

UNIVERSITY PARTNER



Hireability: Information Extraction from Resume

Student Id : 2050196
Student Name : Prasuna Pokharel

Declaration Sheet

Award Title: *BSc (Hons) Computer Science*

Declaration Sheet

(Presented in partial fulfillment of the assessment requirements for the above award.)

This work or any part thereof has not previously been presented in any form to the University or to any other institutional body whether for assessment or for other purposes. Save for any express acknowledgements, references and/or bibliographies cited in the work. I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

It is acknowledged that the author of any project work shall own the copyright. However, by submitting such copyright work for assessment, the author grants to the University a perpetual royalty-free license to do all or any of those things referred to in section 16(1) of the Copyright Designs and Patents Act 1988. (viz. to copy work; to issue copies to the public; to perform or show or play the work in public; to broadcast the work or to make an adaptation of the work).

Student Name: Prasuna Pokharel

Student ID Number: 2050196

Signature:



Date: 27th April 2022

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 in identifying 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, 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.

1. Introduction.....	
1.1. Problem Statement.....	
1.2. Implementation of module.....	
1.2.1. Reason behind implementing NLP.....	
1.2.2. Work Flow of NLP.....	
1.3. Aims.....	
1.4. Objectives.....	
1.5. Artefacts.....	
1.6. Scope and Limitations of the project.....	
1.7. Structure of the report.....	
1.8. Academic Questions.....	
2. Literature review.....	
2.1. End-to-End Resume Parsing and Finding Candidates for a Job Description using.....	
BRET.....	
2.2. Information Extraction from Free-Form CV Documents in Multiple Languages.....	
2.3. Resume Information Extraction with a Novel Text Block Segmentation Algorithm.....	
2.4. Information Extraction from Resume Documents in PDF Formats.....	
2.5. Study of Information Extraction in Resume.....	
2.6. Automatic Extraction of Usable Information from Unstructured Resumes to Aid.....	
Search.....	
2.7. Analysis and Findings.....	
2.8. Conclusion of literature review.....	

3. Project Methodology.....	
3.1 Scrum Methodology.....	
3.1.1. Reason behind Selection.....	
3.2. Initial Gantt Chart.....	
3.3 Additional info.....	
3.3.1 Resources.....	
3.3.2 Client.....	
4. Technology and Tools.....	
5. Artefact Design.....	
5.1. System Requirement Specification.....	
5.2. Functional Decomposition Diagram.....	
5.3. Legend.....	
5.4. Admin Management System (AMS).....	
6. Diagrams.....	
6.1. Activity Diagram.....	
6.2. Use Case Diagram.....	
6.3. Sequence Diagram.....	
6.4. Wireframe Designs.....	
6.4.1. Home Page.....	
6.4.2. Login Page.....	
6.4.3. Parsing Page.....	
6.4.4. Information Extraction Page.....	
6.4.5. FAQ Page.....	

6.4.6. Terms and Condition Page.....	
6.5. Design Process.....	
6.6. Development.....	
6.7. Product Backlog.....	
6.8. Accessibility.....	
6.9. System Work Flow.....	
6.10. Testing and Evaluation.....	
6.10.1. Accessibility Testing.....	
6.10.2. Black Box Testing.....	
6.11. Module Implementation.....	
6.11.1 Important Necessary Libraries and Module.....	
6.11.2. Data Collection.....	
6.11.3. Data Pre-Processing.....	
6.11.4. Code and its explanation.....	
6.11.5. Output.....	
7. Conclusion.....	
8. Critical Evaluation.....	
8.1. Final Report.....	
8.2. Finding and process.....	
8.3. System.....	
8.4. Future Work.....	
8.5. Self-reflection.....	
9. Evidence of Project Management.....	

9.1. Gantt Chart.....

9.2. Project Management tool.....

10. References.....

Figure 1: Single Column Resume	2
Figure 2: Double Column Resume	2
Figure 3: Free Format Resume (Image).....	3
Figure 4: Generic Workflow of NLP	5
Figure 5: A system diagram showing data transmission and task completion.	11
Figure 6: The Information Extraction System from Free-form CVs: A High-Level Overview	12
Figure 7: Workflow of Information Extraction from Resume	13
Figure 8: Workflow of the System	15
Figure 9: Skillate Software	17
Figure 10: DaXtra Software.....	18
Figure 11: Sprint Backlog	20
Figure 12: Initial Gantt Chart	20
Figure 13: Functional Decomposition Diagram	23
Figure 14: Activity Diagram of admin management system	25
Figure 15 activity diagram of parsing system	26
Figure 16: Use Case Diagram	27
Figure 17: Admin Management System	28
Figure 18: File Upload System	28
Figure 19: Design Part	35
Figure 20: Invert Color	37
Figure 21: Increasing and Decreasing Text size	38
Figure 22: System Work Flow	38
Figure 23: Light House Testing	39
Figure 24: Color Contrast Test	40
Figure 25: Screenshot of Test 1	41
Figure 26: Screenshot of test 2	42
Figure 27: Screenshot of test 3	43
Figure 28: Screenshot of test 3	44
Figure 29: Screenshot of test 4	46
Figure 30: Tokenization.....	50
Figure 31: Stop words	51
Figure 32: Lemmatization.....	51
Figure 33: Lower Casing	52
Figure 34: Extract the text from pdf	53

Figure 35: Extract Text from Doc files	53
Figure 36: Detect the file extension	54
Figure 37: Extract the entities	54
Figure 38: Extract the email	55
Figure 39: Extract Name from Resume	56
Figure 40: Extract Mobile Number	57
Figure 41: Extract Skills from the resume	57
Figure 42: Extract Technical Skills	58
Figure 43: Extract Education and Year	59
Figure 44: Extracting address	59
Figure 45: Extract Experience	60
Figure 46: Output	60
Figure 47: Final Gantt Chart.....	81
Figure 48: Project Management	81

1. Introduction

Daily, corporate firms and recruiting agencies have to process a large number of resumes. Working with a large volume of text data is usually time consuming and stressful. Data gathered from different resumes can be in a various form, including .pdf, .docx, single column resumes, double-column resumes, free formats, and so on. And these formats might not be suitable for the particular application. So, questions may arise in our mind that, what is resume parsing? The process of converting the unstructured form (.pdf/ .docx / .jpeg etc.) of resume data into a structured format is known as resume parsing.

Subsequently, converting a resume into prepared text or structured information makes studying, analyzing, and comprehending easier. As a result, many organizations and institutions depend on Information Extraction, where unstructured data and vital information are extracted and converted to make information more readable and organized data forms. The completion of this task takes a long time for humans. So, it is necessary to develop an automated intelligent system that can extract all relevant information to determine whether an applicant is suitable for a particular job profile (Kurama, 2021).

The foundation of this project is a resume automation system. Concerning the project, there will be an admin panel at first, into which the administrator must initially log in. Following that, there will be a section for uploading gathered CVs, which the admin will manage. After that, Regex, NLTK, and Spacy's phrase matcher will extract necessary information such as Name, Address, Email, phone number, Nationality, Skills [Hard Skills, Soft Skills], Education, Experience, Experience Year, Languages etc. in. json format. Finally, the extracted information or dump JSON file is saved in the database by admin if necessary.

Below is the wireframe of Unstructured Data (Single-Column, Double-Column, Free Format)

Elizabeth Fraser

EDUCATION

WORK EXPERIENCE

RESEARCH

SKILLS

HOBBIES

LANGUAGES

Figure 1: Single Column Resume

Elizabeth Fraser

SUMMARY

WORK EXPERIENCE

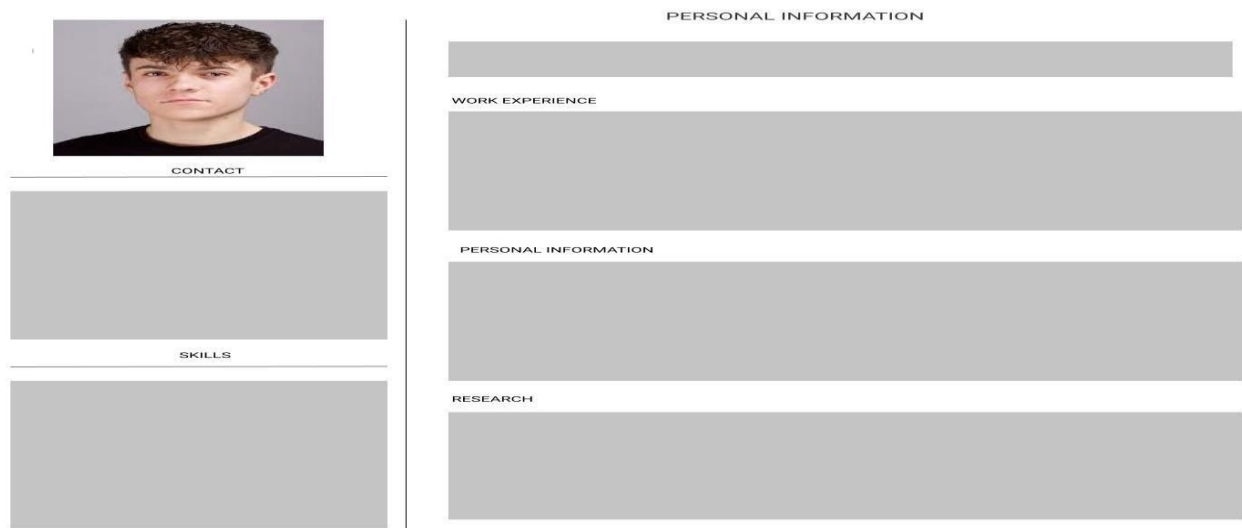
SKILLS

LANGUAGES

EDUCATION

RESEARCH

Figure 2: Double Column Resume



The image shows a resume template with a profile picture of a young man. Below the picture is a section labeled 'CONTACT' with a large gray placeholder box. To the right of the 'CONTACT' section is a section labeled 'PERSONAL INFORMATION' with a large gray placeholder box. Below the 'CONTACT' section is a section labeled 'SKILLS' with a large gray placeholder box. To the right of the 'PERSONAL INFORMATION' section is a section labeled 'PERSONAL INFORMATION' with a large gray placeholder box. Below the 'SKILLS' section is a section labeled 'RESEARCH' with a large gray placeholder box.

Figure 3: Free Format Resume (Image)

Structured Data (Name, Address, Email, phone_number, Skills [Hard Skills, Soft Skills], Education, Experience, Languages)

Name	
Address	
Email	
phone_number	
Skills [Hard Skills, Soft Skills]	
Education	
Experience	

The structured data comes in. json format which makes the HR department easy to read the resume.

1.1. Problem Statement

People with diverse personalities come from a variety of fields and backgrounds. In the same way, their resume writing style varies. They've worked on a variety of projects, and each of them has unique writing style. As a result, each resume is unique.

Some people work in the human resources department. They will have to go through hundreds of resumes on the internet. Executives summarize the resume, enter specific information into their database, and then call the applicant for job counselling after obtaining the resume. An executive spent around 10-15 minutes on each resume, summarizing it and entering the information into the database. This project will help in the automation of this procedure.

1.2. Implementation of module

By collecting the relevant information from the resume, a **natural language processing** technique is employed to create a Hire ability system. Different NLP libraries, such as NLTK and Spacy, are utilized for extraction. Natural language networks are usually taught as **unsupervised** techniques, which implies that no previous tagging or labeling is done before the model is trained.

Natural Language Processing is a branch of AI that integrates languages and computer science to study the patterns and architecture of language. It helps to develop intelligent systems that are based on machine learning and NLP algorithms which can read, analyze, and extract meaning from text and voice.

1.2.1. Reason behind implementing NLP

The key reason I chose NLP for Hireability is that it can handle massive volumes of data in seconds or minutes that would take days or weeks to analyze manually. NLP technologies can instantly scale up or down to match demand, allowing us to have as much or as little processing capacity as per requirement.

Aside from that, humans are prone to errors or may have personal biases that might distort the findings while conducting repeated jobs like examining resumes one by one and other textual data. In this case, NLP-powered solutions can be taught to understand company's need and requirements in only a few steps. So, once they're up and going, they perform better in terms of accuracy (Wolff, 2020).

1.2.2. Work Flow of NLP

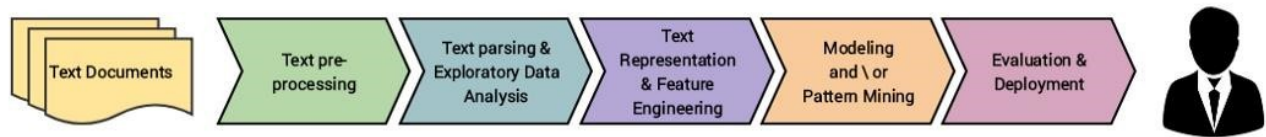


Figure 4: Generic Workflow of NLP

Natural language processing comprises a wide variety of methods for analyzing human language, based on machine learning techniques as well as rules-based and computational approaches. Tokenization, lemmatization and stemming, parsing, part-of-speech tagging, language identification are some basic NLP tasks. NLP tasks, in general, break downs the language into smaller, essential components, attempt to comprehend links between the pieces, and then examine how the components combine to form meaning (C, 2020).

The procedures given above represent the standard workflow for an NLP task.

The initial stage is generally text wrangling and pre-processing on the collection of documents. Then there's parsing and some basic exploratory data analysis. The next stage is to represent text using word embeddings and then do feature engineering. Following that, we must select a model based on whether we are dealing with a supervised or unsupervised learning scenario. The final step of any ML workflow is model testing and deployment. The early processes of text pre-processing and EDA are covered in ([6.11.3. Data PreProcessing](#))

1.3. Aims

- To take the help of the cutting-edge and latest NLP technology to enhance their business processes.
- To extract the required information about candidates without having to go through each resume manually.
- To replace slow and expensive human processing of resumes with extremely fast and cost-effective software.

1.4. Objectives

- For information extraction, NLP model will be configured and reconfigured.
- A system for uploading resumes will be developed.
- The unstructured data will be transformed into structured form.

