

CHAPTER 1

INTRODUCTION

A brief introduction about the project, hardware and software platforms, used for the development purpose is discussed in this chapter.

1.1 PROBLEM STATEMENT

Resume parser with NLP is a software that can help both recruiter and job seekers. The objective of this project is to design a software to extract information from unstructured resume and transform that information to structured format. Resumes are ranked based on the information extracted from the resumes against the job description of the company.

It helps recruiters to parse resume information and convert them into standard JSON format. The standardized JSON format of the resume contains key-value pairs of the desired details of candidate like Personal details, educational qualification, experience and technical skills. The candidate can upload their resume in multiple formats like .docx, .pdf, .txt, etc.

Resume parser will go through all the resumes stored in JSON format and suggest only the eligible candidates based on the constraints given by the recruiter and display it to the concerned authority. The candidate can also filter the jobs posted by the recruiter based on designation, location, company. The main aim of developing this project is to reduce the burden of recruiters in Online Recruitment System (ORS).

1.2 DEVELOPMENT ENVIRONMENT

The following are the hardware and software specifications used in this project.

1.2.1 MINIMUM SYSTEM SPECIFICATION

HARDWARE CONFIGURATION

CPU	: Intel® Core™ i5 – 8250U 1.80GHz
RAM	: 8 GB
HDD	: 1024 GB

SOFTWARE CONFIGURATION

Operating system	: Windows 10
Development Languages	: Python, HTML
Tools & Technologies	: Spyder
DB	: SQLite

1.3 TECHNOLOGIES USED

Resume parser is built with the help of following software technologies and packages.

1.3.1 PYTHON

Python is a widely used general-purpose, versatile and modern programming language. Python is an interpreted, high level programming language created by Guido Van Rossum. Its design philosophy emphasizes code readability and its syntax allows programmers to express concepts in fewer lines of code than other

languages such as C. Python language provides constructs intended to enable clear programs on both a small and large scale. It supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. Python is dynamically typed and it follows automatic memory management approach called garbage collection. Python is often used as a scripting language also. Python code can be packages into a standalone executable program. Python interpreters are available for many operating system.

1.3.2 SPYDER

Spyder is an open source cross platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open source software.

Features of Spyder

- An editor with syntax highlighting, introspection, code completion
- Support for multiple IPython consoles
- The ability to explore and edit variables from a GUI
- A help pane is able to retrieve and render classes and methods automatically
- A debugger linked to IPdb, for step-by-step execution
- A run-time profiler to benchmark code
- Allowing to work on multiple development efforts simultaneously
- A built-in file explorer for interacting with the filesystem
- A "Find in Files" feature, allowing full regular expression
- An online help browser, allowing users to search package documentation

- A history log recording every user command entered in each console
- Spyder-Notebook is a plugin allowing for editing of Jupyter Notebooks

1.3.3 SQLITE

SQLite in general, is a server-less database that can be used within almost all programming languages including Python. Server-less means there is no need to install a separate server to work with SQLite one can connect directly with the database. SQLite is a lightweight database that can provide a relational database management system (RDBMS) with zero-configuration because there is no need to configure or setup anything to use it.

Features of SQLite

- Extremely light-weighted
- It is serverless which means no need of any separate server for availing its services
- No complex setup
- Fully transactional and concurrency control

1.3.4 FLASK

Flask is a web application framework written in Python. A web application framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management. Flask has many configuration values, with sensible defaults, and a few conventions when getting started. By convention, templates and static files are stored in sub directories within the application's Python source tree, with the names templates and static respectively.

1.3.5 NLP

Natural Language Processing is manipulation or understanding text or speech by any software or machine. An analogy is that humans interact, understand each other views, and respond with the appropriate answer. In NLP, this interaction, understanding, the response is made by a computer instead of a human.

1.3.6 NLTK

NLTK stands for Natural Language Toolkit. This toolkit is one of the most powerful NLP libraries which contains packages to make machines understand human language and reply to it with an appropriate response. NLTK includes graphical demonstrations and sample data. NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities

The next chapter will discuss the drawbacks of the existing system and the advantages of the proposed system.

CHAPTER 2

SYSTEM ANALYSIS

This chapter describes the analysis of the existing system and the proposed system. This chapter deals in detail about the objectives of the system and then gives a detailed description about the system.

2.1 EXISTING SYSTEM

Resume plays an important role in recruitment process. It is a document used by persons to present their educational qualifications and skills. It is an unstructured document without any standard format. There are various online job portals available. In these sites the candidate must sign-up first by a unique email address and a valid password. After sign-up the candidate has to manually fill a form regarding their personal details. Then the candidate has to upload their resume. Although, the personal details are available in the resume itself, the candidate has to manually fill the form without ignoring. It is a time-consuming process. In the existing job portals, there is no standard format for the resume. So, each resume will be in a different format. Because of that, the recruiter has to go through every parameter of resume one by one which is a difficult task. Also resumes are written in different format so it is difficult for companies to store the resume data into a huge database.

2.2 PROPOSED SYSTEM

Resume parser will eliminate the difficulties arising in the existing system. In this project candidate has to sign-up first by providing their unique email address and a password. Then the candidate has to upload only the resume. No need to fill

the personal details manually. A user interface screen is developed for the candidates to upload the resume. Once it is uploaded, it will be parsed and then transformed into a standardized format. The candidate can upload his resume in multiple formats like .doc, .pdf, .txt. The system will go through all the resumes and suggest the eligible candidates based on the constraints and display it to the concerned authority.

The “Resume parser” is an application which helps recruiters to parse resume information and convert them into standard JSON format. The standardized format of the resume contains desired details of candidate like Personal details, educational qualification, technical skills and work experience. Resume parsing is the process of analysing the resume and extracting elements of the resume. It is also known as CV Extraction, CV Parsing or Resume Extraction. This project is designed to extract information from unstructured resume and transform that information to structured format and ranking those resumes based on the information extracted, according to the skill sets of the candidate and based on the job description of the company.

