Mini-Project Report On

SRA: Smart Resume Analyzer

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

By

Navneeth Sunil (RET20CS142)
Nikhil M V (RET20CS149)
Rishikesh Sivaji (RET20CS163)
Pranav Sridhar (RET20CS157)

Under the guidance of
Mr.Harikrishnan M

RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***"SRA: Smart Resume Analyzer"*** *is a bonafide work done by* ***Navneeth Sunil (RET20CS142), Nikhil M V (RET20CS149), Rishikesh Sivaji (RET20CS163), Pranav Sridhar (RET20CS157)****, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.	**Mr. Sajanraj T D**	**Mr. Harikrishnan M**
Head of Department	Mini-Project Coordinator	Mini-Project Guide
Dept. of CSE	Research Assistant	Asst. Professor
RSET	Dept. of CSE	Dept. of CSE
	RSET	RSET

ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, "Smart Resume Analyzer".

We are highly indebted to our mini-project coordinators, **Mr. Sajanraj T D** ,Research Assistant, Department of Computer Science and Engineering, and **Ms. Anita John**, Assistant Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Mr. Harikrishnan M**, for his patience and all the priceless advice and wisdom he has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Navneeth Sunil

Nikhil M V

Rishikesh Sivaji

Pranav Sridhar

ABSTRACT

This project "SRA: Resume Analyzer" is about to analyze the resume of any person, based on resume our smart system will give the smart recommendation to the user. It is very easy to use and smart application, in this application user just need to upload the resume. It will automatically analyze the resume. After that it will give you the recommendations. Technology is playing a very vital role in changing our life in many ways. There are many sectors which benefited after implementing technology in it. Now it's time for evolution in very field. With the use of Artificial Intelligence every field is growing widely. With this project our aim is to use AI in job sectors. This system can be use by any fresher, graduate or experienced people. Our future aim to create same module for companies oriented also, so with this application companies can directly shortlist the people using AI. We are using the core concepts of Natural Language Processing for analyzing the resume. This is just first step for implementing Artificial Intelligence in the field of Job Recruitment sector.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

1.1.1 Purpose

The main purpose of building this project is to create a smart technology for corporate hiring world. Nowadays there are thousands of people are unemployed or they are finding the jobs. Our Aim is to provide a smart system that can give the perfect recommendation based on the resume of user, It will give the recommendation of Job working industry, tools and Technology, courses, and resume writing techniques. So with this help user can create a better resume so it will increase the chances of getting a job in good company.

1.1.2 Current Situation

In current situation there are thousands of unemployed people finding the job but they are can't able to select in the companies, the reason is they don't know the trend of current technology and fields. May be they are lacking with the skills of resume writing. It is chances that they didn't have done any particular certifications in particular field. So, our ideas is to implement Artificial Intelligence in this field. AI Technology is playing a very vital role in changing our life in many ways. There are many sectors which benefited after implementing AI technology in it. Now it's time for evolution in very field. With the use of Artificial Intelligence every field is growing widely. With this project our aim is to use AI in job sectors. This system can be use by any fresher, graduate or experienced people. Our future aim to create same module for companies oriented also, so with this application companies can directly shortlist the people using AI. We are using the core concepts of Natural Language Processing for analyzing the resume. This is just first step for implementing Artificial Intelligence in the field of Job Recruitment sector.

1.1.3 Tools and Technologies

- Python Programming
- NLTK and Pyparser for NLP
- Streamlit for Backend
- PDFMiner for extracting PDF
- Base64 for PDF displaying
- Pillow
- Numpy
- Pycharm IDE
- BS4 and requests for Web Scraping

Chapter 2

Literature Review

According to [1], In this paper authors have explained Automated Resume Screening System using Natural Language Processing. They have used stemming, lemmatization, POS Tagging, Chunking and Tokenization. They have used Cosine Similarity to get the ranking of the Candidates. In [2], Paper explained different techniques for the Resume Parsing using Natural Language Processing. They have used OCR to fetch to text from the resume. They are using the Tokens and Semantic Analysis to pre-process the text of resume. They are using Ranking algorithms to select the candidate. According to [3], Paper explained Resume Parser with Natural language Processing technique. They have used Lexical analysis, Tokens, Syntactic Analysis, and parse tree for the processing. They are using Json format to fetch the output. According to [4] According to paper, Authors are using Stop words removal, Stemming, Lemmatization to fetch the data. In machine learning they are using Random forest, Naïve Bayes, Logistic regression and Linear Support Vector Classifier.

Chapter 3

System Analysis

3.1 Expected System Requirements

The system of user which is a smart phone is expected to have the following features:

- Android platform with a version above 4.
- Requirement of Internet connection

3.2 Study of Current System

In the current system, we have a resume uploader that allows users to upload their resumes. After the resume is uploaded, the system stores it and converts all resume pages into images. The next step is to extract text from these images using natural language processing (NLP). Based on the extracted text, the system provides recommendations to the user regarding skills, courses, and resume writing. Users also have the option to remove their resumes from the system, although their data will be stored for future reference and training purposes.

3.2.1 Problem of Current System

The current system has several limitations. Firstly, it can only accept resumes in a specific format and does not support other formats such as Word, JPG, or any image format. Secondly, the system is currently tailored for the IT industry and may not work well for non-IT professionals. Thirdly, the system is currently hosted on a localhost server and has not been deployed, making it difficult to predict its performance after deployment. Additionally, the user interface occasionally experiences hanging issues during resume viewing, especially on mobile devices.

3.2.2 Feasibility Study

The feasibility analysis begins by defining the project goals and generating possible solutions to provide an indication of the new system's potential. This phase requires creativity and imagination to explore new ways of doing things and generate ideas. It is important to provide enough information for reasonable cost estimates and to evaluate how the new system fits into the organization without investing excessive effort at this stage.

3.2.3 Technical Feasibility

The application is developed as a web application and requires specific tools and technologies. The main technology used is natural language processing (NLP), implemented in Python programming language. Adobe XD is used for design purposes, and Star UML is used for creating diagrams. The PyCharm IDE is utilized for development.

3.2.4 Financial Feasibility

The application is freely available for all users, without any charges for usage. There are no monetary services associated with the application, allowing every user to access it freely.

3.2.5 Operational Feasibility

Operational feasibility measures how well the proposed system solves problems and takes advantage of opportunities identified during scope definition. It ensures that the system satisfies the requirements identified in the requirements analysis phase. In our application, all operations, such as PDF processing and system recommendations, are functioning properly. The admin can access all user data, and users can remove their resumes from the system.

3.3 Hardware Requirements

The following are the system requirements to develop Smart Resume Analyzer.

- Processor: Intel Core i5
- Hard Disk: Minimum 100GB

- RAM: Minimum 8GB

3.4 Software Requirements

The following are the softwares used in the development of the app.

Operating System: Windows or Linux

3.4.1 Python Programming Language

Python is advanced, higher level and easy to learn programming language, We have chosen Python to develop Desktop GUI, We will create a desktop application using Python. In python everything is related to the Image processing is very easy to implement. The documentation and community of the python is very big, so we will get the help in development from the community.

3.4.2 PyCharm IDE

PyCharm is professional IDE which is developed by JetBrains. The features which are offered by PyCharm Community version is Intelligent Python editor, Code debugger, Code inspection, VCS support and many more related to the User Interface.

3.4.3 Streamlit

Streamlit is special framework for handling the Data intensive applications, we are using it because it's have amazing UI components, we don't need to implement HTML+ CSS, everything will be handled by python code. It is very easy to use. In this deployment is also very easy.

3.4.4 NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language. We are using NLTK to preprocess the resume.

3.4.5 XAMP

It is open source control panel which is easy to use and maintain the apache, database, and PhpMyAdmin. It is very easy to operate. We can operate apache, MySQL easily from it. It is easily available for windows and other platforms.

3.4.6 GitHub

It is free and open source backup platform for your running project. It is maintaining by GIT. It's very easy to upload your work on free. There is no storage limitation. You can do development in group and separately also. It is very easy to maintain the Information Technology work with other teammates also

3.5 Literature Survey

- According to [1], In this paper authors have explained Automated Resume Screening System using Natural Language Processing. They have used stemming, lemmatization, POS Tagging, Chunking and Tokenization. They have used Cosine Similarity to get the ranking of the Candidates.

- In [2], Paper explained different techniques for the Resume Parsing using Natural Language Processing. They have used OCR to fetch to text from the resume. They are using the Tokens and Semantic Analysis to pre-process the text of resume. They are using Ranking algorithms to select the candidate.

- According to [3], Paper explained Resume Parser with Natural language Processing technique. They have used Lexical analysis, Tokens, Syntactic Analysis, and parse tree for the processing. They are using Json format to fetch the output.

- According to [4] According to paper, Authors are using Stop words removal, Stemming, Lemmatization to fetch the data. In machine learning they are using Random forest, Naïve Bayes, Logistic regression and Linear Support Vector Classifier.

Chapter 4

System Requirement Study

In this chapter, we will learn about the system requirement, specification and functionality.

4.1 User Characteristics

In our system, there will be two types of user

Admin: - In this system admin can only access the all user's data who are previously used our system.

Normal User: - Normal user can visit our web application they can upload the resume, they will get the recommendations based on resume characteristics. They can delete the resume from the system.

4.2 Hardware Software Requirements

	Hardware	**Software**
Developers	4 GB RAM, 256 GB Storage , Intel i5 5th Gen + Processor	Anaconda or Python Pycharm IDE Streamlit, PDFMiner Pyparser, NLTK
Users	Mobile, Tablet/PC, Laptop	Any Browser

4.3 System Main Module

- Resume Storing: User can upload the resume into system, we need to store that resume into our local system/server for future preprocessing.

- PDF Extracting:User can only upload the PDF, now we need to processed PDF into number of Images, so this module will convert the PDF pages into Images
- Text Extracting :After the converting PDF pages into Images, now our next step is to fetch all the text from the images, because for the NLP we need to have the text, so this module will fetch the text from all the Images
- Smart Recommendation: After the getting the text from the resume, we need to create recommendations for the particular data like giving data science skills and courses suggestions to person who have skills of data science. It will display the recommendations to the user.
- Data Store: - After the giving recommendations to the user, our system will store the all user's data, it will be secure with our system. We are storing it for making our system better in future.