1.5. Artefacts

A subsystem is a fundamental component of a larger system. Dividing the system into subsystems will aid in the development of a more advanced version. The following are the three key artefacts that will be developed for this project in order to create a functional model.

1. Admin management system

This sub-system is one of the most secretive aspects of the project. As it is a carefully guarded part of the project, only the administrator can access using the admin email ID and password. After the admin logged in, there'll be given the option to upload the resume. Following the submission of the resume, there will have two sub-models: resume parsing and information extraction.

The following is a brief list of how this artefact will be created and the tools required to do it.

Methodology: SCRUM **Design Tool:** Figma

Logo Design: Figma

Frontend: HTML, CSS, JavaScript

Backend: Flask/Django

IDE: VsCode/ Sublime text **Version**

Control: Git

Browser: Google Chrome or Mozilla (Alternate)

2. File upload and Parsing system

This system is at the center of the overall project. In this sub system the admin will upload the resume and go for further processing. The parser subsystem will use different libraries to transform the submitted unstructured resume to a structured format. This will make it much easier to examine, analyze, and grasp the data.

The following is a brief list of how this artefact will be created and the tools required to do it.

Methodology: SCRUM

Data Requirement: dataset collected from Kaggle

Data analysis and evaluation: Jupyter Notebook/VsCode

IDE: VsCode/Sublime text

Version Control: Git

Browser: Google Chrome or Mozilla

Build Tools: Pip

3. Information extraction system

After parsing the data, we employ the NLTK, Spacy phrase matchers, regex to extract the essential information. The required extracted data will be saved in JSON format.

Finally, the extracted data or dump JSON file is stored in a MySQL database for further usage if it is needed.

The following is a brief list of how this artefact will be created and the tools required to do it.

Methodology: SCRUM **Design Tool:** Figma **Programming Language:** Python 3 Technologies

Libraries: NLTK, Spacy

Data analysis and evaluation: Jupyter Notebook/VsCode

IDE: VsCode

Version Control: Git **Build Tools:** Pip

1.6. Scope and Limitations of the project

A purpose system can help in resolving the challenge of obtaining useful information from a resume in a structured format. By resolving this issue, recruiters will be able to save hours each day by eliminating manual resume screening. Bias in hiring is still prevalent, thus this method may also address the bias hiring process and strengthen a non-bias policy.

Purpose system is not able to solve the complex network issue such as:

- Excessive web traffic can significantly slow down or restrict access to a website entirely. This occurs when the server receives more file requests than it can handle.
- Latency issues

1.7. Structure of the report

Introduction: It includes the description of purpose system, its types, problem domain and project as the solution.

Literature Review: It includes the related project descriptions, algorithms used and the diagram.

Answer to academic question: It includes answers to the academic question with the proof.

Project methodology: It includes the methods taken to complete the project.

Technology and Tools: It includes the different types of tools and technology used while building the system.

Artefact Designs: It includes the major artefacts of the system with its description and diagrams.

Conclusion: It includes what has been discovered while writing and developing the project.

References and the Bibliography: It includes all the sources from where the project is developed and report is written.

1.8. Academic Questions

- What would be the best approach to parse and extract useful information from a resume in an effective manner using AI?

Explanation of academic question

Hiring an employee often causes complications if they hire based on a resume since they have to check through all of the candidates' data such as education, talents, job experiences, and so on. As a result, people may begin to wonder how they would get through all of the resumes. They are always looking for a solution to the problem. Machine learning or artificial intelligence (AI) could be used to extract information from resumes as a solution.

2. Literature review

Agencies and different high-level companies have to deal with an extreme number of new jobs seeking employees with different resumes. However, looking after those large numbers of text data and filtering out the needed candidates is a burden on the brain and more time consuming. Therefore, the essence of this literature review is on studying resumes in different formats such as single-column resumes, double-column resumes with extension.pdf,.docx, and how the suggested Information Extraction System converts that unstructured information into structured layout through Parsing.

Accordingly, this review also helps to understand and apply several in-use and well recognized algorithms currently being used in industries to reduce human labor. Depending upon the Company's preference to hire employees, the Extraction System will manage the gathered information with more readability and organized data forms. Furthermore, the analysis of various machine learning algorithms and natural language processing techniques would be equally carried out along with their proper implementation and evaluation. The reviews from multiple research publications and journals are included below.

2.1. End-to-End Resume Parsing and Finding Candidates for a Job Description using BRET

For evaluating the suitable candidates for various job openings in consideration to their compatibility, the use of deep learning-based system has provided the end-to-end solution for people seeking employment (Bhatia, et al., 2019). They may be done in two stages: first, by developing a resume parser that extracts all necessary information from candidate resumes, and second, by conducting ranking using BERT phrase pair classification. The BERT algorithm, which was used to classify sentence pairs, predicted the measure of the correlation between both the job description and candidate profiles with 72.77 percent accuracy. In this research, they explored the possibility of building a standard parser for all types of resumes and determined that it was impossible to do so without losing information in all situations, resulting in the unfair rejection of specific candidates' resumes. Instead, they used LinkedIn-style resumes to scan without losing any information. They wanted to investigate the vision-based site segmentation technique in the future in order to improve structural comprehension of resumes. In addition, the study also creates a firm foundation and a feasibility study that can lead to advancements in deep learning and language representation being used in the hiring process. The system diagram below represents the data flow and the completed task.

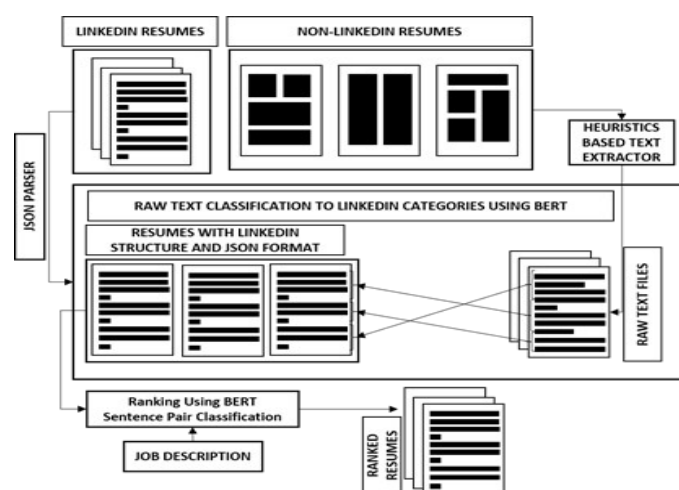


Figure 5: A system diagram showing data transmission and task completion.

2.2. Information Extraction from Free-Form CV Documents in Multiple Languages

The use of two natural language processing algorithms to extract important data from an unstructured multilingual CV has provided a solution for selecting relevant document parts and the similar particular information at the low hierarchy level (VUKADIN, et al., 2021). It uses a machine learning method in NLP to obtain a high level of extraction accuracy. In their practice, authors used the transformer architecture and its application of the encoder part in the BERT language model. A dual model was developed to extract both section and item level information from a CV document. A self-assessment model of skill proficiency categorizes the retrieved Skills section from the dual model. The authors claim that they have solved the CV parsing challenge by building an NLP system. The recently introduced tokens [NEW LINE] and [SKILL] are shown to have trained to perform as expected. In the future, adding additional CVs in other languages to the dataset would improve performance in other languages.

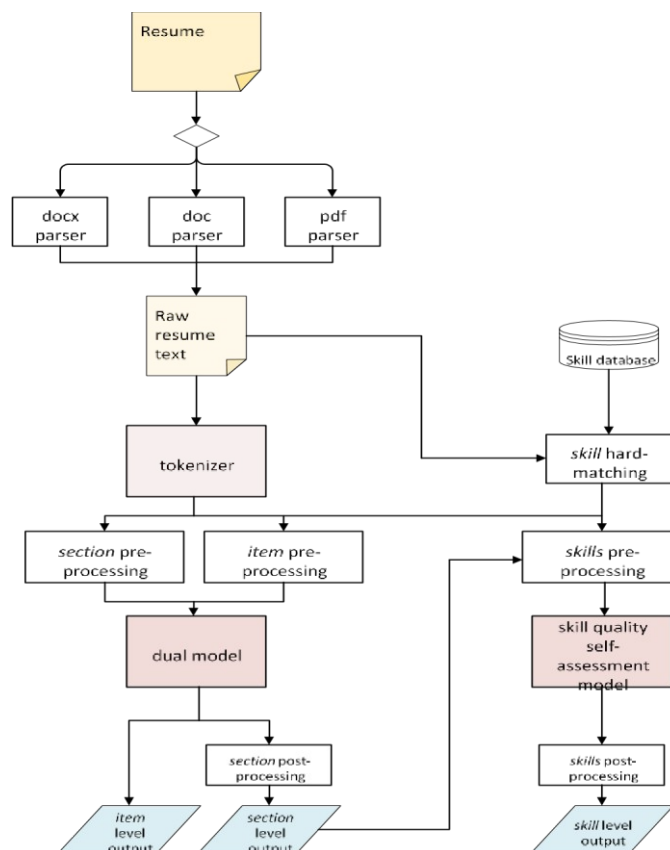


Figure 6: The Information Extraction System from Free-form CVs: A High-Level Overview

2.3. Resume Information Extraction with a Novel Text Block Segmentation Algorithm

Recruitment has evolved rapidly in the last decade, from traditional job fairs to web-based recruitment platforms. As a result, (Wang & Zu, 2019) presented a resume parsing pipeline that uses neural network-based classifiers and distributed embeddings from beginning to end. The pipeline eliminates the laborious process of manually creating several handcrafted elements. The proposed system combines text block segmentation with resume fact identification using position-wise line information and integrated word representations and named entity recognition using multiple sequence labeling classifiers inside labeled text blocks. The ablation experiment was conducted by eliminating the CNN layer from BLSTMCNN's-CRF and comparing various word embeddings testifying. Personal information, job experience, education, project experience, publications, and professional abilities are the six main areas they have focused on to normalize the resume parsing process. Finally, the authors created an online resume parser based on the proposed resume information extraction method, and it turned out that the system works effectively in practice. In the future, they hope to improve the functionalities of online resume parsers by embracing the ontology notion.

2.4. Information Extraction from Resume Documents in PDF Formats

(Chen, et al., 2016) focuses on the problem of extracting data from PDF-format resumes and proposes a hierarchical extraction technique. Resume papers break a page into blocks using heuristic criteria, categorize each block using a Conditional Random Field (CRF) model, and approach the detailed information extraction problem as a sequence labeling issue. According to the authors, the layout-based features have shown to be especially beneficial for semi structured information extraction challenges, increasing the average F1score by more than 20% in testing. In comparison to HTML resumes, PDF

resumes usually include more detailed information. They want to experiment with various page segmentation techniques in the future to improve their understanding of the document's layout and content. The report's method is explained below.

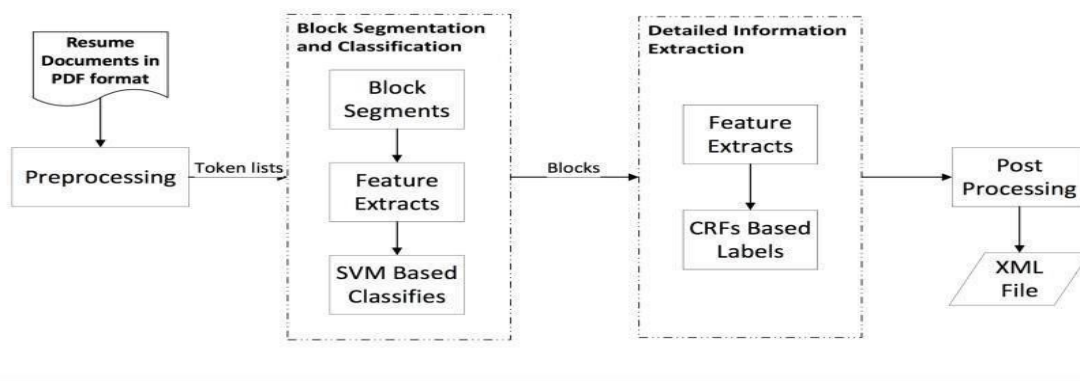


Figure 7: Workflow of Information Extraction from Resume

2.5. Study of Information Extraction in Resume

(Nguyen, et al., 2018) proposes Text Segmentation, Rule-based Named Entity Recognition, Deep Neural Network Find Name Entities, and Text Normalization approaches that automatically retrieve and process multiple resumes formats. To label the sequence in the resume and then extract Name Entities from the labeled line, the author defined the Deep Learning model as a mix of Convolutional Neural Networks, Bidirectional Long Short-Term Memory, and Conditional Random Field. The developer collected promising findings with over 81 percent F1 for NER when experimenting on a medium-sized collection of CV information and compared this model to other systems. However, there are some flaws in this system that might be addressed. The quantity of data to train the model is insufficient. Thus, the more data, the more accurate the model will be. In addition, the model's calculations necessitate the use of sophisticated computers. The author wishes to improve the hardware systems to improve performance in the future.

2.6. Automatic Extraction of Usable Information from Unstructured Resumes to Aid Search

(Kopparapu, 2015) proposes a natural language processing (NLP) system that focuses on automated information extraction from resume to facilitate speedy resume search and management for structured and unstructured resumes. According to them, it is a two-pass method; in the first pass, the resume is divided into a group of successive labeled blocks that indicate the broad data that the block includes. Then, in the second pass, information is retrieved in detail. They applied a variety of heuristics and pattern matching algorithms for the extraction procedure. According to the results of experiments conducted on many resumes, the suggested system can grip a wide range of resumes in various document formats with an accuracy of 91% and a recall of 88%. In the future, they expect to enhance the accuracy with which a CV is picked for high-skilled employment. The data flow and completed tasks are represented in the system diagram below.

