

AUTOMATED SKILL-EXTRACTING WEBSITE AND RESUME ANALYZER

**A Project Report submitted in partial fulfillment of the requirements for
the award of the degree of**

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING**

Submitted by:

**B AKSHITH-222010303005
ABHISHT NARAYAN-222010303010
G KARTHIK REDDY-222010303050
CHETNA TIKARIHA-222010304010**

Under the esteemed guidance of

**DR. K. Dhanasree Devi,
Associate Professor,
CSE Department,
GITAM (Deemed to be University), Hyderabad.**



**Department of Computer Science and Engineering
GITAM School of Technology
GITAM (Deemed To Be University)
Hyderabad Campus-502329
MARCH-2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM (Deemed to be University)**



DECLARATION

We, hereby declare that the project report entitled “**AUTOMATED SKILL EXTRACTING WEBSITE AND RESUME ANALYZER**” is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No(s).	Name(s)	Signature(s)
222010303005	B AKSHITH	
222010303010	ABHISHT NARAYAN	
222010303050	G KARTHIK REDDY	
222010304010	CHETNA TIKARIHA	

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM (Deemed to be University)
Hyderabad Campus-502329
JUNE-2023**



CERTIFICATE OF COMPLETION

This is to certify that the project report entitled “**AUTOMATED SKILL EXTRACTING WEBSITE AND RESUME ANALYZER**” is a Bonafide record of work carried out by B Akshith (222010303005); Abhisht Narayan (222010303010); G Karthik Reddy (222010303050); Chetna Tikariha (222010304010) students submitted in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

Project Guide

**DR.K.Dhanasree Devi
Associate Professor
Dept. of CSE**

Head of the Department

**Mahaboob Basha Shaik
Professor & Head
Dept of CSE**

ACKNOWLEDGEMENT

Our project report would not have been successful without the help of several people. We would like to thank the personalities who were part of our seminar in numerous ways, those who gave us outstanding support from the birth of the seminar.

We are extremely thankful to our honorable Pro-Vice-Chancellor, **Prof. D. Sambasiva Rao**, for providing the necessary infrastructure and resources for the accomplishment of our seminar. We are highly indebted to **Prof. N. Seetha Ramaiah**, Associate Director, School of Technology, for his support during the tenure of the seminar.

We are very much obliged to our beloved **Prof. Mahaboob Basha Shaik**, Head of the Department of Computer Science & Engineering, for providing the opportunity to undertake this seminar and encouragement in the completion of this seminar.

We hereby wish to express our deep sense of gratitude to **Dr. K. Dhanasree Devi**, Associate Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support, and invaluable advice provided for the success of the project report.

We are also thankful to all the Computer Science and Engineering department staff members who have cooperated in making our seminar a success. We would like to thank all our parents and friends who extended their help, encouragement, and moral support directly or indirectly in our seminar work.

Sincerely,

B Akshith - 222010303005

Abhisht Narayan-222010303010

G Karthik Reddy-222010303050

Chetna Tikariha-222010304010

TABLE OF CONTENTS

S.NO	CONTENT	PGNO
1	Introduction:	
	1.1 Objective	8
	1.2 Problem Statement	8
	1.3 Software requirements	9
	1.4 Hardware requirements	10
2	Feasibility study:	
	2.1 Economic feasibility	12
	2.2 Technical feasibility	12
	2.3 Social feasibility	12
3	Literature survey	14
4	System analysis:	
	4.1 Existing system	18
	4.1.1 Disadvantages of existing system	18
	4.2 Proposed system	19
	4.2.1 Advantages of the proposed system	19
	4.3 Functional requirements	20
	4.4 Non-Functional requirements	20
5	System design:	
	5.1 System architecture	21
	5.2 UML diagrams	23
6	Implementation	
	6.1 Modules	31
	6.2 Sample code	31
7	Software environment	42
8	System testing:	
	8.1 Testing strategies	47
	8.2 Test cases	50

9	Screens	51
10	Conclusion	60
11	References	61

ABSTRACT

The most qualified applicant for a position must be found through careful consideration of job applications, which is done during the Automated Evaluation of Resumes Using NLP stage of the hiring process. [1] Automated resume screening is now a practical alternative to the manual screening procedure because of developments in deep learning and natural language processing (NLP) [7]. In this paper, we examine a few contemporary methods for screening automated resumes. To increase the precision and effectiveness of the screening process, these approaches employ a variety of methods including hybrid deep learning frameworks, transfer learning, genetic algorithms, and multisource data. Also, some research investigates the use of job descriptions to improve resume screening precision. This research's experimental findings show that the suggested strategies are more effective than conventional ones. The results of this study can help human resource managers and recruiters automate the hiring process and efficiently and impartially identify viable applicants.

1. INTRODUCTION

An essential step in the hiring process is the automatic review of resumes, which entails assessing job applications to find the applicant most suited for a given position. This procedure may take a long time and be prone to human mistakes, which could lead to the loss of qualified individuals. Automated resume screening has grown in popularity recently as a solution to this problem. Automatic resume screening uses several methods to enhance accuracy and efficiency, including deep learning algorithms, machine learning, and natural language processing (NLP). Several studies have suggested various methods for automating the screening of resumes. Li et al. (2020) introduced a hybrid deep learning framework that makes use of long short-term memory (LSTM) networks and convolutional neural networks (CNNs) [6].

1.1 OBJECTIVE

The main goal of using NLP algorithms for resume screening, such as cosine similarity and S-BERT, is to ensure that the most qualified individuals are found and given further consideration while automating the hiring process. The specific goal of the recruiting process is to become more effective by automating the screening of job applications. to provide a more objective method to reduce the possibility of biases in manual screening by utilizing cutting-edge NLP algorithms such as cosine similarity and S-BERT to improve resume screening accuracy. to increase the number of resumes processed while saving time and money by eliminating the need for human screening. To improve the candidate's experience, a faster and more effective screening procedure is offered. Improving the quality of the hiring process.

1.2 PROBLEM STATEMENT

The creation of an automated system that can efficiently filter and score job applications based on their resemblance to a given job description is the main objective of the resume screening project that uses NLP techniques like cosine similarity and S-BERT [9]. The talents listed on the resumes are then ascertained. The essential information from the resume is extracted using the resume parser package.

1.3 SOFTWARE REQUIREMENTS:

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Platform – In computing, a platform describes some sort of framework, either in hardware or software, that allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries.

The operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of the same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, the software is designed using newer features of Linux Kernel v2.6 and generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

APIs and drivers – Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

Web browser – Most web applications and software depending heavily on Internet technologies make use of the default browser installed on the system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

- 1) **FRAMEWORK - FLASK/DJANGO**
- 2) **DATABASE - MYSQL WORK BENCH/SQLITE 3**
- 3) **USER INTERFACE LANGUAGES- HTML, CSS, JS, BOOTSTRAP**
- 4) **FRONTEND LANGUAGES - PYTHON**
- 5) **SOFTWARE INSTALLED - PYTHON IDLE (3.7.0)**

1.4 HARDWARE REQUIREMENTS:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in the case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Architecture – All computer operating systems are designed for a particular computer architecture. Most software applications are limited to operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power – The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture defines processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speeds often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity and are often mentioned in this category.

Memory – All software, when run, resides in the random-access memory (RAM) of a computer. Memory requirements are defined after considering the demands of the application, operating system, supporting software and files, and other running processes. The optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary storage – Hard disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display adapter – Software requiring a better-than-average computer graphics display, like graphics editors and high-end games, often defines high-end display adapters in the system requirements.

Peripherals – Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

1) Operating System: Windows Only

2) Processor: i5 and above

3) RAM: 8GB and above

4) Hard Disk: 25 GB in local drive

2. FEASIBILITY STUDY

Feasibility Study:

A feasibility study evaluates a project's or system's practicality. As part of a feasibility study, the objective and rational analysis of a potential business or venture is conducted to determine its strengths and weaknesses, potential opportunities and threats, resources required to carry out, and ultimate success prospects. Two criteria should be considered when judging feasibility: the required cost and expected value.

Types Of Feasibility Study:

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility studies—separate areas that a feasibility study examines, described below.

1. Technical Feasibility:

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team can convert the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to put Star Trek's transporters in their building—currently, this project is not technically feasible.

2. Economic Feasibility:

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

3. Legal Feasibility:

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts, or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal location isn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

4. Operational Feasibility:

This assessment involves undertaking a study to analyze and determine whether—and how well—the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

5. Scheduling Feasibility:

This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

- **Internal Project Constraints:** Technical, Technology, Budget, Resource, etc.
- **Internal Corporate Constraints:** Financial, Marketing, Export, etc.
- **External Constraints:** Logistics, Environment, Laws, and Regulations, etc.

3. LITERATURE SURVEY

3.1 Automated resume screening and evaluation using machine learning techniques:

<https://www.sciencedirect.com/science/article/pii/S187705092030750X>

ABSTRACT: Finding suitable candidates for an open role could be a daunting task, especially when there are many applicants. It can impede team progress for getting the right person on the right time. An automated way of “Resume Classification and Matching” could really ease the tedious process of fair screening and shortlisting, it would certainly expedite the candidate selection and decision-making process. This system could work with a large number of resumes for first classifying the right categories using different classifiers, once classification has been done then as per the job description, top candidates could be ranked using Content-based Recommendation, cosine similarity and by using k-NN to identify the CVs that are nearest to the provided job description.

3.2 A study on the extraction of competencies from job postings and their correlation with resumes using natural language processing:

https://www.researchgate.net/publication/361772014_RESUME_PARSER

ABSTRACT: Agencies and various high-level firms must deal with a large number of new jobs seeking people with various resumes. However, managing large amounts of text data and selecting the best-fit candidate is more difficult and time-consuming. This paper provides an overview of an ongoing Information Extraction System project that helps recruiters identify the best candidate by extracting relevant information from the resume. This project presents a system that uses Natural Language Processing (NLP) techniques to extract minute data from a resume, such as education, experience, skills, and experience. The recruiting process is made easier and more efficient by parsing the resume. The proposed system is made up of three modules: an administration management system, a File upload and parser system, and an information extraction system. The administrator will upload the applicant's resume into the system, and the relevant information will be extracted in a structured format. Using the parsed information from the Resume, HR can select the best candidate for the job based on the company's needs.

3.3 Resume screening using deep learning and natural language

processing:

https://link.springer.com/chapter/10.1007/978-981-19-0011-2_58

ABSTRACT: Recruitment, a worldwide industry with millions of resumes being uploaded on N-number of hiring websites for countless jobs on a daily routine, with the great evolution in the online-based hiring process. Job requirement is considered one of the major activities for humans which is a very strenuous job to find a fruitful talent. Every job seeker will have their novel to organize the data blocks with unique style and format for representation, thereby the resumes become an extraordinary illustration of data that is unstructured. Choosing a fantastic resume from a collection of unstructured resumes is very simple with a resume parser, which adapts the process of extracting structured data from a collection of unstructured resumes. Machine-understandable output is obtained using the set of instructions that investigates and abstracts resume/CV data. This helps to store and analyze data automatically. This paper proposes a smart recruitment system (SRS) which includes resume classification using deep learning along with natural language processing (NLP) techniques and automatic questionnaires' which measure technical proficiency of an applicant, and by using syntactical similarity and semantic similarity measurements to identify the suitable candidate according to the requirement of the company skill set. To abstract the significant data for the hiring process using the natural language process, we propose to use named entity recognition. The proposed procedure irrespective of format whether it is a PDF or Word document of the resume, will pick up the data associated with the candidate-like work experience and education. Parsing and ranking the resume makes the hiring process easy and efficient.

3.4 Resume Screening Using Semantic Similarity and Clustering Algorithms:

https://www.researchgate.net/publication/347633082_AN_AUTOMATED_RESUME_SCREENING_SYSTEM_USING_NATURAL_LANGUAGE_PROCESSING_AND_SIMILARITY

ABSTRACT: A typical job posting on the Internet receives a massive number of applications within a short window of time. Manually filtering out the resumes is not practically possible as it takes a lot of time and incurs huge costs that the hiring companies cannot afford to bear. In addition, this process of screening resumes is not fair as many suitable profiles don't get enough consideration which they deserve. This may result in missing out on the right candidates or selection of unsuitable applicants for the job. In this paper, we describe a solution that aims to solve these issues by automatically suggesting the most appropriate candidates according to the given job description. Our system uses Natural Language Processing to extract relevant information like skills, education, experience, etc. from the unstructured resumes and hence creates a summarized form of each application. With all the irrelevant information removed, the task of screening is simplified, and recruiters can better analyze each resume in less time. After this text-mining process is completed, the proposed solution employs a vectorization model and uses cosine similarity to match each resume with the job description. The calculated ranking scores can then be utilized to determine the best-fitting candidates for that particular job opening.

