackJaipur - Googl...
Anjuman I islams K...
Resume Matcher
Logout
Upload
Calculate Similarity
Upload csv file, column name as below:
A
B
1
job-description
your-resume
Minimum Qualifications
Master's degree in
Statistics, Mathematics,
Economics, Engineering,
applied science, or
Lectures/Speech: Requester
Text Mining in Practice with R
Institute for Information
Industry
Data Visualization with R


127.0.0.1:5000
Apps
Slack | team4-ad04...
Version Control wit...
Slack | ibmhackchal...
Flutter - From Zero...
deepanshu1422/Lis...
HackJaipur - Googl...
Anjuman I islams K...
Resume Matcher
Logout
Upload
Your resume matched 29.15 % of the job-description!
Upload csv file, column name as below:
A
B
1
job-description
your-resume
Minimum Qualifications
Master's degree in
Statistics, Mathematics,
Economics, Engineering,
applied science, or
Lectures/Speech: Requester
Text Mining in Practice with R
Institute for Information
Industry
Data Visualization with R



5. Flowchart

Insert the following nodes into the flow in Node-RED.

- UI
- Login/ SignUp
- Upload a Resume
- Calculate Similarity

6. Results

Web based UI was been developed using HTML with jinja Template.

Run app.py (command :python app.py)

After that visit given url below

http://127.0.0.1:5000/

7. Advantages & Disadvantages

Advantages

1. Reduces Man Power.

2. Cost Efficient.

3. Take Less time .

Disadvantages

1.Supported extension is .csv.

2.Web Design is not much responsive.

8. Applications

It can be used in every various industry as well as a startup it will reduce the cost and benefit for the startup as well as big MNC's.

9. Conclusion

It will be created in a flask that uses WSGI server that is a lite server and makes application faster.

10. Future Scope

In the future, various other Watson services like Text-To-Speech and Speech-To-Text can be integrated our Web app

and we can implement Watson Assistant Chatbot. We can use a visual recognition or NLP also in our project.

11. Bibliography

Resume matcher model:-

https://www.youtube.com/watch?v=bkigzpBLN6o

flask documentation:-

https://flask.palletsprojects.com/en/1.1.x/

Appendix

Source Code

App.py

```
from scripts import tabledef

from scripts import forms

from scripts import helpers

from flask import Flask, redirect, url_for, render_template, request, session

import json

import sys

import os

import pandas as pd

from werkzeug.utils import secure_filename

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import Ridge

from sklearn.model_selection import train_test_split

from sklearn.pipeline import make_pipeline

from sklearn.metrics.pairwise import cosine_similarity

from sklearn.feature_extraction.text import CountVectorizer

import numpy as np

import pandas as pd

import jieba

import jieba.analyse

import csv

import ast


app = Flask(__name__)

app.secret_key = os.urandom(12)



@app.route('/', methods=['GET', 'POST'])

def login():
```

```
    if not session.get('logged_in'):
        form = forms.LoginForm(request.form)
        if request.method == 'POST':
            username = request.form['username'].lower()
            password = request.form['password']
            if form.validate():
                if helpers.credentials_valid(username, password):
                    session['logged_in'] = True
                    session['username'] = username
                    return json.dumps({'status': 'Login successful'})
                return json.dumps({'status': 'Invalid user/pass'})
            return json.dumps({'status': 'Both fields required'})
        return render_template('login.html', form=form)
    user = helpers.get_user()
    return render_template('home.html', user=user)


@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if not session.get('logged_in'):
        form = forms.LoginForm(request.form)
        if request.method == 'POST':
            username = request.form['username'].lower()
            password = helpers.hash_password(request.form['password'])
            email = request.form['email']
            if form.validate():
                if not helpers.username_taken(username):
                    helpers.add_user(username, password, email)
                    session['logged_in'] = True
                    session['username'] = username
                    return json.dumps({'status': 'Signup successful'})
                return json.dumps({'status': 'Username taken'})
            return json.dumps({'status': 'User/Pass required'})
```

```
        return render_template('login.html', form=form)
    return redirect(url_for('login'))


@app.route('/settings', methods=['GET', 'POST'])
def settings():
    if session.get('logged_in'):
        if request.method == 'POST':
            password = request.form['password']
            if password != "":
                password = helpers.hash_password(password)
            email = request.form['email']
            helpers.change_user(password=password, email=email)
            return json.dumps({'status': 'Saved'})
        user = helpers.get_user()
        return render_template('settings.html', user=user)
    return redirect(url_for('login'))


@app.route('/login_page', methods=['POST'])
def login_page():
    if session.get('logged_in'):
        user = helpers.get_user()
        try:
            user.active = True
            return render_template('home.html', user=user)
        except error:
            return render_template('error.html')


@app.route("/logout")
def logout():
    session['logged_in'] = False
```

```
    return redirect(url_for('login'))



@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files['file']

        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        df = pd.read_csv(file_path)

        seg_list01 = df['job-description']
        seg_list02 = df['your-resume']

        item01_list = seg_list01
        item01 = ','.join(item01_list)

        item02_list = seg_list02
        item02 = ','.join(item02_list)

        documents = [item01, item02]

        count_vectorizer = CountVectorizer()
        sparse_matrix = count_vectorizer.fit_transform(documents)

        doc_term_matrix = sparse_matrix.todense()
        df = pd.DataFrame(doc_term_matrix,
                          columns=count_vectorizer.get_feature_names(),
                          index=['item01', 'item02'])
```

```
        answer = cosine_similarity(df, df)
        answer = pd.DataFrame(answer)
        answer = answer.iloc[[1], [0]].values[0]
        answer = round(float(answer), 4)*100

        return "Your resume matched " + str(answer) + " %" + " of the job-description!"
    return None


if __name__ == "__main__":
    app.run(debug=True, use_reloader=True)
```

https://github.com/SmartPracticeschool/SBSPS-Challenge-2083-AI-resume-Selector