4.4 Functional Requirement

- - User need to visit the site.
- - User Upload Resume PDF into the System.
- -System will store the PDF.
- -PDF will be processed into Images.
- -Our System will fetch the Text from Images.
- -System will apply NLP on the text.
- -It will send the recommendation based on the resume to the system.
- -System will display the recommendations on the web applications.

.

4.5 Non - Functional Requirement

- **Accessibility**: This can be used by any one because it is basic in the structure and usage.
- **Efficiency**: This project efficiency as a first trial is not great by numbers but with more learning and hard work the app can be perfect and very efficient.
- **Scalability**: This is now only on computers, so it requires a good internet connection to work properly, still it will take a time to do pre-processing.
- **Security**: We are not providing any Login/Registration for user, still user can upload the resume, resume will be deleted from the system, still user data will be secured with us for future training purpose.

4.6 Project Planning

We have built this project planning in Jira.

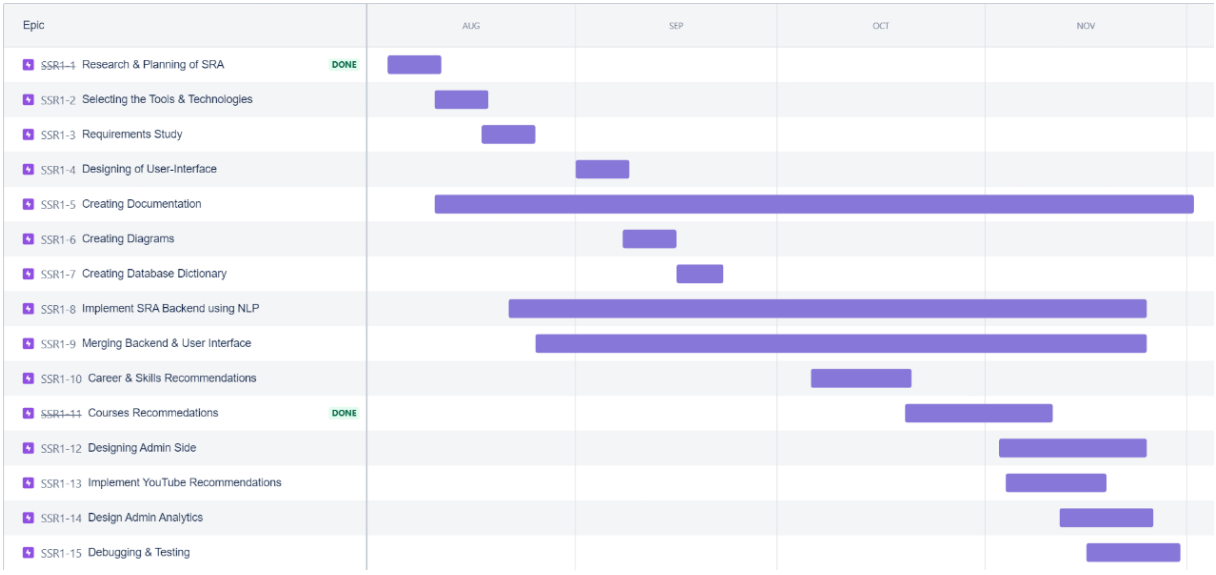


Figure 4.1: Fig1 Project Planning

∩

Chapter 5

Methodology

In this chapter we are going to learn about working of Smart Resume Analyzer.

5.1 Workflow of System

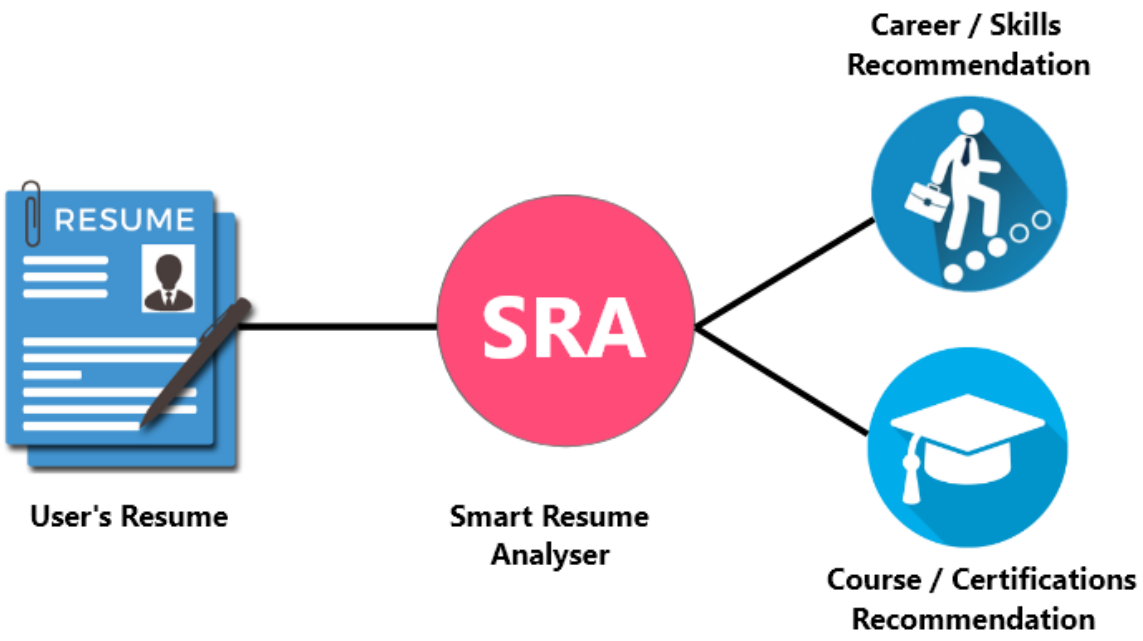


Figure 5.1: Workflow of System

- We are going to see how our SRA system is analyzing the resume, and giving the career, skills, and courses/certifications recommendations.

5.2 Working of Smart Resume Analyzer

- We are going to see how actually our system is working behind, we have divided our work in separate tasks, let's understand each steps of it.

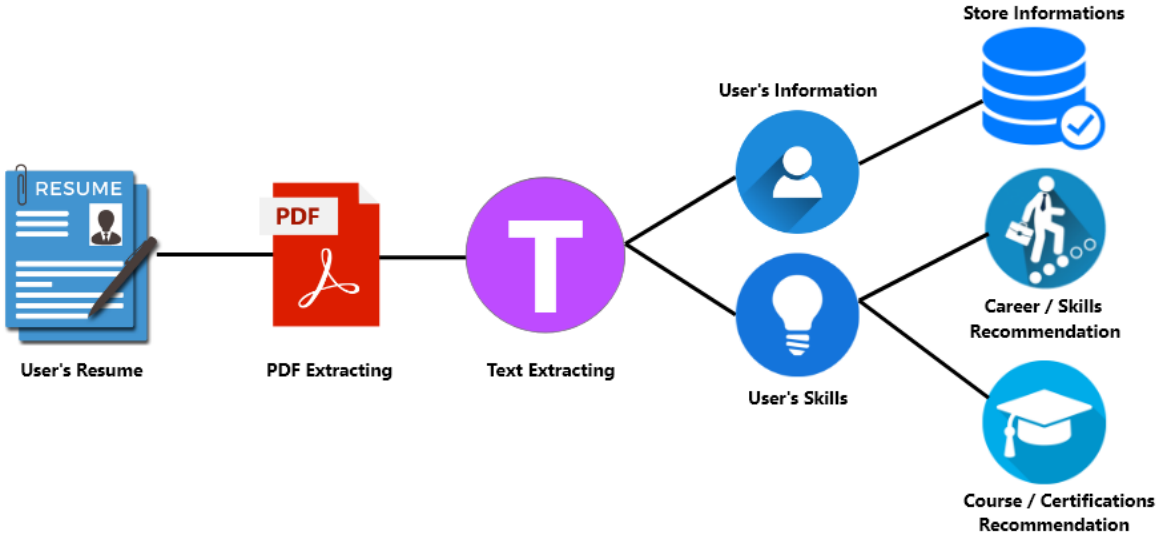


Figure 5.2: Working of Smart Resume Analyzer

5.2.1 PDF Extracting

PDF Extracting is a module that automatically retrieves the user's resume, provided that the resume is in PDF format. This module extracts the user's data from the resume.

5.2.2 Text Extracting

Text Extracting is a module that fetches the text information from the resume. This text data is used for language processing in further tasks, such as recommendations and fetching the user's personal information.

5.2.3 User's Data

After the text extraction, the next module involves fetching the user's information, such as their full name, contact details, email, mobile number, and skills, from the extracted text data.

5.2.4 Career/Skills Recommendations

Based on the user's current skills, this module provides career path and skills recommendations. For example, if the user has skills in Machine Learning, it will offer career, tools, and technology recommendations in that field.

5.2.5 Course/Certifications Recommendations

Similarly, based on the user's skills, this module suggests courses and certifications. For instance, if the user has skills in Machine Learning, it will recommend both free and paid courses and certifications in that domain.

5.2.6 Data Analytics

A new module called Data Analytics has been added. With a substantial amount of user data available, visualization becomes an effective technique for understanding patterns. We create visualizations, particularly pie charts, for the admin side to facilitate easy data comprehension.

5.2.7 YouTube Video Recommendation

Another new module called YouTube recommendations has been incorporated. The system recommends two types of videos to the user: one focusing on resume preparation and the other on interview preparation. These videos are directly scraped from YouTube without using any official API.

Chapter 6

System Design

In this chapter, we will learn about the system designs, diagram and DFDs.