3.5 Automated Resume Screening Using Semantic Similarity-Based Sentence Embeddings:

<https://arxiv.org/pdf/2307.08624.pdf>

ABSTRACT: Many companies and organizations have started to use some form of AI-enabled automated tools to assist in their hiring process, e.g. screening resumes, interviewing candidates, performance evaluation. While those AI tools have greatly improved human resource operations efficiency and provided conveniences to job seekers as well, there are increasing concerns on the unfair treatment of candidates, caused by underlying bias in AI systems. Laws around equal opportunity and fairness, like GDPR, and CCPA, are introduced or under development, to

regulate AI. However, it is difficult to implement AI regulations in practice, as technologies are constantly advancing and the risk pertinent to their applications can fail to be recognized. This study examined deep learning methods, a recent technological breakthrough, with a focus on their application to automated resume screening. One impressive performance of deep learning methods is the representation of individual words as low-dimensional numerical vectors, called word embedding, which are learned from aggregated global word-word co-occurrence statistics from a corpus, like Wikipedia or Google News. The resulting word representations possess interesting linear substructures of the word vector space and have been widely used in downstream tasks, like resume screening. However, word embedding inherits and reinforces the stereotyping from the training corpus, as deep learning models essentially learn a probability distribution of words and their relations from historical data. Our study finds out that if we rely on such deep-learning-powered automated resume screening tools, it may lead to decisions favoring or disfavoring certain demographic groups and raise ethical, even legal, concerns. To address the issue, we developed bias mitigation methods. Extensive experiments on real candidate resumes are conducted to validate our study.

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

The current system for screening resumes employs a manual process in which recruiters or human resource managers evaluate job applications based on their qualifications, experience, and other factors.

Among the existing systems are:

Taleo: This system is a cloud-based recruitment tool that evaluates resumes and selects the best candidates for a given job using AI-powered algorithms. Using natural language processing and machine learning, it compares resumes and job descriptions based on similarities [10].

Job scan: is an online resume scanner that uses ATS (Applicant Tracking System) technology to evaluate resumes by specific job descriptions [5]. It examines the keywords, talents, and other relevant data to determine whether the job description and resume are compatible. Current automated resume screening systems evaluate job applications for relevance to a given job description using a variety of NLP approaches, such as entity identification, semantic search, and machine learning. The accuracy of these algorithms still needs to be improved, particularly when it comes to identifying the best candidates for a position.

4.1.1 DISADVANTAGES OF THE EXISTING SYSTEM:

1. **Insufficient customization:** Many current resume screening tools rely on pre-set criteria or algorithms that may not be the best fit for specific job roles or industries. Because of a high proportion of false positives and false negatives, qualified candidates may be passed over in favor of less qualified individuals.
2. **Narrow focus:** Certain resume screening tools may only consider a few factors, such as keywords or years of experience, leaving out critical information about a candidate's abilities or accomplishments.
3. **Language prejudice:** The lack of diversity in the candidate pool is caused by resume screening tools that are biased towards certain languages, keywords, or cultural norms [2].

4. **Poor parsing precision:** The accuracy of the NLP algorithms used to analyze resumes may be impacted by formatting issues or consistency issues, which could result in inaccurate information extraction.
5. **Without context:** Current resume screening methods may be unable to consider the context of a candidate's education, work experience, or talents, resulting in inaccurate assessments.

4.2 Proposed System:

We are extracting skills, qualifications, and personal information from resumes using Resume Parser and NLP API in this project. This is a very useful extraction for companies that are not supposed to manually scan every resume. Once a resume is uploaded, a score based on the applicant's skills and the required skills will be calculated; if the applicant's score is high, the company will shortlist them and schedule interviews.

4.2.1 Advantages of the proposed system:

1. **Improved precision:** NLP algorithms such as SBERT and cosine similarity excel at identifying resumes that are most relevant to a specific job description. These algorithms are designed to comprehend the context of the text and decipher the intended meanings of the words.
2. **Improved effectiveness:** NLP algorithms can evaluate hundreds or thousands of resumes in a matter of minutes, making them far faster than hand screening. Recruiters will save a lot of time and money as a result of this [3]. NLP algorithms such as SBERT and cosine similarity can be tailored to specific businesses, positions, or organization resulting in more accurate resume assessments.
3. **More accurate candidate matching:** The algorithms S-BERT and cosine similarity are created to match candidates with job descriptions based on the relevance and similarity of their abilities, experience, and qualifications.
4. **Language autonomy:** Employing managers will find it easier to evaluate resumes from candidates with different linguistic backgrounds thanks to NLP algorithms' ability to interpret resumes written in a range of languages.

5. **Working knowledge of unstructured data:** NLP algorithms can pull relevant data from unstructured data, like resumes, making it easier for recruiters to evaluate resumes that do not follow a standard pattern.

4.3 FUNCTIONAL REQUIREMENTS:

1. Admin
2. User

4.4 NON-FUNCTIONAL REQUIREMENTS:

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allow you to impose constraints or restrictions on the design of the system across the various agile backlogs. For example, the site should load in 3 seconds when the number of simultaneous users is > 10000. The description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

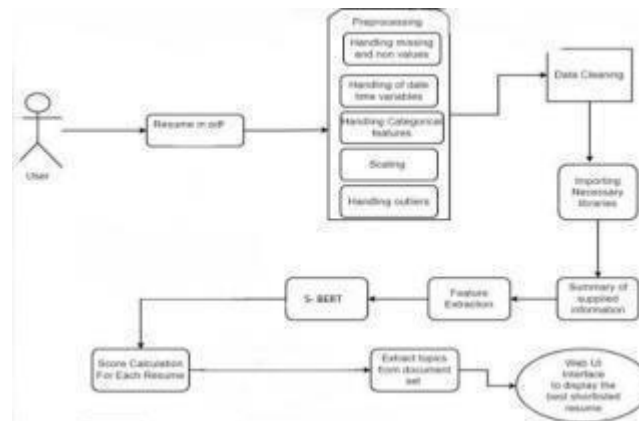
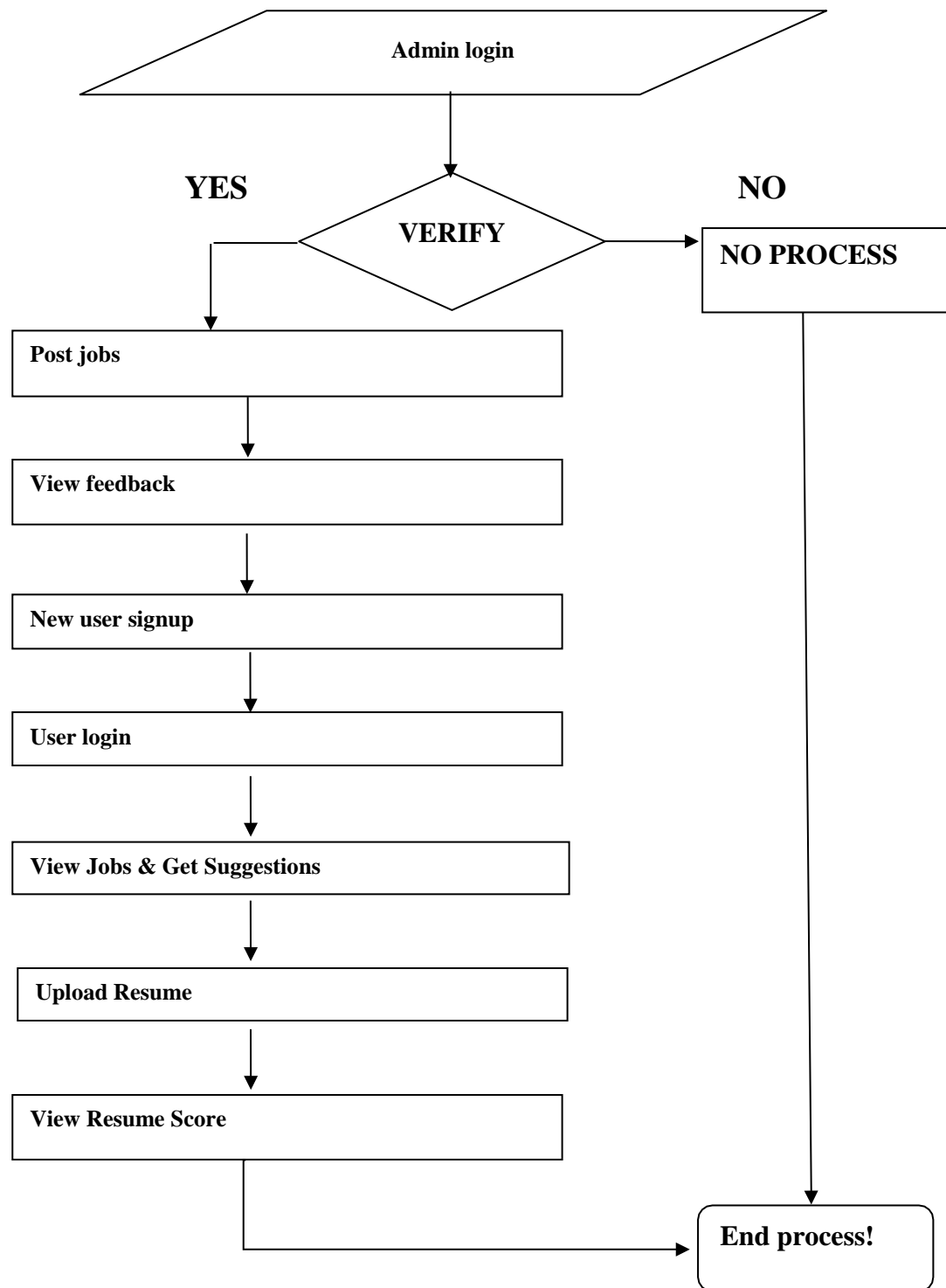


Fig.5.1.1 System architecture

DATA FLOW DIAGRAM:

1. The DFD is also called bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information that flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as a bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



5.2 UML DIAGRAMS:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

UML is a very important part of developing object-oriented software and the software development process. UML uses mostly graphical notations to express the design of software projects.

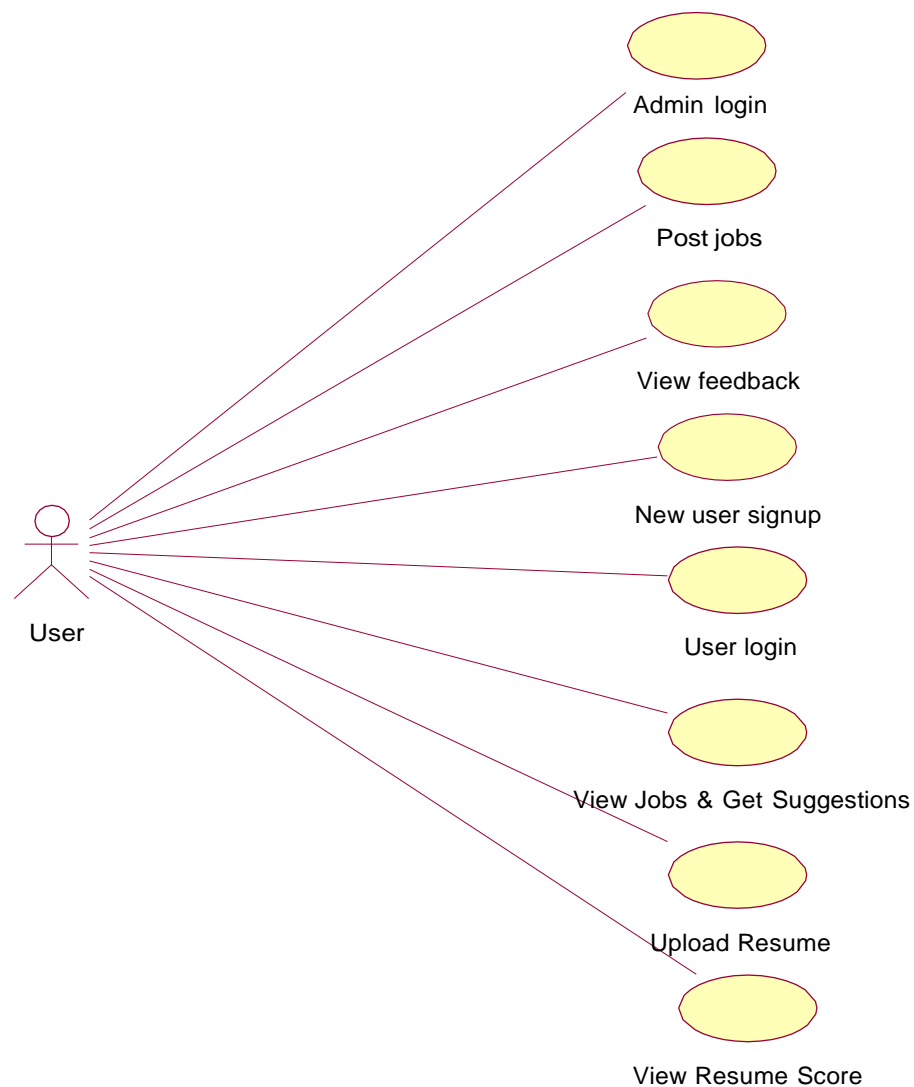
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users with a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
7. Integrate best practices.