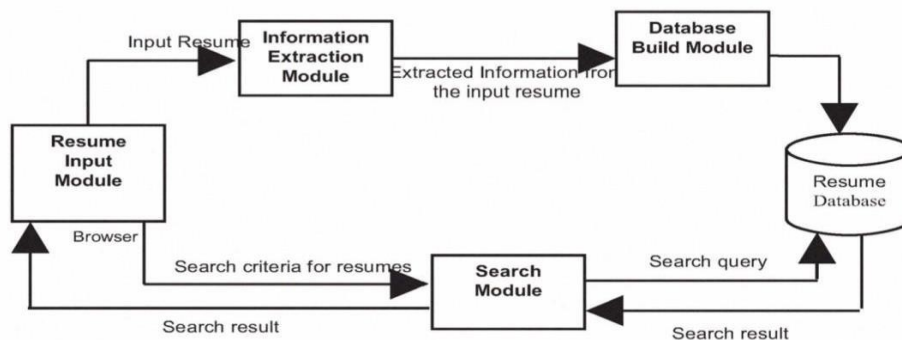


Figure 8: Workflow of the System

2.7. Analysis and Findings

The research and execution of several algorithms to extract the legitimate number of applicants based on their resumes has been raised following the detailed analysis of the proposed literature review. Based on the studies and research papers, it has been seen that the implementation of various algorithms like Bidirectional Encoder Representations

from Transformers (Bert) Model, Neural Network, NLP have been fruitful for information extraction. BERT is a language representation pre-training approach that involves training a general-purpose "language understanding" model on a considerable text corpus, such as Wikipedia, and then using that model to downstream NLP tasks. Because it is profoundly bidirectional and the first unsupervised system for pre-training NLP, the BERT technique beats RNN, LSTM, and Bi-LSTM results. BERT was trained unsupervised with just a plain text corpus, which is noteworthy because a massive quantity of simple text corpus is significant because a large amount of primary text data is publicly accessible on the web in different types of languages.

Even though the research makes extensive use of different approaches, extracting information without losing candidate information from resumes is seen to be a challenging task that requires complex analysis. However, compared to other methodologies described in the papers, the NLP method is simple to adopt and provides meaningful results. It focuses on allowing computers to understand natural language content in a human-like way and develop intelligent systems capable of comprehending, evaluating, and retrieving information from the text. We can also see the ratio of correct predictions made against the size of the test data. The recall and f1 scores are likewise good in terms of overall comparison. Its capability also astounds the researcher since it effectively extracts relevant information from unstructured resumes. So, after evaluating all of the data, I also discovered how NLP is used to extract useful information from resumes. On the other hand, Cosine similarity will measure the score based on cv and job description. After performing the cosine, similarity recall is measured. The score based on high probability will quickly help in the job match.

2.8. Conclusion of literature review

Modern science has developed an excellent learning and working environment in the twenty-first century. Likewise, AI-based solutions are gaining worldwide recognition and implementation for solving the challenge of filtering necessary applicants for a company. As a result, different information extraction systems are created in the present time, replacing the traditional and time-consuming one-by-one approach with a simple filtering out of what is required for the Company's post.

The reviewed literature addressed a wide range of information extraction concepts and research-based procedures and their main approaches. The majority of the research

material uses a combination of ML and DL-based methods. After analyzing all of the review findings, it is easier to conclude that extracting meaningful information from resumes using NLP and its different techniques like Regex and Spacy simplifies the recruiting process significantly minimizes time complexity in a larger way

2.10. Competitive analysis

The two applications listed below have been evaluated for doing competitive analysis as well as studying the system and services they deliver.

1. Skillate

In general, Skillate helps in the optimization of the complete recruiting value chain, from job request creation through resume matching and applicant participation. Deep Learning is used by the system to extract information from even the most complex resumes. This application allows users to upload resumes in a various format, including .PDF, DOC, DOCX, and ZIP. The most complicated resumes are processed in 1-3 seconds by the AI enabled parser.

The system is a powerful decision-making tool that makes recruiting simple, quick, and fair. When it comes to the system's drawbacks, the initial login to the home page takes time. There could be features that cater to various areas of the hiring process. The ability to deliver answers in multiple nations and languages is also essential in features that are not available in the system.

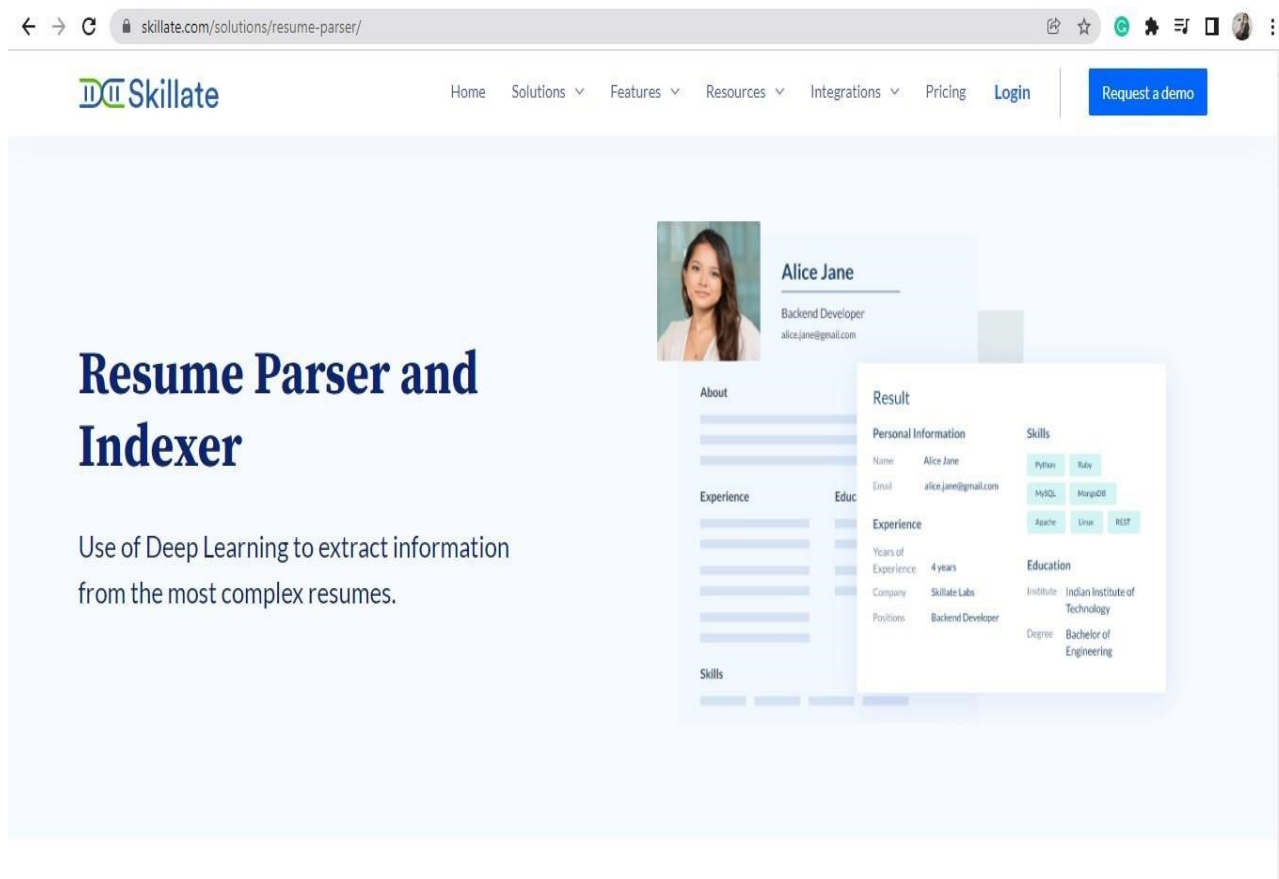


Figure 9: Skillate Software

2. DaXtra

This program continues the same pattern as the last one, quickly, effectively, and accurately extracting information from CVs. The organized data generated from the system follows a well-defined XML or JSON format. Personal and contact information, work history, and education background are all included in CV data. Another component is the Native talents taxonomy, including industry skills, job titles, and academic and professional credentials. The system also supports custom taxonomies. One notable feature of this system is it utilizes multilingual text-understanding technologies. This provides worldwide users with a thorough understanding of the geographical name, address, and number forms across a variety of document types.

When it comes to the system's disadvantages, there are a lot of bells and whistles, which might be confusing for non-technical recruits. The searches aren't very good, and because each search platform uses various search inputs, a single search entry for many platforms isn't very useful.

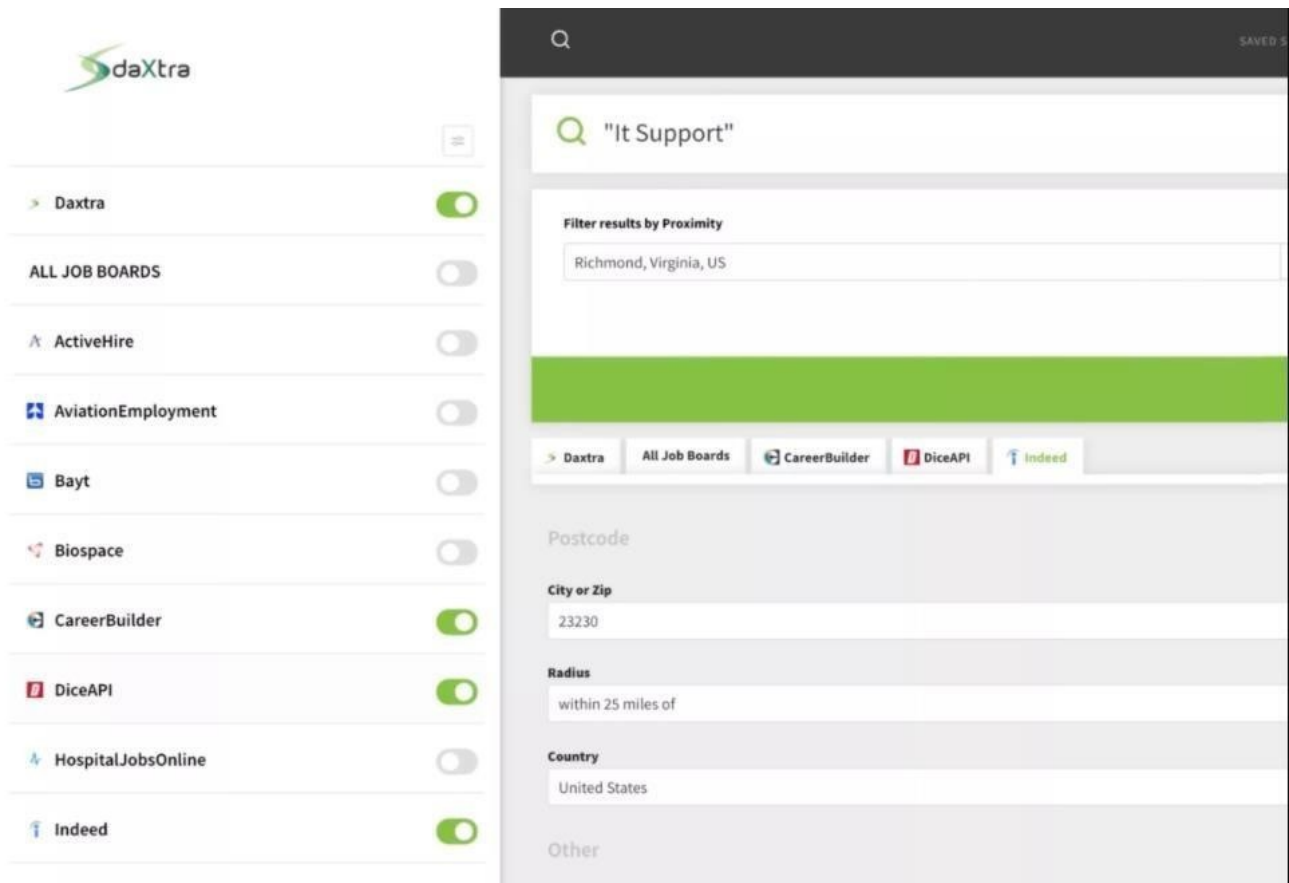


Figure 10: DaXtra Software

3. Project Methodology

A collection of guiding concepts and practices for planning, managing, and executing projects is known as project management methodology. The project management approach that we select affects how work is prioritized and finished. Here, are a few well-known project management approaches in use today.

1. Waterfall methodology
2. Agile methodology
3. Kanban methodology

Agile is a software development process that is continuous and responsive. High levels of communication and cooperation, rapid and convenient reactions to change, adaptable planning, and continuous improvement are all characteristics of Agile development. Scrum, Kanban are just a couple of the techniques and sub-frameworks that have resulted

in the emergence of the agile project management concept. Each framework has its own set of advantages and disadvantages.

3.1 Scrum Methodology

To do Hire ability project as yearlong project I have Chosen scrum. It is the most widely used Agile methods framework due to its ease of use and efficiency.

3.1.1. Reason behind Selection

The major reason I chose this method is:

- It is a transparent approach to project management.
- The tasks and the progress of each task are constantly visible during the sprint.
- It is easy to learn and use.
- It even reduces time taken to get product to market.

Scrum divides work into small cycles known as "sprints," which typically run approximately one to four weeks. For each sprint iteration, work is taken from the backlog. For the course of the sprint, small teams are supervised by a Scrum Master, after which they analyze their work in a "sprint retrospective" and make any required changes before beginning the next sprint. Testing thoroughly, paying close attention to detail, and learning from prior Sprints all contribute to a high level of quality (teamwork, 2021) .