6.1 Use Case Diagram

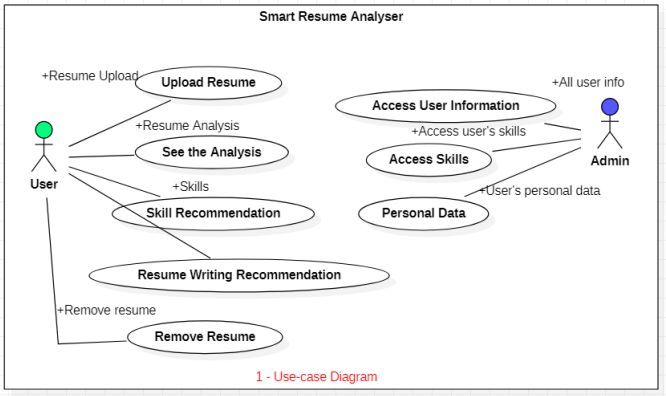


Figure 6.1: Use Case Diagram

- In this diagram, we seen what the functionality can user and admin can use. We can see user and admin have the different access of the functionality.

6.2 Class Diagram

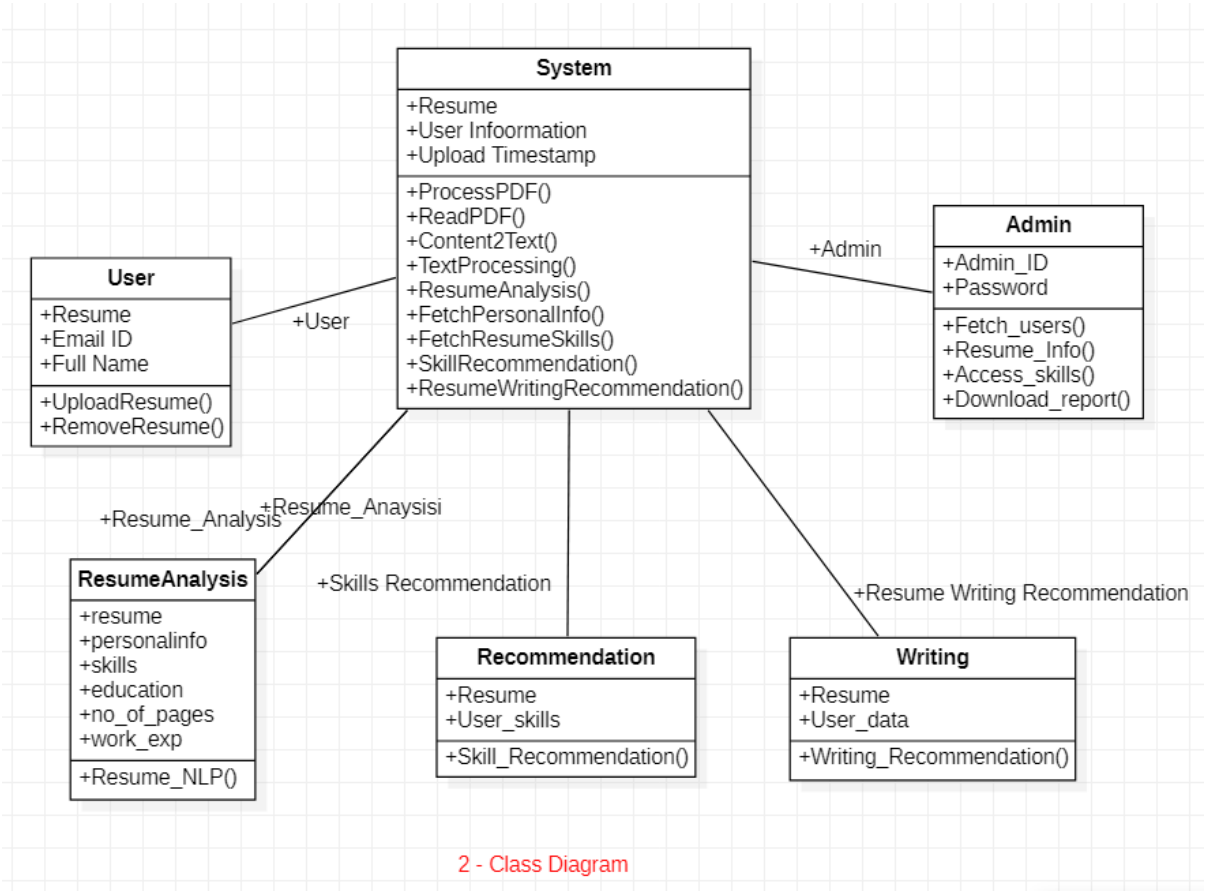


Figure 6.2: Class Diagram

- In this diagram, we seen different classes and functions according to the functionality. We can see how system is associated with the user and admin functionality.

6.3 Sequence Diagram

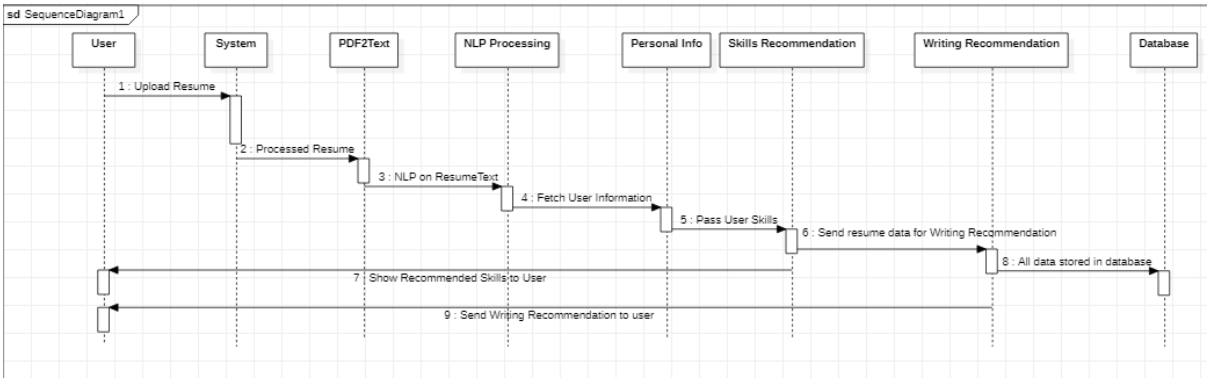


Figure 6.3: Sequence Diagram

- In this diagram, we have seen the step-by-step procedure of Resume Processing. It is showing the sequence from Upload resume to get the Recommendations.

6.4 Activity Diagram

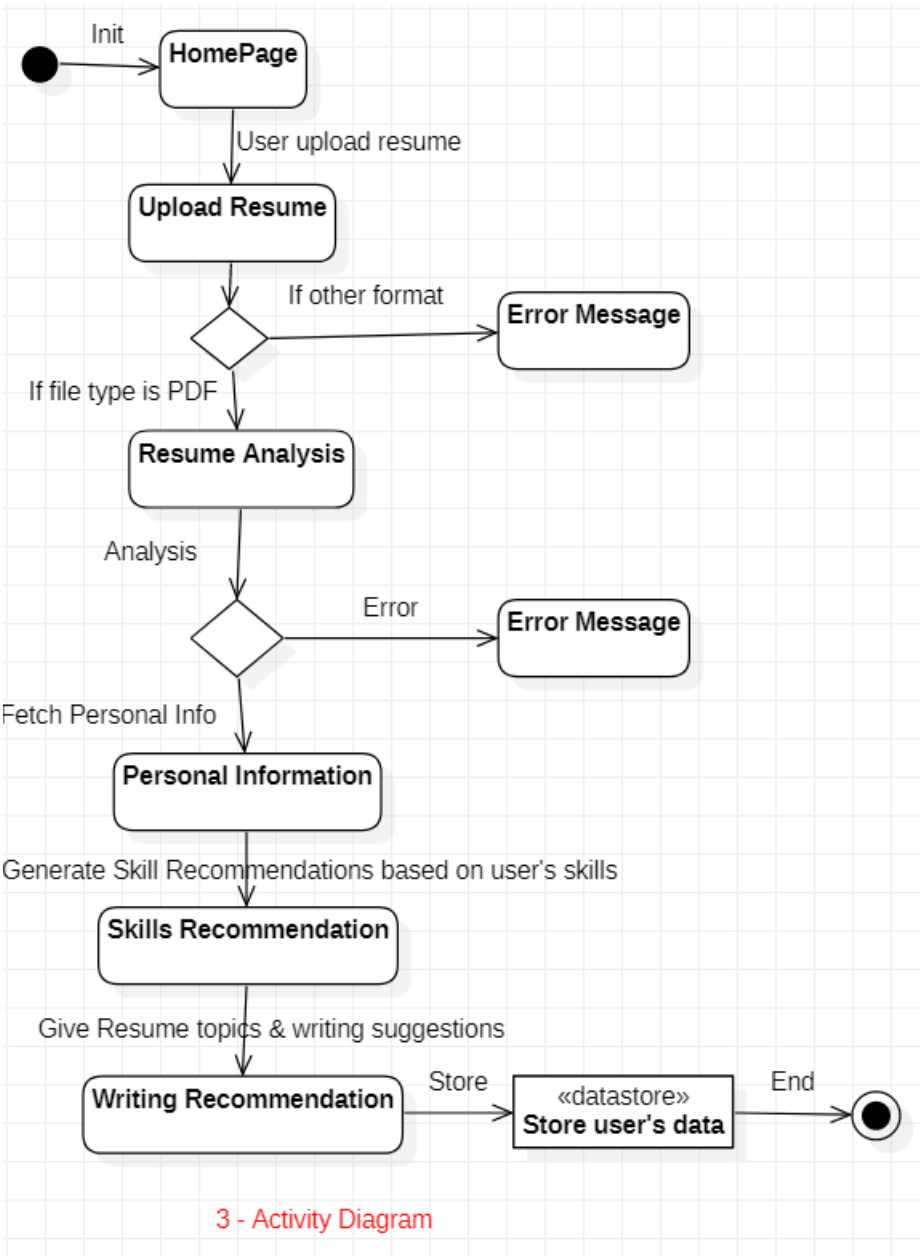


Figure 6.4: Activity Diagram

- In this diagram, we can see that the activity of Smart Resume Analyzer, if everything works well, we will get the output, otherwise it will be error.