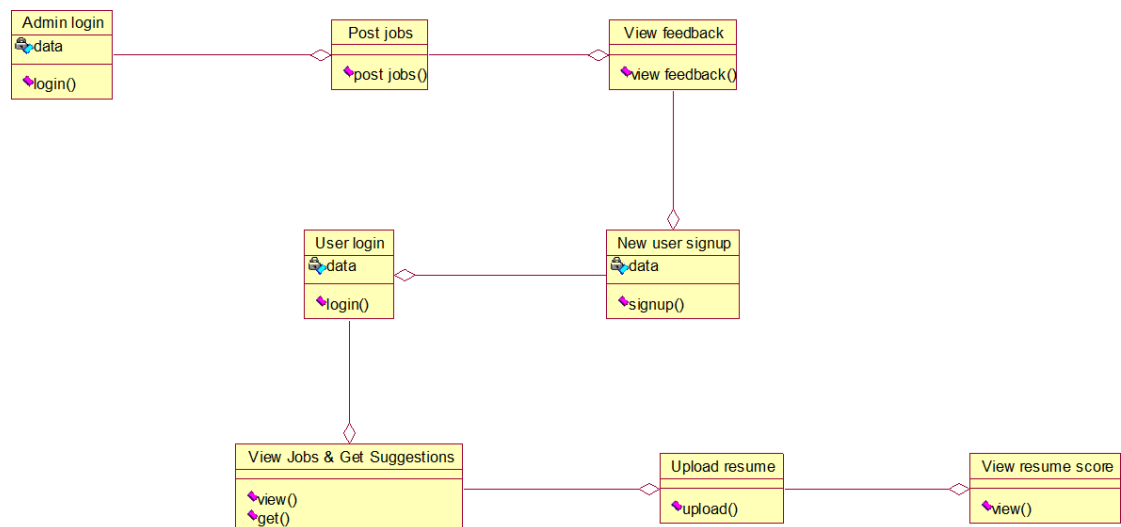
Use case diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. The roles of the actors in the system can be depicted.



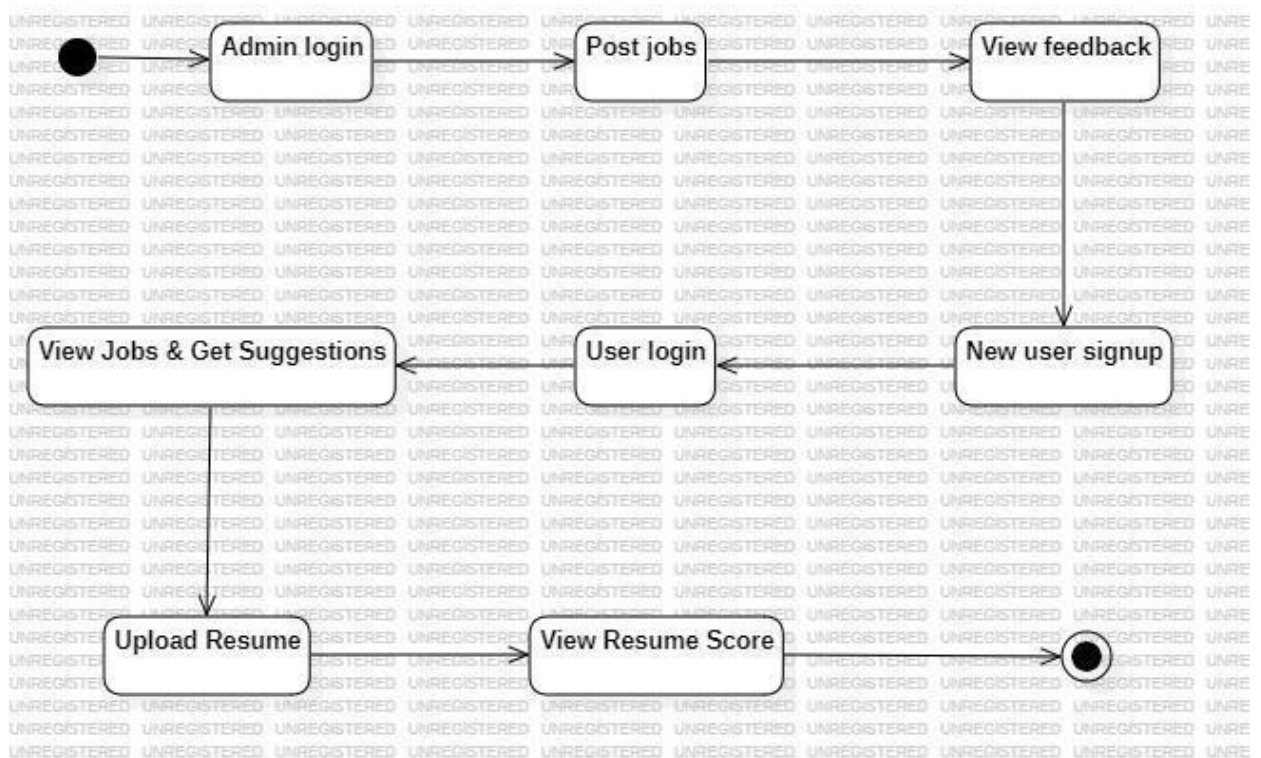
Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.



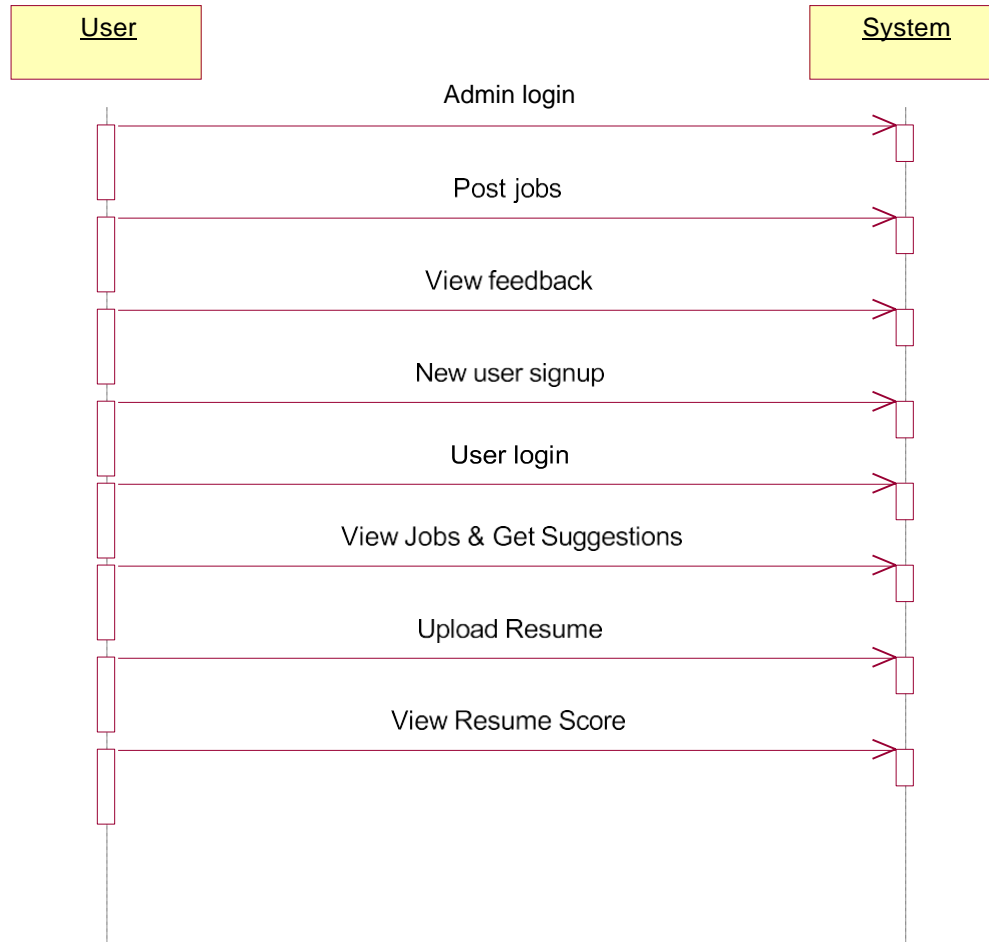
Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.



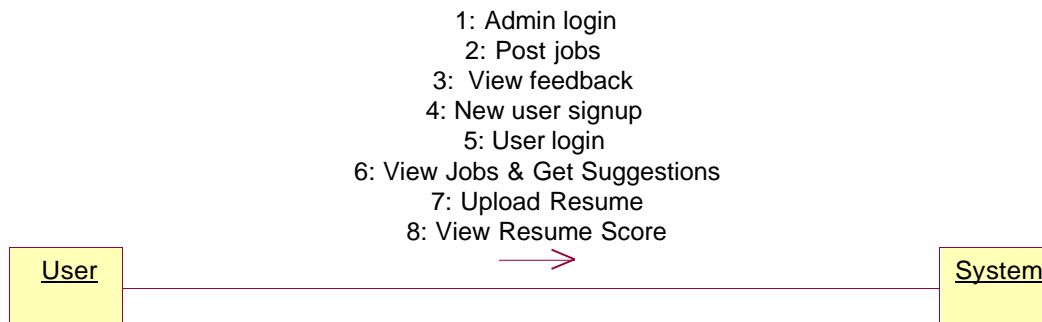
Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



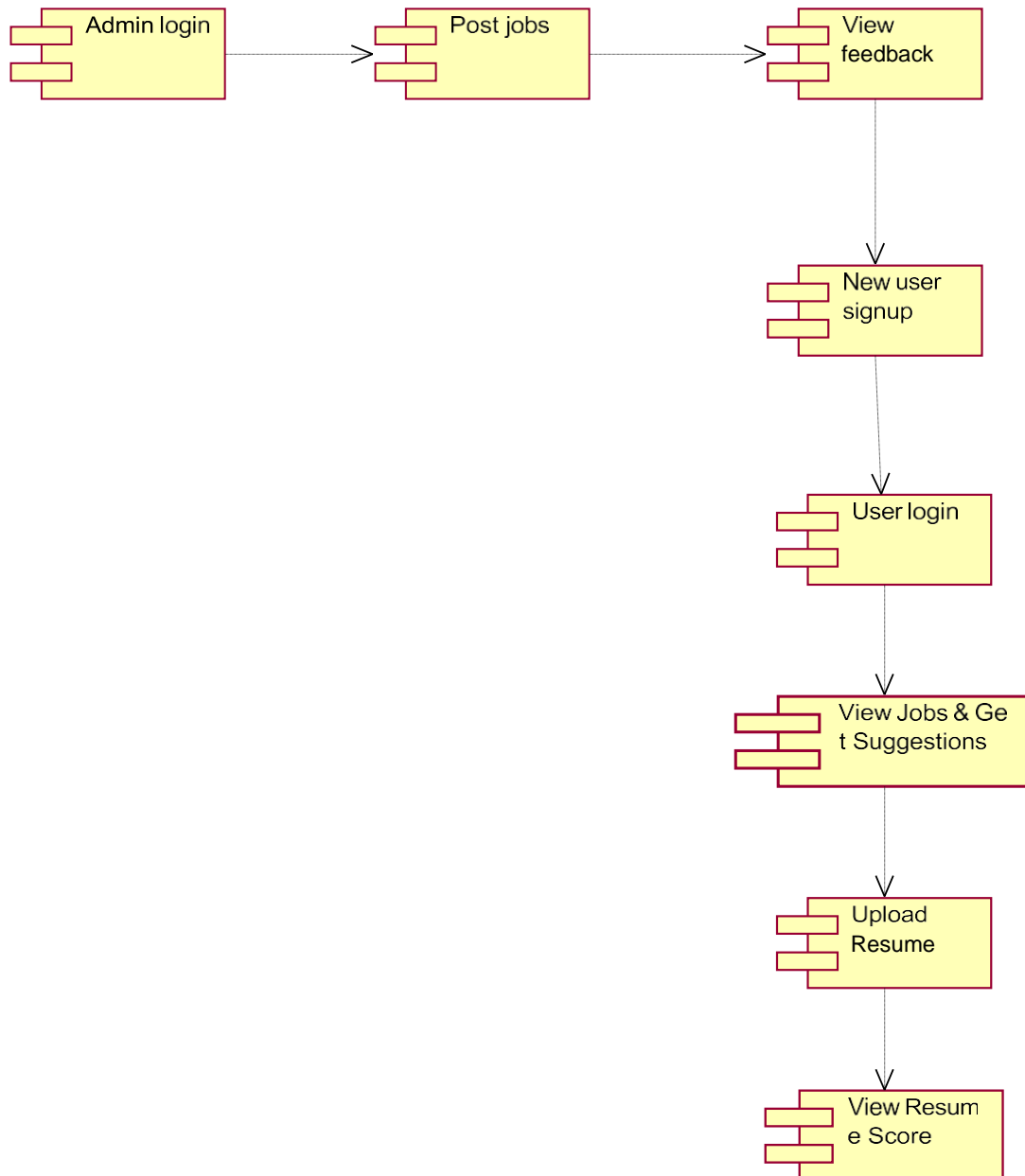
Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.



Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far the most useful when a system is built and ready to be deployed.



6. IMPLEMENTATION

In this project using Resume Parser and NLP API we are parsing resumes to extract details like skills, qualifications, and personal details and this extraction is very helpful for companies where they are not supposed to manually scan every resume. Once the resume is uploaded based on the required skills and applicant's skills score will be calculated if the score is high then the company will shortlist that applicant and call for interviews.

To run the project, install Python 3.7.0 and then install the below

packages
Pip install Django==2.1.7

Pip install numpy==1.19.2
Pip

install pyresparser==1.0.6
Pip

install PyMySQL==0.9.3
Pip

install matplotlib==3.1.1

6.2 SAMPLE CODE:

```
from flask import Flask,request,jsonify
```

```
from flask_restful import reqparse,Resource,Api
```

```
import numpy as np
```

```
import json
```

```
import time
```

```
import pandas as pd
```

```
import requests
```

```
from clustering.clustering import getVector,predict_classification
```

```
from resumeParser import parseResume
```

```

from flask_cors import CORS

# import hindi modules

#creating a flask app

app=Flask(__name__)

CORS(app)

api=Api(app)

uri = 'http://localhost:3000/getAll'

STOPWORDS = ['Engineering', 'English', 'Programming', 'Computer science', 'Research','Technical',
'System', "Java"]

class JsonAccumulator(Resource):

def post(self):

    print('post function called')

    json_data = request.get_json(force=True)

        # print(json_data)

    url = json_data['url'] result =

    parseResume(url)

    return result

class JsonGetResumeGivenConstraints(Resource):def

    get(self):

        return "hello"

```



```

def post(self):

    json_data = request.get_json(force=True)

    requirement = json_data['requirement']

    nresume= json_data['noOfResume']

    nresume = int(nresume)

    resume = requests.get(uri)

    resume = resume.json()

    mydict = { }

    keys = list(list(resume[0].keys()))

    for key in keys:

        mydict[key] = []

    for res in list(resume):for

        key in keys:

            mydict[key].append(res[key])

    df = pd.DataFrame(mydict)

    if nresume > df.shape[0]:

        nresume = df.shape[0] - 1

    ids = df['_id'].values.tolist()

    rawResume = df['resume'].values.tolist()

```

```

vec = getVector([requirement] + rawResume, ids)

result = predict_classification(vec[1:], vec[0], nresume)

finalResult = { }

df['skills'] = df['skills'].apply(lambda x: x.split('$'))

skillsList = df['skills'].values.tolist()

skillDict = createDict(skillsList)

finalResult['skills'] = skillDict

df['total_experience'] = df['total_experience'].apply(lambda x: int(float(x)))

expDict = createDict(df['total_experience'].values.tolist(), False)

finalResult['Experience'] = expDict

finalResume = []

for item in result:

    temp = df[df['_id'] == item[0] ].to_dict()

    temp = cleanDict(temp)

    temp['score'] = item[1]

    finalResume.append(temp)

finalResult['result'] = finalResume

return finalResult

def cleanDict(mydict):

```

```

cleanDict = { }

keys = list(mydict.keys())
for
key in keys:

    cleanDict[key] = list(mydict[key].values())[0]

return cleanDict

def createDict(array, flag=True):
    if
    flag == True:

        array = [item for sublist in array for item in sublist]

        array = [w for w in array if not w in STOPWORDS]

    mydict = { }

    for item in array:

        if item in list(mydict.keys()):

            mydict[item] = mydict[item] + 1

        else:

            mydict[item] = 1

    newDict = sorted(mydict.items(), key=lambda x: x[1], reverse=True)

    newDict = list(newDict)

    if flag == True:

        newDict = newDict[:5]

    result = []

```

```

for item in newDict:

    temp = {

        "key": item[0],

        "value": item[1]

    }

    result.append(temp)

return result

api.add_resource(JsonAccumulator,'/prediction') api.add_resource(JsonGetResumeGivenConstraints,"/")

if __name__ == '__main__':

    app.run(debug=True)

from pyresparser import ResumeParser

import io

import os

import re

import nltk

from pdfminer.converter import TextConverter

from pdfminer.pdfinterp import PDFPageInterpreter from

pdfminer.pdfinterp import PDFResourceManagerfrom

pdfminer.layout import LAParams

from pdfminer.pdfpage import PDFPage

```

```

from pdfminer.pdfparser import PDFSyntaxError

import wget

import pdfx

url = 'https://viditkhemka00.s3.amazonaws.com/Vidit.pdf'

def extractLink(path): pdf
    = pdfx.PDFx(path)

    urls = pdf.get_references_as_dict()if

    'url' in list(urls.keys()):

        return urls[url]

    return []

def serializeJSON(json_data):

    result = { }

    for key in list(json_data.keys()):

        if type(json_data[key]) == type([]):

            result[key] = "$".join(json_data[key])

        else:

            result[key] = str(json_data[key])

    return result

def parseResume(url): fileName

```

```

    = downloadFile(url)data =

    extractInfo(fileName)

    data['resume'] = pdf_to_text(fileName)

    data = serializeJSON(data)

    os.remove(fileName)

    return data

def downloadFile(url):

    filename = url.split('/')[-1]

    wget.download(url, filename)

    return filename

def extractInfo(path):

    data = ResumeParser(path).get_extracted_data()

    return data

def pdf_to_text(path):

    mygen = extract_text_from_pdf(path)

    resume = []

    for item in mygen:

        resume.append(item)

    return ''.join(resume)

def extract_text_from_pdf(pdf_path):

    if not isinstance(pdf_path, io.BytesIO):

```

```

# extract text from local pdf file

with open(pdf_path, 'rb') as fh:

    try:

        for page in PDFPage.get_pages(

            fh,

            caching=True,

            check_extractable=True

        ):
            resource_manager = PDFResourceManager()

            fake_file_handle = io.StringIO()

            converter = TextConverter(

                resource_manager,

                fake_file_handle,

                laparams=LAParams()

            )

            page_interpreter = PDFPageInterpreter(

                resource_manager,

                converter

            )

            page_interpreter.process_page(page)

            text = fake_file_handle.getvalue()

```

```

        yield text

    # close open handles

    converter.close()

    fake_file_handle.close()

except PDFSyntaxError:

    return

else:

    # extract text from remote pdf file

    try:

        for page in PDFPage.get_pages(

            pdf_path,

            caching=True, check_extractable=True

        ):
            resource_manager = PDFResourceManager()

            fake_file_handle = io.StringIO()

            converter = TextConverter(
                resource_manager,

                fake_file_handle,

                laparams=LAParams()

            )

            page_interpreter = PDFPageInterpreter(

```



```
        resource_manager,  
  
        converter  
  
    )  
  
    page_interpreter.process_page(page)  
  
    text = fake_file_handle.getvalue()  
  
    yield text  
  
    # close open handles  
  
    converter.close()  
  
    fake_file_handle.close()  
  
except PDFSyntaxError:  
    return
```

7. SOFTWARE ENVIRONMENT

PYTHON LANGUAGE:

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components. Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source-level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language. For example, `x = 10` Here, x can be anything such as String, int, etc.

Features in Python:

There are many features in Python, some of which are discussed below as follows:

1. Free and Open Source

Python language is freely available at the official website, and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open source, this means that source code is also available to the public. So you can download it, use it as well as share it.

2. Easy to code

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python's syntax is straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

4. Object-oriented Language

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

5. GUI Programming Support

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

6. High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

7. Extensible feature

Python is an Extensible language. We can write some Python code into C or C++ language and also, we can compile that code in C/C++ language.

8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.

9. Python is a Portable language.

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

10. Python is an integrated language.

Python is also an integrated language because we can easily integrate Python with other languages like C, C++, etc.

11. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code which makes it easier to debug our code. The source code of Python is converted into an immediate form called byte code.

12. Large Standard Library

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

13. Dynamically Typed Language

Python is a dynamically typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

14. Frontend and backend development:

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do front-end development work in Python like JavaScript. Backend is the strong forte of Python. It's extensively used for this work because of its frameworks like Django and Flask.

15. Allocating Memory Dynamically:

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write `int y = 18` if the integer value 15 is set to y. You may just type `y=18`.

LIBRARIES/PACKAGES:

Tensor flow

Tensor Flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high- performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code.

- Useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to integrate with a wide variety of databases seamlessly and speedily.

Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tools using its powerful data structures. Python was majorly used for data munging and preparation. It made very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few line of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc., via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn:

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

8. SYSTEM TESTING

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

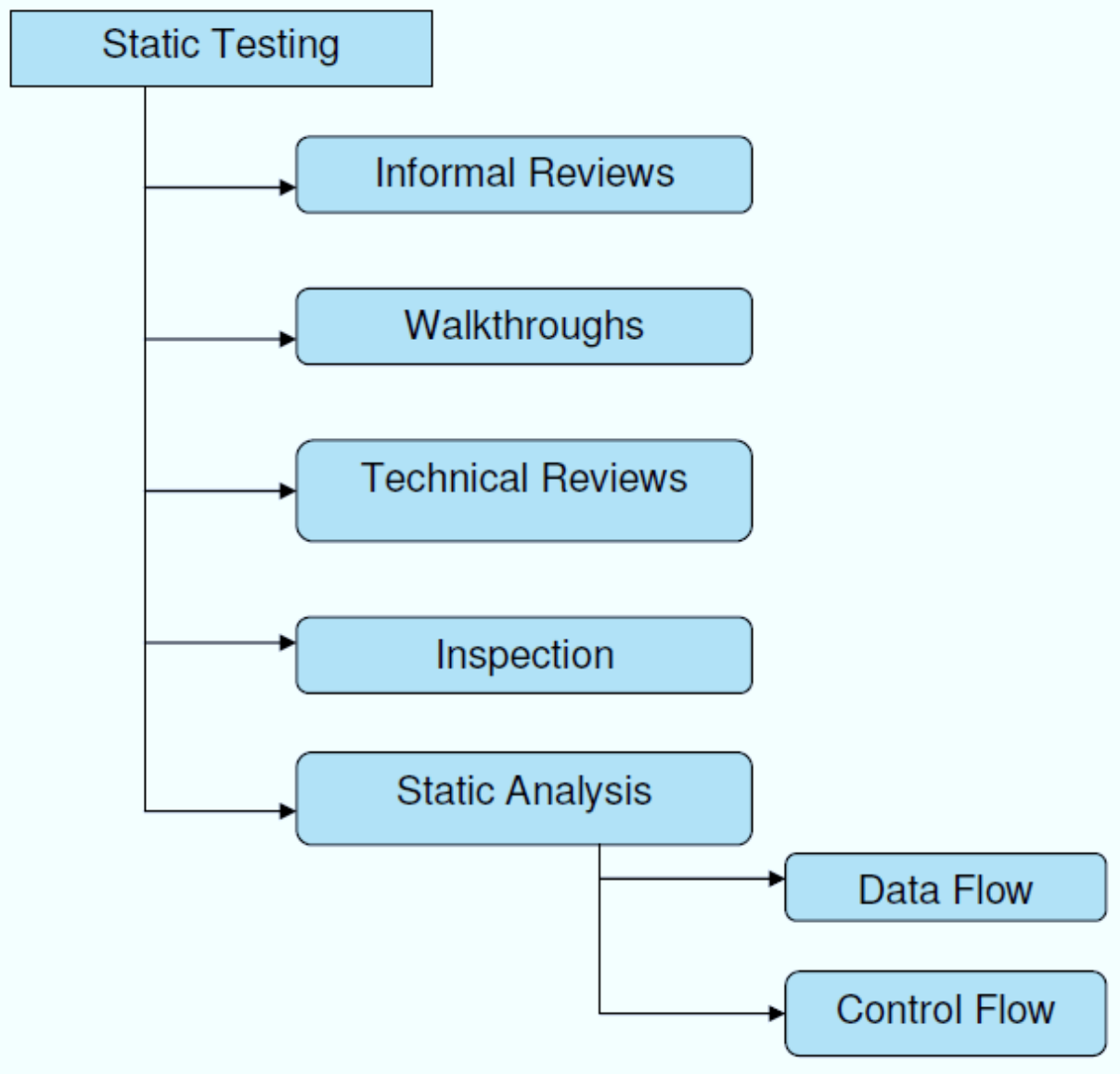
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects and establishes what kinds and amount of defects are tolerable.