Sprint Backlog					
Task Number	Description of the task	Start date	End date	Sprint Goal	Sprint
1	Project Planning	12/18/2021	12/25/2021	Requirements gathering	1
2	Wireframe	12/25/2021	12/30/2021	Draw wireframe	
3	UML Diagram	1/1/2022	12/30/2021	Draw UML Diagram	
4	UI/UX Design	1/1/2022	1/8/2022	Draw Dessign	
5	Machine Learning Model	1/11/2022	2/10/2022	Build, train and Test MI Model	2
6	Frontend Development	2/11/2022	2/25/2022	development of frontend	
7	Backend Development	2/25/2022	03/12/2022	connecting system using backend coding	
8	Model Integrate	3/13/2022	04/10/2022	Adding build MI model to system	
9	Testing	04/11/2022	4/15/2022	Test the system	
10	Documentation	4/16/2022	4/26/2022	Final Report Writing	

Figure 11: Sprint Backlog

3.2. Initial Gantt Chart



Figure 12: Initial Gantt Chart

3.3 Additional info

3.3.1 Resources

The resources necessary for the entire project's development are listed below.

Hardware Requirements:

- PC with 128 GB of SSD, Python Development Environment and a stable internet connection
- Windows and Linux for testing
- Figma for UI/UX design

Software Requirements:

- Figma for logo design
- Google Chrome or Mozilla (Alternate) for research and debugging
- Visual Studio Code for Python Development
- Git for version control
- Google Drive for the document (Word/ PDF) controls
- Microsoft Visio/draw.io for creating WBS and other diagrams
- Excel for creating Gantt Chart
- MySQL for storing data if needed
- Pip for installing python packages
- NLTK and SpaCy for resume parsing and extraction
- Flask for developing web application
- Jupyter Notebook or Vscod for evaluating and exploring data

3.3.2 Client

Mr. Sarjil Napit would act as the project's supervisor, while Mr. Yogesh Bikram Shah, a teacher at Herald College Kathmandu, would act as the project's reader.

4. Technology and Tools

During the project's development, a variety of tools were used. The list of Software tools used in the project is divided into two categories: hardware and software tools.

1. Programming Language:

For this project, I employed the **Python programming language**. The major reason for using Python is because of its simple syntax, which makes it versatile, simple to use, and quick to develop. Another reason to prefer Python is that it can be used for web development and is ideal for ML and AI applications.

2. Framework:

When I began exploring frameworks, **Django** was at the top of the list. Flask, Laravel, Ruby on Rails, and CakePHP were among the other alternatives, but according to

research and evaluations, the Django framework was more reliable, fast-simple, and, most importantly, perfect for any web application project. It's a Python web frame. As a result, I began to focus on Django for the project's backend by watching videos, reading manuals, and so on.

3. Web Technologies:

For frontend development, **HTML**, **CSS**, **JavaScript**, and Bootstrap were chosen. HTML provides the foundation for a website's structure, which is then improved and updated by CSS and JavaScript. CSS is used to manage the appearance, style, and layout of a page, whereas JavaScript is used to control the behavior of certain components.

4. Design Tool:

The design is the most key component of any project. So, I used **Figma** for the various design sections since it is a free graphic design application that is simple to use. Another advantage of using Figma is we can create, produce, and export files very quickly. I've also used **draw.io**, a free-to-use online tool, for additional diagrams like activity and use-case. It also comes with lots of Templates for creating simple diagrams and flowcharts.

5. IDE:

For the coding side, I used **vscode**. As, VS Coding is a highly capable code editor with built in git integration. It attracted to me since it is simple to use, write code in, add files, and most importantly, it provides syntax highlighting and bracket-matching.

6. Version Control:

When working on any project, using Version Control is essential since it allows individuals to rapidly track changes made to the project and ensures that everyone is working on the most recent version. So, using a variety of commands, I uploaded my project to **Github** and committed it every time I made a change to the code.

7. Browser:

I choose **Google Chrome** as my browser since it is simple to use and is faster. It also loads web pages with a single click and can open many tabs at once.

5. Artefact Design

5.1. System Requirement Specification

5.2. Functional Decomposition Diagram

A chart called FDD is used to represent the separate components of the method and their varied levels relationships to one another. The diagram depicts a technique in a top-down order. It has a rectangle for the function name and a connecting arrow.

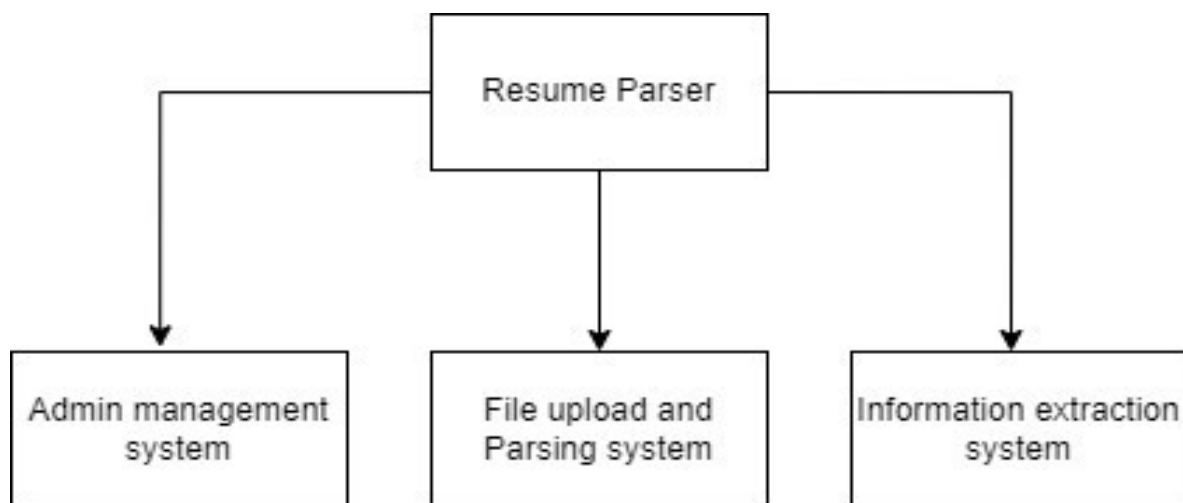
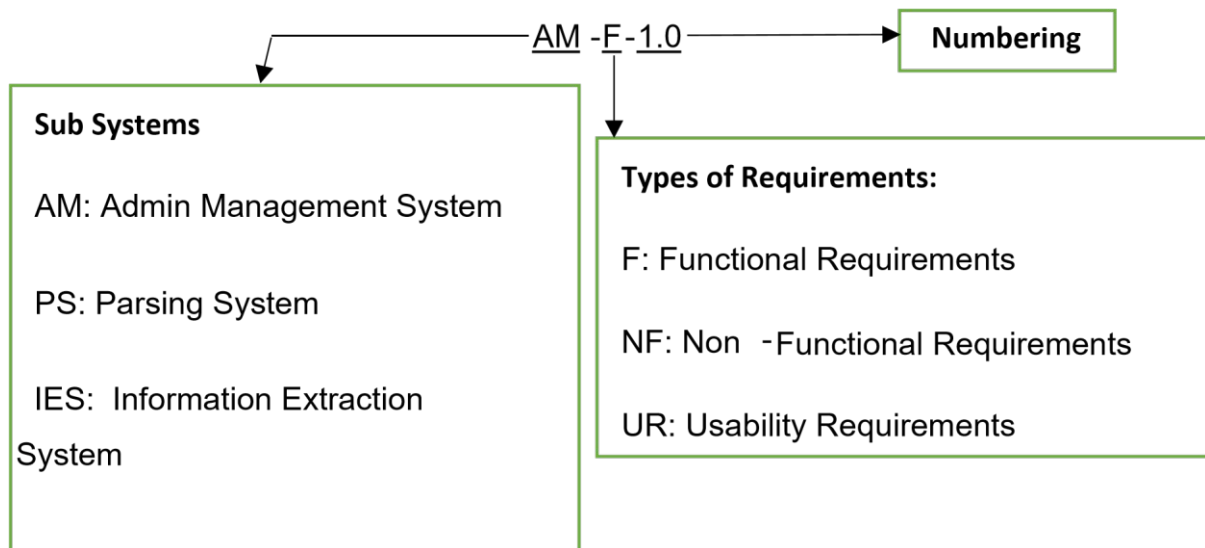


Figure 13: Functional Decomposition Diagram

5.3. Legend



5.4. Admin Management System (AMS)

Requirement- Code	Requirement Specification	MOSCOW
AMS-F-1	The admin should be able to click the Login button on the system.	Must Have
AMS-NF-1.1	Login form should be opened.	Must Have
AMS-F-2	System should allow admin to login.	Must Have
AMS-F-2.1	Email should be valid.	Must Have
AMS-F-2.2	Password should be valid.	Must Have
AMS-F-2.3	For a successful login, the admin must have a valid username and password.	Must Have
AMS-F-2.4	The entered username and password must be validated by the system.	Must Have
AMS-UR-2.1	If the login and password are incorrect or empty, the system should produce an error message.	Must Have

6. Diagrams

6.1. Activity Diagram

The behavior of a system is depicted in an activity diagram. The control flow from a start point to a completion point is represented in an activity diagram, which shows the numerous decision routes that exist while the activity is being performed. The activity diagram for the admin management system is shown below (geeksforgeeks, 2018).

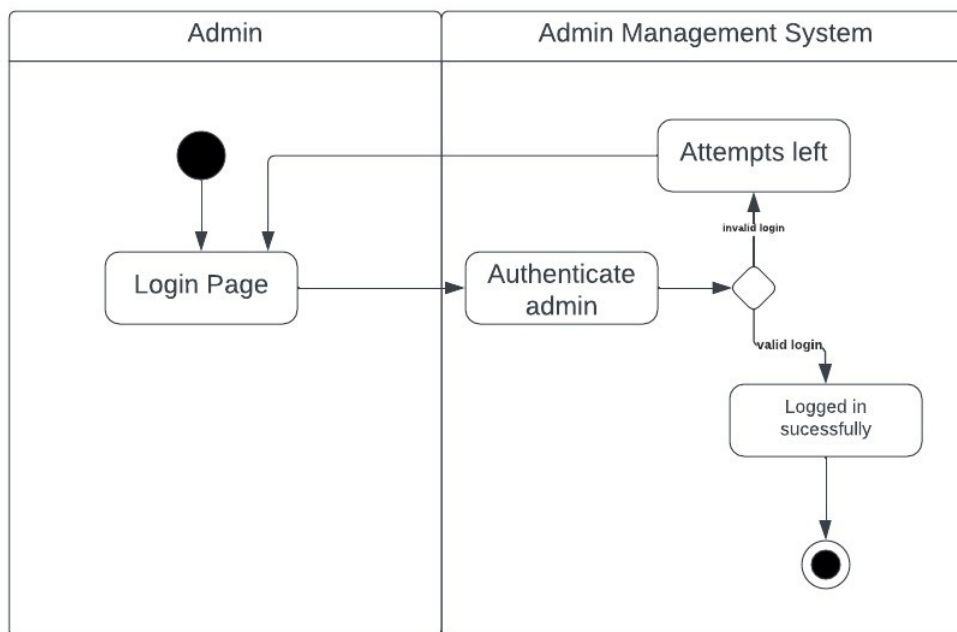


Figure 14: Activity Diagram of admin management system

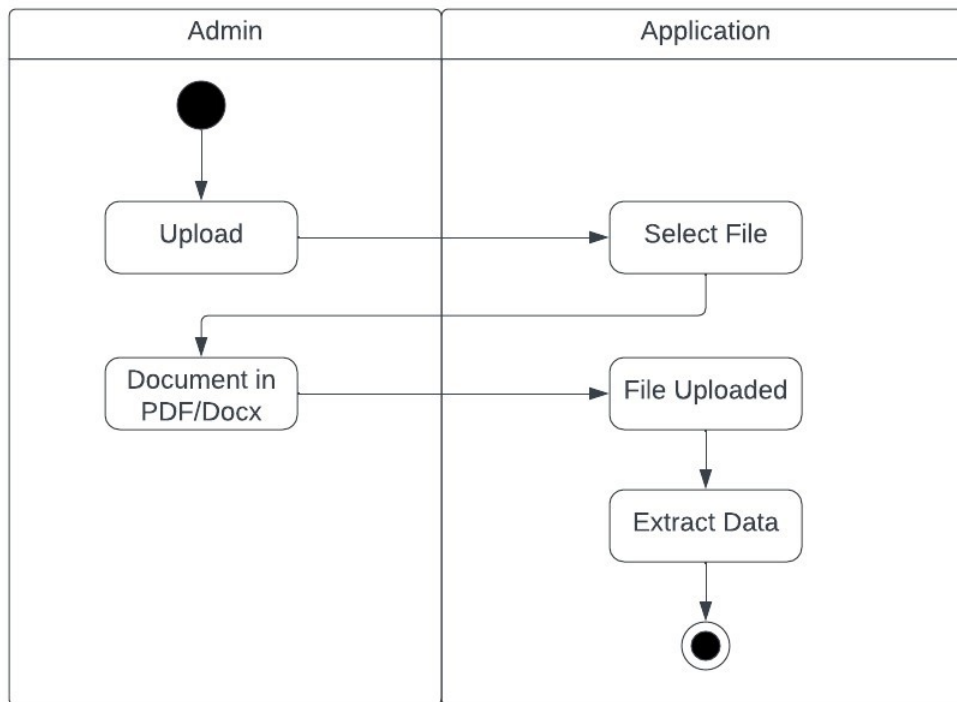


Figure 15 activity diagram of parsing system

6.2. Use Case Diagram

The interactions between the system and its actors are mostly identified with the use case diagram. In use-case diagrams, use cases and actors define what the system does and how the actors interact with it. There is just one user in the diagram below, and that is the admin. Admin can log in, navigate to the upload page, upload their resume, and view the extracted data (ibm, 2021).