6.5 Data-Flow diagram

6.5.1 Level-0 Diagram

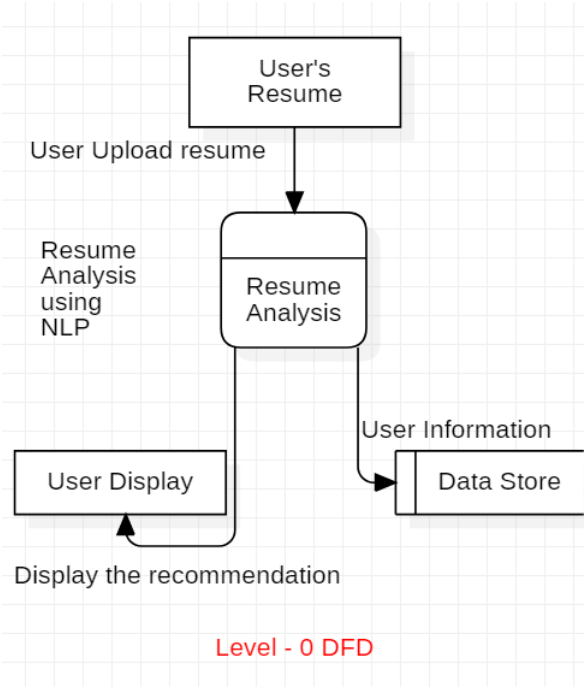


Figure 6.5: Level-0 Diagram

- In this data-flow diagram, we can see that how system is associated with the Resume Analysis function and the database. We can see the data flow from resume to system, system to user and database.

6.5.2 Level-1 Diagram

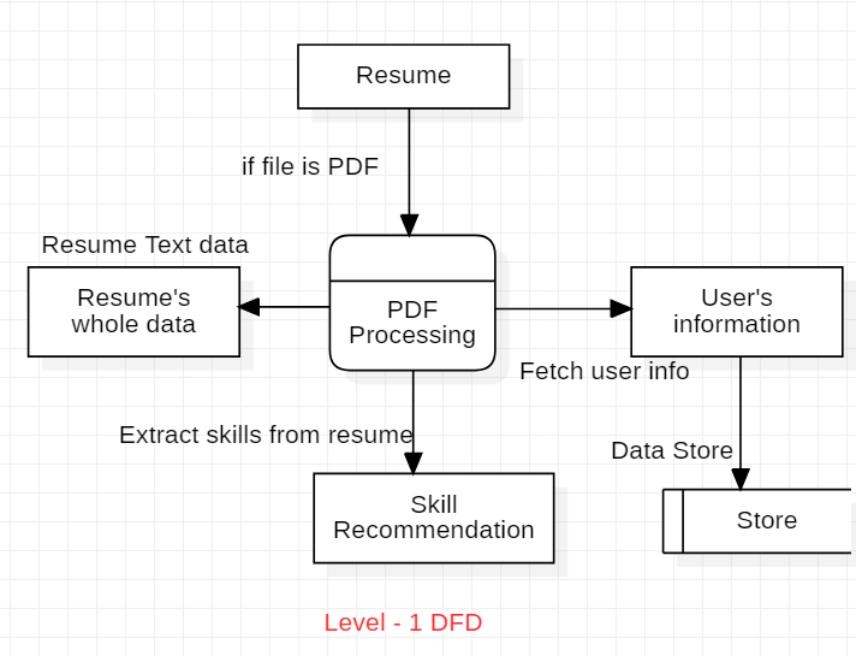


Figure 6.6: Level-1 Diagram

- In this diagram, we can see that how PDF processing is working into the user's resume, it will fetch the text data from the resume.

6.5.3 Level-2 Diagram

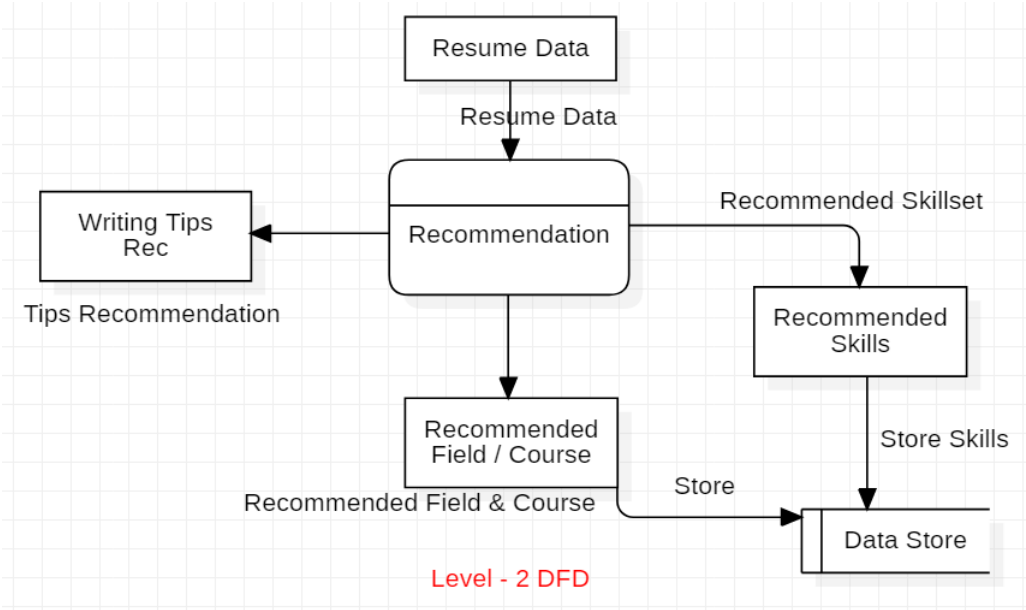


Figure 6.7: Level-2 Diagram

- In this diagram, we can see that how recommendation is working into our system.

Chapter 7

DATABASE DICTIONARY

In this chapter we are going to understand the different tables of our database.

7.1 Data Dictionary

Database Name: - SRA

7.1.1 User Data Table

Table Name: - user data

Column Name	**Data Type**	**Description**
ID	Auto Increment	Unique User ID
User Name	Varchar(50)	Full Name of user
Email ID	Varchar (50)	User Email ID
Resume_score	Varchar (10)d	User's auto generated score
Time_stamp	Varchar (50)	Time stamp
Page_no	Varchar (5)	Number of pages
User_Level	Varchar(10)	User Experience
Actual_skills	Varchar (300)	User's actual skills
Recommended_skills	Varchar (300)	User's Recommended skills
Recommended_courses	Varchar (600)	Recommended courses

Chapter 8

System Implementation

In this chapter we are going to discuss about the coding & implementation of it.

8.1 Coding

The coding is the process of transforming the design of a system into a computer language format. This coding phase of software development is concerned with software translating design specification into the source code. It is necessary to write source code & internal documentation so that conformance of the code to its specification can be easily verified. Coding is done by the coder or programmers who are independent people than the designer. The goal is not to reduce the effort and cost of the coding phase, but to cut to the cost of a later stage. The cost of testing and maintenance can be significantly reduced with efficient coding.

8.2 Module Specification

In our system there will be two modules, admin & user. Admin will be only one. User can be multiple. Let see what user & admin can do.

user	**admin**
Upload the Resume	Do Login
Check the own information	Check the all user's report
Check the skills/career recommendations.	Can export the report in CSV.
Check the Courses/Certifications recommendations	Admin can see the Data Analytics

8.3 Sample coding

The code which is mentioned below it is just main logic of giving the recommendations to the user. This code is only just functional part, not the full part with the backend.

8.3.1 PDF extracting code

This code is used for the PDF extraction; it will extract the Pdf of resume which is uploaded by the user.

```
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
```

8.3.2 Recommender Code

This code is used for the course recommender, it will fetch the user's skills, based on that skills it will give the recommendations.

```
def course_recommender(course_list):
    st.subheader("**Courses & Certificates🎓 Recommendations**")
    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
```

8.3.3 PDF Showing Code

This code is used to show the uploaded resume in the user interface, it simply allows to display the uploaded resume into the system.

```
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)
```

8.3.4 Insert user's Data Code

This code is used to insert the user data into our database.

```
Definsert_data(name,email,mobile,timestamp,no_of_pages,reco_field,
cand_level,skills,recommended_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(mobile),
timestamp,str(no_of_pages), reco_field, cand_level,
skills,recommended_skills,courses)
    cursor.execute(insert_sql, rec_values)
    connection.commit()
```

8.3.5 Visualization Code (Pie Chart)

This code is used display the Pie Chart for data analytics.

```
labels = plot_data.Predicted_Field.unique()
values = plot_data.Predicted_Field.value_counts()
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)
```

There are different codes running for different task, we have just seen the basic functions that are working for our system.

Chapter 9

Testing

In this chapter, we will learn & implement the testing in our application.

