

# Summer Project On Resume Extractor

By

**Jasmit Gharat (2021510017)**

Under the guidance of

**Prof.**

**Nikhita Mangaonkar**



Department of Master Of Computer Application  
Sardar Patel Institute of Technology  
Autonomous Institute Affiliated to Mumbai University  
2022-23

## **CERTIFICATE OF APPROVAL**

This is to certify that the following students

**Jasmit Gharat (2021510017)**

Have satisfactorily carried out work on the project  
entitled

**“Resume Extractor”**

Towards the fulfilment of project, as laid down  
by

Sardar Patel Institute of Technology  
during year  
2022-23.

Project Guide:  
Nikhita Mangaonkar

## PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

**Jasmit Gharat (2021510017)**

Have successfully completed the Project report on

**“Resume Extractor”,**

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,  
ANDHERI (W), MUMBAI

INTERNAL EXAMINER

EXTERNAL EXAMINER

Dr. Pooja Raundale  
HEAD OF DEPARTMENT

PRINCIPAL

# Contents

<b>Abstract</b>	<b>i</b>
<b>Objectives</b>	<b>i</b>
<b>List Of Figures</b>	<b>ii</b>
<b>List Of Tables</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Definition . . . . .	1
1.2 Objectives and Scope . . . . .	1
1.2.1 Objectives . . . . .	1
1.2.2 Scope . . . . .	1
1.3 Existing System . . . . .	2
1.4 Proposed System . . . . .	2
1.5 System Requirements . . . . .	4
1.6 Technology Used . . . . .	4
<b>2 Software Requirement Specification and Design</b>	<b>6</b>
2.1 Introduction . . . . .	6
2.1.1 Product Functions . . . . .	6
2.1.2 User Characteristics . . . . .	7
2.1.3 Dependencies And Assumptions . . . . .	7
2.1.4 Functional Requirements . . . . .	7
2.1.5 Non-Functional Requirements . . . . .	7
<b>3 Project Analysis and Design</b>	<b>8</b>
3.1 Methodologies Adapted . . . . .	8
3.2 Modules . . . . .	9
3.2.1 Activity diagram . . . . .	9
3.2.2 Deployment Diagram . . . . .	10
3.2.3 PERT Chart . . . . .	11
3.2.4 Gantt Chart . . . . .	11
3.2.5 Use-Case . . . . .	13
<b>4 Project Implementation and Testing</b>	<b>14</b>
4.1 Web UI . . . . .	14
4.2 Output 1 . . . . .	15
4.3 Output 2 . . . . .	16
4.4 Code 1 . . . . .	17
4.5 Code 1 . . . . .	18
4.6 Code 2 . . . . .	19
4.7 Code 3 . . . . .	20
<b>5 Test Cases</b>	<b>21</b>

<b>6</b>	<b>Limitations</b>	<b>22</b>
<b>7</b>	<b>SYSTEM MAINTENANCE</b>	<b>22</b>
<b>8</b>	<b>Future Enhancements</b>	<b>22</b>
<b>9</b>	<b>User Manual</b>	<b>23</b>
<b>10</b>	<b>Bibliography</b>	<b>23</b>
10.1	Web References . . . . .	23

## Abstract

A multi-national company goes through lot of headache in order to hire the best talents across the nation or across the world. To view resumes of thousands of candidates is a no human task. This is the reason they have been shortlisting resumes by ATS(application tracking system). But even after this the HR has to view the resumes of all the shortlisted candidates one by one manually. This is a lot for HR as he or she has to look upto a lot more things in the company. So in order to solve the problem we have came across a solution to build resume extractor that will fetch the relevant data of these resumes and display all of them in a well-defined manner that will help the HR to distinguish between the good and best candidates.

## Objectives

- To fetch or extract relevant data from resumes.
- To Fetch data such as Education, Skills, Experience, Achievements, Projects.
- To help HR to easily compare resumes between multiple candidates and select the best suite for the company.
- To provide a platform to keep personal information updated

## List of Figures

3.1.1Diagrammatic Representation of Waterfall Model . . . . .	8
3.2.1Activity Diagram . . . . .	9
3.2.2Work Breakdown Structure . . . . .	10
3.2.3PERT Chart . . . . .	11
3.2.4Gantt Chart . . . . .	12
3.2.5Use-Case Diagram . . . . .	13
4.1.1 Upload Resume . . . . .	14
4.3.1Output 2 . . . . .	16

## List of Tables

1.5.3 Software Requirements on Client Side . . . . .	4
1.5.3 Software Requirements on Client Side . . . . .	4
6.1 Test Case - Login and Register . . . . .	21

# 1 Introduction

## 1.1 Problem Definition

To design a model this can parse information from unstructured resumes and transform it to a structured JSON format. Also, present the extracted resumes to the employer based on the job description.