The system should satisfy the requirements of both employer and candidate. Resume parser Application should be able to reduce lot of burden on the head of candidate/employee in Online Recruitment System (ORS). The recruiter cannot read each and every resume line by line to know about the candidate.

2.3 SYSTEM FLOW

A system flowchart symbolically shows how data flows throughout a system and how event-controlling decisions are made. System flow diagrams, also known as process flow diagrams or data flow diagrams. In a system flow diagram, the goal is to present a visual representation of some component of the business model. The system flow of resume parser is clearly depicted in Figure 2.1.

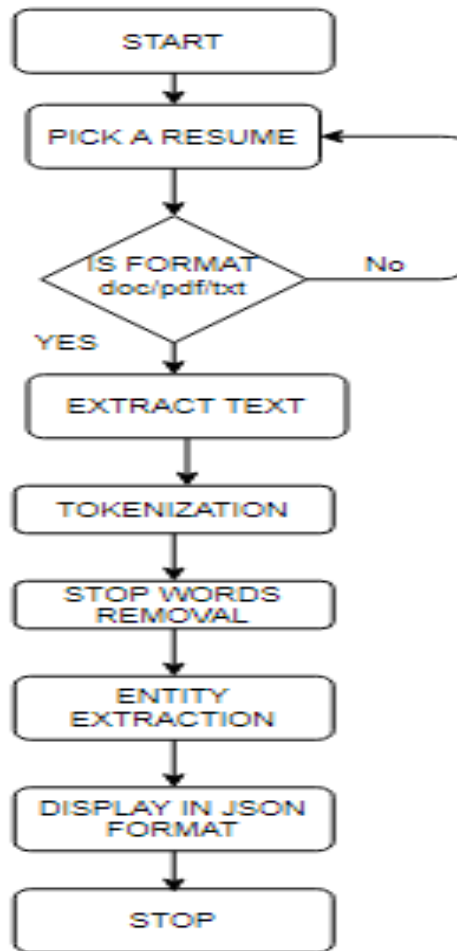


FIGURE 2.1 SYSTEM FLOW

Resume parser can parse the resumes stored in multiple formats like pdf, docx and txt. Once a candidate/job seeker uploads a resume the system will check whether the format is docx or pdf or txt. If so, the system will extract the text from the resumes and perform tokenization. After tokenization the resume parser will do stop word removal. Then the system will execute Named Entity Extraction (NEE) algorithm. The extracted content will be displayed in JSON format into the SQLite database.

2.4 MODULE DESCRIPTION

A module is a part of a program. Modules make programmer's job easy by allowing the programmer to focus on only one specific task. The following are the list of modules used in this project.

2.4.1 EXTRACTING TEXT FROM PDF FORMAT

PDF stands for **Portable Document Format**. It uses **.pdf** extension. It is used to present and exchange documents reliably, independent of software, hardware, or operating system. PyPDF2 is a pure-python PDF library capable of splitting, merging together, cropping, and transforming the pages of PDF files. It can also add custom data, viewing options, and passwords to PDF files.

It is capable of:

- extracting document information (title, author, ...)
- splitting or merging or cropping documents page by page
- merging multiple pages into a single page
- encrypting and decrypting PDF files

The following code represents the syntax to extract text from a pdf resume.

```
import PyPDF2

pdfFileObj = open(filename,'rb')

pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

num_pages = pdfReader.numPages

text = ""
```

```

for i in range(0,num_pages):

    pageObj = pdfReader.getPage(i)

    text += pageObj.extractText()

print(text)

```

In the above code a PDF file is opened using filename with the absolute path. numPages() is a function which returns the number of pages in the PDF file. For every page in the PDF file, extract the text and append to the variable text.

2.4.2 EXTRACTING TEXT FROM DOCX FORMAT

Docx2txt is a tool that attempts to generate equivalent text files from Microsoft .docx documents, preserving some formatting and document information along with appropriate character conversions for ASCII text experience. It is a platform independent solution consisting of Perl and Unix/Windows shell scripts and a configuration file to control the output text appearance to fair extent. It can very conveniently be used to build a web based docx document conversion service. This tool can extract text from corrupted docx documents also.

The following code represents the syntax to extract text from a docx resume.

```

import Docx2txt

doc=docx.Document(filename)

text=""

for para in doc.paragraphs:

    text+=para.text()

```

In the above code a docx file with filename is given as input to the function Document(). For every page in the docx file the content is extracted using the function text() and stored in the variable text.

2.4.3 TOKENIZATION

Tokenization is a very common task in NLP, it is basically a task of chopping a character into pieces, called as token, and throwing away the certain characters at the same time, like punctuation. **Tokenization** is the process of tokenizing or splitting a string, text into a list of tokens.

It can tokenize the following,

- Text into sentences tokenization
- Sentences into words tokenization
- Sentences using regular expressions tokenization

The following code represents the syntax for tokenization in Python.

```
import nltk

sentTokens=nltk.sent_tokenize(text)

wordTokens=nltk.word_tokenize(text)
```

To use functionalities defined in nltk it must be imported to the code first as shown in the above code. The text extracted from the resume will be the input and it is splitted into sentences using the function sent_tokenize() and then stored in the variable sentTokens. Nltk can also split the text into words using word_tokenize() function.