9.1 Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

9.2 Testcases

9.2.1 Test-Case for App Running

id	**Testcases**	**Input Data**	**Steps of execution**	**Expected Output**	**Result**
1	User is able to access the web-application	Request from browser	Go to URL	Site UI should be visible	Pass
2	All the UI should be working fine	Request from browser	Click on any component	Every component should do task	Pass

9.2.2 Test-Case Role Switching

id	Testcases	Input Data	Steps of execution	Expected Output	Result
1	User is able to go to "Normal User"	-	Click on the left panel of user's selection	User should redirect to normal user section	Pass
2	User is able to go to "Admin User"	-	Click on the left panel of user's selection	User should redirect to Admin user section	Pass

9.2.3 Test-Case for Upload Resume

id	Testcases	Input Data	Steps of execution	Expected Output	Result
1	User is able to upload Resume	Resume's PDF	Click on the Upload or Drag and Drop	Execution should be start of SRA	Pass
2	User get the message if they are not registered in system	Id, Password	Click on the Login.	No account found	Pass

9.2.4 Test-Case for Smart Resume Analyzer (SRA)

id	Testcases	Input Data	Steps of execution	Expected Output	Result
1	System is able to fetch the User's information from the uploaded resume	User's Resume	Automatic	User's Data	Pass
2	Page numbers of resume should be generated	User's Resume	Automatic	Experience according to the resume pages	Pass
3	User's skills should be generated	User's Resume	Automatic	User's Skills	Pass
4	Recommended Career path	User's Skills	Automatic	Recommend Career Path	Pass
5	Courses & Skills should be recommended	User's Skill	Automatic	Recommend skills & courses	Pass

6	Resume Writing tips & suggestions should be generated	User's Skills	Automatic	Resume writing tips and suggestions	Pass
7	Resume Score should be generated	User's Skills	Automatic	Resume Score	Pass
8	YouTube Videos should be recommended	User's Skills	Automatic	YouTube Videos related to resume/interview tips	Pass

9.2.5 Test-Case for Admin Login

id	Testcases	Input Data	Steps of execution	Expected Output	Result
1	Admin is able to login using Admin Credentials	Admin Id, Password	Click on the Admin in role's selection in left panel.	Admin should redirect to Admin Panel	Pass
2	Admin get the message when ID is correct but password is wrong	Admin Id, Password	Click on the Admin Login	Wrong password	Pass
3	Admin can see the all user's data into UI(Table)	User's Data	Automatic	User's Data	Pass
4	Admin can see the Data Analytics (Pie Charts) based on the Data	User's Data	Automatic	Visualizations	Pass

5	Admin can download user's data into CSV	User's Data	Automatic	CSV file should be downloaded	Pass

Chapter 10

Result Screenshots

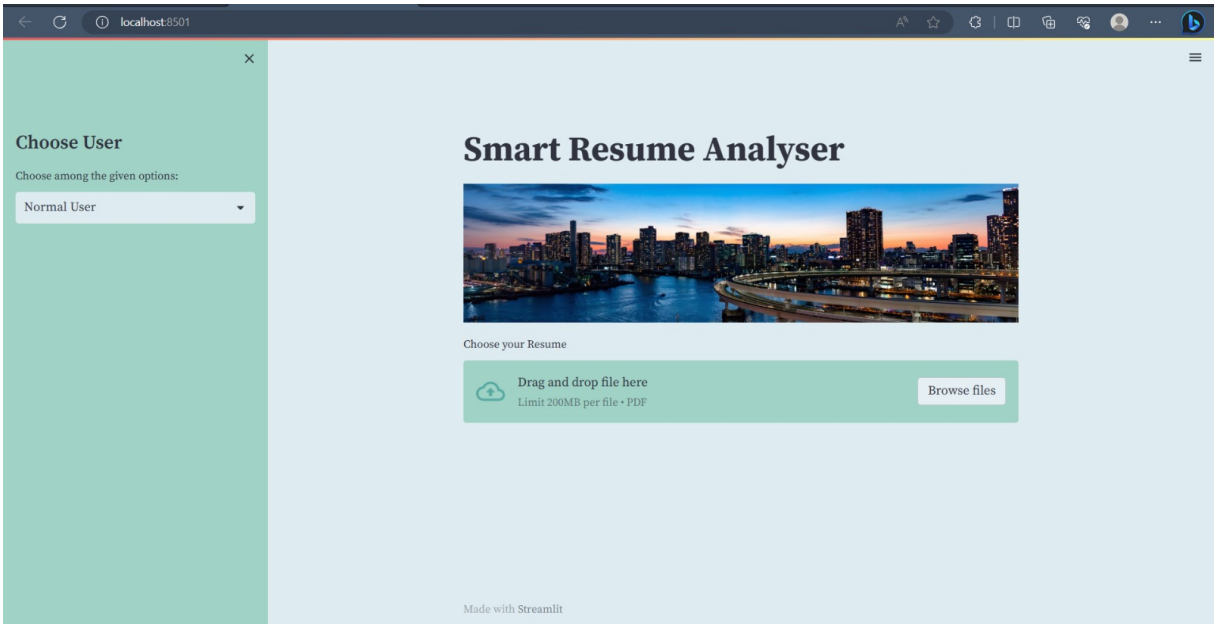
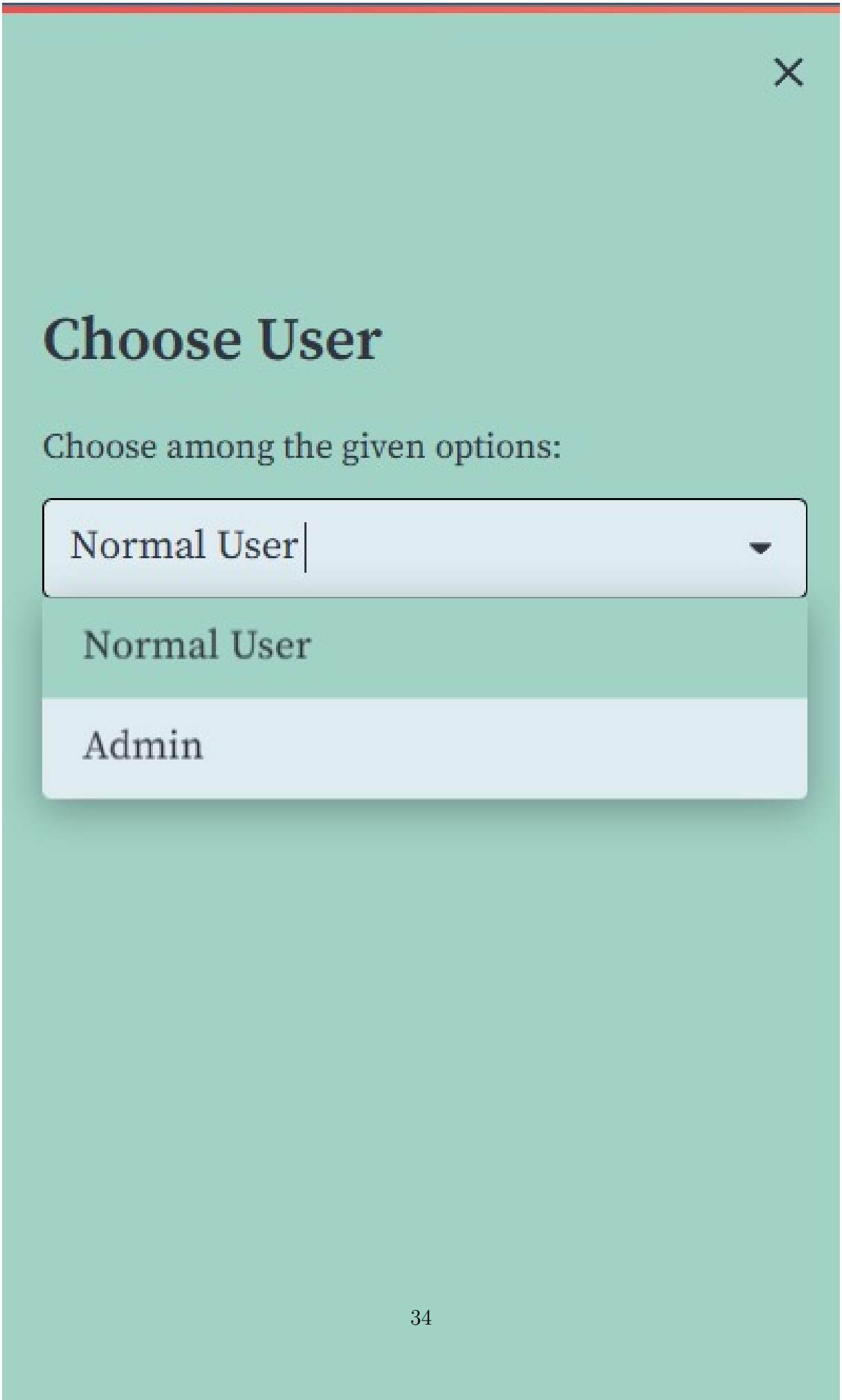