## 1.2 Objectives and Scope

### 1.2.1 Objectives

The Android based application "Placement App" is

- To provide training and placement system in which communication, application and check of the process is uncomplicated.
- To keep required details regarding Placements handy
- To provide students a easy and convenient process of application for the upcoming internships and placements.
- To provide a better connection between the distanced roles.

### 1.2.2 Scope

If you are running the app on windows, then you can only extract .docs and .pdf files. If you want to parse DOC files you can install textract for your OS (Linux, MacOS). You just have to install textract (and nothing else) and doc files will get parsed easily. Anyone from HR to recruiter to manager can use this software to extract valuable data from the resume that they get everyday.



### 1.3 Existing System

The process of hiring has evolved over the period of time. In the first generation hiring model, the companies would advertise their vacancies on newspapers and television. The applicants would send in their resumes via post and their resumes would be sorted manually. Once shortlisted, the hiring team would call the applicants for further rounds of interview. Needless to say, this was a time-consuming procedure. But the industries started growing and so did the hiring needs. Hence the companies started outsourcing their hiring process. Hiring consultancies came into existence. These agencies required the applicants to upload their resumes on their websites in particular formats. The agencies would then go through the structured data and shortlist candidates for the company. This process had a major drawback. There were numerous agencies and each had their own unique format. To overcome all the above problems an intelligent algorithm was required which could parse information from any unstructured resumes, sort it based on the clusters and rank it finally. The model uses natural language processing to understand the resume and then parse the information from it. Once information is parsed it is stored in the database. When the employer posts a job opening, the system ranks the resumes based on keyword matching and shows the most relevant ones to the employer.

### 1.4 Proposed System

Data preprocessing is the first and foremost step of natural language processing. Data preprocessing is a technique of data mining which transforms raw data into a comprehensible format. Data from the real world is mostly inadequate, conflicting and contains innumerable errors. The method of Data preprocessing has proven to resolve such issues. Data preprocessing thus further processes the raw data. Data is made to pass through a series of steps in the time of preprocessing.

- Data Cleaning:  
Processes, like filling in missing values, smoothing noisy data or resolving inconsistencies, cleanses the data.
- Data Integration:  
Data consisting of various representations are clustered together and the clashes between the data are taken care of.
- Data Discretization:  
In this step, the number of values of an uninterrupted characteristic is reduced by division of the range of intervals of characteristics.
- Data Transformation:  
Data is distributed, assembled and theorized.

- Data Reduction:  
The objective of this step is to present a contracted model in a data warehouse.
- Tokenization:  
Tokenization is the task of chopping off a provided character sequence and a detailed document unit. It does away with certain characters like punctuation and the chopped units are further called tokens.

## 1.5 System Requirements

- Hardware Requirements

Table 1.5.3: Software Requirements on Client Side

Processor	Dual Core i3 Processor or Above
Ram	4 GB or above
Storage	30 GB Hard-disk space and above

- Software Requirements

Table 1.5.3: Software Requirements on Client Side

Operating System	Windows, Linux, Mac
Tools used	Visual Studio Code, Pyresparser, Docker

## 1.6 Technology Used

1. Django:  
Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.
2. Spacy:  
spaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython. The library is published under the MIT license and its main developers are Matthew Honnibal and Ines Montani, the founders of the software company Explosion. It will be used to build information extraction, natural language understanding systems.
3. Docker:  
Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production. Running app in Docker

Install docker-compose

Execute the following commands from the root of the project

Build our images

`docker-compose build`

Starting our containers and services

`docker-compose up -d`

Visit `localhost:8080` in your browser to run the app

#### 4. Pyresparser:

A simple resume parser A simple resume parser used for extracting information from resumes. A Resume parsing technology converts an unstructured form of resume data into a structured format. A Resume parser analyses resume data and extract it into machine-readable output such as XML, JSON. A CV/resume parser helps automatically store, organize, and analyze resume data to find the best candidate. `pip install pyresparser`

## 2 Software Requirement Specification and Design

### 2.1 Introduction

Corporate companies and recruitment agencies process numerous resumes daily. This is no task for humans. An automated intelligent system is required which can take out all the vital information from the unstructured resumes and transform all of them to a common structured format which can then be ranked for a specific job position. Parsed information include name, email address, social profiles, personal websites, years of work experience, work experiences, years of education, education experiences, publications, certifications, volunteer experiences, keywords and finally the cluster of the resume (ex: computer science, human resource, etc.). The parsed information is then stored in a database (NoSQL in this case) for later use. Unlike other unstructured data (ex: email body, web page contents, etc.), resumes are a bit structured. Information is stored in discrete sets. Each set contains data about the person's contact, work experience or education details. In spite of this resumes are difficult to parse. This is because they vary in types of information, their order, writing style, etc. Moreover, they can be written in various formats. Some of the common ones include '.txt', '.pdf', '.doc', '.docx', '.odt', '.rtf' etc. To parse the data from different kinds of resumes effectively and efficiently, the model must not rely on the order or type of data.