2.4.4 STOP WORD REMOVAL

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that is occurred most often in a text. These words will not bring a necessary meaning. But these words also consume memory space. Figure 2.2 shows list of stop words in nltk.

```
{‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’, ‘about’, ‘once’, ‘during’, ‘out’,  
‘very’, ‘having’, ‘with’, ‘they’, ‘own’, ‘an’, ‘be’, ‘some’, ‘for’, ‘do’, ‘its’, ‘yours’, ‘such’, ‘into’, ‘of’,  
‘most’, ‘itself’, ‘other’, ‘off’, ‘is’, ‘s’, ‘am’, ‘or’, ‘who’, ‘as’, ‘from’, ‘him’, ‘each’, ‘the’, ‘themselves’,  
‘until’, ‘below’, ‘are’, ‘we’, ‘these’, ‘your’, ‘his’, ‘through’, ‘don’, ‘nor’, ‘me’, ‘were’, ‘her’, ‘more’,  
‘himself’, ‘this’, ‘down’, ‘should’, ‘our’, ‘their’, ‘while’, ‘above’, ‘both’, ‘up’, ‘to’, ‘ours’, ‘had’, ‘she’,  
‘all’, ‘no’, ‘when’, ‘at’, ‘any’, ‘before’, ‘them’, ‘same’, ‘and’, ‘been’, ‘have’, ‘in’, ‘will’, ‘on’, ‘does’,  
‘yourselves’, ‘then’, ‘that’, ‘because’, ‘what’, ‘over’, ‘why’, ‘so’, ‘can’, ‘did’, ‘not’, ‘now’, ‘under’,  
‘he’, ‘you’, ‘herself’, ‘has’, ‘just’, ‘where’, ‘too’, ‘only’, ‘myself’, ‘which’, ‘those’, ‘i’, ‘after’, ‘few’,  
‘whom’, ‘t’, ‘being’, ‘if’, ‘theirs’, ‘my’, ‘against’, ‘a’, ‘by’, ‘doing’, ‘it’, ‘how’, ‘further’, ‘was’, ‘here’,  
‘than’}
```

FIGURE 2.2 STOP WORDS IN NLTK

The following code represents the syntax for stop words removal in python.

```
from nltk.corpus import stopwords  
  
stop_words = set(stopwords.words('english'))  
  
filtered_sentence = [w for w in text if not w in stop_words]
```

In the above code the set of stop words in english language is imported into the variable stop_words. Then the stop words are removed from the original text and stored into the variable filtered_sentence.

2.4.5 NAMED ENTITY EXTRACTION

Named entity extraction or Named entity recognition (NER) is probably the first step towards information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. NER is used in many fields in Natural Language Processing (NLP), and it can help answering many real-world questions.

The following code is used for Named Entity Extraction (NEE).

```
from nltk.corpus import treebank_chunk

from nltk.chunk import ne_chunk

ne_chunk(tr; kleebank_chunk.tagged_sents()[0])
```

In the above code the text is separated into a predefined set of tags such as Person, Organization, Place etc.

2.4.6 CONVERT TEXT INTO JSON FORMAT

JavaScript Object Notation (JSON) is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format, with a diverse range of applications, such as serving as replacement for XML in AJAX systems.

JSON is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data. JSON filenames use the extension .json.

Next chapter deals with the complete design of the project.

CHAPTER 3

SYSTEM DESIGN

System design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization. This chapter will discuss the detailed design of Resume parser software.

3.1 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

There are three main users or actors in Resume Parser

- i) Candidate or Job seeker
- ii) Recruiter
- iii) Admin

Candidate will upload the resume into the system and the Recruiter will post the job description into the system. The system will go through the resumes and suggest best jobs to the candidates and also suggest best candidates to the recruiters.

Figure 3.1 clearly shows the user's interactions with the resume parser system.