Figure 10.1: Working of Smart Resume Analyzer


Choose User
Choose among the given options:
Normal User
Normal User
Admin

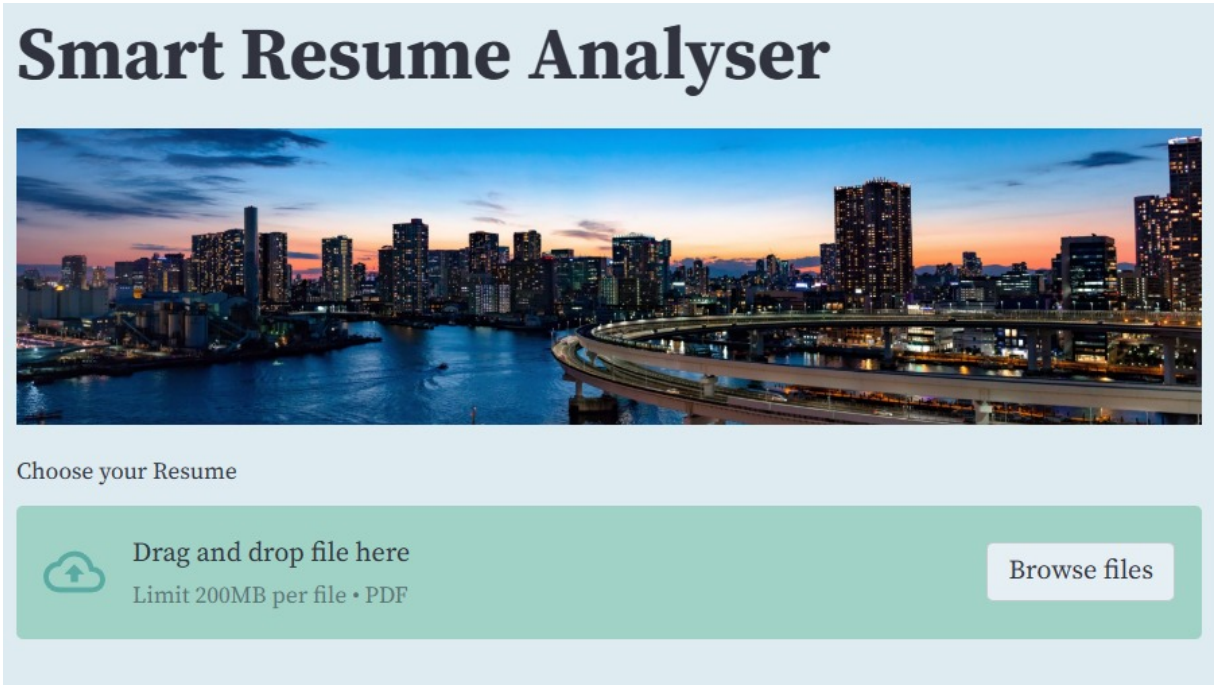


Figure 10.3: Working of Smart Resume Analyzer

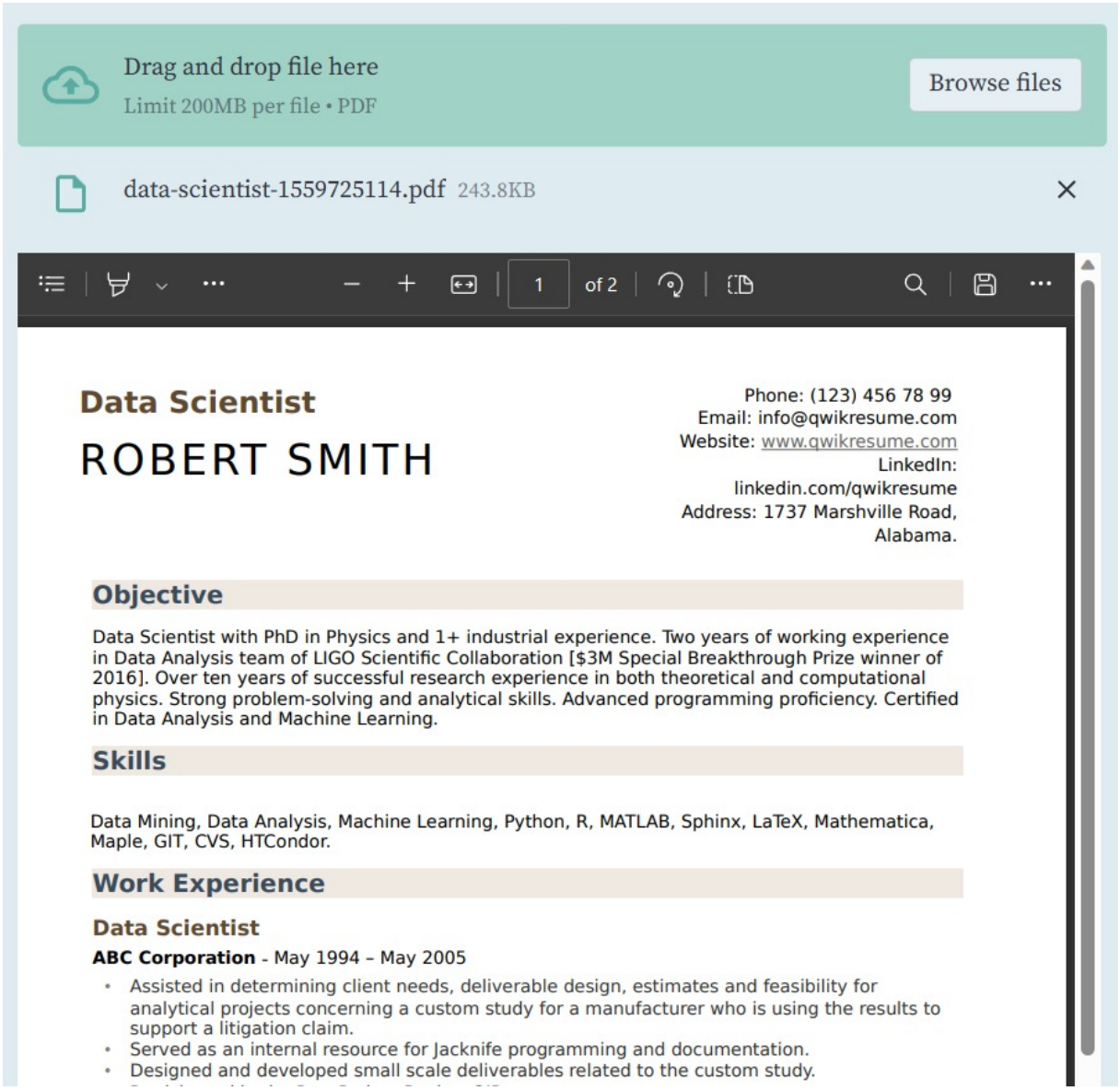


Figure 10.4: Working of Smart Resume Analyzer

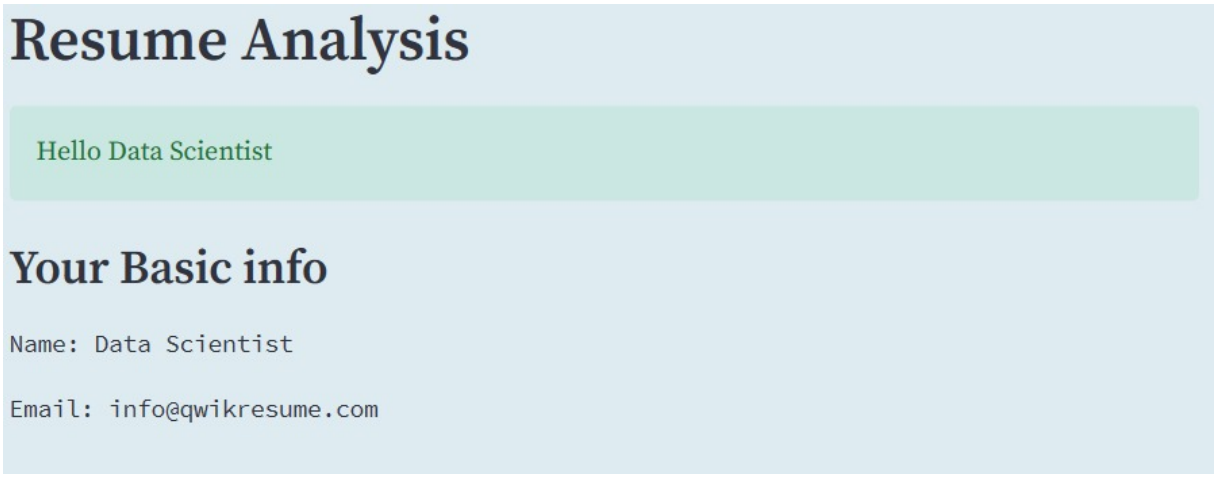


Figure 10.5: Working of Smart Resume Analyzer

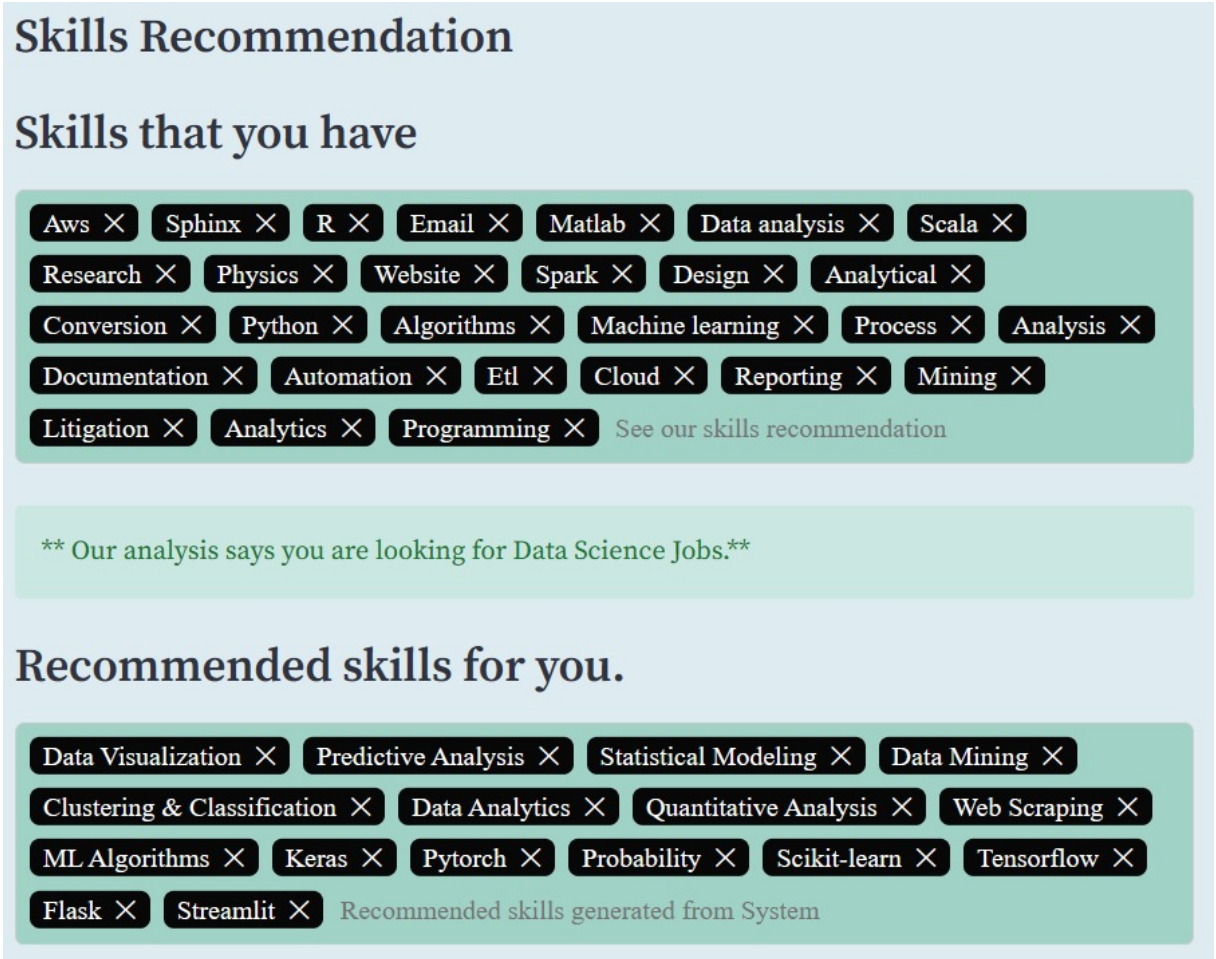


Figure 10.6: Working of Smart Resume Analyzer

Figure 10.7: Working of Smart Resume Analyzer

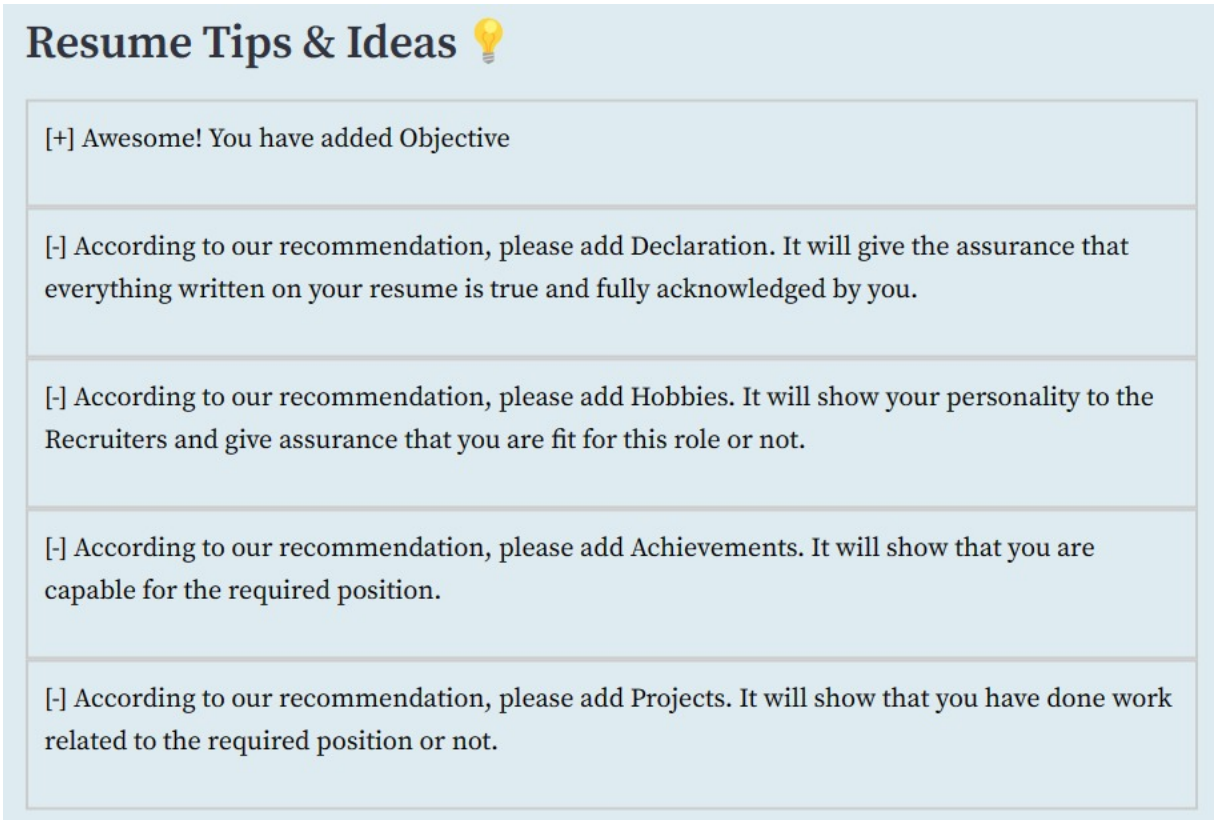


Figure 10.8: Working of Smart Resume Analyzer

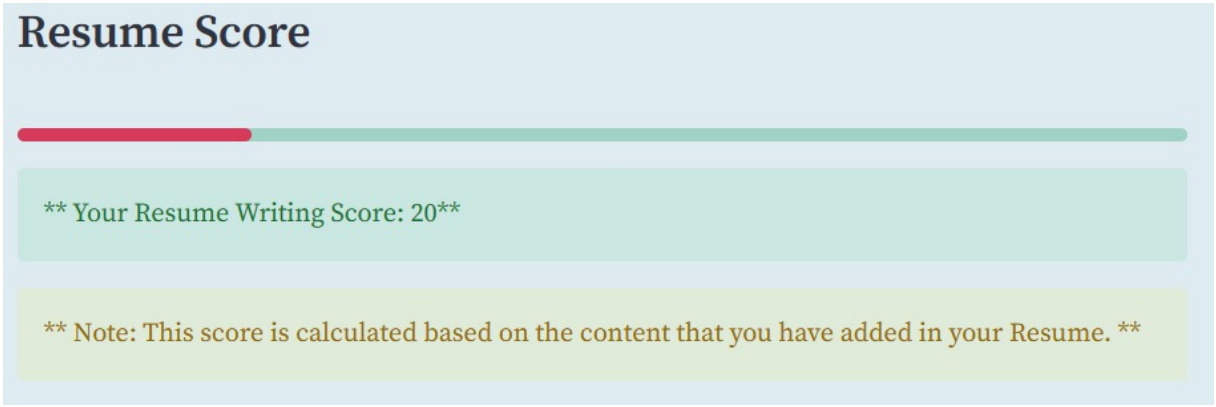


Figure 10.9: Working of Smart Resume Analyzer

Figure 10.10: Working of Smart Resume Analyzer

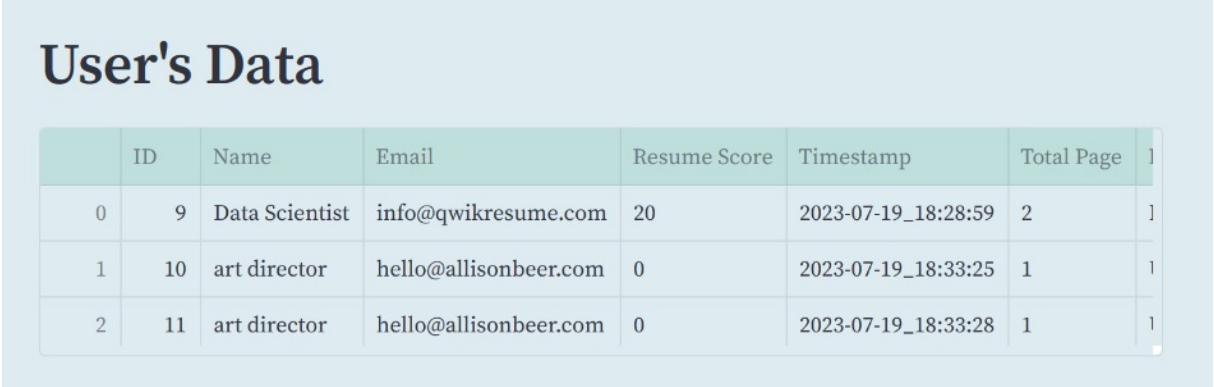


Figure 10.11: Working of Smart Resume Analyzer