#### 2.1.1 Product Functions

The product function includes:

1. Extract name
2. Extract email
3. Extract mobile numbers
4. Extract skills
5. Extract total experience
6. Extract college name
7. Extract degree
8. Extract designation
9. Extract company names

### 2.1.2 User Characteristics

User can be the HR or the recruiter who want to hire candidates for the relevant role in the company.

### 2.1.3 Dependencies And Assumptions

The system is dependent on the internet. Needs Python 2.7. There are no other known dependencies or assumptions of this system. Rest dependencies are subject to minimum hardware requirements to run this application. It is assume that at times the system may fail to extract 100

### 2.1.4 Functional Requirements

- Extract Accurate Data: The software is expected to extract accurate data from the uploaded resume by the HR or the recruiter. Though there may be some expectation in some cases.
- Displaying Relevant Details: Once the HR uploads the resume then he or she should be able to see data in well standardized manner.

### 2.1.5 Non-Functional Requirements

- Performance: It basically deals with How fast does the system return results? How much will this performance change with higher workloads. The system should perform all the operations in less amount of time especially the loading time after uploading a resume is not more than 3 seconds.
- Modifiability: It deals with How much effort is required to make changes to the software? The system in easily modifiable.
- Maintainability: This system is easy to maintain.
- Legal: There should not be legal issues around data privacy, intellectual property rights and more. The system in legal in terms of government policies.
- Security: The system should be reliable in terms of security and is safe from various cyber attacks.

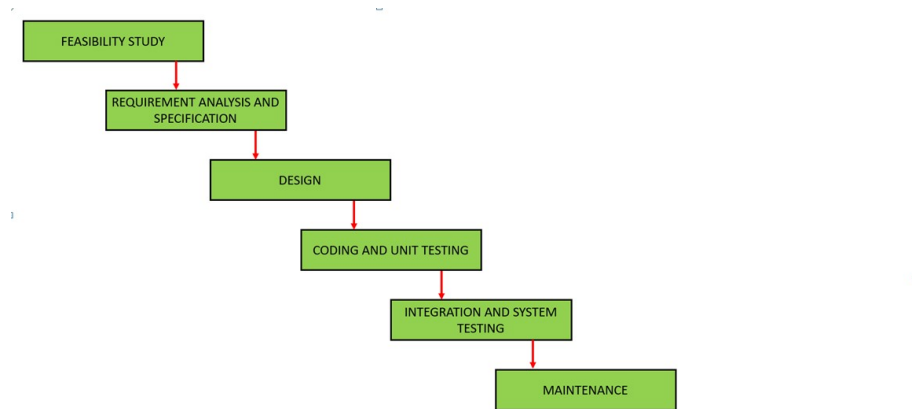
### 3 Project Analysis and Design

#### 3.1 Methodologies Adapted

The waterfall methodology is a project management approach that emphasizes a linear progression from beginning to end of a project. This methodology, often used by engineers, is front-loaded to rely on careful planning, detailed documentation, and consecutive execution. Methodology involves dividing software development work into distinct stages and coming up with tasks or activities aimed at achieving better planning and time management.

Waterfall Model:

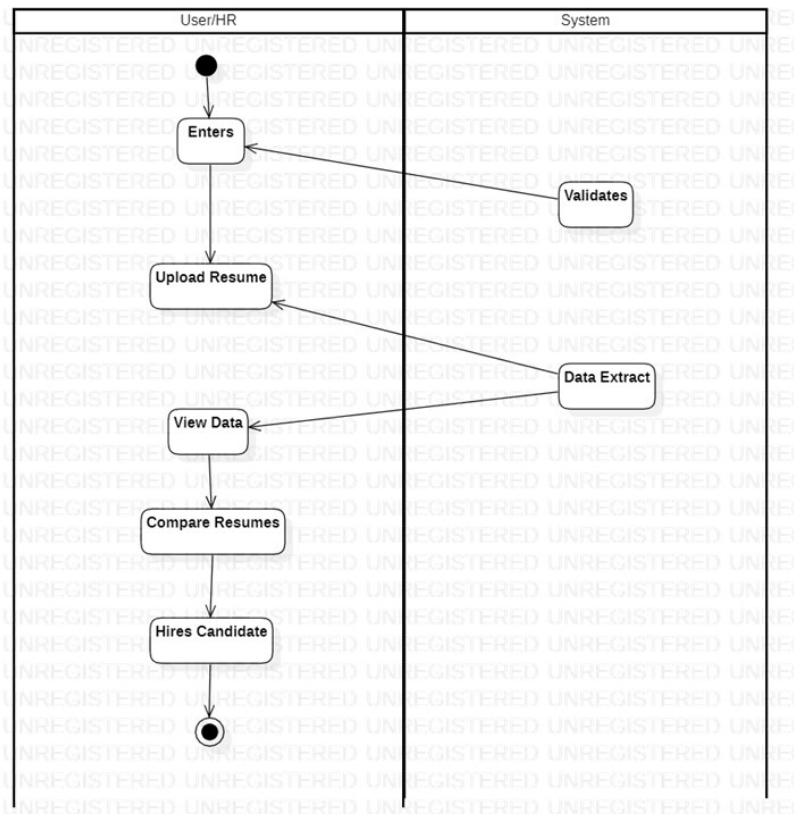
The waterfall model depicts the software development process in a linear sequential flow; due to this, it is also referred to as a linear-sequential life cycle model, which indicates that any development process steps can start only after the previous one has finished. The stages are always done in this order and never overlap. Before moving on to the next step, the developer must finish the present one. The model is called a waterfall because it progresses from one phase to the next logically. There is an emphasis on the natural succession of these phases. During the SDLC phase, each step is meant to execute specific tasks. The requirements were well-defined and unchangeable, The technology is well-known and well-understood plus the technology has huge community over the internet, the project isn't long, and the risk is either nil or negligible. It is best for these scenarios because it is straightforward to comprehend and apply. It's simple to use. Tasks can be simply organized. Both the process and the outcomes are thoroughly recorded as a part of this model.



