

# Smart Resume Analyser

## **Introduction**

Any recruiter will find it difficult to choose the best prospects from a vast pool of applicants for that employment vacancy. The chore of manually sorting through thousands of resumes to find the best candidates for the position is incredibly challenging for recruiters. Although the methods employed by job websites have produced some accuracy and precision, one of the main drawbacks is the intricacy of the time component. The time complexity for getting the results is very significant if every candidate resume is compared to every other job posting provided on the online recruitment site. In the last several years, more than 50,000 e-recruitment websites have been created. These online recruitment services creators have employed a variety of strategies to find potential applicants for a specific job profile. Some of them have been successful in using approaches for categorizing resumes of applicants into different groups for each job posting provided by each employer. These methods attempt to match each applicant's resume with the specific job posting. To determine which resumes are closest to the specified job description, top candidates could be sorted using Content-based Recommendation, cosine similarity, and KNN. Although the methods employed by these websites have produced great levels of accuracy and precision, one drawback is the time required to find potential candidates for a given job description. Some of them have been successful in using approaches for categorizing resumes of applicants into different groups for each job posting provided by each employer. These strategies attempt to match each resume with the specific job posting. High accuracy and precision have been achieved by these sites' procedures, however one of the main drawbacks is the intricacy of the time component.

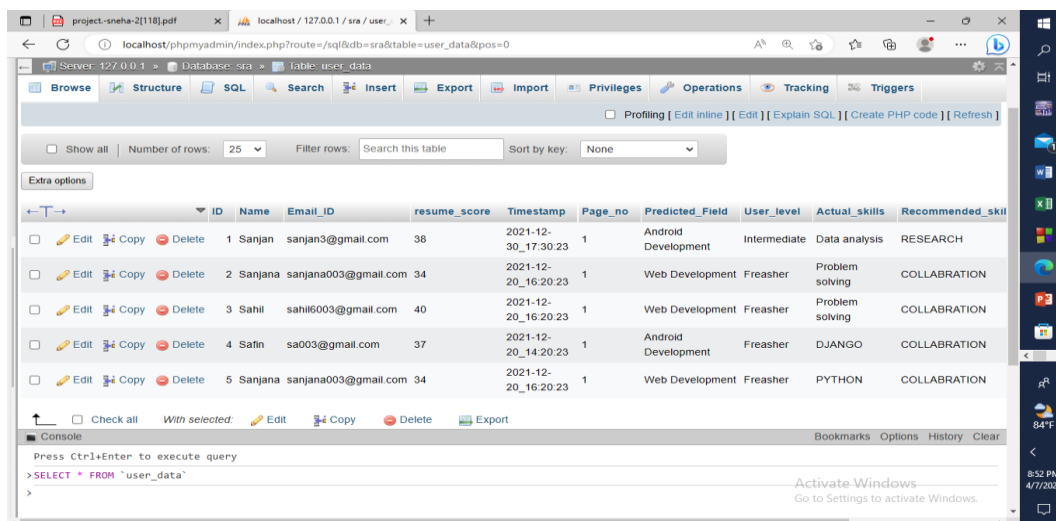
## **Survey**

Resume Classification and Ranking using KNN The KNN Algorithm is used to category the resumes into the appropriate categories, and Cosine Similarity is used to determine how well the candidate's resume matches the job description. The resumes are then sorted in accordance with their classifications. Web Application for Screening Resume: Semi-supervised learning is used by the suggested system, which is now being put into use, to attain high accuracy. The recruiter's workload will be lighter because they won't have to manually go through the extensive pool of applicants' resumes. A Machine Learning approach for automation of Resume Recommendation system. The screening and shortlisting processes may be greatly facilitated by an automated method of Resume Classification and Matching, and the candidate selection and decision-making processes would undoubtedly be sped up. III. RELATED WORK Existing system: The existing system uses Naïve Bayes algorithm. This method uses traditional machine learning and has lower accuracy rates. It is less effective and inaccurate data. The potential of people may be lost due to this system. The recruiters cannot differentiate between the similarity of the resumes of the candidates as they might have taken a copy of someone else's resume . There are other approaches being tried, such KNN, however they are ineffective for huge datasets. Another popular approach is text parsing; however it can only be applied to resumes with structure. If the outcome is positive, the resume matched the job description, then the test of hypothesis gives a statistical representation by comparing the variable data to the specified data. Manually checking the resumes can be time-consuming for the recruiters [5]. Although semi-supervised learning based on machine learning can be , it can only forecast the quality of

the resume based on the job description. Proposed system: Our approach uses, machine learning and natural language processing (NLP) techniques to evaluate resumes contextually. Artificial intelligence is employed along with these tools to go beyond keywords. Following resume screening, the software assesses candidates in real-time based on the employment requirements of the recruiter. This online application seeks to organize the resumes by comparing the resumes that best fits the specified Job Descriptions that is intelligently read as input. This software uses NLP for the instantaneous comparison and ranking of provided resumes [6]. In order to match candidates with job descriptions, we perform the extraction of abilities and relevant criteria from resume material, which is a different approach from other tasks. This application uses instead of generic processes for string matching and cosine similarity for overlapping co-efficient. Job seekers will be able to submit their resumes and apply for any open positions via the interactive web application. The applicants can view the skills that they need to attain additionally in order to meet the recruiter's criteria. Moreover, the user is provided with sources that can acknowledge them regarding the required skills along with the resume building tips for increasing their resume score. The resumes can then be scored, and the scores can be sorted from best match to worst match. Only the firm recruiter who is interested in choosing the top prospects from a vast pool of applicants is given access to this ranking. The recruiter can also view the insights like predicted field and the level of applicant from all resumes uploaded by using analytic statistical methods. The best candidates for that particular job vacancy can then be selected using the calculated ranks. By moving through all these mechanisms our method provides accurate results with greater efficiency, precision, and accuracy. This is done with the intention of saving recruiters at any organization time and effort from having to read through and evaluate hundreds of resumes.