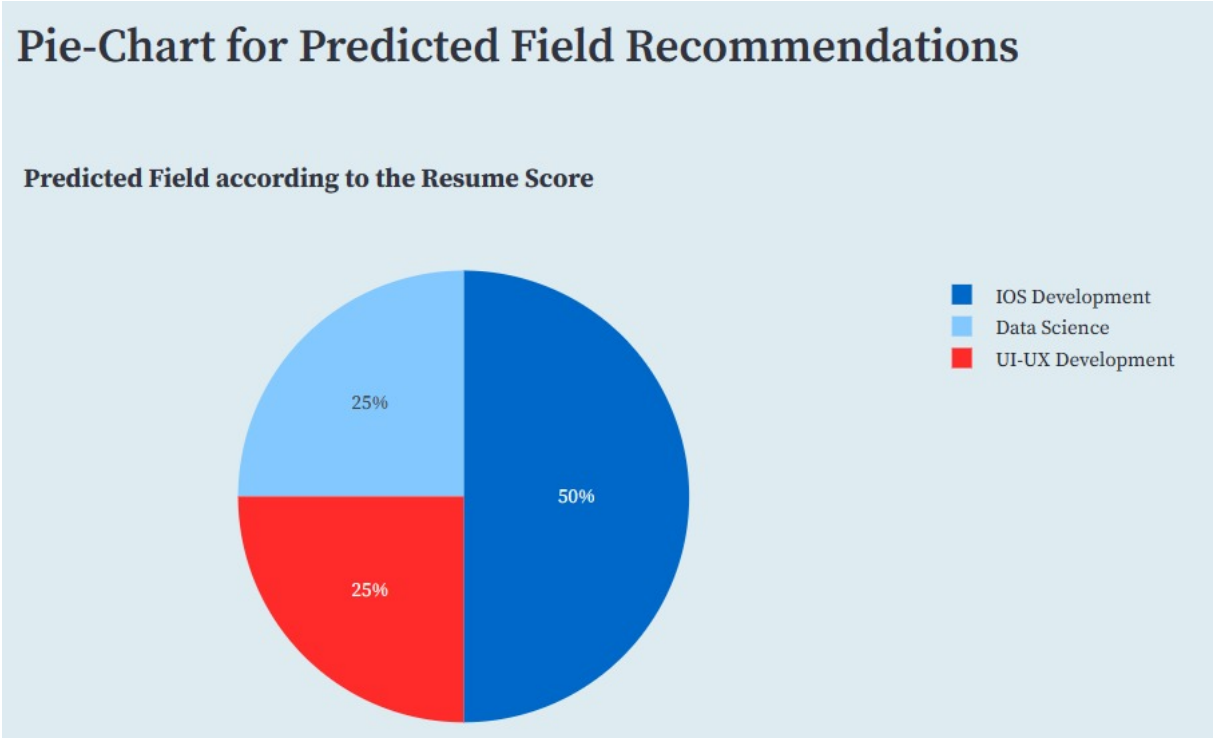


Figure 10.12: Working of Smart Resume Analyzer

Chapter 11

Risks and Challenges

1. Reliance on Keyword-based Analysis: One of the key risks associated with the smart resume analyzer application is its heavy reliance on keywords for assessing candidate skills. While keywords can provide a valuable indication of a candidate's expertise, there is a risk of oversimplification and misinterpretation. Since the application primarily focuses on keywords, there is a possibility that candidates who possess significant field knowledge but use fewer keywords to represent their skills might be inaccurately classified as beginners or intermediates. This limitation can lead to a lack of nuance in evaluating the true capabilities of candidates, potentially resulting in mismatches between job requirements and candidate qualifications.