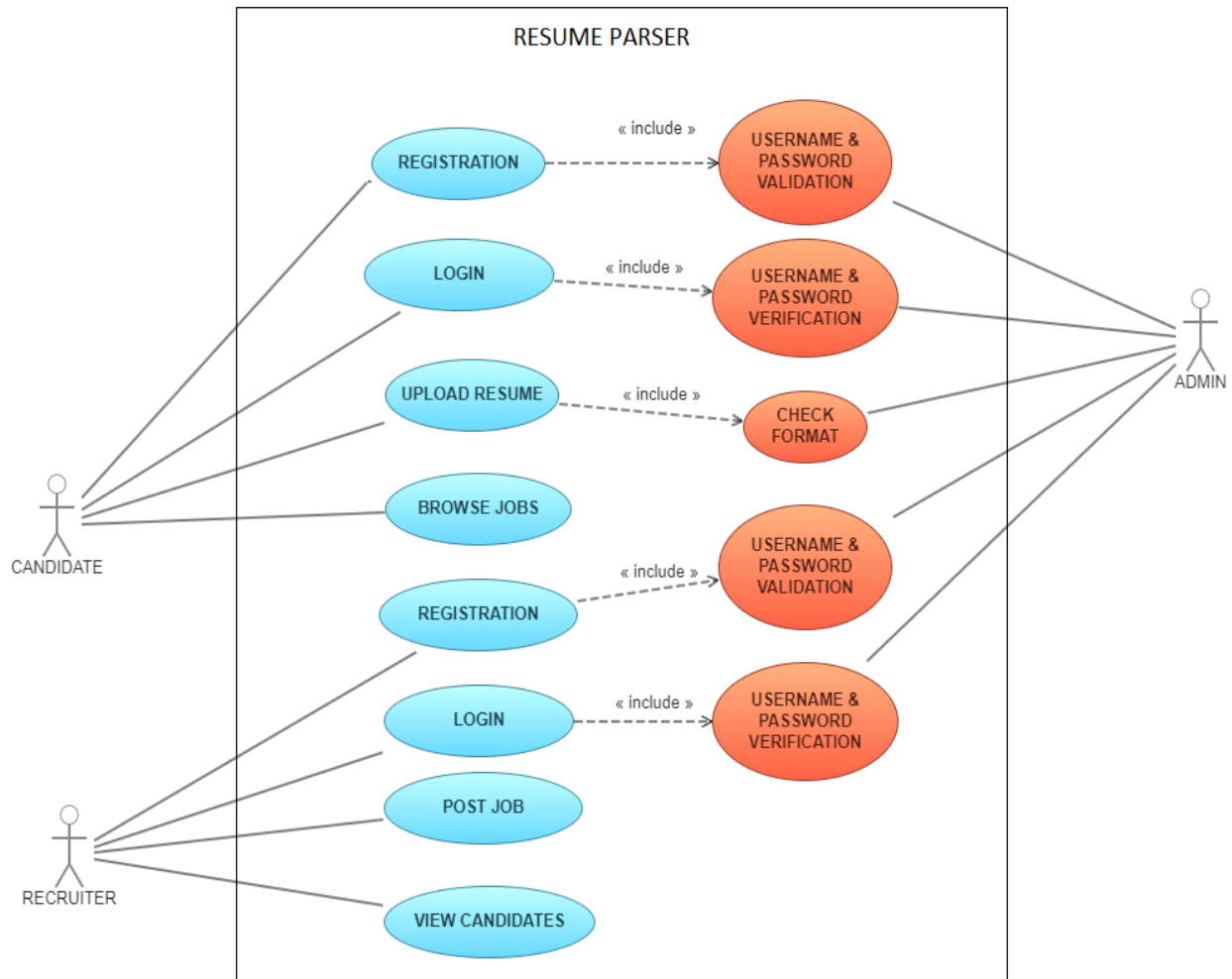


FIGURE 3.1 USE CASE DIAGRAM

The admin is the resume parser system that will do all types of verification and validation process like username and password validation.

Candidate can browse the jobs based on their own interest like designation based, company based, location based.

Similarly, recruiter can view the eligible candidates based on educational qualification, years of experience.

3.2 ENTITY RELATIONSHIP DIAGRAM

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. Figure 3.2 shows the ER diagram for resume parser.

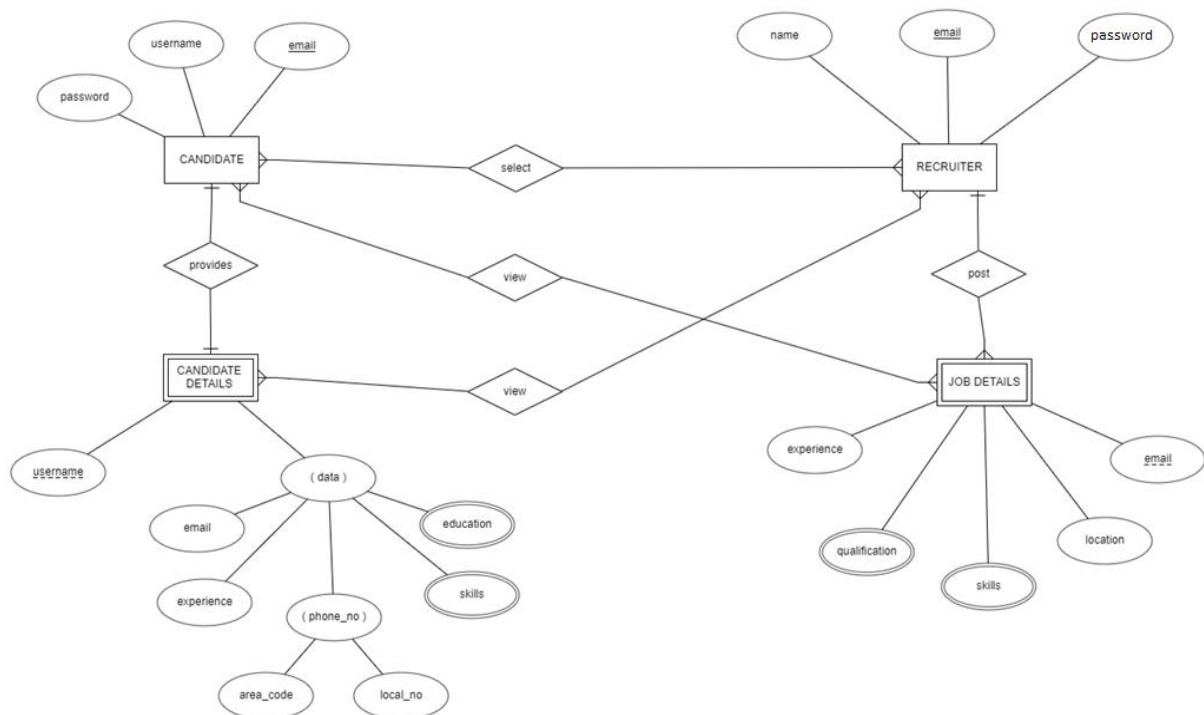


FIGURE 3.2 ENTITY RELATIONSHIP DIAGRAM

Resume parser required four tables to achieve its goal. They are,

- i) Candidate
- ii) Recruiter
- iii) CandidateDetails
- iv) JobPosting

CANDIDATE TABLE:

Each candidate has to register with the system by providing their username, email address and a password. Email address must be unique between candidates. Two candidates cannot have same email address. Figure 3.3 shows the schema description of Candidate table.

```
conn.execute("CREATE TABLE Candidate (USER_NAME TEXT NOT NULL,  
EMAIL TEXT NOT NULL PRIMARY KEY,  
PASSWORD CHAR(20) NOT NULL);")
```

FIGURE 3.3 SCHEMA FOR CANDIDATE TABLE

CANDIDATE DETAILS TABLE:

CandidateDetails is a weak entity which does not have primary key. It depends on the Candidate table's primary key. It consists of only 2 fields email address and the data extracted from resumes. The data field is of JSON data type which in turn contains key-value pairs for education, email address, skillset, experience., phone number, email. Figure 3.4 shows the schema description of CandidateDetails table.

```
c.execute("CREATE TABLE CandidateDetails (  
email varchar(30) PRIMARY KEY, data json NOT NULL)")
```