## Dataset

Collecting the Resumes was the most important task in this process. Around fifty resumes were collected from belonging to Python Developer and Project Manager Roles which were in doc and docx formats. The resumes collected were then converted into pdf format using bash script for easy handling of the data.



The screenshot shows a web application interface for a database. The table displayed has the following columns: ID, Name, Email\_ID, resume\_score, Timestamp, Page\_no, Predicted\_Field, User\_level, Actual\_skills, and Recommended\_skill. The data rows are as follows:

ID	Name	Email_ID	resume_score	Timestamp	Page_no	Predicted_Field	User_level	Actual_skills	Recommended_skill
1	Sanjan	sanjan3@gmail.com	38	2021-12-30_17:30:23	1	Android Development	Intermediate	Data analysis	RESEARCH
2	Sanjana	sanjana003@gmail.com	34	2021-12-20_16:20:23	1	Web Development	Freasher	Problem solving	COLLABRATION
3	Sahil	sahil6003@gmail.com	40	2021-12-20_16:20:23	1	Web Development	Freasher	Problem solving	COLLABRATION
4	Safin	sa003@gmail.com	37	2021-12-20_14:20:23	1	Android Development	Freasher	DJANGO	COLLABRATION
5	Sanjana	sanjana003@gmail.com	34	2021-12-20_16:20:23	1	Web Development	Freasher	PYTHON	COLLABRATION

**Input:** Input on the user side is the job applicant resume which will be further pre-processed to remove any special characters or stop words. During cleaning, all distinct characters, numbers, and words with only one letter are removed. After all these steps we have a clean data collection. The login information

must be entered on the admin page in order to access the admin page. The outcome of the uploaded resumes is available on the admin page.

**Output:** The user-side application shows the necessary skills that users' resumes lack. Even the links to the courses that would help applicants increase their knowledge are provided. Even on the user side, the resume is visible so that person can try to create a better resume. On the dashboard, there are also resume-building suggestions. On the admin side, a highly ranked candidate's resume that most closely matches the job description is returned. Since they would be able to quickly review a huge number of resumes with the right fit, the technique would help the recruiter expedite profile shortlisting while also ensuring the validity of the shortlisting process, something a human would not be able to do in almost real-time. The level of candidates for the requisite talent is acknowledged by a variety of statistical data that is given

## Project Details

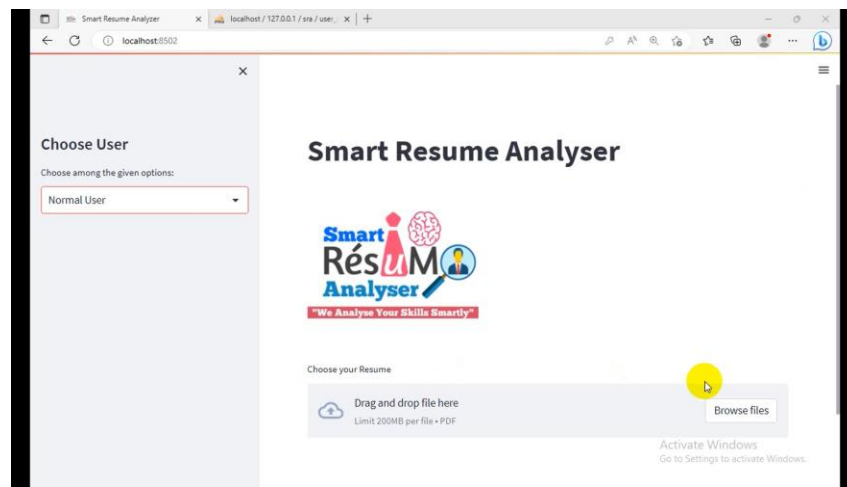


Fig:1.1(Front Page)

In the front page we can see that there is a cv drop box where any people can drop there cv and this cv will save in this box.