8.1 Software Testing Strategies:

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

Static Testing:

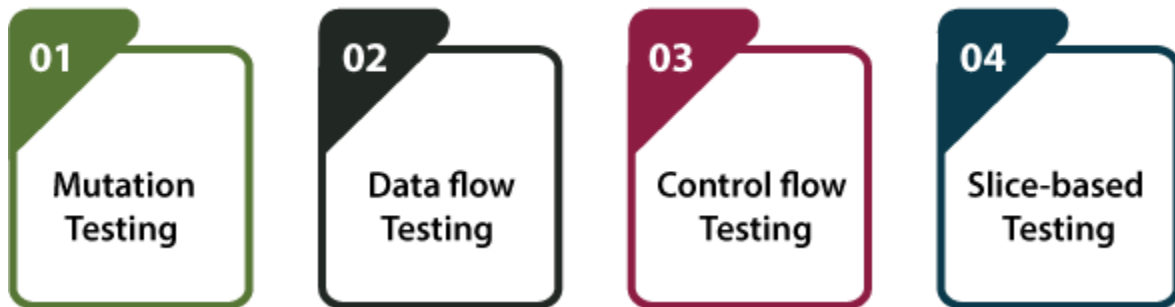
The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



Structural Testing:

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).

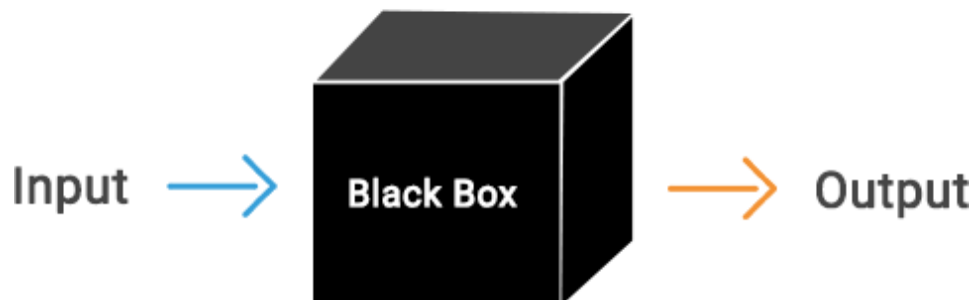
Types of Structural testing



Behavioral Testing:

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.

Black Box Testing



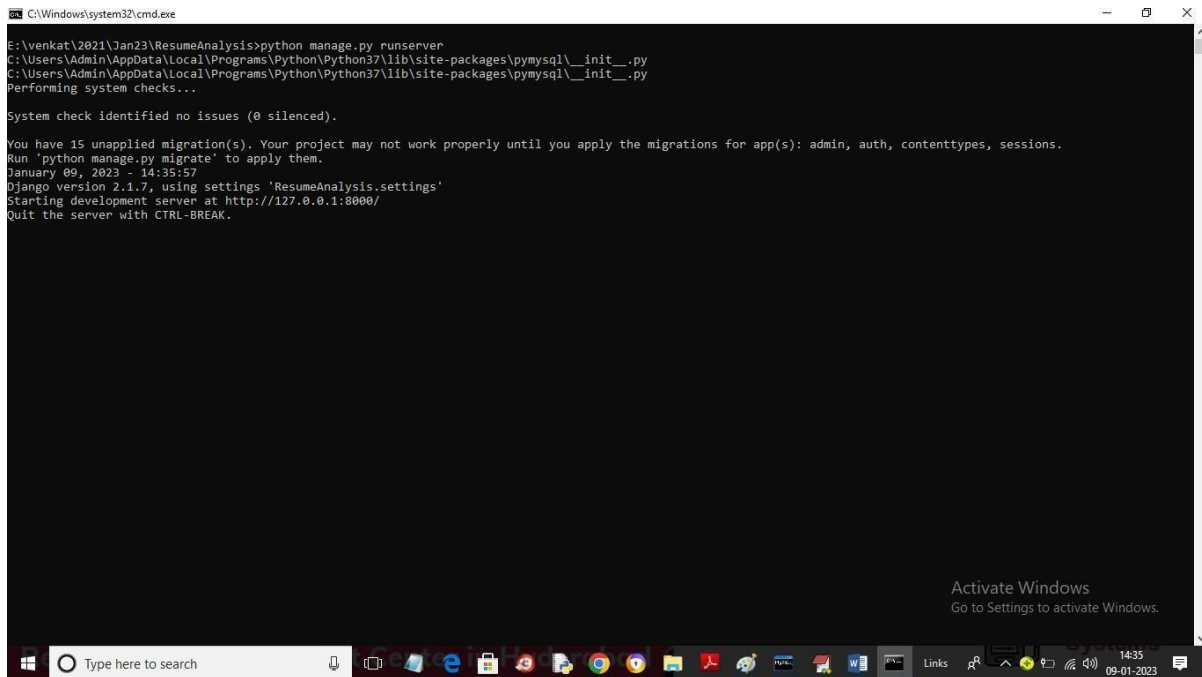
8.2 TEST CASES:

S.NO	INPUT	If available	If not available
1	Admin login	Login into the application	There is no process
2	Post jobs	admin will enter job details and then select require skills	There is no process
3	View feedback	admin can viewFeedback	There is no process
4	signup and login new user	User can register and login into the application	There is no process
5	View Jobs & GetSuggestions	view all jobs and suggestion	There is no process
6	Upload Resume	upload resume and get score	There is no process
7	View Resume Score	view all applicant resumes with score	There is no process

9. SCREENS

SCREENSHOTS:

Now double click on 'run.bat' file to start python DJANGO server and then open browser and enter URL as <http://127.0.0.1:8000/index.html> and press enter key to get below page:



```

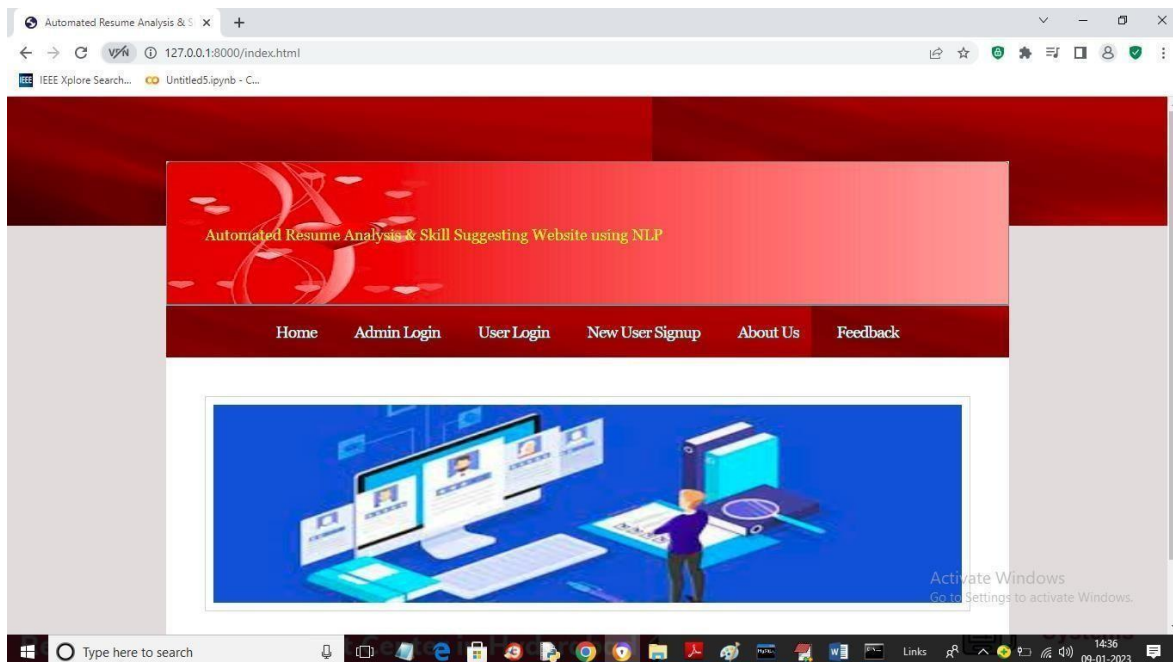
C:\Windows\system32\cmd.exe

E:\venkat\2021\Jan23\ResumeAnalysis>python manage.py runserver
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\mysql\__init__.py
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\mysql\__init__.py
Performing system checks...

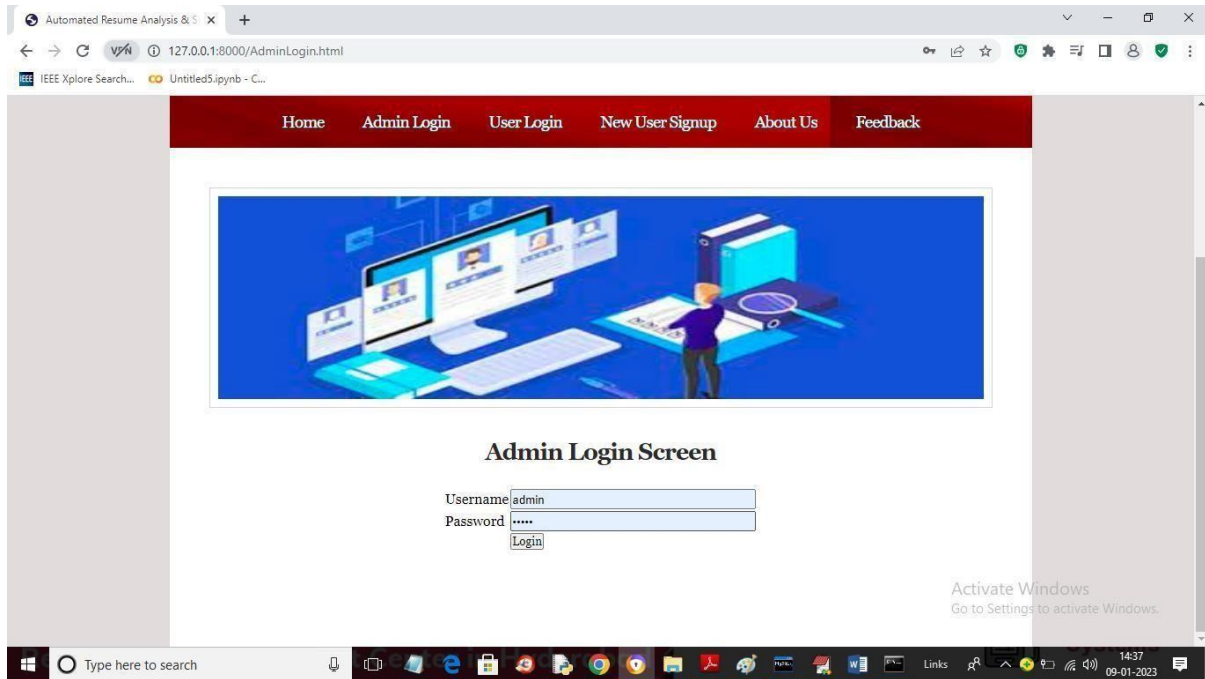
System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
January 09, 2023 - 14:35:57
Django version 2.1.7, using settings 'ResumeAnalysis.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
  