FIGURE 3.4 SCHEMA FOR CANDIDATE DETAILS TABLE

RECRUITER TABLE:

Each recruiter has to register with the system by providing the company's username, email address and a password. Email address must be unique between recruiters. Two recruiters cannot have same email address. Figure 3.5 shows the schema description of Recruiters table.

```
c.execute("CREATE TABLE Recruiter
(COMPANY_NAME TEXT NOT NULL,
EMAIL TEXT NOT NULL PRIMARY KEY,
PASSWORD CHAR(20) NOT NULL)")
```

FIGURE 3.5 SCHEMA FOR RECRUITER TABLE

JOBPOSTINGS TABLE

JobPostings is a weak entity which does not have primary key. It depends on the Recruiters table's primary key. It consists of only 6 fields company name, location, skillset, designation, experience and educational qualification. The data field is of JSON data type which in turn contains key-value pairs for education, email address, skillset, experience., phone number, email. Figure 3.6 shows the schema description of JobPostings table.

```
c.execute("CREATE TABLE JobPostings
(COMPANY TEXT NOT NULL, LOCATION TEXT NOT NULL,
SKILL json NOT NULL, DESIGNATION TEXT NOT NULL,
EXPERIENCE TEXT NOT NULL, EDUCATION TEXT NOT NULL)")
```

FIGURE 3.6 SCHEMA FOR JOBPOSTINGS TABLE

The implementation of this project is described in the next chapter.

CHAPTER 4

IMPLEMENTATION

This chapter will discuss the user interface design of Resume parser to satisfy the needs of Online Recruitment Process (ORS).

4.1 USER INTERFACE DESIGN

In homepage of the resume parser the user can select either candidate or recruiter. Figure 4.1 shows the home page of resume parser.

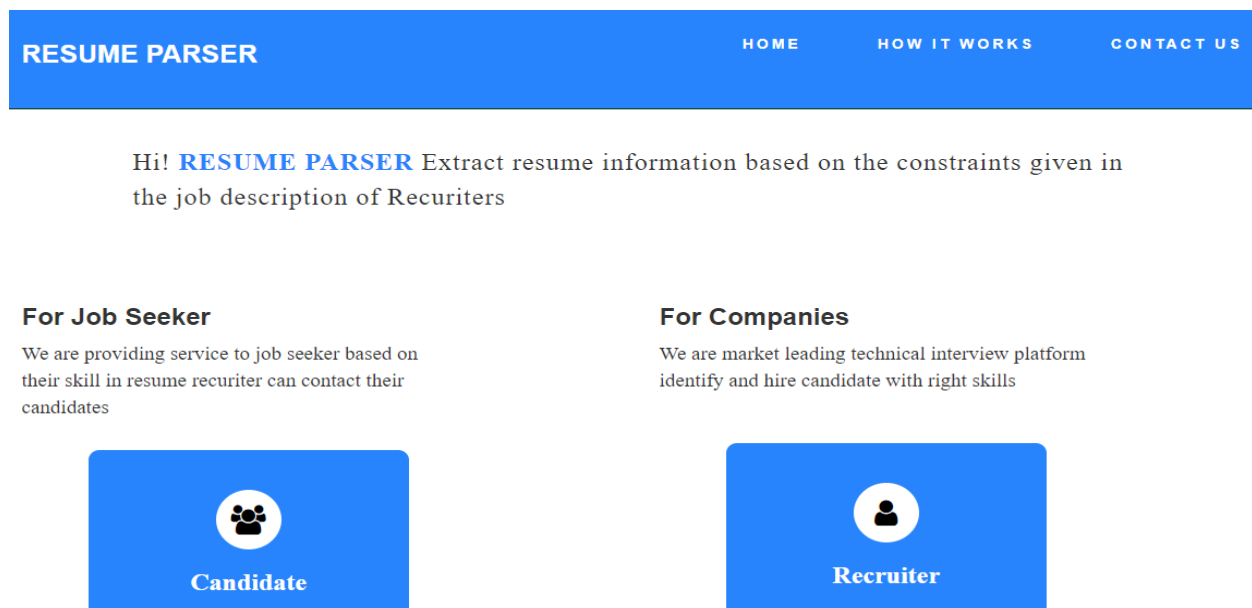
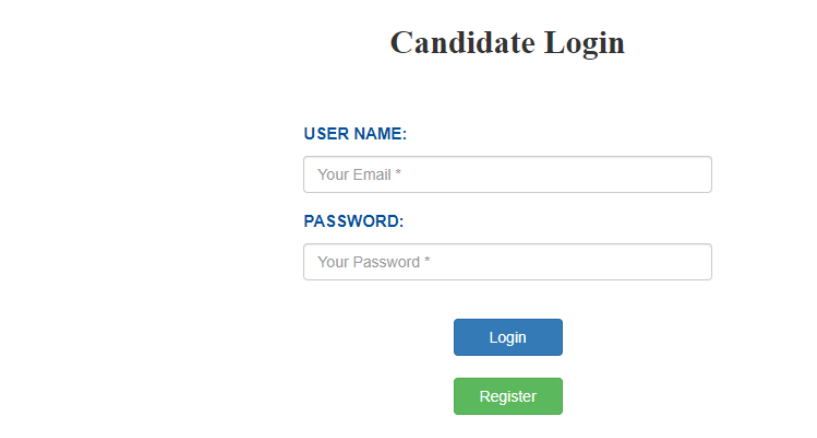