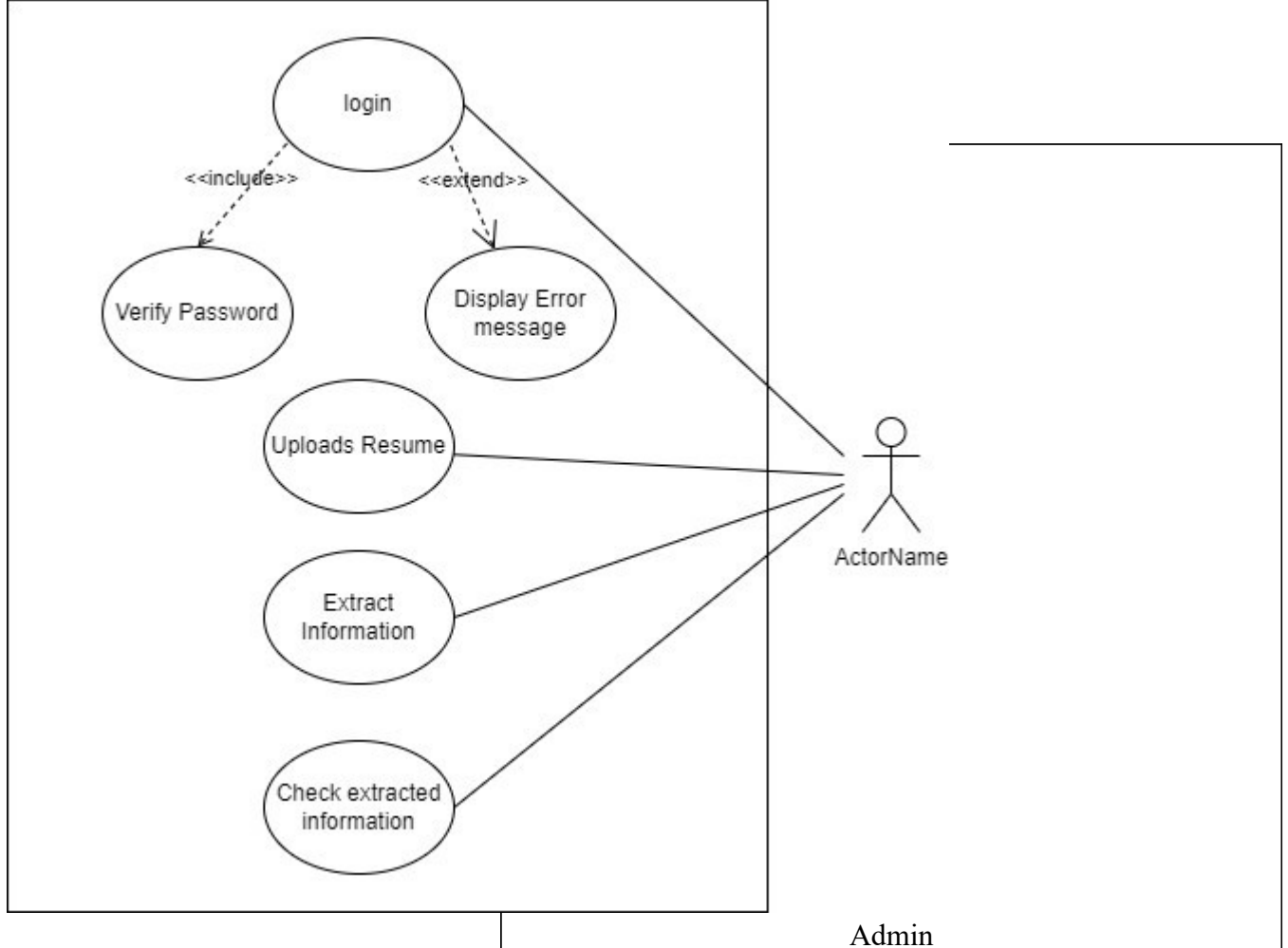


Figure 16: Use Case Diagram

6.3. Sequence Diagram

A sequence diagram displays the order in which items interact in a sequential way. It shows how and in what sequence the components of a system work together. The sequence diagram for the admin management system is shown below (geeksforgeeks, 2018).

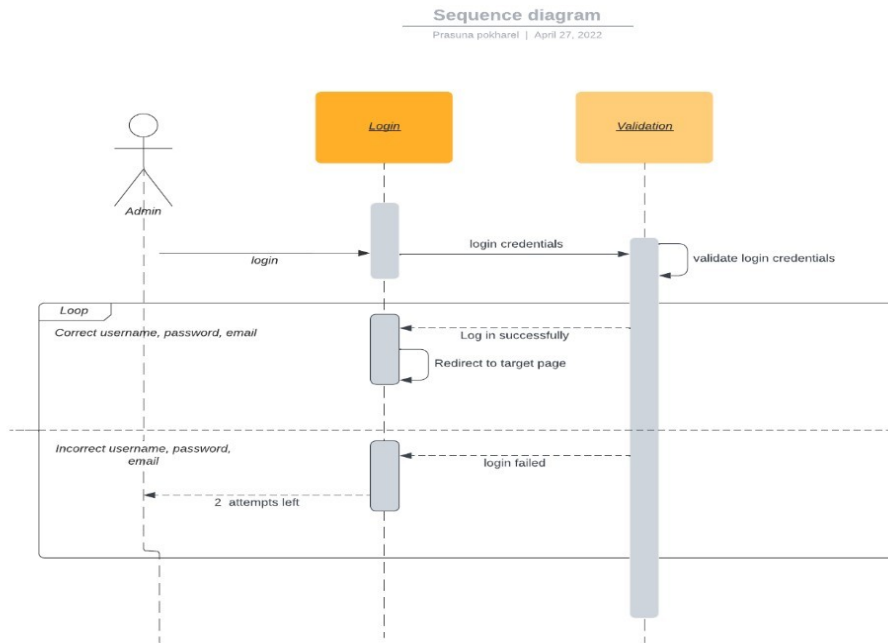


Figure 17: Admin Management System

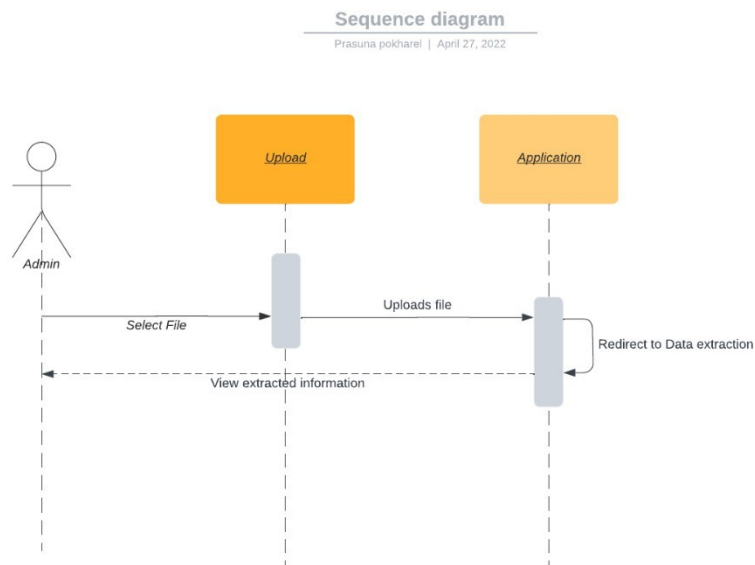


Figure 18: File Upload System

6.4. Wireframe Designs

Many software programs, such as Balsamiq, Figma, and in vision, have been designed to build wireframes. Figma was my tool of choice. According to the website's requirements, I created a wireframe that shows the website's simple layout, which includes a navigation bar, content section, and footer. Wireframes do not contain any of the system's logic. Below is the home page of the application.

6.4.1. Home Page



Below is the login page of application.

6.4.2. Login Page

logo

Icons

Home

Resume Parser

Data Extraction

Login

Ready to upload the resume ?

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore

Login for an account

User name

Enter password

Email Address

Login

Need Help ?

Lorem Ipsum

Lorem Ipsum

Visit Us at:

f

location

o

contact us

e

email

Lorem Ipsum

© 2021 Photo, Created by Prashuna

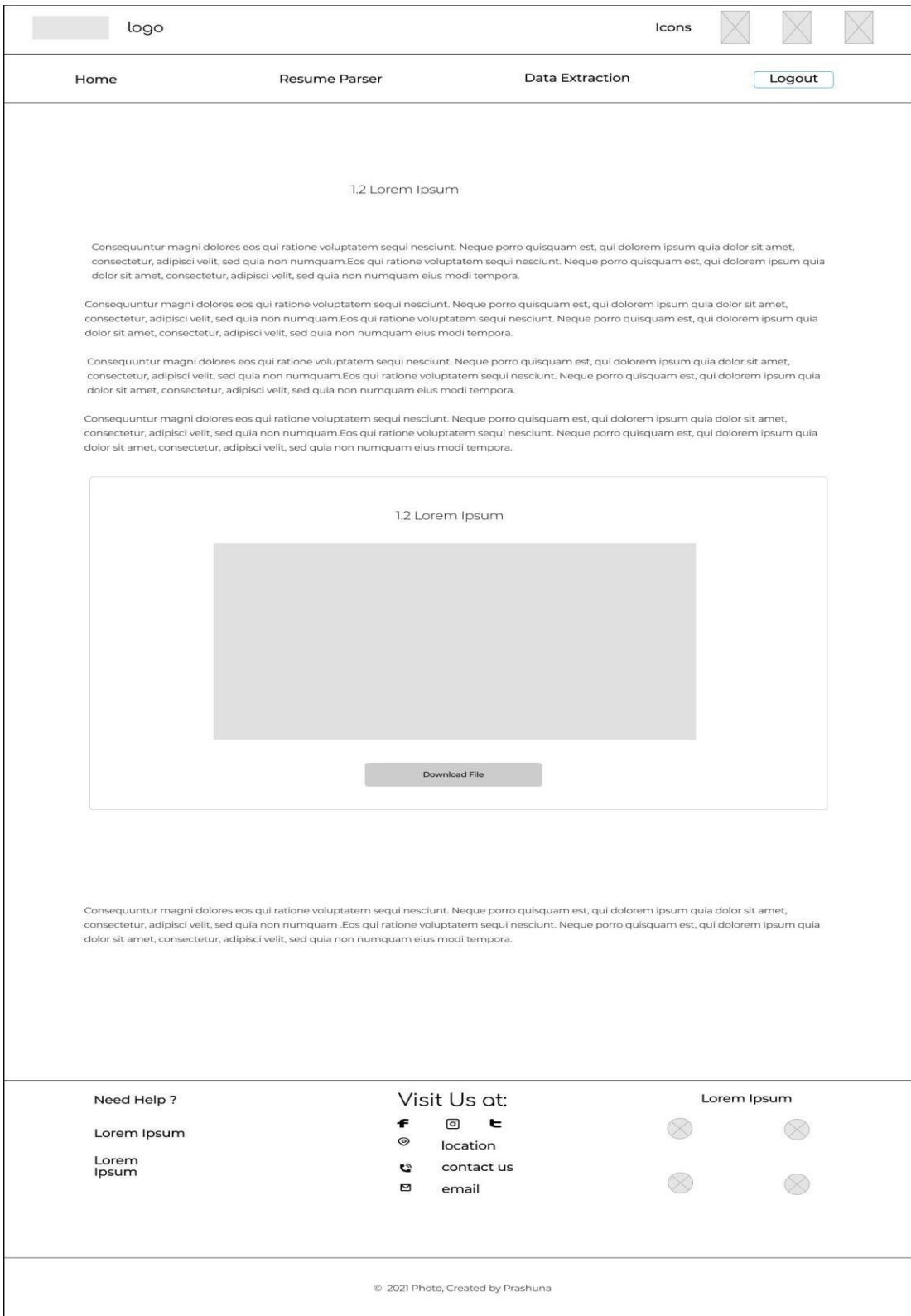
Below is the parsing page of application.

6.4.3. Parsing Page



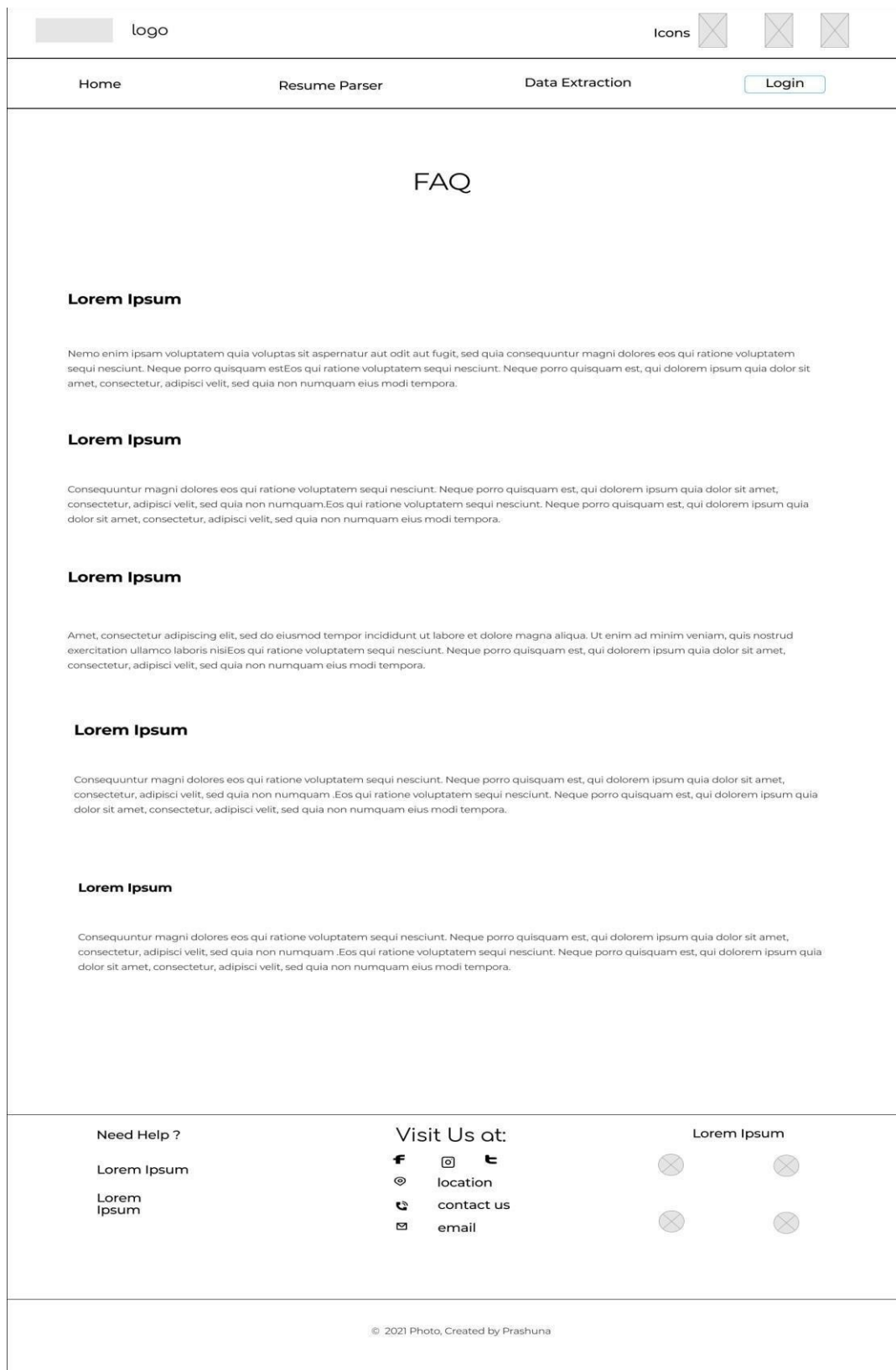
Below is the information Extraction Page of the System.