3.1.1: Diagrammatic Representation of Waterfall Model

## 3.2 Modules

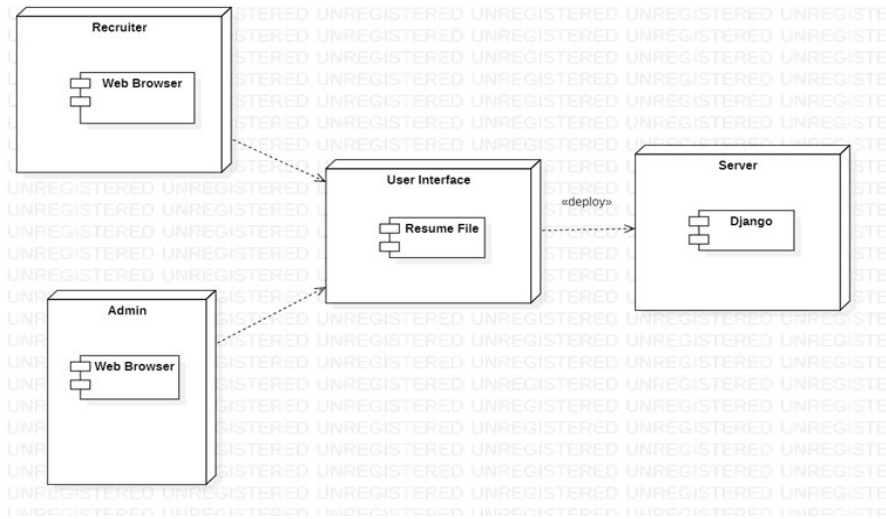
### 3.2.1 Activity diagram



3.2.1: Activity Diagram

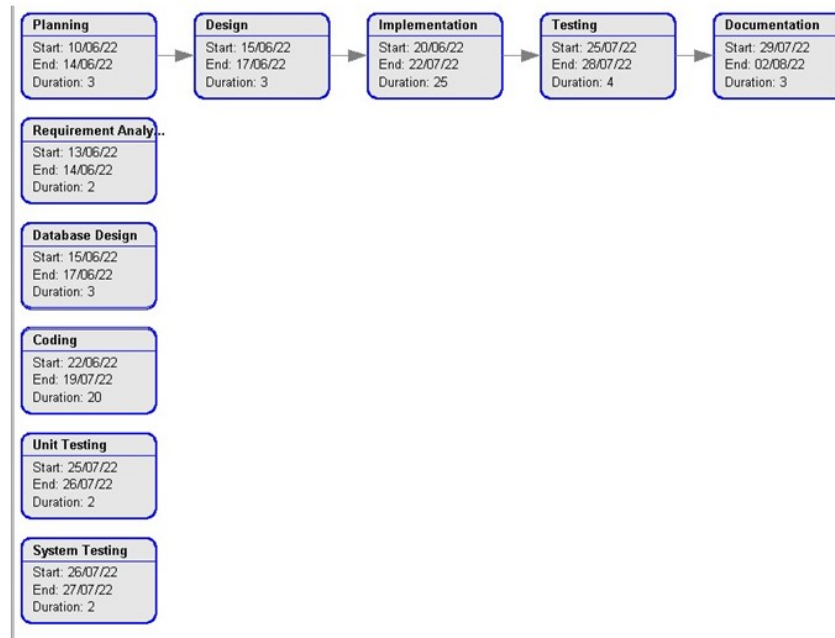


### 3.2.2 Deployment Diagram



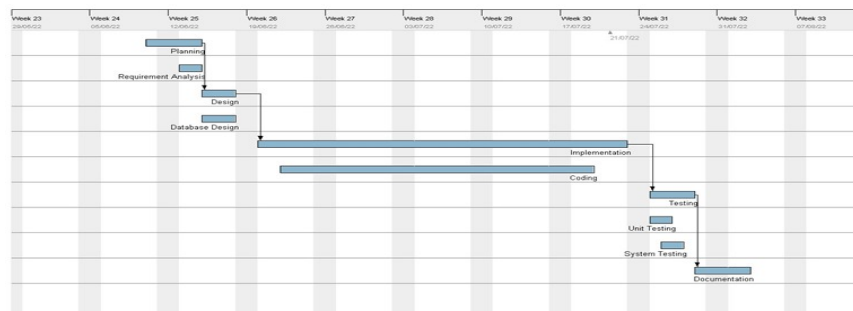
3.2.2: Work Breakdown Structure

### 3.2.3 PERT Chart



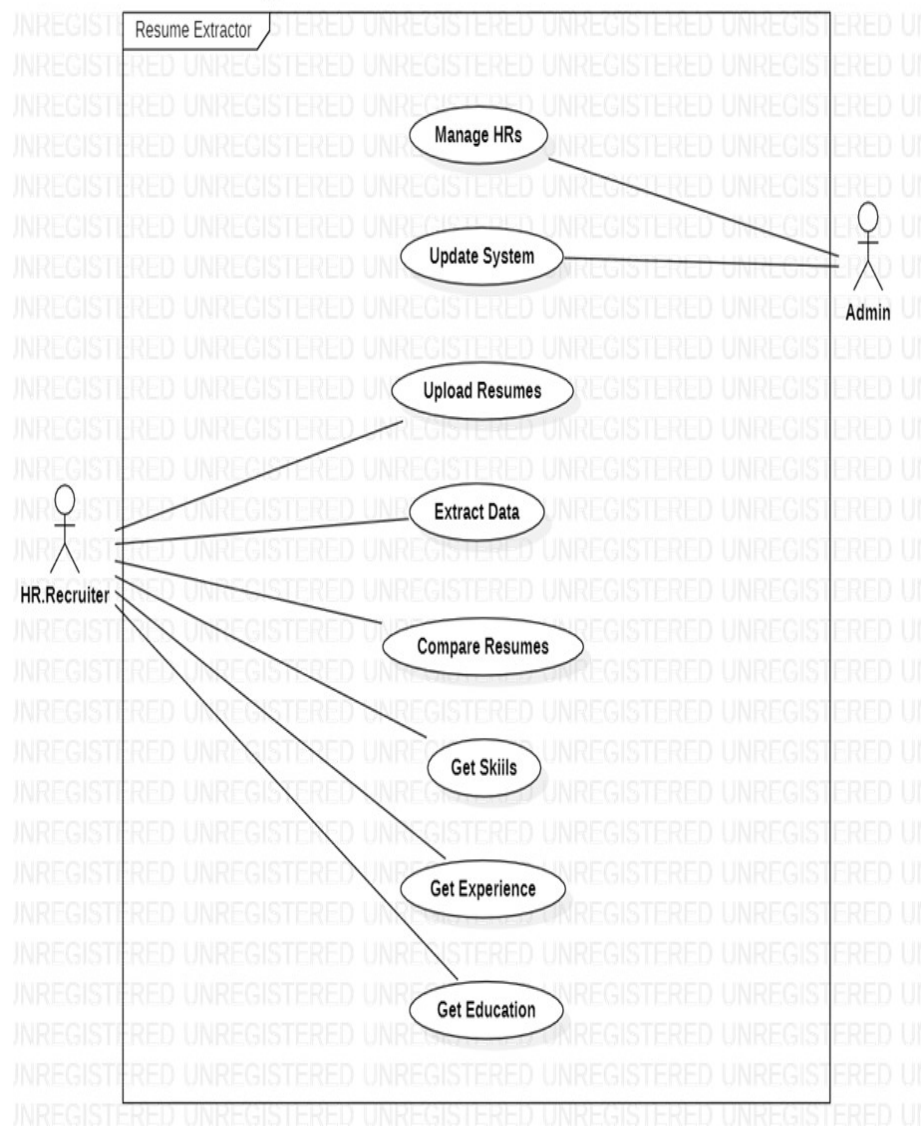
3.2.3: PERT Chart

### 3.2.4 Gantt Chart



3.2.4: Gantt Chart

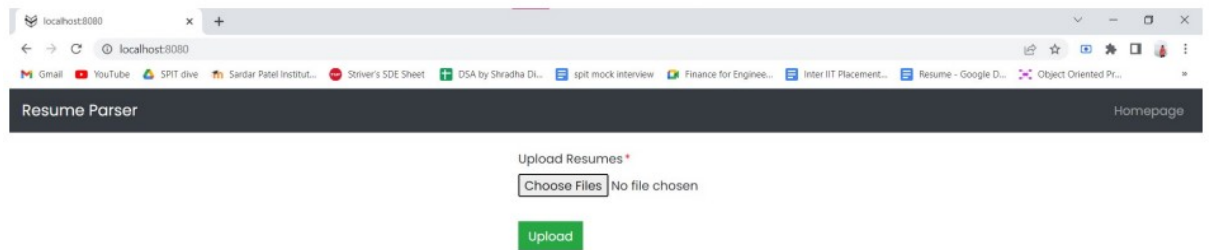
### 3.2.5 Use-Case



3.2.5: Use-Case Diagram

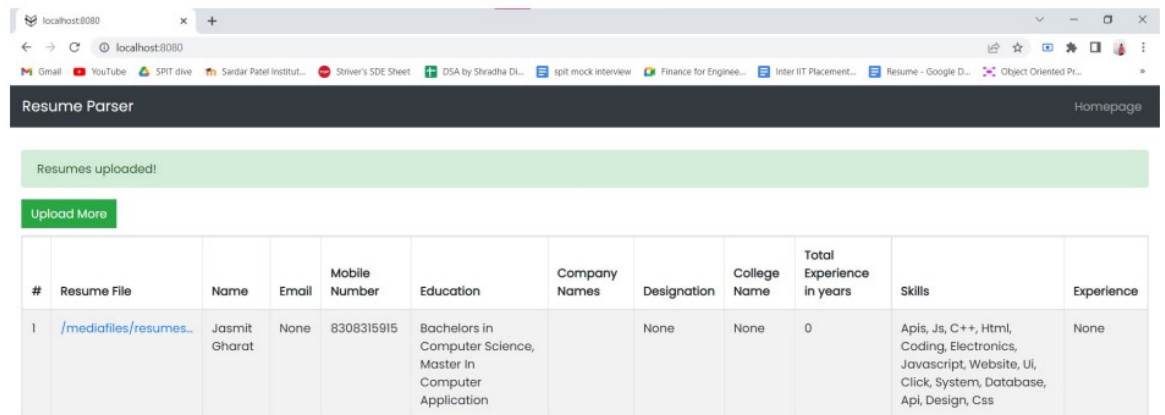
## 4 Project Implementation and Testing

### 4.1 Web UI



#### 4.1.1: Upload Resume

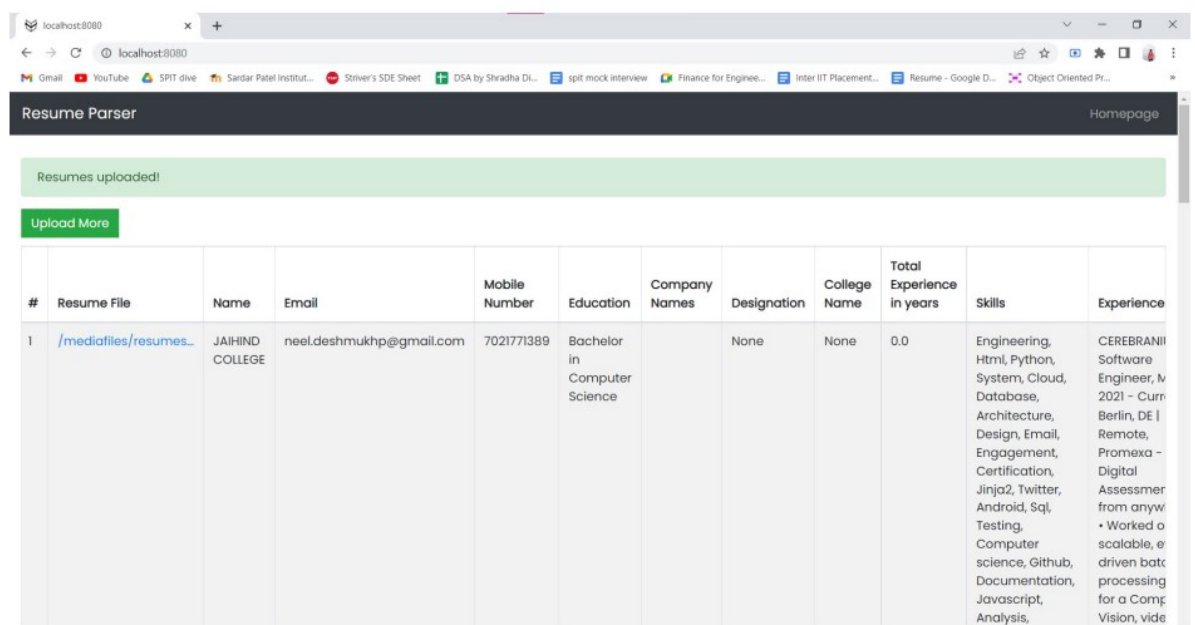
## 4.2 Output 1



The screenshot shows a web browser window with the address bar at localhost:8080. The page title is "Resume Parser" and there is a "Homepage" link in the top right. A green notification bar at the top says "Resumes uploaded!". Below it is a green "Upload More" button. The main content is a table with 11 columns: #, Resume File, Name, Email, Mobile Number, Education, Company Names, Designation, College Name, Total Experience in years, Skills, and Experience. The table contains one row of data for a resume uploaded from /media/files/resumes... belonging to Jasmit Gharat.