2. Limited Applicability to Freshers: Another challenge associated with the smart resume analyzer is its restricted scope, as it is currently only designed to cater to freshers. This limitation can hinder its usefulness for organizations seeking candidates with experience beyond entry-level positions. By exclusively targeting freshers, the application may not adequately address the hiring needs of companies looking to fill mid-level or senior roles. This restriction can limit the application's market appeal and potential user base, posing challenges for its adoption and long-term viability.

3. Variability in Keywords and Industry-Specific Jargon: Different industries and job roles often have their own specific terminologies, acronyms, and jargon. This presents a challenge for the smart resume analyzer as it needs to accurately recognize and interpret industry-specific keywords in order to assess candidate skills effectively. Failure to account for these industry nuances can result in the application misjudging a candidate's qualifications or misaligning them with job requirements. To mitigate this risk, continuous updates and maintenance of the application's keyword database would be necessary to keep up with evolving industry terminology.

4. Potential Bias and Inequality: Automated resume analysis systems, including the smart resume analyzer, can be susceptible to biases. The selection and weighting of keywords could inadvertently favor certain demographics or perpetuate existing inequalities in the job market. For example, if the system predominantly associates specific keywords with certain genders or ethnic backgrounds, it may unintentionally discriminate against qualified candidates from underrepresented groups. Ensuring fairness, diversity, and inclusivity in the application's algorithm and training data should be a priority to minimize such biases and promote equal opportunities.

5. User Adoption and Trust: A significant challenge for any new technology or application is gaining user adoption and establishing trust among potential users. Organizations may be hesitant to fully rely on an automated resume analyzer, especially if they have concerns about its accuracy and potential limitations. Building trust through rigorous testing, providing transparency about the system's limitations, and demonstrating the application's reliability can help address these concerns. Ongoing user feedback and continuous improvement of the system based on user input are also crucial for enhancing user adoption and satisfaction.

6. Technical Limitations and Integration: The smart resume analyzer may face technical challenges related to scalability, performance, and integration with existing HR systems. Processing a large volume of resumes efficiently and providing real-time analysis can be demanding in terms of computational resources. Additionally, integrating the application seamlessly with different applicant tracking systems (ATS) and HR software platforms may require technical adaptations and compatibility considerations. Addressing these technical hurdles is essential for ensuring a smooth user experience and the successful implementation of the smart resume analyzer within various organizational contexts.

Chapter 12

Conclusion

Conclusion + Future scope

12.1 Future scope

- Currently web applications are deployed locally, our future aim is to deploy them on the internet (AWS or Heroku).
- In the future, we will add more formats of resumes. Currently, the system only supports PDF format for uploading resumes.
- This system currently works with a limited set of fields and recommendations, specifically designed for IT professionals. We plan to add more fields and data in the future to provide recommendations for all types of resumes.
- We have achieved good accuracy in fetching user data, but sometimes the displayed data fetched from the PDF is incorrect. We will improve this in the future.
- Currently, we have not implemented the display of charts and visualizations on the user side. We will add this functionality in the future, which will show various charts based on the data.
- In the mobile view of the application, there are occasional UI lags. We will address this issue soon.

12.2 Conclusion

The purpose of this project is to learn the Natural Language Processing. We have research too many skills, tools and technologies for different types of IT jobs. Then we have done NLP processing that can recommend the Skills, Courses and career field. We learn

to create one-page application development using Streamlit python. We have gathered the knowledge of Plotly for the visualization and Data analytics which is created in Admin Side. We have covered this project according to scheduled time. In this project we learn time management, Non-technical skills like documentations, diagrams drawing, Technical skills like NLP, Database handling, Visualization, Web scraping, Web development and many more things from this project.

References

[1] Mr. Chirag Dariyani, Gurmeet Singh Chhabra, Harsh Patel, Indrajit Chhabra. "An Automated Resume Screening Using Natural Language Processing And Similarity." *Ethics and Information Technology.*

[2] Shubham Bhor, Vivek Gupta, Vishak Nair, Harish Shinde, Mansi Kulkarni. "A Resume Parser Using Natural Language Processing Technique." *International Journal of Research in Engineering and Science (IJRES).*

[3] Satyak Sanyal, Souvik Hazra, Neelanjan Ghosh, Soumyashree Adhikary. "Resume Parser with Natural Language Processing." *2017 IJESC.*

[4] Pradeep Roy, Sarabjeet Chaudhary, Rocky Bhatia. "A Machine Learning Approach for Automation of Resume Recommendation System." *ICCIDS 2019.*

[5] Streamlit Documentation. *Documentation from Streamlit Developer Community.*

[6] OpenCV Documentation. *Documentation from OpenCV Developer Community.*

[7] Pillow Documentation. *Official documentation from PIL Developer Community.*

[8] NLTK Documentation. *Official documentation from NLTK Developer Community.*

[9] PyResParser Documentation. *Official documentation from PyResParser Developer Community.*

[10] PDFMiner Documentation. *Official documentation from PDFMiner Developer Community.*

Appendix A: Sample Code

SMART RESUME ANALYZER

```
import streamlit as st
import nltk
import spacy
nltk.download('stopwords')
spacy.load('en_core_web_sm')

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
import yt_dlp as youtube_dl

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

def get_table_download_link(df, filename, text):
    """Generates a link allowing the data in a given panda dataframe to be downloaded
    in:  dataframe
    out: href string
    """
    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
        rec_course.append(c_name)
        if c == no_of_reco:
            break
    output = ""
    for i, course_name in enumerate(rec_course):
        output += f"({i+1}) [{course_name}]({course_list[i][1]})\n"
    st.markdown(output)
    return rec_course



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

```
def insert_data(name, email, res_score, timestamp, no_of_pages, reco_field, cand_level, skills,
recommended_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)"""
   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)

   # st.sidebar.markdown(link, unsafe_allow_html=True)
   img = Image.open('./Logo/pexels-pixabay-356830.jpg')
   #img = img.resize((250, 250))
   st.image(img)

   # Create the DB
   db_sql = """CREATE DATABASE IF NOT EXISTS SRA;"""
   cursor.execute(db_sql)
   connection.select_db("srra")

   # 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 += 20
        st.markdown(
            '''
            <div style="border: 2px solid #ccc; padding: 10px;">
                <p style="color: black;">[+] Awesome! You have added Objective</p>
            </div>
            ''',
            unsafe_allow_html=True)
    else:
        st.markdown(
            '''
            <div style="border: 2px solid #ccc; padding: 10px;">
                <p style="color: black;">[-] According to our recommendation, please add your
```

```
career objective. It will give your career intention to the Recruiters.</p>
                </div>
                ''',
                unsafe_allow_html=True)

        if 'Declaration' in resume_text:
            resume_score += 20
            st.markdown(
                '''
                <div style="border: 2px solid #ccc; padding: 10px;">
                    <p style="color: black;">[+] Awesome! You have added Declaration</p>
                </div>
                ''',
                unsafe_allow_html=True)
        else:
            st.markdown(
                '''
                <div style="border: 2px solid #ccc; padding: 10px;">
                    <p style="color: black;">[-] 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.</p>
                </div>
                ''',
                unsafe_allow_html=True)

        if 'Hobbies' in resume_text or 'Interests' in resume_text:
            resume_score += 20
            st.markdown(
                '''
                <div style="border: 2px solid #ccc; padding: 10px;">
                    <p style="color: black;">[+] Awesome! You have added your Hobbies</p>
                </div>
                ''',
                unsafe_allow_html=True)
        else:
            st.markdown(
                '''
                <div style="border: 2px solid #ccc; padding: 10px;">
                    <p style="color: black;">[-] According to our recommendation, please add Hobbies. It
will show your personality to the Recruiters and give assurance that you are fit for this role or
not.</p>
                </div>
                ''',
                unsafe_allow_html=True)

        if 'Achievements' in resume_text:
            resume_score += 20
            st.markdown(
```

```
                '''
                <div style="border: 2px solid #ccc; padding: 10px;">
                    <p style="color: black;">[+] Awesome! You have added your Achievements</p>
                </div>
                ''',
                unsafe_allow_html=True)
            else:
                st.markdown(
                '''
                <div style="border: 2px solid #ccc; padding: 10px;">
                    <p style="color: black;">[-] According to our recommendation, please add Achievements. It will show that you are capable for the required position.</p>
                </div>
                ''',
                unsafe_allow_html=True)

            if 'Projects' in resume_text:
                resume_score += 20
                st.markdown(
                '''
                <div style="border: 2px solid #ccc; padding: 10px;">
                    <p style="color: black;">[+] Awesome! You have added your Projects</p>
                </div>
                ''',
                unsafe_allow_html=True)
            else:
                st.markdown(
                '''
                <div style="border: 2px solid #ccc; padding: 10px;">
                    <p style="color: black;">[-] According to our recommendation, please add Projects. It will show that you have done work related to the required position or not.</p>
                </div>
                ''',
                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. **")


        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))



        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 == 'admin' and ad_password == 'admin':
        st.success("Welcome Admin")
        # 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(plot_data, values='resume_score', names='Predicted_Field', title='Predicted
Field according to the Resume Score')
            st.plotly_chart(fig)



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


run()
```