FIGURE 4.1 HOME PAGE OF RESUME PARSER

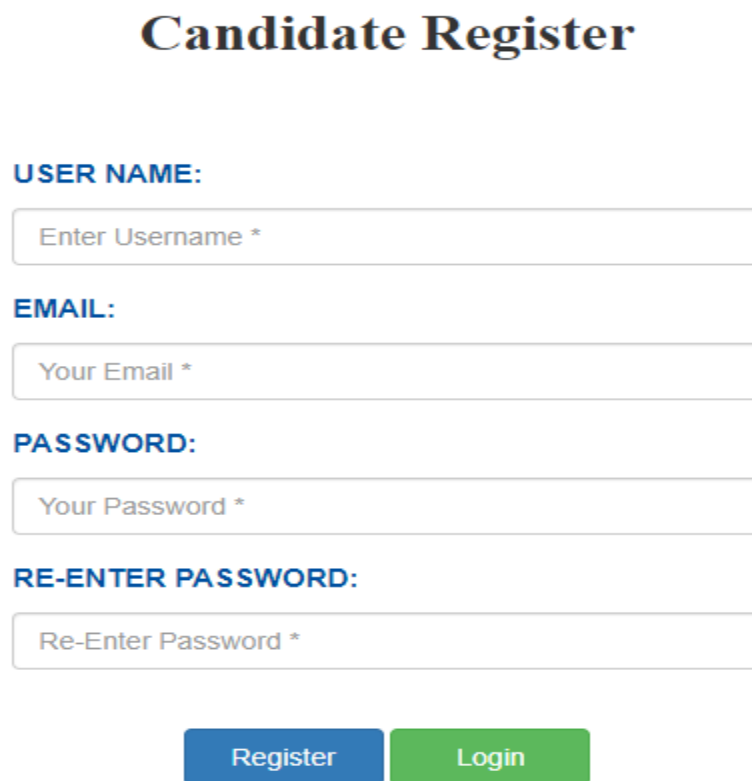
If candidate option is selected it will ask for either new candidate or existing candidate. New candidate can move into registration and existing candidate can choose login option. Figure 4.2 shows the login page for the candidate.



The image shows a 'Candidate Login' page. At the top, the title 'Candidate Login' is centered in a bold, black, serif font. Below the title, there are two labels in blue: 'USER NAME:' and 'PASSWORD:'. Under 'USER NAME:', there is a text input field with the placeholder text 'Your Email *'. Under 'PASSWORD:', there is a text input field with the placeholder text 'Your Password *'. Below these fields, there are two buttons: a blue 'Login' button and a green 'Register' button, both with white text. A vertical line is on the right side of the form.

FIGURE 4.2 CANDIDATE LOGIN PAGE

New candidate can register into the system by providing their own username, email and password, Once the system validates the password it will be moved into login page. Figure 4.3 shows the registration page for the candidate.



The image shows a 'Candidate Register' page. At the top, the title 'Candidate Register' is centered in a bold, black, serif font. Below the title, there are four labels in blue: 'USER NAME:', 'EMAIL:', 'PASSWORD:', and 'RE-ENTER PASSWORD:'. Under 'USER NAME:', there is a text input field with the placeholder text 'Enter Username *'. Under 'EMAIL:', there is a text input field with the placeholder text 'Your Email *'. Under 'PASSWORD:', there is a text input field with the placeholder text 'Your Password *'. Under 'RE-ENTER PASSWORD:', there is a text input field with the placeholder text 'Re-Enter Password *'. At the bottom, there are two buttons: a blue 'Register' button and a green 'Login' button, both with white text.

FIGURE 4.3 CANDIDATE REGISTRATION PAGE

Once the candidate successfully logged in the system will display six options. They are,

- i) Browse all jobs
- ii) Browse jobs by location
- iii) Browse jobs by company
- iv) Browse jobs by designation
- v) Upload resume
- vi) Logout

Figure 4.4 shows candidate options page

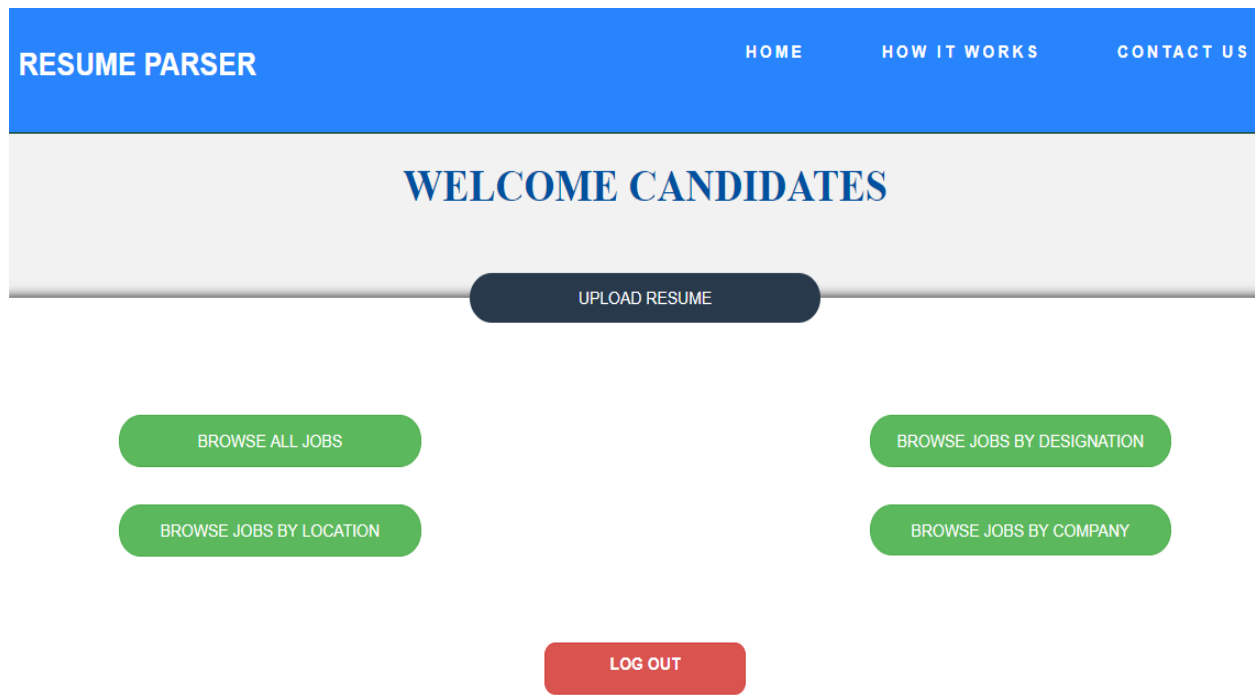


FIGURE 4.4 CANDIDATE OPTIONS

Once the candidate presses the browse jobs button the appropriate jobs will be displayed as shown in Figure 4.5.