#	Resume File	Name	Email	Mobile Number	Education	Company Names	Designation	College Name	Total Experience in years	Skills	Experience
1	/media/files/resumes...	Jasmit Gharat	None	8308315915	Bachelors in Computer Science, Master In Computer Application		None	None	0	Apis, Js, C++, Html, Coding, Electronics, Javascript, Website, Ui, Click, System, Database, Api, Design, Css	None

### 4.3 Output 2



#	Resume File	Name	Email	Mobile Number	Education	Company Names	Designation	College Name	Total Experience In years	Skills	Experience
1	/media/files/resumes...	JAIHIND COLLEGE	neel.deshmukhp@gmail.com	7021771389	Bachelor in Computer Science		None	None	0.0	Engineering, Html, Python, System, Cloud, Database, Architecture, Design, Email, Engagement, Certification, Jinja2, Twitter, Android, Sql, Testing, Computer science, Github, Documentation, Javascript, Analysis,	CEREBRANI Software Engineer, 2021 - Current Berlin, DE   Remote, Promexa - Digital Assesmer from anyw • Worked o scalable, e driven batc processing for a Comp Vision, vide

4.3.1: Output 2

#### 4.4 Code 1

```
import os
import multiprocessing as mp
import io
import spacy
import pprint
from spacy.matcher import Matcher
from . import utils

class ResumeParser(object):
    def __init__(
        self,
        resume,
        skills_file=None,
        custom_regex=None
    ):
        nlp = spacy.load('en_core_web_sm')
        custom_nlp =
spacy.load(os.path.dirname(os.path.abspath(__file__)))
        self.__skills_file = skills_file
        self.__custom_regex = custom_regex