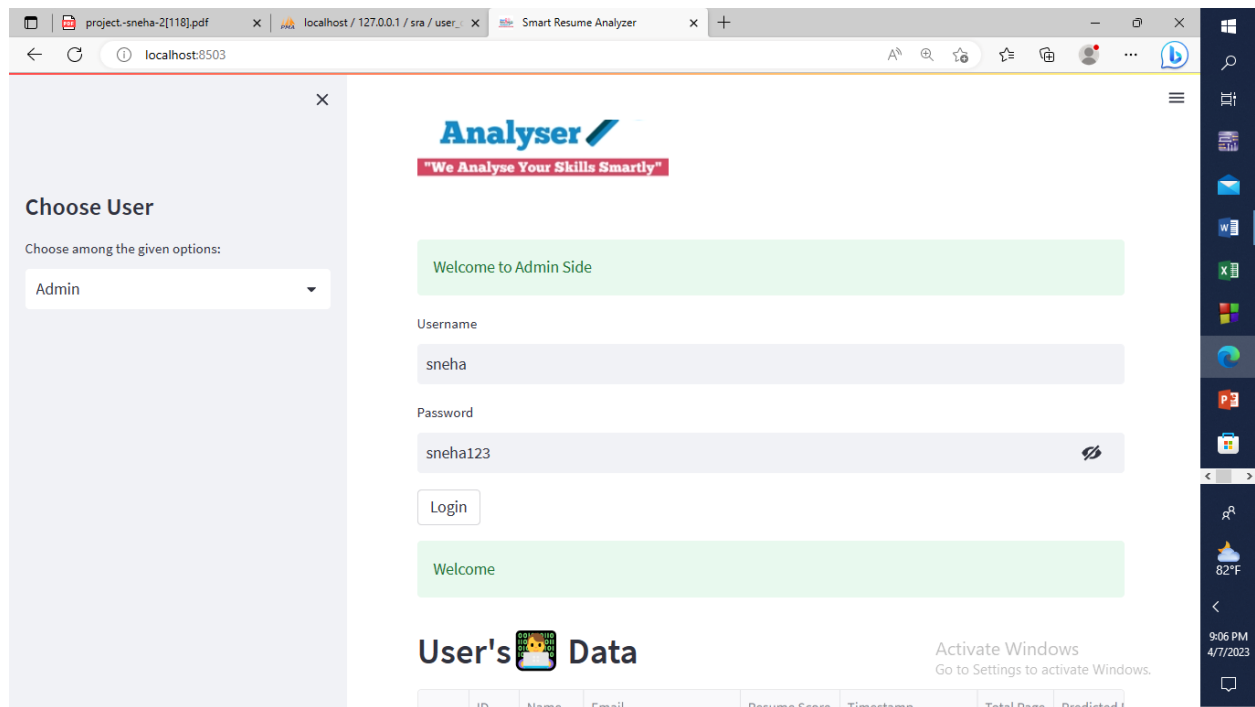


Fig1.2(Admin Sector)

In the admin sector we can see that if the admin give username and password admin can see all the cv details who drop the cv.

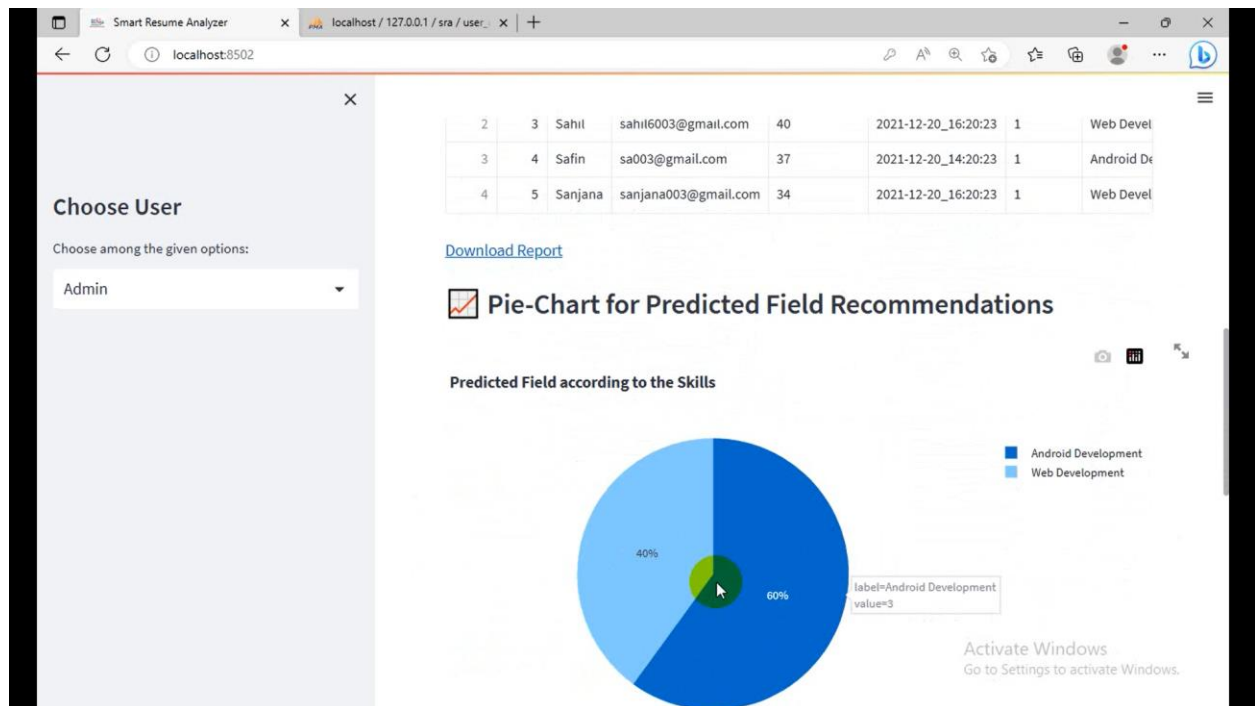


Fig1.3(Skills)

Here we can see the skills of the cv dropper. The android developer skills is 60% and the web

Developer are 40%.