```

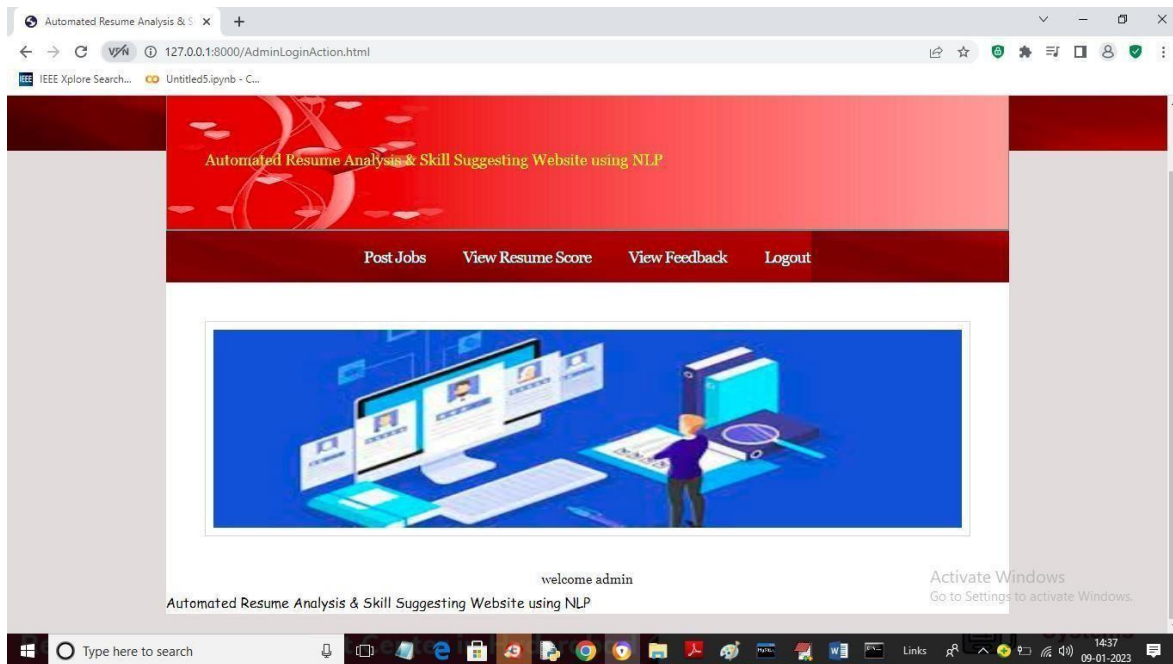
In above screen Django server started and now open browser and enter URL to get below page



In above screen click on 'Admin Login' link to login as admin and then post jobs



In above screen admin is login and after login will get below page



In above screen click on 'Post Jobs' link to post jobs

Post New Job Screen

Job Name: C Programmer

Job Details: Must be proficient in system C programming and web

Company Name: Infosys

Salary: 100000

Skills: Java, Database, Html, C++, Submit

Activate Windows
Go to Settings to activate Windows.

In above screen admin will enter job details and then select require skills and press button to get below output

Post New Job Screen

Job details posted with ID : 1

Job Name:

Job Details:

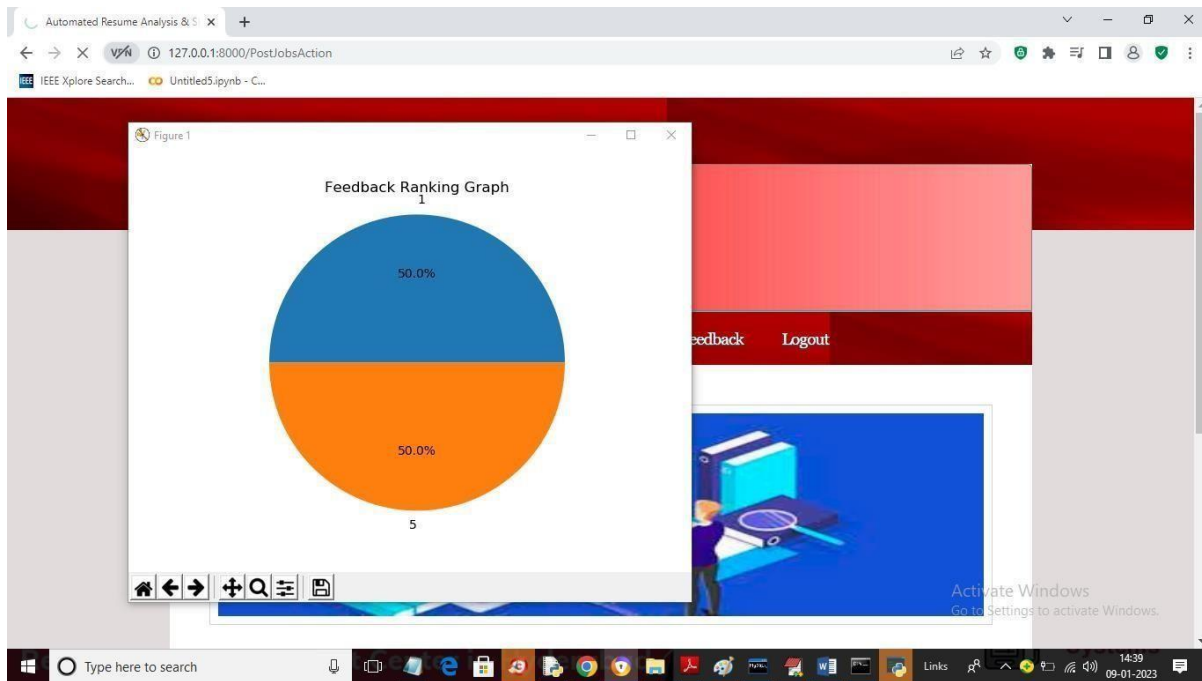
Company Name:

Salary:

Skills: Access, Oracle, Data analysis, Technical, Submit

Activate Windows
Go to Settings to activate Windows.

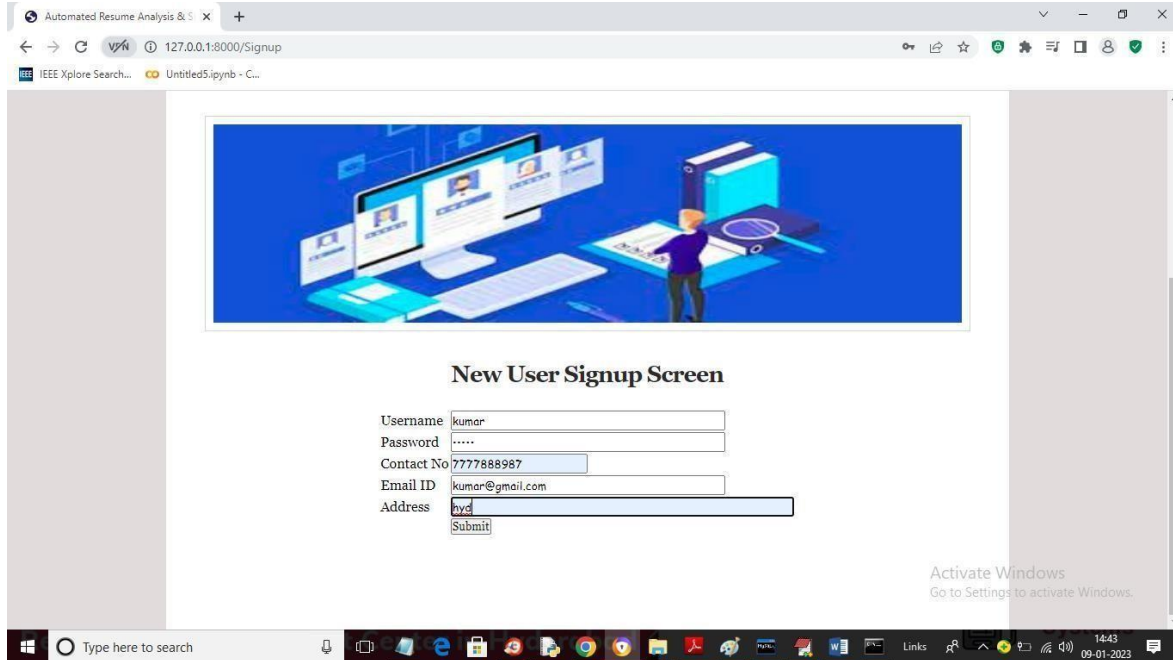
In above screen job details added with JOB ID 1 and now click on ‘View Feedback’ link to get below screen



In above screen admin can view Feedback as pie chart and close above graph to get below page

Feedback	Feedback Date	Feedback Rank
good site to get resume score	2023-01-09	5
worst site	2023-01-09	1

In above screen admin will view all feedback and now logout and signup and login new user



Automated Resume Analysis & S...

127.0.0.1:8000/Signup

IEEE Xplore Search... Untitled5.ipynb - C...

New User Signup Screen

Username: kumar

Password:

Contact No: 7777888987

Email ID: kumar@gmail.com

Address: hyd

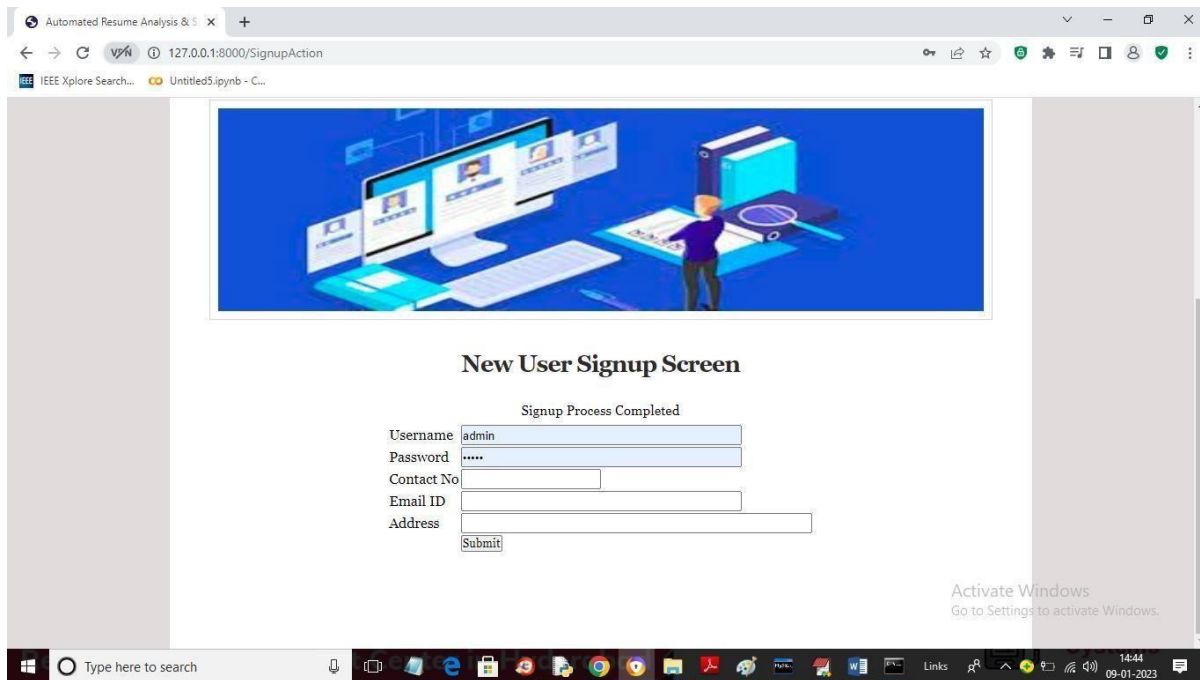
Submit

Activate Windows
Go to Settings to activate Windows.

Type here to search

14:43
09-01-2023

In above screen user is signing up and then press button to get below page:



Automated Resume Analysis & S...

127.0.0.1:8000/SignupAction

IEEE Xplore Search... Untitled5.ipynb - C...