6.4.4. Information Extraction Page



Below is the FAQ Page of the system.

6.4.5. FAQ Page



Below is the terms and condition page of the system.

6.4.6. Terms and Condition Page



6.5. Design Process

The goal of design is to get as near as possible to the final design. As a result, I built a design that featured all of the pages' important content by picking the appropriate colors,

font, and structure. All of the visual material, including images, icons, and buttons has been created. I reduced the number of unnecessary colors that might cause users to become distracted. The design was made with Figma. My main considerations have been usability, layout, and consistency. The design has provided a general notion of what the upcoming product would look like.

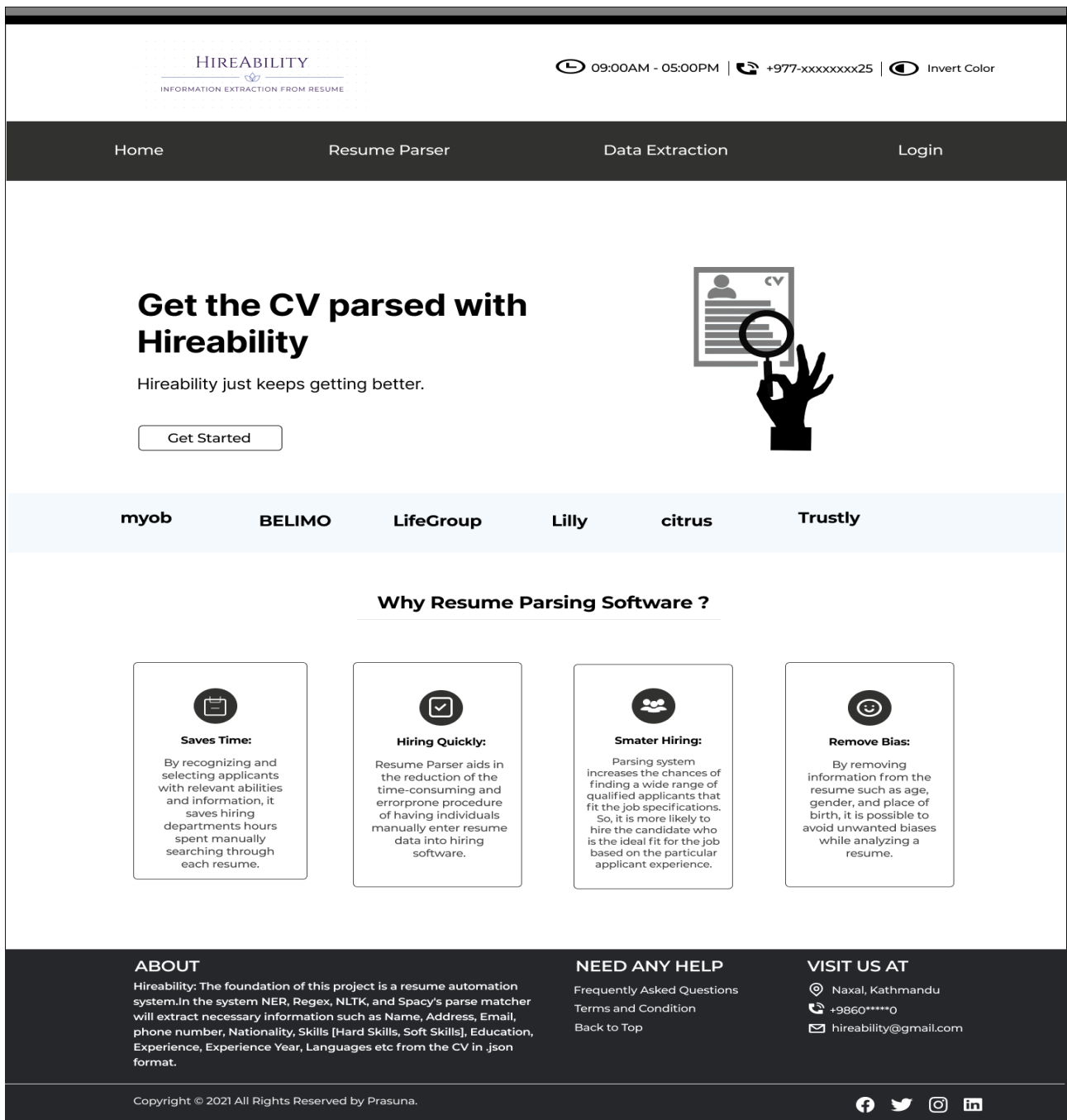


Figure 19: Design Part

6.6. Development

The final and most essential step is development. I built a functional website in development using the graphic components that were created during the wireframe. The home page is created initially, followed by all sub-pages. The static web page elements

modeled during the mock-up were generated. The development languages used were HTML, CSS, JavaScript, Bootstrap, and Flask. The web pages are user-friendly and responsive, with plenty of visuals on screen. To make the website more engaging, various accessibility options have also been included.

6.7. Product Backlog

As a	I want to	So that	Acceptance Criteria	Priority
Admin	Login the system	I can upload the resume	Login credentials	Must Have
Admin	Select files from the Folder	I can parse the resume.	A system to open Folder	Must Have
Admin	See the extracted information.	I can choose the right candidate as per the need.	Parse the resume	Must Have
Admin	Reupload an file from the files	I can reupload any Files to parse the information.	A system to open Folder	Must Have

6.8. Accessibility

The most significant factor in determining a site's worth is its accessibility. It is important that the Internet be accessible to all in order to provide those with disabilities with equal access and opportunities. People with impairments can also benefit from a web that is accessible. Two accessibility symbols have been used on these websites.

a) Invert Color:

Inverting the color helps to boost visibility without raising the screen's brightness when the screen is in direct sunlight. Another significant benefit is that it helps those who are colorblind or suffer from night blindness to use their gadget without becoming anxious.

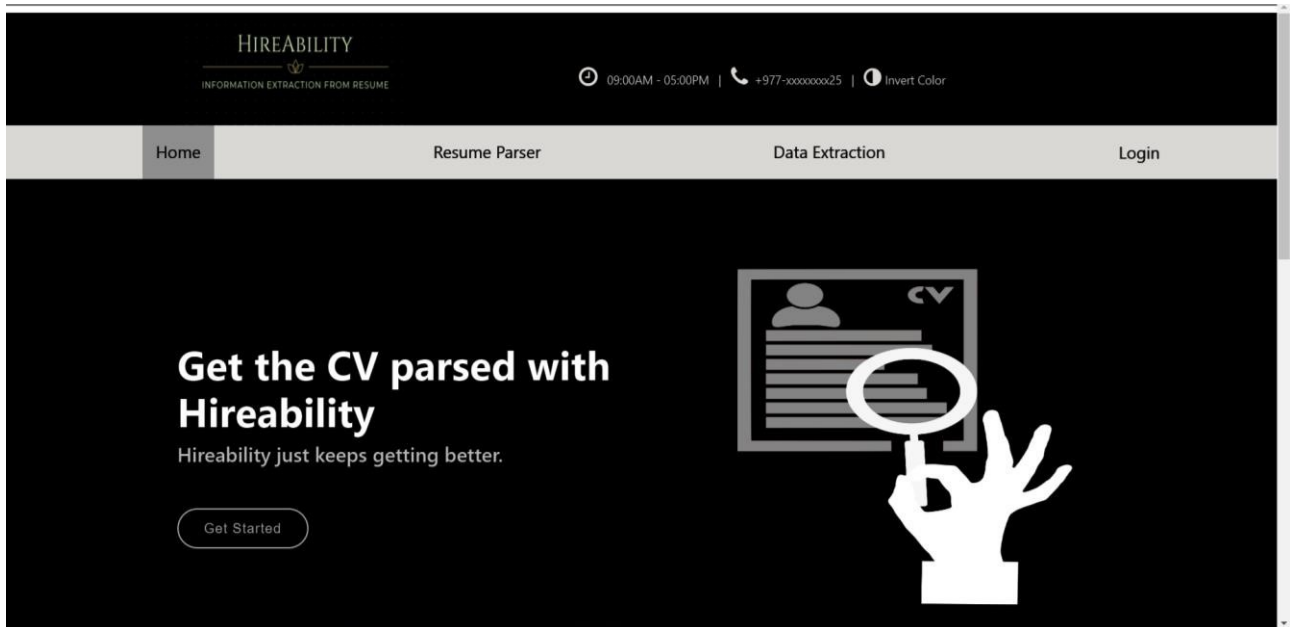


Figure 20: Invert Color

b) Increasing and decreasing the text size:

Many persons with low vision do not use magnifying software and are unaware of how to change the font size on their browser. Larger fonts may also attract to certain persons with cognitive impairments. To use this technique, there are buttons that enable users to gradually increase, decrease, or keep the text size of all the text on the page at the same level.

Terms And Conditions



Figure 21: Increasing and Decreasing Text size

6.9. System Work Flow

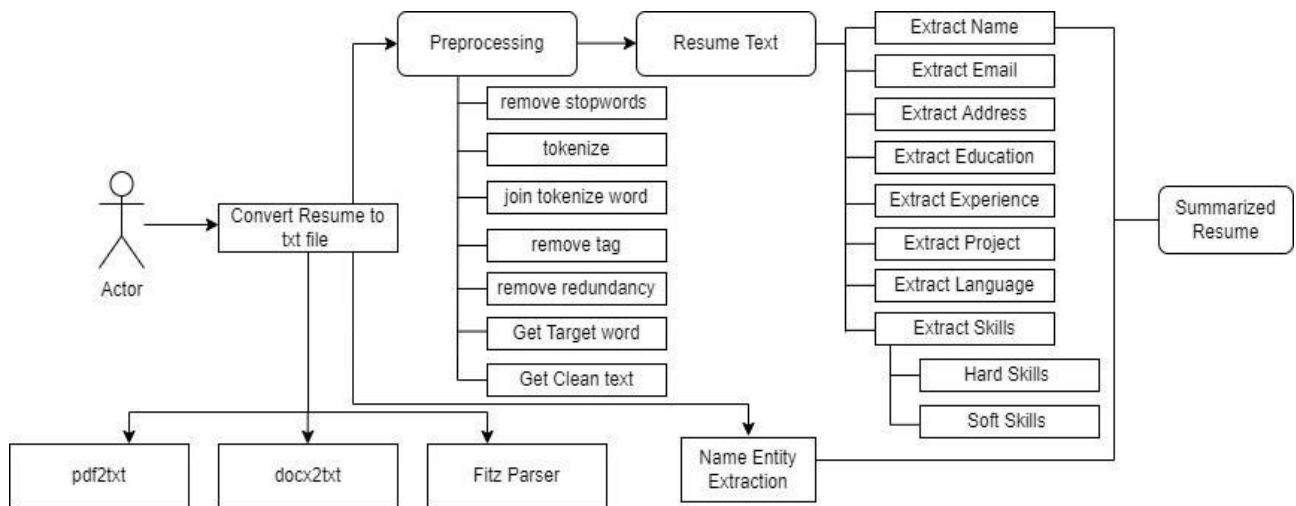


Figure 22: System Work Flow

6.10. Testing and Evaluation

6.10.1. Accessibility Testing

A) Lighthouse

To do my testing, I employed a range of tools. Lighthouse is one of them. Lighthouse is a free open source, automated web page quality optimization tool that is mostly used for accessibility testing. Lighthouse provides the Chrome DevTools Audits panel. To run a study, we must do the following:

- First, we'll need to download Google Chrome for Desktop, and then we'll need to open Chrome DevTools.
- We'll need to choose inspect.

- DevTools provides a list of audit groups after selecting Lighthouse. We may look at many options based on our preferences, but accessibility is a must.
- Select "Generate Report" from the drop-down menu. After 30 to 60 seconds, Lighthouse provides a page report.

The lighthouse then shows us the error fields, which we can utilize to fix our mistakes.