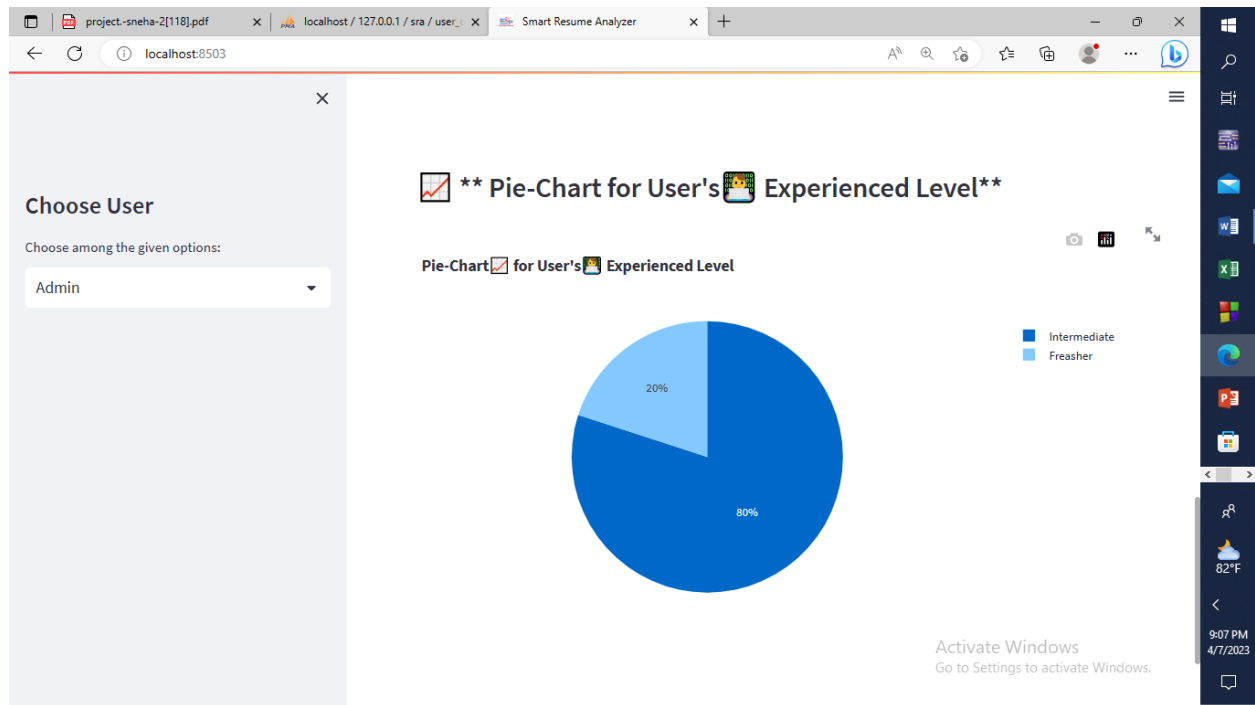


Fig:1.4(Experienced Level)

Here we can see the experienced level of the cv dropper. The intermediate is 80% and the web freasher are 20%.

### Code

```
import streamlit as st
import pandas as pd
import base64,random
import time,datetime
from pyresparser import ResumeParser
from pdfminer3.layout import LAParams, LTTextBox
from pdfminer3.pdfpage import PDFPage
from pdfminer3.pdfinterp import PDFResourceManager
from pdfminer3.pdfinterp import PDFPageInterpreter
from pdfminer3.converter import TextConverter
import io,random
```

```

from streamlit_tags import st_tags

from PIL import Image

import pymysql

from                               Courses                               import
ds_course,web_course,android_course,ios_course,uiux_course,resume_videos,interview_videos

import pafy

import plotly.express as px


def fetch_yt_video(link):
    video = pafy.new(link)
    return video.title


def get_table_download_link(df,filename,text):

    csv = df.to_csv(index=False)

    b64 = base64.b64encode(csv.encode()).decode() # some strings <-> bytes conversions
    necessary here

    # href = f'<a href="data:file/csv;base64,{b64}">Download Report</a>'

    href = f'<a href="data:file/csv;base64,{b64}" download="{filename}">{text}</a>'

    return href


def pdf_reader(file):
    resource_manager = PDFResourceManager()
    fake_file_handle = io.StringIO()
    converter = TextConverter(resource_manager, fake_file_handle, laparams=LAParams())
    page_interpreter = PDFPageInterpreter(resource_manager, converter)
    with open(file, 'rb') as fh:

```

```

    for page in PDFPage.get_pages(fh,
                                   caching=True,
                                   check_extractable=True):
        page_interpreter.process_page(page)
        print(page)
    text = fake_file_handle.getvalue()

# close open handles
converter.close()
fake_file_handle.close()
return text

def show_pdf(file_path):
    with open(file_path, "rb") as f:
        base64_pdf = base64.b64encode(f.read()).decode('utf-8')

    # pdf_display = f'<embed src="data:application/pdf;base64,{base64_pdf}" width="700"
    height="1000" type="application/pdf">'

    pdf_display = F'<iframe src="data:application/pdf;base64,{base64_pdf}" width="700"
    height="1000" type="application/pdf"></iframe>'

    st.markdown(pdf_display, unsafe_allow_html=True)

def course_recommender(course_list):
    st.subheader("**Courses & Certificates 📖 Recommendations**")
    c = 0
    rec_course = []
    no_of_reco = st.slider('Choose Number of Course Recommendations:', 1, 10, 4)
    random.shuffle(course_list)
    for c_name, c_link in course_list:

```



```

        c += 1

        st.markdown(f'({c}) [{c_name}][c_link]')

        rec_course.append(c_name)

        if c == no_of_reco:

            break

    return rec_course


connection = pymysql.connect(host='localhost',user='root',password='',db='sra')
cursor = connection.cursor()


def
insert_data(name,email,res_score,timestamp,no_of_pages,reco_field,cand_level,skills,recomm
ended_skills,courses):

    DB_table_name = 'user_data'

    insert_sql = "insert into " + DB_table_name + "
    values (0,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"

    rec_values = (name, email, str(res_score), timestamp,str(no_of_pages), reco_field, cand_level,
skills,recommended_skills,courses)

    cursor.execute(insert_sql, rec_values)

    connection.commit()


st.set_page_config(
    page_title="Smart Resume Analyzer",
    page_icon='./Logo/SRA_Logo.ico',
)

def run():

    st.title("Smart Resume Analyser")

    st.sidebar.markdown("# Choose User")