LIST OF ALL JOBS

S.No	Company name	Job Location	Skillset	Designation	Experience	Education
1	dell	Coimbatore	C C++ Java	Software Developer	0	B.Sc
2	google	Coimbatore	C C++	Software Developer	0	B.Sc
3	dell	Chennai	C C++	Software Developer	0	B.Sc
4	dell	Chennai	C C++ Java	Software Developer	0	B.Sc
5	dell	Karnataka	C C++ Java	Java Developer	0	B.Sc

FIGURE 4.5 BROWSE ALL JOBS

The candidate can upload the resume many times. If a new resume uploaded then the old resume will be automatically deleted. Figure 4.6 shows the resume upload page.

RESUME PARSER

HOMEHOW IT WORKSCONTACT US

WELCOME CANDIDATES

Please Upload your Resume

Select file(s) to upload

Choose Files

No file chosen

SUBMIT

FIGURE 4.6 RESUME UPLOAD PAGE

Once Choose Files button is pressed it will be redirected to file explorer window and the candidate can select the resume as shown in Figure 4.7

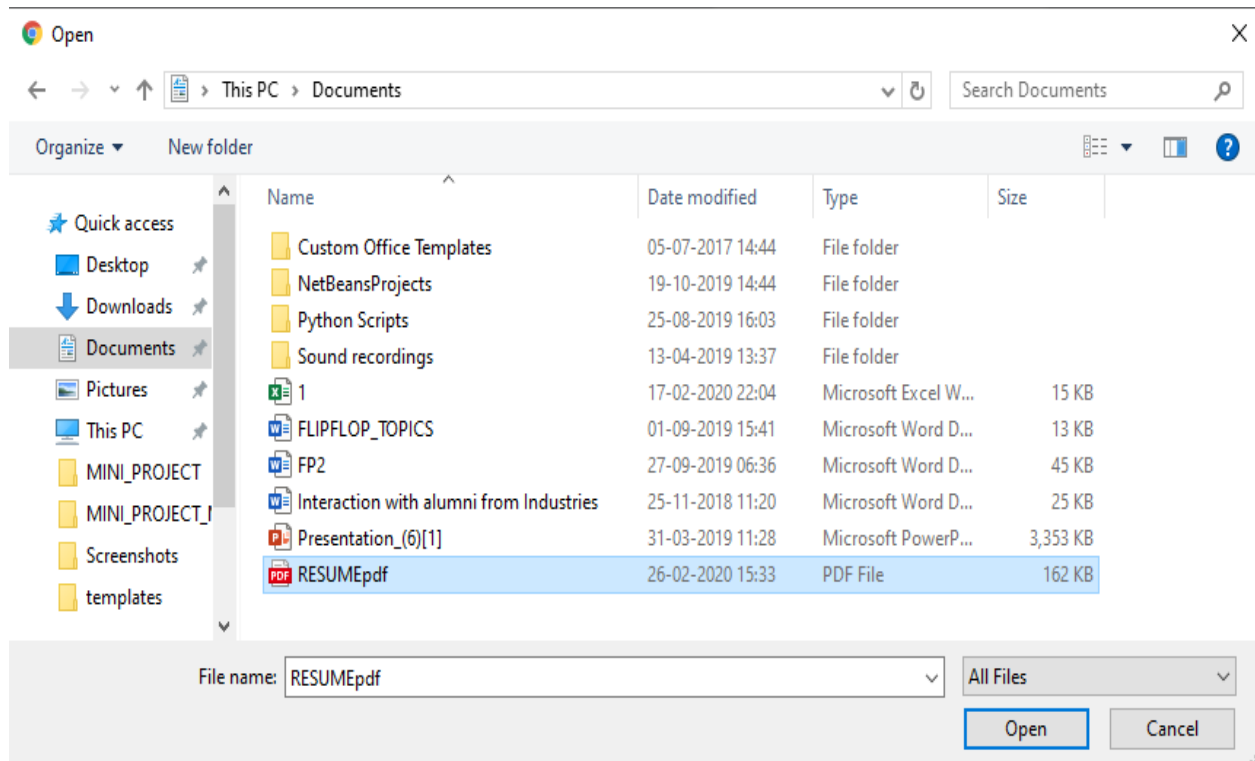


FIGURE 4.7 FILE EXPLORER WINDOW

If the user selects recruiter option in the home page it will be redirected to recruiter login page as shown in Figure 4.8.

Recuriters Login

USER NAME:

PASSWORD:

FIGURE 4.8 RECRUITER LOGIN PAGE

If the recruiter is not registered with the system then they have to register by providing the details as shown in Figure 4.9. Each company will have a unique company name. One company can register only once. The email address and the password must be matched to the predefined format. All the fields are mandatory.

Recuriter Register

COMPANY NAME:

EMAIL:

PASSWORD:

RE-ENTER PASSWORD:

FIGURE 4.9 RECRUITER REGISTRATION PAGE

Once the recruiter is successfully logged in he/she can post the job by providing the necessary details as shown in Figure 4.10.

Company Name	dell
Designation	Software Developer
Choose Job Location ▼	
choose Experience ▼	
Choose Education ▼	
SKILLS	Additional Skills
SEARCH ELIGIBLE CANDIDATES	

FIGURE 4.10 JOB DESCRIPTION PAGE

Once the job is posted the recruiter can view the eligible candidates who are all matched with the specified job description.