        self.__matcher = Matcher(nlp.vocab)
        self.__details = {
            'name': None,
            'email': None,
            'mobile_number': None,
            'skills': None,
            'college_name': None,
            'degree': None,
```



## 4.5 Code 1

```
        'designation': None,
        'experience': None,
        'company_names': None,
        'no_of_pages': None,
        'total_experience': None,
    }
    self.__resume = resume
    if not isinstance(self.__resume, io.BytesIO):
        ext =
os.path.splitext(self.__resume)[1].split('.')[1]
    else:
        ext = self.__resume.name.split('.')[1]
    self.__text_raw = utils.extract_text(self.__resume, '.'
+ ext)
    self.__text = ' '.join(self.__text_raw.split())
    self.__nlp = nlp(self.__text)
    self.__custom_nlp = custom_nlp(self.__text_raw)
    self.__noun_chunks = list(self.__nlp.noun_chunks)
    self.__get_basic_details()

    def get_extracted_data(self):
        return self.__details

    def __get_basic_details(self):
        cust_ent = utils.extract_entities_wih_custom_model(
            self.__custom_nlp
        )
        name = utils.extract_name(self.__nlp,
matcher=self.__matcher)
        email = utils.extract_email(self.__text)
        mobile = utils.extract_mobile_number(self.__text,
self.__custom_regex)
        skills = utils.extract_skills(
            self.__nlp,
            self.__noun_chunks,
```