```

```
activities = ["Normal User", "Admin"]

choice = st.sidebar.selectbox("Choose among the given options:", activities)

# link = '[©Developed by Spidy20](http://github.com/spidy20)'

# st.sidebar.markdown(link, unsafe_allow_html=True)

img = Image.open('./Logo/SRA_Logo.jpg')

img = img.resize((250,250))

st.image(img)


# Create the DB

db_sql = """CREATE DATABASE IF NOT EXISTS SRA;"""

cursor.execute(db_sql)


# Create table

DB_table_name = 'user_data'

table_sql = "CREATE TABLE IF NOT EXISTS " + DB_table_name + """

    (ID INT NOT NULL AUTO_INCREMENT,

    Name varchar(100) NOT NULL,

    Email_ID VARCHAR(50) NOT NULL,

    resume_score VARCHAR(8) NOT NULL,

    Timestamp VARCHAR(50) NOT NULL,

    Page_no VARCHAR(5) NOT NULL,

    Predicted_Field VARCHAR(25) NOT NULL,

    User_level VARCHAR(30) NOT NULL,

    Actual_skills VARCHAR(300) NOT NULL,

    Recommended_skills VARCHAR(300) NOT NULL,

    Recommended_courses VARCHAR(600) NOT NULL,

    PRIMARY KEY (ID));
```

```

        """

cursor.execute(table_sql)

if choice == 'Normal User':

    # st.markdown("""<h4 style='text-align: left; color: #d73b5c;'>* Upload your resume, and get
    smart recommendation based on it."</h4>""",

    #         unsafe_allow_html=True)

    pdf_file = st.file_uploader("Choose your Resume", type=["pdf"])

    if pdf_file is not None:

        # with st.spinner('Uploading your Resume....'):

        #     time.sleep(4)

        save_image_path = './Uploaded_Resumes/'+pdf_file.name

        with open(save_image_path, "wb") as f:

            f.write(pdf_file.getbuffer())

        show_pdf(save_image_path)

        resume_data = ResumeParser(save_image_path).get_extracted_data()

        if resume_data:

            ## Get the whole resume data

            resume_text = pdf_reader(save_image_path)


            st.header("**Resume Analysis**")

            st.success("Hello " + resume_data['name'])

            st.subheader("**Your Basic info**")

            try:

                st.text('Name: '+resume_data['name'])

                st.text('Email: ' + resume_data['email'])

                st.text('Contact: ' + resume_data['mobile_number'])

                st.text('Resume pages: '+str(resume_data['no_of_pages']))

```

```

except:
    pass

cand_level = ""

if resume_data['no_of_pages'] == 1:
    cand_level = "Fresher"

    st.markdown( '''<h4 style='text-align: left; color: #d73b5c;'>You are looking
Fresher.</h4>''',unsafe_allow_html=True)

elif resume_data['no_of_pages'] == 2:
    cand_level = "Intermediate"

    st.markdown( '''<h4 style='text-align: left; color: #1ed760;'>You are at intermediate
level!</h4>''',unsafe_allow_html=True)

elif resume_data['no_of_pages'] >=3:
    cand_level = "Experienced"

    st.markdown( '''<h4 style='text-align: left; color: #fba171;'>You are at experience
level!''',unsafe_allow_html=True)


st.subheader("**Skills Recommendation💡**")

## Skill shows

keywords = st_tags(label='### Skills that you have',
text='See our skills recommendation',
value=resume_data['skills'],key = '1')


## recommendation

ds_keyword = ['tensorflow','keras','pytorch','machine learning','deep
Learning','flask','streamlit']

web_keyword = ['react', 'django', 'node js', 'react js', 'php', 'laravel', 'magento',
'wordpress',

'javascript', 'angular js', 'c#', 'flask']

android_keyword = ['android','android development','flutter','kotlin','xml','kivy']

```

```

ios_keyword = ['ios','ios development','swift','cocoa','cocoa touch','xcode']

uiux_keyword = ['ux','adobe xd','figma','zeplin','balsamiq','ui','prototyping','wireframes','storyframes','adobe photoshop','photoshop','editing','adobe illustrator','illustrator','adobe after effects','after effects','adobe premier pro','premier pro','adobe indesign','indesign','wireframe','solid','grasp','user research','user experience']

recommended_skills = []

reco_field = ""
rec_course = ""

## Courses recommendation
for i in resume_data['skills']:

    ## Data science recommendation
    if i.lower() in ds_keyword:

        print(i.lower())

        reco_field = 'Data Science'

        st.success("** Our analysis says you are looking for Data Science Jobs.**")

        recommended_skills = ['Data Visualization','Predictive Analysis','Statistical Modeling','Data Mining','Clustering & Classification','Data Analytics','Quantitative Analysis','Web Scraping','ML Algorithms','Keras','Pytorch','Probability','Scikit-learn','Tensorflow',"Flask",'Streamlit']

        recommended_keywords = st_tags(label='### Recommended skills for you.',
        text='Recommended skills generated from System',value=recommended_skills,key
        = '2')

        st.markdown('"<h4 style="text-align: left; color: #1ed760;">Adding this skills to resume will boost the chances of getting a Job</h4>"',unsafe_allow_html=True)

        rec_course = course_recommender(ds_course)

        break

## Web development recommendation

```

```

elif i.lower() in web_keyword:

    print(i.lower())

    reco_field = 'Web Development'

    st.success("** Our analysis says you are looking for Web Development Jobs **")

    recommended_skills = ['React','Django','Node JS','React JS','php','laravel','Magento','wordpress','Javascript','Angular JS','c#','Flask','SDK']

    recommended_keywords = st_tags(label='### Recommended skills for you.',
    text='Recommended skills generated from System',value=recommended_skills,key
    = '3')

    st.markdown("""<h4 style='text-align: left; color: #1ed760;'>Adding this skills to
    resume will boost🚀 the chances of getting a Job👛</h4>""",unsafe_allow_html=True)

    rec_course = course_recommender(web_course)

    break

```

## ## Android App Development

```

elif i.lower() in android_keyword:

    print(i.lower())

    reco_field = 'Android Development'

    st.success("** Our analysis says you are looking for Android App Development Jobs
    **")

    recommended_skills = ['Android','Android development','Flutter','Kotlin','XML','Java','Kivy','GIT','SDK','SQLite']

    recommended_keywords = st_tags(label='### Recommended skills for you.',
    text='Recommended skills generated from System',value=recommended_skills,key
    = '4')

    st.markdown("""<h4 style='text-align: left; color: #1ed760;'>Adding this skills to
    resume will boost🚀 the chances of getting a Job👛</h4>""",unsafe_allow_html=True)

    rec_course = course_recommender(android_course)

    break