New User Signup Screen

Signup Process Completed

Username: admin

Password:

Contact No:

Email ID:

Address:

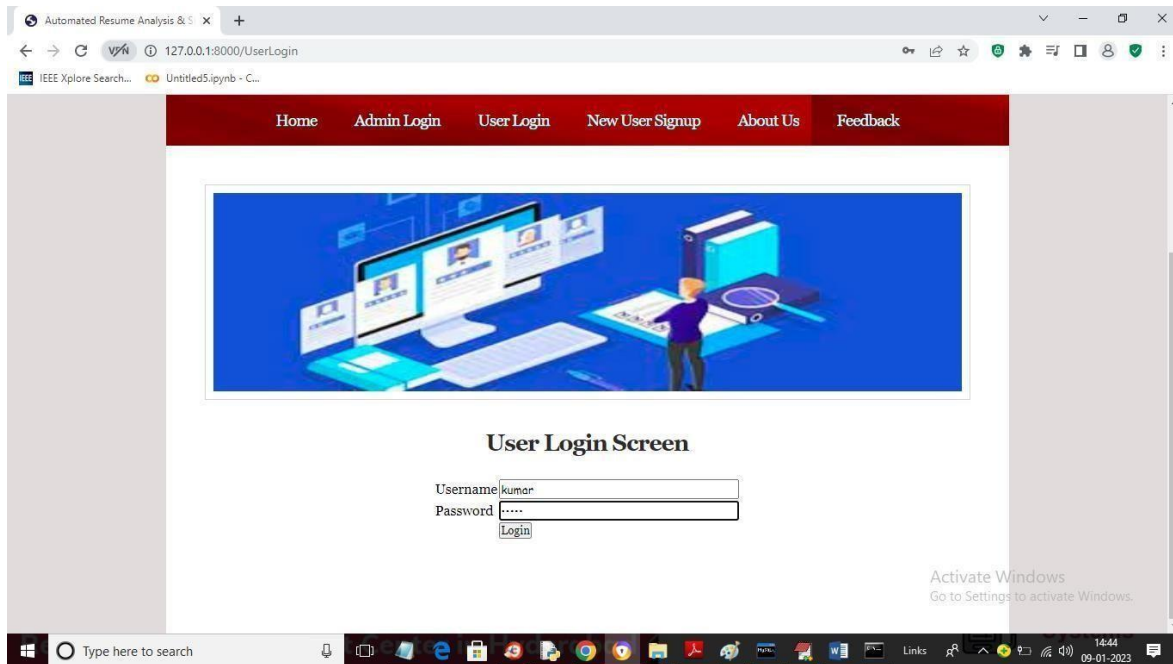
Submit

Activate Windows
Go to Settings to activate Windows.

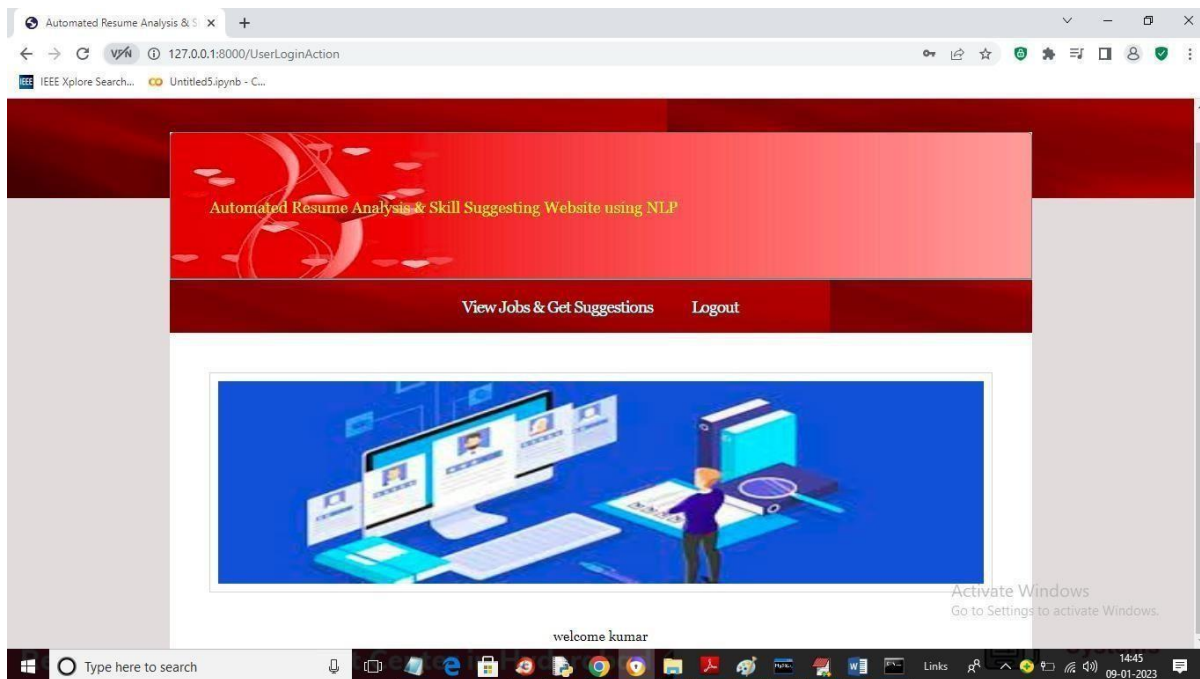
Type here to search

14:44
09-01-2023

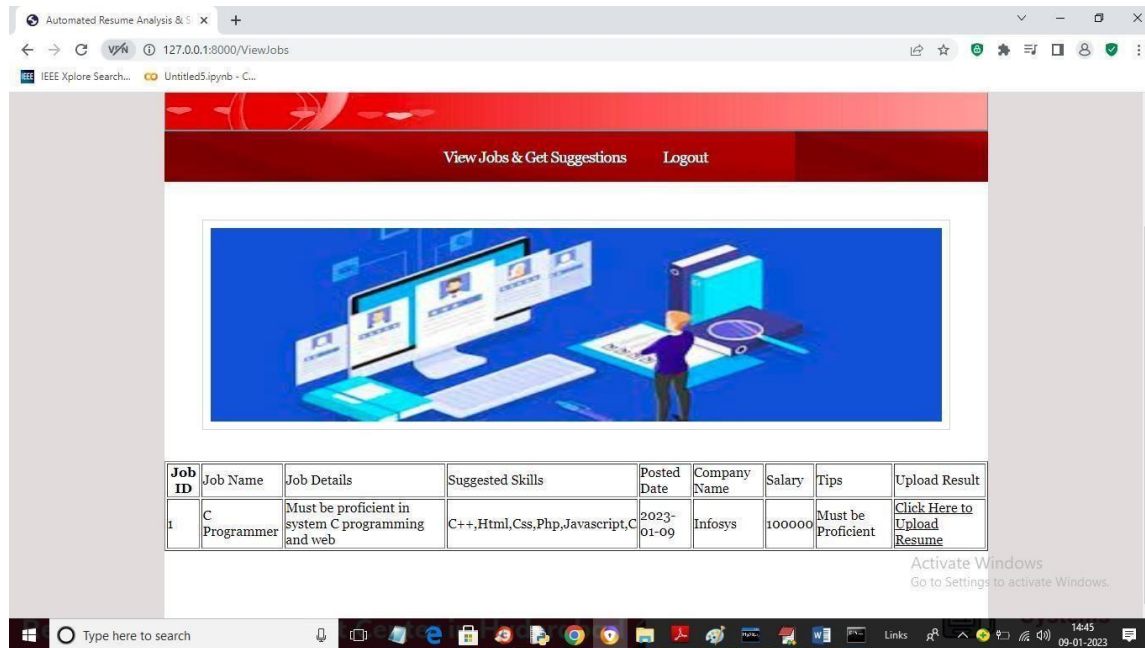
In the above screen signup process completed and now click on 'User Login' link to get Below login page.



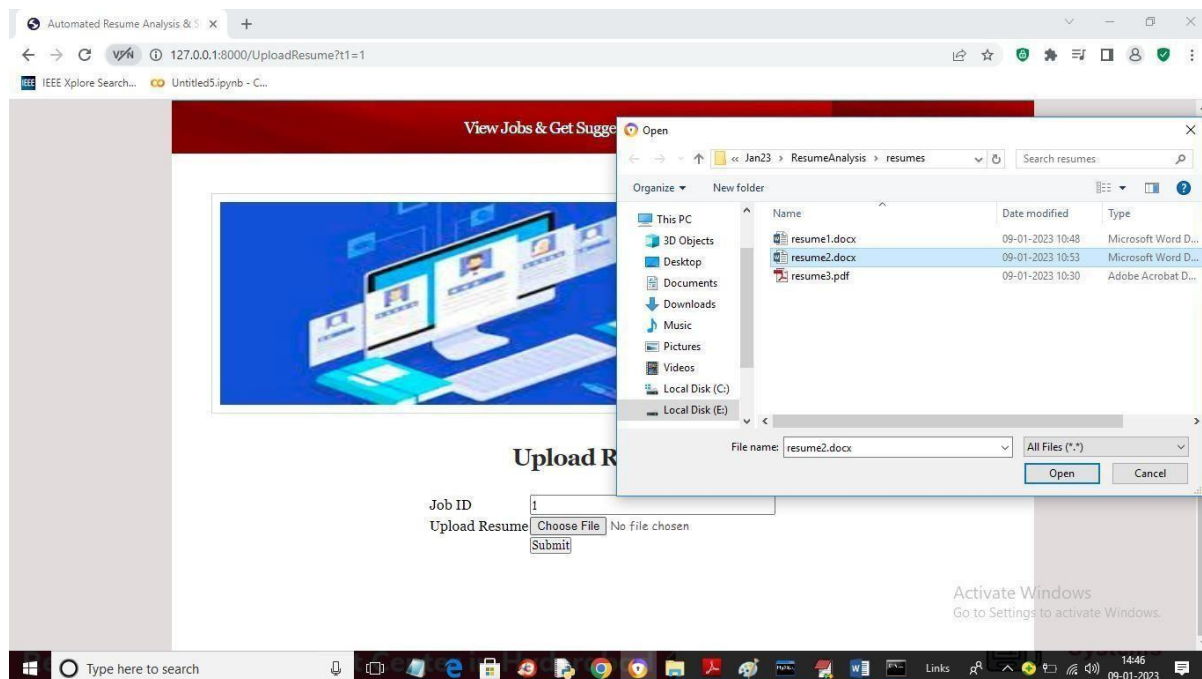
In above screen user is login and after login will get below page



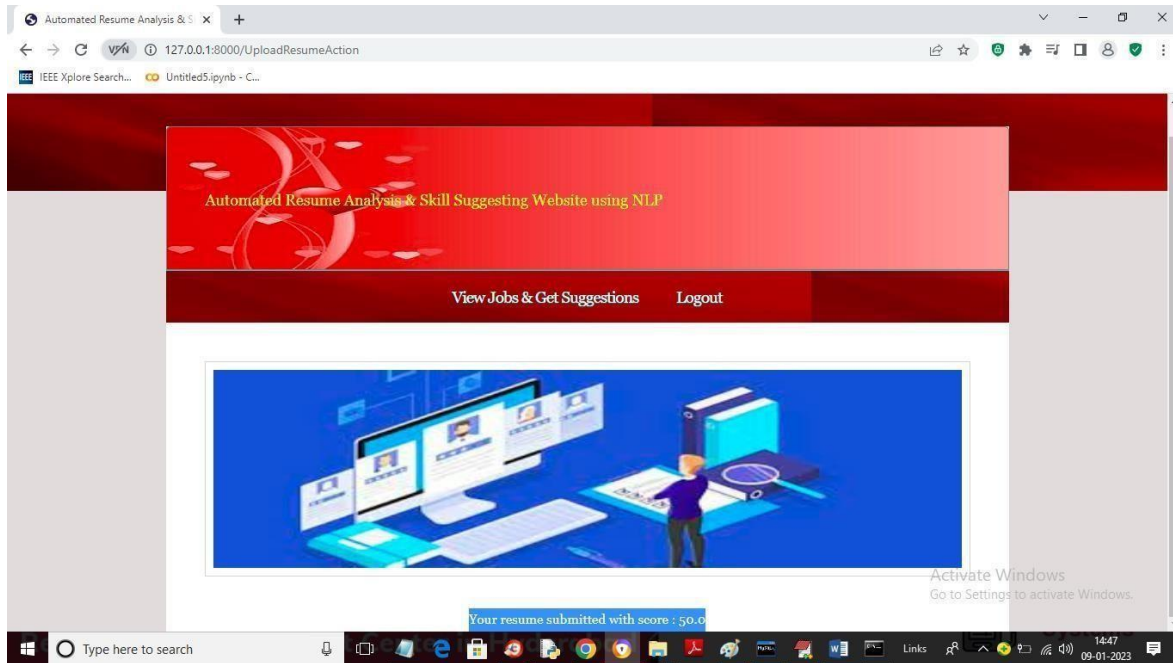
In above screen click on 'View Jobs & Get Suggestions' link to view all jobs and suggestion