Next chapter will discuss the various testing techniques done to ensure the quality and performance of the project.

CHAPTER 5

TESTING

The testing process focuses on the logical structures of the software assuring that all the statements have been tested and also on the functional components by conducting tests to uncover errors. This process also ensures that the defined input will produce the accurate results.

Testing is an important part of Software Development Life Cycle (SDLC). The amount of testing required is related to the size and complexity of the application. In this chapter, various testing strategies adopted in testing are explained.

5.1 UNIT TESTING

Unit testing concentrates on each unit of the software implemented in the source code. The developer at the module level carries this out and this testing is white-box oriented. The module interface is tested to ensure that information properly flows into and out of the program unit under test. The scripts that are intended for testing workflow are ensured that they work similar to the way the manual test works. It's by designing test suites as the application should work and cross check it during execution and no false positives or true negatives exist. The error handling routines, custom scripts, clean up scripts, file reading scripts are checked whether they perform the intended functions. The coding standards are checked in terms of standard and meaningful names are employed for variables, functions. Since the naming conventions are mandatory requirement for any framework for that manner.

In this project each module is tested separately to ensure it is perfectly working as per the needs. The list of modules testes independently is described below.

- ConvertPdfToText
- ConvertDocxToText
- Tokenization
- StopWordRenoval
- NamedEntityExtraction
- ConvertToJSON

In this project each web page is written using HTML and CSS to satisfy the requirements of candidates and recruiters. Each web page is tested separately by giving its own absolute path in the chrome web browser. The following are the list of web pages tested separately.

- Home.html
- How_it_works.html
- Upload.html
- PostJobDescription.html
- FilterCandidates.html
- RecruiterLogin.html
- RecruiterRegister.html
- DisplayCandidates.html
- BrowseAllJobs.html
- BrosweJobsByLocation.html
- BrowseJobsByCompany.html
- BrowseJobsByDesignation.html

- CandidateRegister.html
- CandidateLogin.html

5.2 INTEGRATION TESTING

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. That is the program is constructed and tested in small segments, which makes it easier to isolate and correct. This testing focuses on the design and construction of the framework. The objective is to take unit tested modules and build a program structure that has been dictated by the design. Both integration and testing is done in this phase. Units are combined one at a time and testing, till entire framework gets integrated.

In this project all the above mentioned web pages are linked properly using hyperlinks in html. The entire application is loaded in the chrome web browser and tested whether all links are linked to a particular html file or not.

5.3 VALIDATION TESTING

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfils its intended use when deployed on appropriate environment. At the culmination of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected, and a final series of software tests- validation testing begins. The function or performance characteristics conform to specification and are accepted. A deviation from the specification is undiscovered and a deficiency list is

created. The deficiency list is properly implemented as the next phase of implementation.

In this project all input validations are done using regular expressions in python. Email address should contain proper prefix with '@' symbol and followed by a domain name. Password must include alphanumeric characters such as alphabets, numbers and special characters. Candidate must upload the resume. Without selecting a proper resume it will not move to next web page. Resume must be in acceptable formats such as pdf, txt or docx. Job location, experience and education must be selected from the predefined drop down menu. These validations helps the recruiters to match the eligible candidates easily.

5.4 LOAD TESTING

Load testing is a type of non-functional testing. A load test is type of software testing which is conducted to understand the behaviour of the application under a specific expected load. Load testing is performed to determine a system's behaviour under both normal and at peak conditions. Load testing one among the different kinds of performance testing that determines the performance of the system in real time load conditions. It is used to find if the performance of the application is sustainable when it is at the peak of its user load It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation.

In this project 50 candidate's resumes are collected in different formats. Then these resumes are converted into text format and then necessary informations extracted from that. Similarly, 50 recruiters details also loaded into the database and tested successfully.

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENTS

CONCLUSION:

Resume parser successfully implemented to satisfy the needs of candidates and recruiters. The candidate can upload the resumes in multiple formats. All those resumes will be converted into text format and then the system parsed relevant information from that. This project uses JSON for storing the resumes into the database. So, it is very efficient in optimizing the storage space. Once the recruiter uploads the job description it will be searched against the JSON database and the eligible candidates will be reported back to the recruiter. Candidate can change their resumes often. The latest resume submission will be considered as final in this project.

FUTURE ENHANCEMENTS:

The resumes can be extracted from different social networking sites including Stack Overflow, LinkedIn, etc. Resumes can be categorized based on the genre (example: Computer science, Management, Sales, etc). The system should be able to find similarities between the resumes. Future work also includes analysing information about the candidate from social networking sites like Face book and Twitter so that it can decide more accurately and authentically whether or not to offer the candidate, a job.

BIBLIOGRAPHY

- [1]. F. Ciravegna, “Adaptive information extraction from text by rule induction and generalisation,” in Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI2001), 2001.
- [2]. A. Chandel, P. Nagesh, and S. Sarawagi, “Efficient batch top-k search for dictionary-based entity recognition,” in Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE), 2006.
- [3]. S. Chakrabarti, Mining the Web: Discovering Knowledge from Hypertext Data. Morgan-Kauffman, 2002
- [4]. M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni, “KnowItNow: Fast, scalable information extraction from the web,” in Conference on Human Language Technologies (HLT/EMNLP), 2005.
- [5]. M. J. Cafarella and O. Etzioni, “A search engine for natural language applications,” in WWW, pp. 442–452, 2005.
- [6]. [https://www.ijircce.com/upload/2016/april/218_ Intellig ent.pdf](https://www.ijircce.com/upload/2016/april/218_Intellig ent.pdf)
- [7]. [https://www.tutorialspoint.com/compiler_design/images /token_passing.jpg](https://www.tutorialspoint.com/compiler_design/images/token_passing.jpg)