```

```

## IOS App Development

elif i.lower() in ios_keyword:

    print(i.lower())

    reco_field = 'IOS Development'

    st.success("** Our analysis says you are looking for IOS App Development Jobs **")

    recommended_skills = ['IOS','IOS Development','Swift','Cocoa','Cocoa
Touch','Xcode','Objective-C','SQLite','Plist','StoreKit','UI-Kit','AV Foundation','Auto-Layout']

    recommended_keywords = st_tags(label='### Recommended skills for you.',
    text='Recommended skills generated from System',value=recommended_skills,key
= '5')

    st.markdown("""<h4 style='text-align: left; color: #1ed760;'>Adding this skills to
resume will boost🚀 the chances of getting a Job📁</h4>""",unsafe_allow_html=True)

    rec_course = course_recommender(ios_course)

    break


## Ui-UX Recommendation

elif i.lower() in uiux_keyword:

    print(i.lower())

    reco_field = 'UI-UX Development'

    st.success("** Our analysis says you are looking for UI-UX Development Jobs **")

    recommended_skills = ['UI','User Experience','Adobe
XD','Figma','Zeplin','Balsamiq','Prototyping','Wireframes','Storyframes','Adobe
Photoshop','Editing','Illustrator','After Effects','Premier
Pro','Indesign','Wireframe','Solid','Grasp','User Research']

    recommended_keywords = st_tags(label='### Recommended skills for you.',
    text='Recommended skills generated from System',value=recommended_skills,key
= '6')

    st.markdown("""<h4 style='text-align: left; color: #1ed760;'>Adding this skills to
resume will boost🚀 the chances of getting a Job📁</h4>""",unsafe_allow_html=True)

```

```

rec_course = course_recommender(uiux_course)

break

#
## Insert into table

ts = time.time()

cur_date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
cur_time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
timestamp = str(cur_date+'_'+cur_time)

### Resume writing recommendation

st.subheader("**Resume Tips & Ideas💡**")

resume_score = 0

if 'Objective' in resume_text:

    resume_score = resume_score+20

    st.markdown("""<h4 style='text-align: left; color: #1ed760;'>[+] Awesome! You have
added Objective</h4>""",unsafe_allow_html=True)

else:

    st.markdown("""<h4 style='text-align: left; color: #fab100;'>[-] According to our
recommendation please add your career objective, it will give your career intension to the
Recruiters.</h4>""",unsafe_allow_html=True)

if 'Declaration' in resume_text:

    resume_score = resume_score + 20

    st.markdown("""<h4 style='text-align: left; color: #1ed760;'>[+] Awesome! You have
added Delcaration</h4>""",unsafe_allow_html=True)

else:

```



```
st.markdown("""<h4 style='text-align: left; color: #fab100;'>[-] According to our recommendation please add Declaration. It will give the assurance that everything written on your resume is true and fully acknowledged by you</h4>""",unsafe_allow_html=True)
```

```
if 'Hobbies' or 'Interests' in resume_text:
```

```
    resume_score = resume_score + 20
```

```
    st.markdown("""<h4 style='text-align: left; color: #1e8449;'>[+] Awesome! You have added your Hobbies</h4>""",unsafe_allow_html=True)
```

```
else:
```

```
    st.markdown("""<h4 style='text-align: left; color: #fab100;'>[-] According to our recommendation please add Hobbies. It will show your personality to the Recruiters and give the assurance that you are fit for this role or not.</h4>""",unsafe_allow_html=True)
```

```
if 'Achievements' in resume_text:
```

```
    resume_score = resume_score + 20
```

```
    st.markdown("""<h4 style='text-align: left; color: #1e8449;'>[+] Awesome! You have added your Achievements ✨ </h4>""",unsafe_allow_html=True)
```

```
else:
```

```
    st.markdown("""<h4 style='text-align: left; color: #fab100;'>[-] According to our recommendation please add Achievements ✨ . It will show that you are capable for the required position.</h4>""",unsafe_allow_html=True)
```

```
if 'Projects' in resume_text:
```

```
    resume_score = resume_score + 20
```

```
    st.markdown("""<h4 style='text-align: left; color: #1e8449;'>[+] Awesome! You have added your Projects 🧑🏻💻 </h4>""",unsafe_allow_html=True)
```

```
else:
```

```
    st.markdown("""<h4 style='text-align: left; color: #fab100;'>[-] According to our recommendation please add Projects 🧑🏻💻. It will show that you have done work related the required position or not.</h4>""",unsafe_allow_html=True)
```

```

st.subheader("**Resume Score🎉**")
st.markdown(
    """
    <style>
        .stProgress > div > div > div > div {
            background-color: #d73b5c;
        }
    </style>""",
    unsafe_allow_html=True,
)
my_bar = st.progress(0)
score = 0
for percent_complete in range(resume_score):
    score +=1
    time.sleep(0.1)
    my_bar.progress(percent_complete + 1)
st.success("** Your Resume Writing Score: " + str(score)+"**")
st.warning("** Note: This score is calculated based on the content that you have added in your Resume. **")
st.balloons()

insert_data(resume_data['name'], resume_data['email'], str(resume_score),
timestamp,
str(resume_data['no_of_pages']), reco_field, cand_level,
str(resume_data['skills']),
str(recommended_skills), str(rec_course))