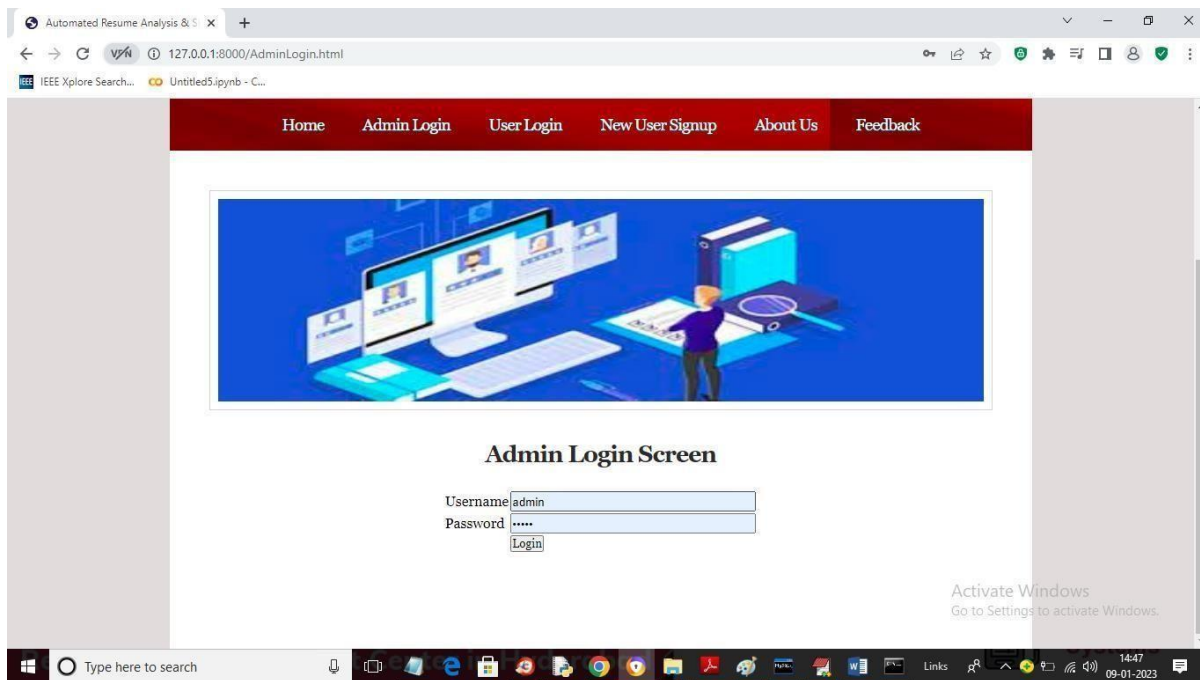
In above screen user can view available jobs and suggested skills and if job is suitable for user then he can click on 'Click Here to Upload Resume' link to upload resume and get score



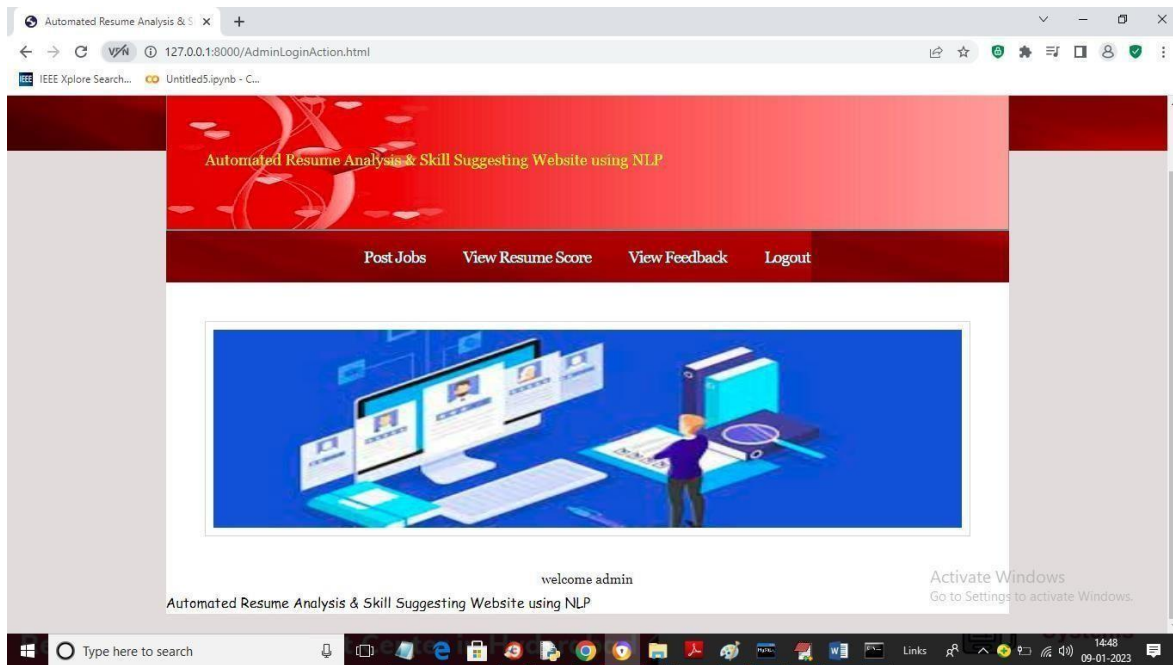
In above screen user is selecting and uploading resume and press 'Open' and 'Submit' button to get below score value



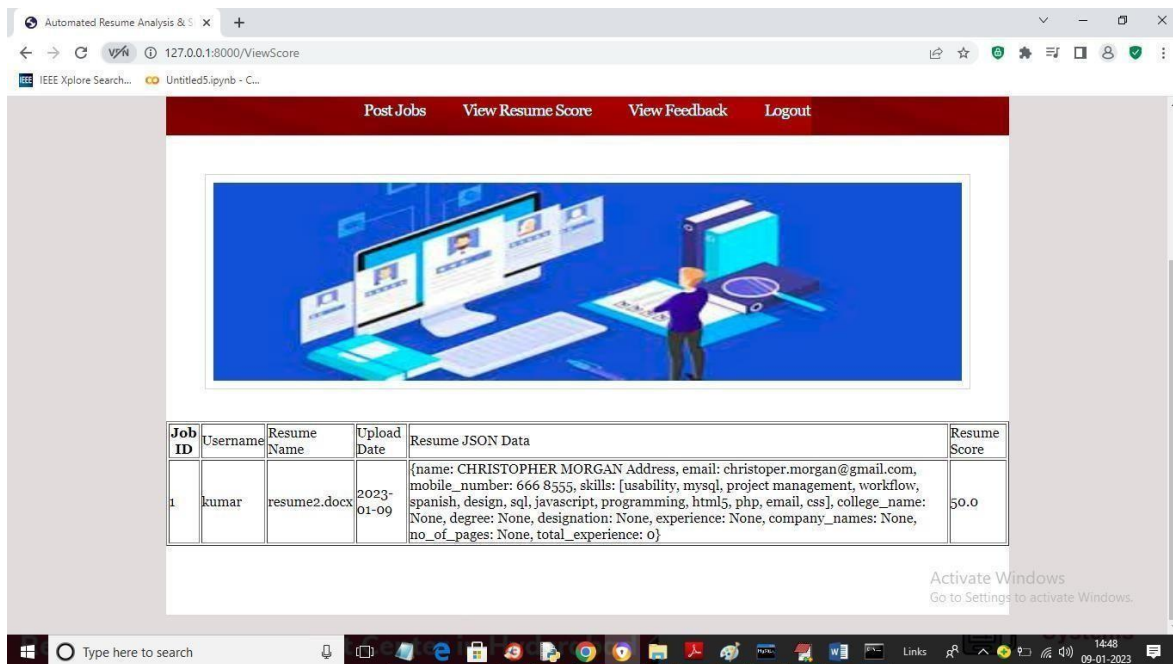
In above screen in blue colour text we can see user resume got 50% score and now login as admin to view all resume details



In above screen admin is login to get below page



In above screen admin can click on 'View Resume Score' to view all applicants resumes with score



In the above screen admin can view resume file name which he can read from server, and he can view all resume extracted details in JSON format and in last column he can see the score values.

Similarly, you can upload and extract details and score from resume and supported formats are only PDF and DOCX files.

CONCLUSION

By drawing this conclusion, we'll say that applying NLP algorithms for resume screening—like SBERT and cosine similarity—offers several benefits over more traditional methods. These algorithms are exceedingly precise, efficient, and adaptive, and they can handle unstructured data, such as resumes written in many languages. They can also minimize prejudice among people and enhance candidate matching, improving recruiting processes. It is critical to remember that these algorithms have limitations and are not optimal in all circumstances [11]. So, it is crucial to use these algorithms as a part of a larger hiring strategy that also includes human judgement and arbitrary criteria. The use of NLP algorithms in recruiting, such as SBERT and cosine similarity, is a promising development that has the potential to fundamentally alter how businesses screen and select job candidates.

Future versions of this application will include more types of resumes or templates, and it will accept input in any extended format, such as .pdf or .docx, so that the object detection range can be expanded. For optimal use, this application can be combined with eyewear [12]. Additionally, we work to manage job descriptions that require more complex evaluation criteria, such as soft skills or qualifications relevant to a certain industry. More complex evaluation criteria may be added in the future to increase the accuracy of candidate matching.

BIBLIOGRAPHY

REFERENCES:

- [1] Singh, A. K., & Shukla, P. (2020). "Automated resume screening and evaluation using machine learning techniques". *Journal of Intelligent & Fuzzy Systems*, 39(4),5947-5960.
- [2] Oh, J., & Lee, S. (2019). "A study on the extraction of competencies from job postings and their correlation with resumes using natural language processing". *Expert Systems with Applications*, 115, 475-486.
- [3] Xu, C., Lu, J., Liu, J., & Wei, X. (2021). "Resume screening using deep learning and natural language processing". *Knowledge-Based Systems*, 215, 106864.
- [4] Bhowmik, R., Garg, N., & Gupta, A. (2021). "Resume Screening Using Semantic Similarity and Clustering Algorithms". In *Proceedings of the 2021 3rd International Conference on Communication, Devices and Computing*.
- [5] Elakkiya, R., & Muthurajkumar, S. (2021). "Automated Resume Screening System using Semantic Similarity". In *2021 International Conference on Computing, Electronics & Communications Engineering (ICCECE)*.
- [6] Garg, N., Bhowmik, R., & Gupta, A. (2021). "Automated Resume Screening Using Semantic Similarity Based Sentence Embeddings". In *2021 International Conference on Smart Electronics and Communication (ICOSEC)*.
- [7] Huang, S., Li, W., Wang, L., & Huang, H. (2021). "Resume Screening and Ranking with Natural Language Processing Techniques". *Applied Sciences*, 11(5), 2095.
- [8] Kang, Y., & Lee, J. (2020). "Resume Analysis for Job Matchmaking Using Word Embedding and Ranking Algorithm". In *Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication*.
- [9] Li, X., & Shen, X. (2021). "Resume Ranking and Classification Based on SBERT". In *2021 International Conference on Computer, Information and Telecommunication Systems (CITS)*.

- [10] Liu, J., Zhang, R., Yang, W., & Guan, R. (2021). "A Semantic Similarity-Based Resume Screening System". *Journal of Intelligent & Fuzzy Systems*, 40(1), 787- 797.
- [11] Ma, Z., Wang, Y., & Zhao, Y. (2021). "Automated Resume Screening with Semantic Similarity and Gradient Boosting". In *Proceedings of the 2021 3rd International Conference on Cybernetics, Robotics and Control*.
- [12] Mandviwalla, M., & Kappelman, L. A. (2021). "Automated Resume Screening Using Semantic Similarity and Machine Learning". *Journal of Information Systems Education*, 32(1).
- [13] Natarajan, R., & Rajaraman, V. (2021). "Resume Analysis and Matching using NLP Techniques". In *2021 International Conference on Smart Intelligent Computing and Applications (ICSICA)*.
- [14] Wang, X., Shen, Y., Huang, X., & Zhang, Y. (2020). "An intelligent resume screening system based on NLP and machine learning". *Future Generation Computer Systems*, 105, 789-799