## 4.6 Code 2

```
        self.__skills_file
    )

    entities =
utils.extract_entity_sections_grad(self.__text_raw)

    # extract name
    try:
        self.__details['name'] = cust_ent['Name'][0]
    except (IndexError, KeyError):
        self.__details['name'] = name

    # extract email
    self.__details['email'] = email

    # extract mobile number
    self.__details['mobile_number'] = mobile

    # extract skills
    self.__details['skills'] = skills

    # extract college name
    try:
        self.__details['college_name'] = entities['College
Name']
    except KeyError:
        pass

    # extract education Degree
    try:
        self.__details['degree'] = cust_ent['Degree']
    except KeyError:
        pass
```

## 4.7 Code 3

```
        try:
            self.__details['designation'] =
cust_ent['Designation']
        except KeyError:
            pass

        # extract company names
        try:
            self.__details['company_names'] =
cust_ent['Companies worked at']
        except KeyError:
            pass

        try:
            self.__details['experience'] =
entities['experience']
            try:
                exp = round(
utils.get_total_experience(entities['experience']) / 12,
                2
            )
            self.__details['total_experience'] = exp
        except KeyError:
            self.__details['total_experience'] = 0
        except KeyError:
            self.__details['total_experience'] = 0
        self.__details['no_of_pages'] =
utils.get_number_of_pages(
            self.__resume
        )
    return

def resume_result_wrapper(resume):
```

## 5 Test Cases

Table 6.1: Test Case - Login and Register

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	Name Extract	Extraction System extract correct name	successful	Extraction successful	Pass
2	Education Extract	System extract correct Education	System extract perfect Education	System extract perfect Education	Pass
3	Skills Extract	System extract correct Skills	System partially extracts correct Skills	Extraction successful	Fail
4	Experience Extract	System extract correct Experience	Extraction successful	Extraction successful	Pass
5	College name Extract	System extract correct College name	Partial College name extracted	Extraction successful	Fail
6	Phone number Extract	System Extract correct phone number	Extraction successful	Extraction successful	Pass

## 6 Limitations

- Requires Internet connection.
- Requires a web browser
- Sometimes can fetch wrong data.
- If you are running the app on windows, then you can only extract .docs and .pdf files
- If you want to parse DOC files you can install textract for your OS (Linux, MacOS)

## 7 SYSTEM MAINTENANCE

- System Maintenance is needed when a new version of Resume-Extractor is released.
- Entire System Maintenance work is also carried out when the system fails to work properly.
- Sometimes can fetch wrong data.
- Application will require maintenance when there is a major release or change of functionality if some new functionality is added to the website.
- System Maintenance will be carried out at least quarterly to check whether everything is working fine.

## 8 Future Enhancements

We successfully converted different formats of resumes to text and parse relevant information from there. We also were able to scrape keywords from different social networking sites including Stack Overflow, LinkedIn, etc and find the similarity between them with which we could determine the genre of the resume (e.g: Computer science, Management, Sales, human resource, etc). Future work includes ranking the resume and analysing information about the candidate from social networking sites like Face book and Twitter so that we can decide more accurately and authentically whether or not to offer the candidate, a job.

## 9 User Manual

### Step 1

Visit The Website

### Step 2

Upload Your Resume or The Resume From Which You want to Extract Data

### Step 3

User can now see the data that the system has extracted from the Resume

## 10 Bibliography

### 10.1 Web References

- [1.] <https://medium.com/@divalicious.priya/information-extraction-from-cv-acec216c3f48>
- [2.] <https://www.kaggle.com/code/nirant/hitchhiker-s-guide-to-nlp-in-spacy/notebook>
- [3.] <https://blog.apilayer.com/build-your-own-resume-parser-using-python-and-nlp/>
- [4.] [https://www.tutorialspoint.com/compiler\\_design/images/token\\_passing.jpg](https://www.tutorialspoint.com/compiler_design/images/token_passing.jpg)
- [5.] [https://www.ijircce.com/upload/2016/april/218\\_Intelligent.pdf](https://www.ijircce.com/upload/2016/april/218_Intelligent.pdf)