```

```
## Resume writing video
```

```
st.header("***Bonus Video for Resume Writing Tips💡**")
```

```
resume_vid = random.choice(resume_videos)
```

```
res_vid_title = fetch_yt_video(resume_vid)
```

```
st.subheader("✔️ ***"+res_vid_title+"**")
```

```
st.video(resume_vid)
```

```
## Interview Preparation Video
```

```
st.header("***Bonus Video for Interview👤💼 Tips💡**")
```

```
interview_vid = random.choice(interview_videos)
```

```
int_vid_title = fetch_yt_video(interview_vid)
```

```
st.subheader("✔️ ***" + int_vid_title + "**")
```

```
st.video(interview_vid)
```

```
connection.commit()
```

```
else:
```

```
st.error('Something went wrong..')
```

```
else:
```

```
## Admin Side
```

```
st.success('Welcome to Admin Side')
```

```
# st.sidebar.subheader('***ID / Password Required!***')
```

```
ad_user = st.text_input("Username")
```

```
ad_password = st.text_input("Password", type='password')
```

```
if st.button('Login'):
```

```
if ad_user == 'sneha' and ad_password == 'sneha123':
```

```

st.success("Welcome")

# Display Data
cursor.execute("""SELECT*FROM user_data""")
data = cursor.fetchall()
st.header("**User's👤💻 Data**")

df = pd.DataFrame(data, columns=['ID', 'Name', 'Email', 'Resume Score', 'Timestamp',
'Total Page',
                                'Predicted Field', 'User Level', 'Actual Skills', 'Recommended Skills',
                                'Recommended Course'])

st.dataframe(df)

st.markdown(get_table_download_link(df,'User_Data.csv','Download      Report'),
unsafe_allow_html=True)

## Admin Side Data
query = 'select * from user_data;'
plot_data = pd.read_sql(query, connection)

## Pie chart for predicted field recommendations
labels = plot_data.Predicted_Field.unique()
print(labels)
values = plot_data.Predicted_Field.value_counts()
print(values)

st.subheader("📊 **Pie-Chart for Predicted Field Recommendations**")

fig = px.pie(df, values=values, names=labels, title='Predicted Field according to the
Skills')

st.plotly_chart(fig)

### Pie chart for User's👤💻 Experienced Level
labels = plot_data.User_level.unique()

```

```

values = plot_data.User_level.value_counts()

st.subheader("📊 ** Pie-Chart for User's👤💻 Experienced Level**")

fig = px.pie(df, values=values, names=labels, title="Pie-Chart📊 for User's👤💻 Experienced Level")

st.plotly_chart(fig)

else:

    st.error("Wrong ID & Password Provided")

run()

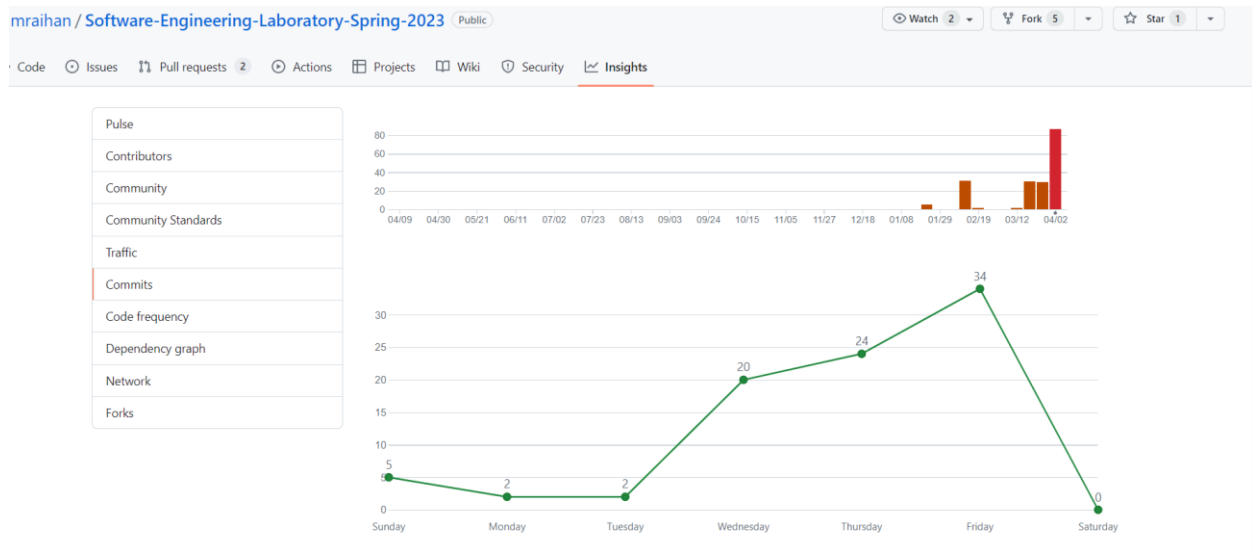
```

### Social Benefits

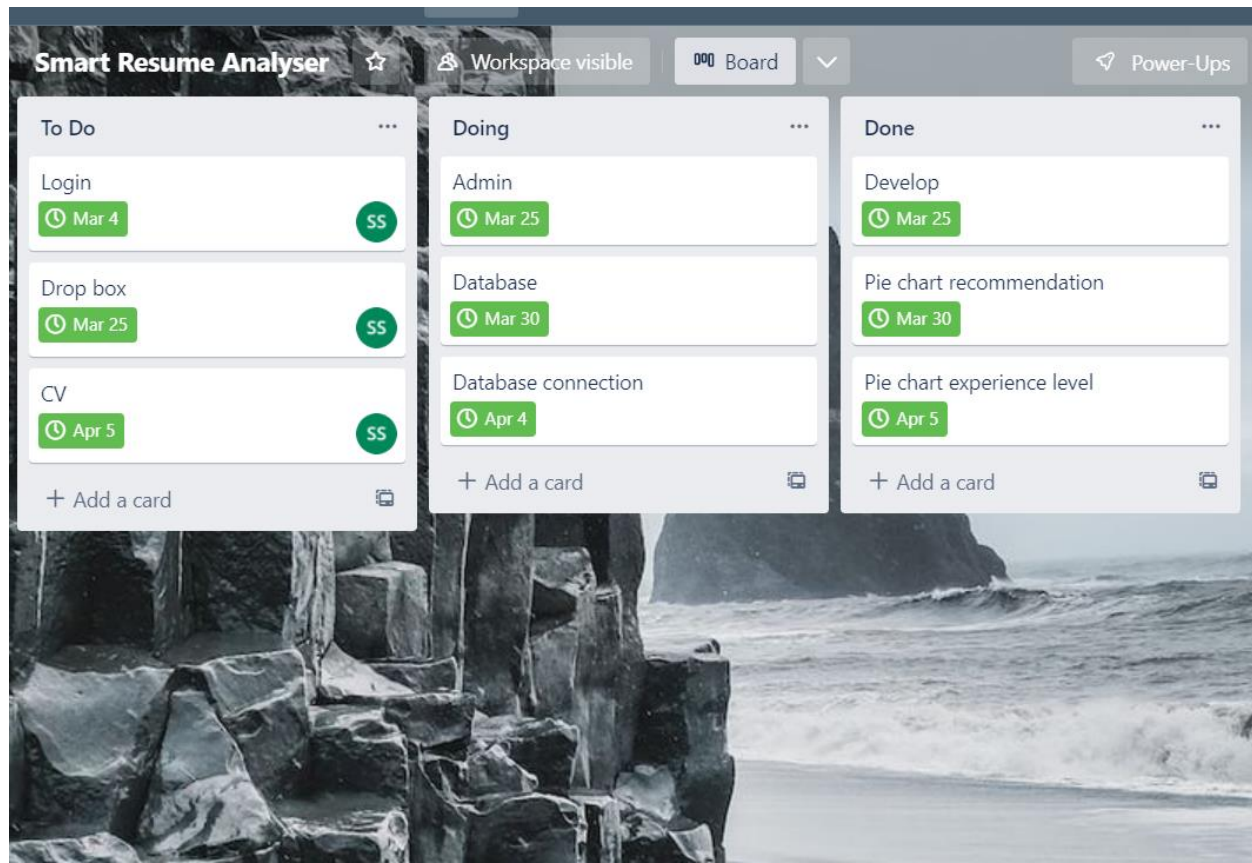
Future development for this approach entails mining applicant social networking data and using this social behaviour data in conjunction with the content of resumes to produce even better suggestions. Using a collaborative filtering-based technique is another option, which can match the present applicant with a job based on how well other candidates who are similar to them are rated for it. As a part of our ongoing research, we intend to leverage the information gleaned from resumes of applicants to dynamically create user profiles that will be used to suggest jobs to job seekers. Involving subject matter experts, such as HR professionals, will aid in the development of a more accurate model, and the HR professionals' feedback will aid in iteratively improving the model. The use of latent semantic analysis in calculating the semantic similarity of the documents and comparing it with the outcomes of the term frequency-based similarity technique is another area of future development. The shortlisted applicant's personalities can be examined utilising the social media data included in their resumes. This evaluation will help determine whether the candidate's personality, as seen in his or her social life, aligns with the demands of the position.

## GitHub

[https://github.com/mraihan/Software-Engineering-Laboratory-Spring-2023/tree/main/Id-20211009010 20211014010 2021101100/resume%20analysere](https://github.com/mraihan/Software-Engineering-Laboratory-Spring-2023/tree/main/Id-20211009010%2020211014010%202021101100/resume%20analysere)



## Task Schedule



## **Conclusion**

In this paper, we investigated the crucial yet understudied issue of automatic resume quality assessment (RQA). The classification of the applicant's resume is a laborious, time-consuming, and resource-wasting process. We have developed a machine learning-based automated algorithm to address this problem by recommending HR the resumes of qualified candidates based on the provided job description. With the use of Natural Language Processing techniques our algorithm was able to screen and shortlist the most qualified candidates. When displaying the top-selected resume on Web user interface, Latent Dirichlet Allocation produced incredibly precise results. By using job descriptions as input the best-fit resume can be selected by matching the skills in the resume of the applicant. Along with this, various visualization and data analytics are provided on the admin side. The linear classifiers are used to provide a statistical model of skill similarity. The user is provided with a proper guidance about the skills they need to upgrade and the sources for getting acknowledged. They are also equipped with essentials tips that can enhance their resume. This indeed, helps the user to know their potential and improve their proficiency in the fields that the company is looking for in them.

## **References**

- [1]. Rajath V; Riza Tanaz Fareed; Sharadadevi Kaganurmam, "Resume Classification and Ranking Using KNN And Cosine Similarity", international JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY, Volume 10, Issue 08, AUGUST 2021
- [2]. Akshay Kulkarni; Adarsha Shivananda; Anoosh Kulkarni, "Creating a resume Parsing, Screening and Shortlisting System", Natural Language Processing Projects pp 125-155, December 2021
- [3]. Sujit Amin; Nikita Jayakar; Sonia Sunny; Pheba Babu; M. Kiruthika; Ambarish Gurjar, "Web Application for Screening Resume", 2019 International Conference on Nascent Technologies in Engineering (ICNTE), January 2020.
- [3]. Sujit Amin; Nikita Jayakar; Sonia Sunny; Pheba Babu; M. Kiruthika; Ambarish Gurjar, "Web Application for Screening Resume", 2019 International Conference on Nascent Technologies in Engineering (ICNTE), January 2020
- [4]. Jitendra Purohit; Aditya Bagwe; Rishabh Mehta; Ojaswini Mangaonkar; Elizabeth George, "Natural Language Processing based Jaro-The Interviewing Chatbot", 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), August 2019
- [5]. Tejaswini K; Umadevi V; Shashank M Kadiwal; Sanjay Revanna, "Design and Development of Machine Learning based Resume Ranking System", Global Transitions Proceedings, October 2021