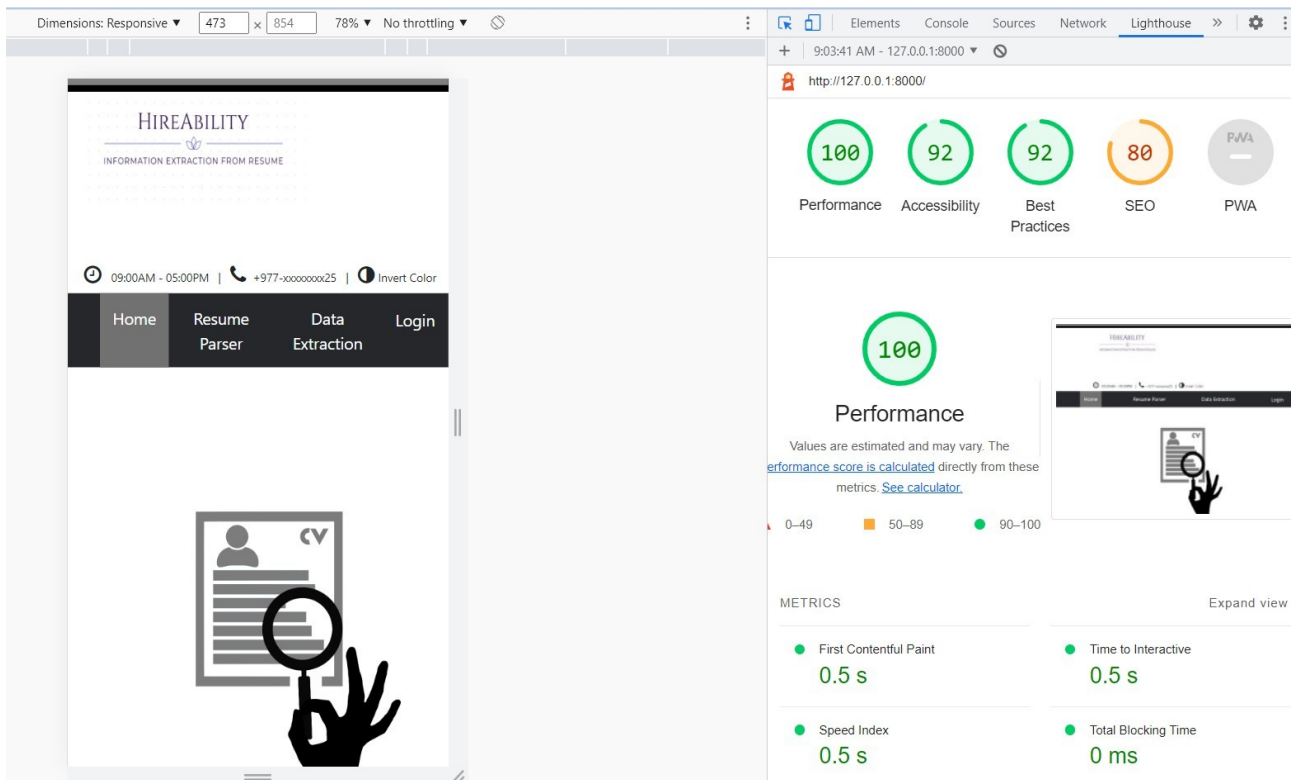


Figure 23: Light House Testing

B) Color Contrast Checker

In Figma, I tested the design element for the second time. Color contrast is a key factor in making the site more accessible to everyone. To make writing simpler to read, the contrast between the text and the context should be significant. So, prior to the development phase, I merely evaluated and fixed all of the color contrast in the design process. To double-check:

- First, we must select plugins.
- Then, choose A11y-Color Contrast Checker.
- Finally, press the check button. And, if there's a color issue, it'll show up on the board, and we'll be able to fix it from there.

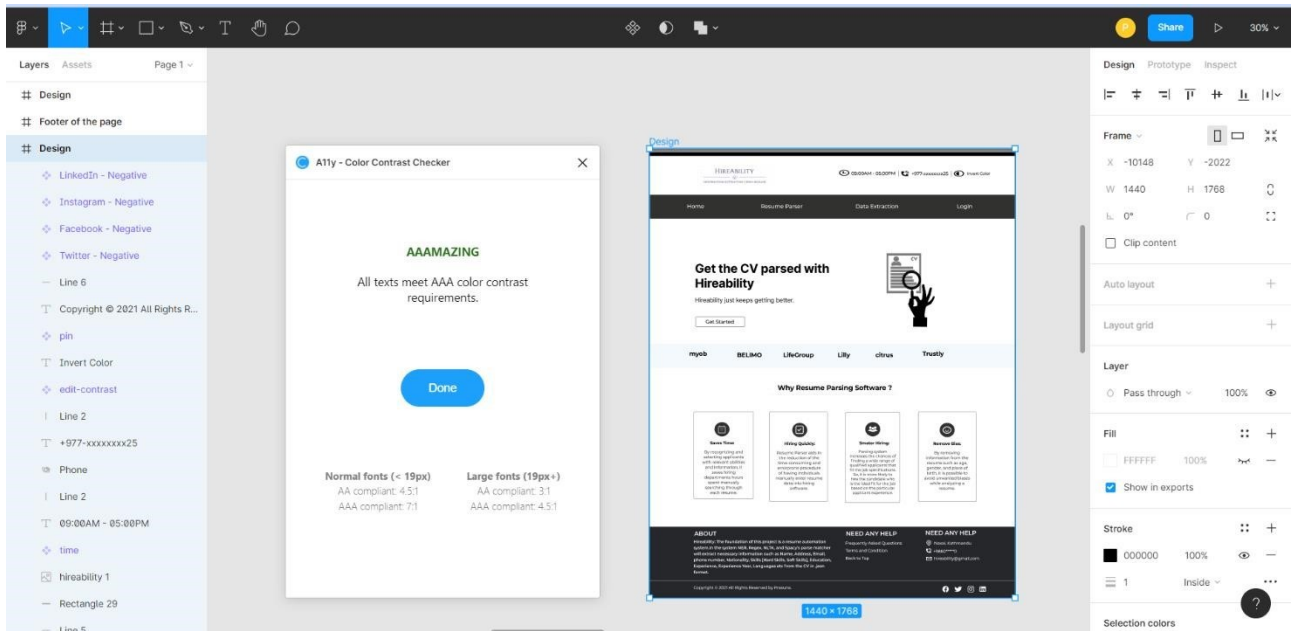


Figure 24: Color Contrast Test

6.10.2. Black Box Testing

Testing must be done once the project has been completed to ensure that the app functions properly. The test runs under various conditions and for each activity developed for the system. The test that will be carried out is known as unit testing. The Black Box's major goal is to determine whether the software is performing as expected in the requirement document and whether it is satisfying user expectations.

a) Login Activity

Test Case No.	1		
Test case Objective	To test whether the application successfully allow login for admin.		
Test Data:	Test Data	Expected Result	Actual Result

	i. Username: Hireability ii. Password: Hireability#123 iii. Email: ps@gmail.com	Login Successfully	Login Successfully
Test Results	The above result indicates that login of admin in the application works well.		

Screenshot of test 1:

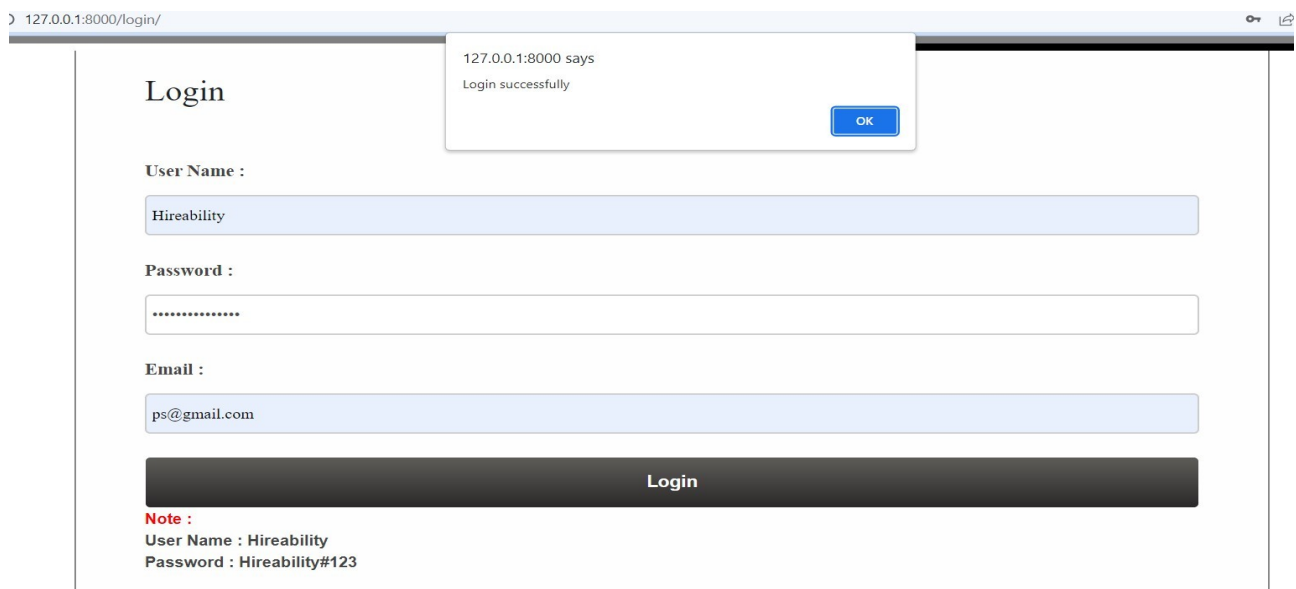


Figure 25: Screenshot of Test 1

Test Case No.	2		
Test case Objective	To test whether the application pass invalid login and left attempt message after invalid input.		
Test Data:	Test Data	Expected Result	Actual Result

	i. Username: Hire ii. Password: Hireability#123 iii. Email: ps@	Left Attempt	Left Attempt
Test Results	The above result indicates that login of admin in the application works well.		

Screenshot of test 2:

127.0.0.1:8000 says
You have left 2 attempt;

OK

Login

User Name :

Password :

Email :

Login

Note :
User Name : Hireability
Password : Hireability#123

Figure 26: Screenshot of test 2

Test Case No.	3
Test case Objective	To test whether the admin is able to upload File from the File explorer application.

Test Data:	Test Data	Expected Result	Actual Result
	Click Upload Button and upload file	Open file explorer and upload file	Open file explorer and upload file
Test Results	The above result indicates that the admin is able click upload button and upload files.		

Screenshot of test 3:

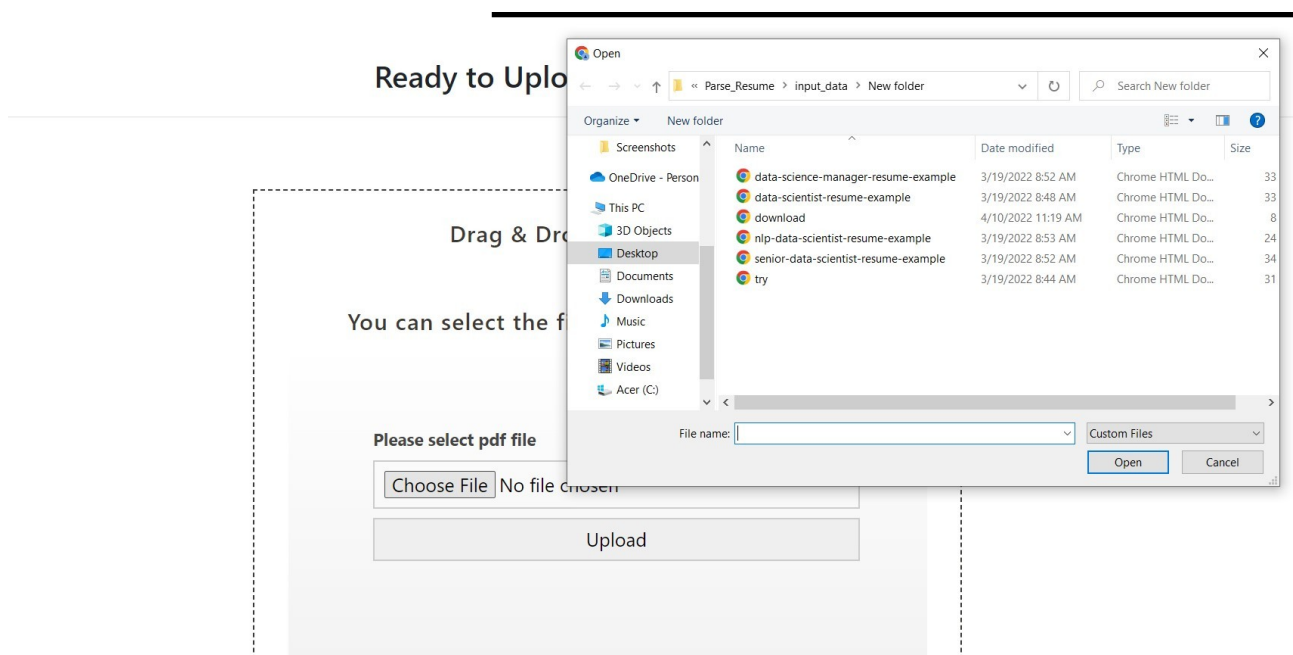


Figure 27: Screenshot of test 3

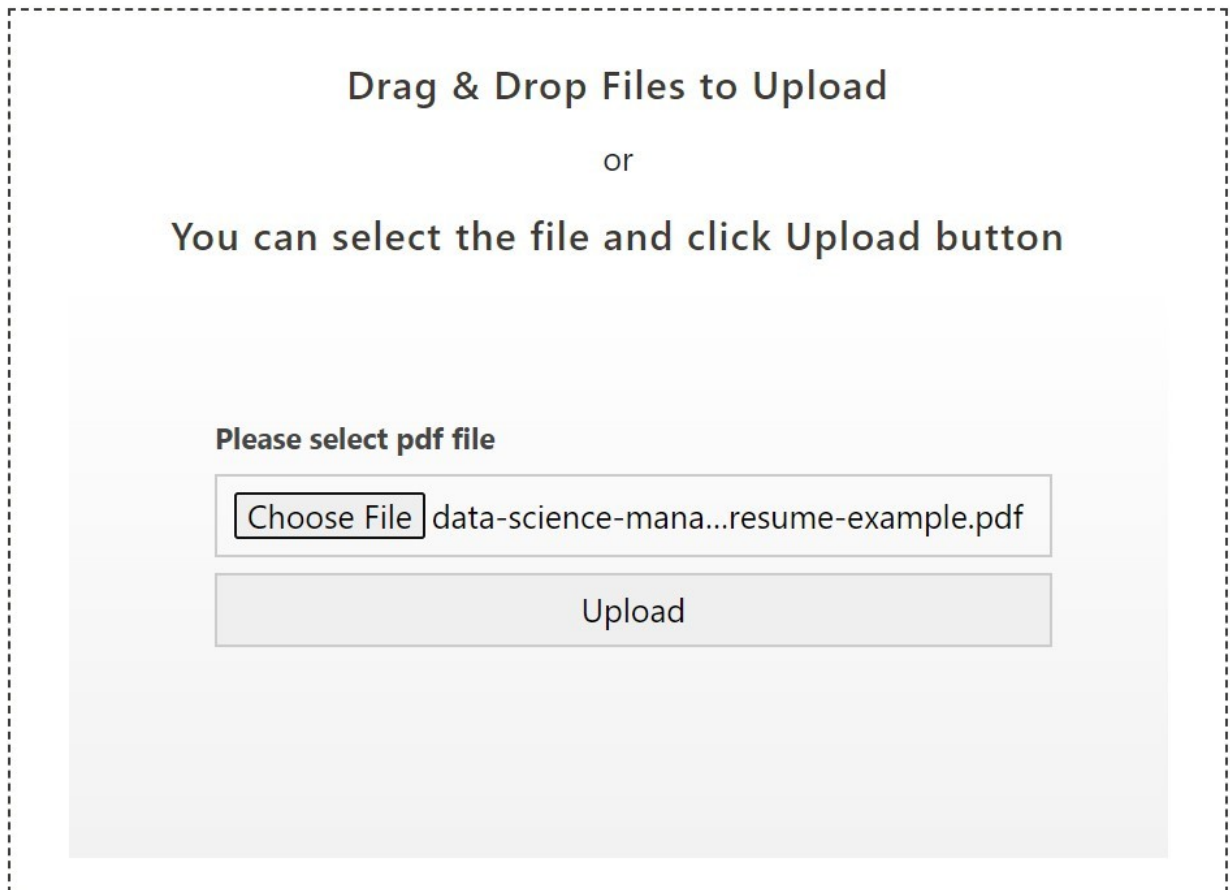


Figure 28: Screenshot of test 3

Test Case No.	4
Test case Objective	To test whether the necessary information from the resume is parsed.

	Test Data	Expected Result	Actual Result
Test Data:	Upload the resume	Show name, email, mobile number, technical skills, soft skills, education, Language, experience, address	Shown name, email, mobile number, technical skills, soft skills, education, Language, experience, address
Test Results	The above result indicates that the necessary information from the resume has been parsed.		

Screenshot of test 4:

KALEB SAXTON
Data Science Manager

✉ ksaxton@email.com
☎ (123) 456-7890
📍 Houston, TX
🌐 LinkedIn

EDUCATION
B.S.
Mathematics and Economics
Rice University
📅 September 2008 - April 2012
📍 Houston, TX

SKILLS
Python (NumPy, Pandas, Scikit-learn, Keras, Flask)
SQL (Redshift, MySQL, Postgres, NoSQL)
Git
Leadership Experience
Recommendation Engines
Customer Segmentation
Propensity Modeling
Productionizing Models

WORK EXPERIENCE
Data Science Manager
Westlake Chemical
📅 February 2018 - current 📍 Houston, TX

- Supervised 3 data scientists and 1 data engineer in developing marketing mix models that led to an ROI improvement of 21% on digital marketing spend over the last 13 months
- Collaborated with the product and marketing teams to identify which pre-client interactions increased chances of conversions, ultimately helping team to increase conversion rates by 32%
- Developed documentation for methodologies and standards for the growing data science team
- Identified duplications across team projects, and reduced instances of this by 92% with increased coordination and accountability

Senior Data Scientist
Energy Transfer Partners
📅 January 2015 - February 2018 📍 Houston, TX

- Worked closely with the product team to build a production prediction engine in Python page, which reduced crude oil profit loss by 16%
- Used random forest models to study well subsets in order to optimize and increase efficiency by 22%
- Compiled and prepared data and reports 2 times per month for stakeholders and functional teams
- Recommended over 100 actionable steps within the first year that improved automation by 7% and profits by 9%

Data Analyst
CenterPoint Energy
📅 April 2012 - January 2015 📍 Houston, TX

- Identified process improvements through client data analysis, which led to a reduced profit loss of 8%
- Owined the reporting using Python and SQL for energy production, saving 30+ hours of manual hours each week
- Built ETL infrastructures and delivered data to Redshift, which increased stakeholder decision making output by 42%
- Created and developed 5 new testing strategies, and created and updated the documentation

Figure 29: Screenshot of test 4

```
Name: KALEB SAXTON
Email: ksaxton@email.com
Mobile_number: (123) 456-7890
Technical_skills: ['Python', 'Numpy', 'Keras', 'Nosql', 'Mysql', 'Flask', 'Pandas']
Soft_skills: ['Leadership']
Education: [['BS', '2008']]
Languages: []
Experience: ['Data Science Manager Westlake Chemical ksaxton @', 'Recommendation Engines Customer Segmentation Propensity Modeling
Productionizing Models Houston']
Address: Houston
```

6.11. Module Implementation

This part contains information about data collecting, how the code is working in the project, different libraries and modules utilized in the project, and so forth.

6.11.1 Important Necessary Libraries and Module

In today's world, everyone wants to complete the tasks as early as possible, but it all relies on the methods they use. There may be a point when we must write many lines of code; if we continue writing code on our own, this will take time. A Library is a group of interconnected modules. It provides code bundles that may be reused in a variety of programs. Modules also play a role in Python. We may define commonly used functions as modules and import them into a code wherever there is a necessity, rather than repeating the same code in multiple programs and making the code complicated. The following are some of the most important libraries and modules:

1. Pandas

Pandas is a Python library which is an open source. It's a tool for analyzing data.

2. OS

In Python, the OS module has methods for creating and deleting folders, retrieving their contents, altering and identifying the current directory, and so on.

3. IO

The `io` module is used to control file-related input and output activities. The advantage of utilizing the `IO` module is that we can extend the capability to allow writing to Unicode data using the classes and methods provided (journaldev, 2022).

4. Utils

Python `Utils` is a collection of simple Python methods and classes that simplify and shorten common patterns.

5. Spacy

`SpaCy` is an open-source package library for advanced natural language processing implemented in Python. `SpaCy` has pre-trained pipelines with tokenization support.

6. pprint

The `pprint` module allows to "pretty-print" any python data structure in a more understandable and well-formatted fashion.

7. Matcher

The `Matcher` assists in the finding of words and phrases by utilizing rules that describe their token properties. After applying the `matcher` on a `Doc`, we can see the matched tokens in context.

8. Multiprocessing

In Python, `multiprocessing` is a built-in package that allows the system to execute many processes at the same time.

9. Warnings

Warning messages are shown using the `warn ()` method from the '`warning`' module.

10. Nltk.corpus

The modules in this package provide functions for reading corpus files in a variety of formats. A corpus is simply a set of texts that are used as input.

11. Re

A RegEx, also known as a Regular Expression, is a string of characters that defines a search pattern. This module's functions allow to see if a given string matches a given regular expression.

12. NLTK

NLTK is a Python toolbox for working with natural language processing. It gives us access to a number of text processing libraries as well as a large number of test datasets.

13. Docx2txt

A pure Python-based library for extracting text from docx files.

14. Constants

The module is just a separate file that contains variables, functions, and other data that is imported into the main file.

15. String

This module will allow to retrieve string constants easily. Cap words is a single utility function in the Python string module.

16. Pdfminer

PDFMiner is a tool that extracts text from PDF files. It focuses completely on gathering and processing text data, unlike other PDF-related tools.

6.11.2. Data Collection

Because training data is not necessary, I only need testing data for my "Hireability" system. So, I got some data from Kaggle's "Hire A Perfect Machine Learning Engineer" and the rest through beam jobs. There were 48 resume templates available in the beam jobs collection.

6.11.3. Data Pre-Processing

Natural Language Processing is a subfield of data science that works with textual data. When it comes to handling the Human language, textual data is one of the most unstructured types of data available. NLP is a technique that operates behind the it, allowing for extensive text preparation prior to any output. Before using the data for analysis in any Machine Learning work, it's critical to analyze the data. To deal with NLP-based problems, a variety of libraries and algorithms are employed. For text cleaning, a regular expression(re) is the most often used library. The next libraries are NLTK (Natural language toolkit) and spacy, which are used to execute natural language tasks like eliminating stopwords.

Preprocessing data is a difficult task. Text preprocessing is done in order to prepare the text data for model creation. It is the initial stage of any NLP project (Deepanshi, 2021). The following are some of the preprocessing steps:

- Removing Stop words
- Lower casing
- Tokenization
- Lemmatization

1. Tokenization

The initial stage in text analysis is tokenization. It enables to determine the text's core components. Tokens are the fundamental units. Tokenization is beneficial since it divides a text into smaller chunks. Internally, spaCy determines if a "." is a punctuation and separates it into tokens, or whether it is part of an abbreviation like as "B.A." and does not separate it. Based on the problem, we may utilize sentence tokenization or word tokenization.

- a. Sentence tokenization: using the `sent_tokenize ()` function, dividing a paragraph into a collection of sentences.
- b. Word tokenization: using the `word_tokenize ()` technique, dividing a statement into a list of words.

```
# word tokenization
word_tokens = nltk.word_tokenize(resume_text)
```

Figure 30: Tokenization

For the Token class, spaCy gives a number of characteristics (chavan, 2020). The following is the most frequently applied attribute:

- `is_stop` determines whether the token is a stop word or not

```
tokens = [token.text for token in nlp_text if not token.is_stop]
```

2. Removing Stop words

To eliminate noise from data, data cleaning is essential in NLP. Stop words are the most frequently repeated words in a text that give no useful information. The NLTK library includes a list of terms that are considered stop words in English. [I, no, nor, me, mine, myself, some, such we, our, you'd, your, he, ours, ourselves, yours, yourself, yourselves, you, you're, you've, you'll, most, other] are only a few of them.

The NLTK library is a popular library for removing stop words, and it eliminates about 180 stop words. For certain difficulties, we can develop a customized set of stop words. Using the add technique, we can easily add any new word to a collection of terms.

```
stop_words = set(stopwords.words('english'))
```

Figure 31: Stop words

3. Lemmatization

The process of reducing inflected forms of a word while verifying that the reduced form matches to the language is known as lemmatization. A lemma is a simplified version or base word. Lemmatization uses a pre-defined dictionary to save word context and verify the word in the dictionary as it decreases. Organizes, organized, and organizing, for example, are all forms of organize. The lemma in this case is organize. The inflection of a word can be used to communicate grammatical categories such as tense (organized vs

organize). Lemmatization is required since it aids in the reduction of a word's inflected forms into a particular element for analysis. It can also assist in text normalization and the avoidance of duplicate words with similar meanings (Agrawal , 2021).

```
wordnet_lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# word tokenization
word_tokens = nltk.word_tokenize(resume_text)

# remove stop words and lemmatize
filtered_sentence = [w for w in word_tokens if not w in stop_words and wordnet_lemmatizer.lemmatize(w) not in stop_words]
sent = nltk.pos_tag(filtered_sentence)
```

Figure 32: Lemmatization

4. Lower casing

When the text is in the same case, a computer can easily read the words since the machine treats lower and upper case differently. Words like Cat and cat, for example, are processed differently by machines. To prevent such issues, we must make the word in the same case, with lower case being the most preferable instance. In python lower () is a function that is mostly used to handle strings. The lower () function accepts no parameters. It converts each capital letter to lowercase to produce lowercased strings from the provided string. If the supplied string has no capital characters, it returns the exact string.

```
token.lower() in link:
    link_types.append(token)

for bi-grams and tri-grams
en in noun_chunks:
    en = token.text.lower().strip()
    token in link:
        link_types.append(token)
[i.capitalize() for i in set([i.lower() for
```

Figure 33: Lower Casing

6.11.4. Code and its explanation

At first, I have imported all the necessary Libraries and modules.

```

import io
import os
import re
import nltk
import spacy
import pandas as pd
import docx2txt
import constants as cs
import string
import re
import os
import utils
import spacy
import pprint
from spacy.matcher import Matcher
import multiprocessing as mp
import warnings
warnings.filterwarnings('ignore')

```

The first step after importing the libraries is to read the resume. As resumes do not have a set file type, they can be in any format, including .pdf, .doc, and .docx. As a result, our key priority is to read the resume and turn it into plain text. I used pdfminer and doc2text Python modules for this. Text can be extracted from .pdf, .doc, and .docx files using these modules.

a) Extract the text from pdf

```

def extract_text_from_pdf(pdf_path):

    with open(pdf_path, 'rb') as fh:
        # iterate over all pages of PDF document
        for page in PDFPage.get_pages(fh,
                                      caching=True,
                                      check_extractable=True):
            # creating a resource manager
            resource_manager = PDFResourceManager()
            # create a file handle
            fake_file_handle = io.StringIO()
            # creating a text converter object
            converter = TextConverter(resource_manager, fake_file_handle, codec='utf-8', laparams=LAParams())
            # creating a page interpreter
            page_interpreter = PDFPageInterpreter(resource_manager, converter)
            page_interpreter.process_page(page)

            text = fake_file_handle.getvalue()
            # extract text
            yield text

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

```

Figure 34: Extract the text from pdf

Here, I have defined a function to extract the plain text from .pdf files. The pdf_path parameter gives path to pdf files that is to be extracted. We will begin by establishing a resource_manager instance. Then, using Python's io module, we will then build a file-like object. The next step is to build a converter. We'll use the TextConverter for that. Finally, we construct a PDF interpreter object that will extract the text from our resource management and converter objects. It returns the iterator of string of extracted text.

b) Extract Text from Doc files

```
def extract_text_from_doc(doc_path):  
    temp = docx2txt.process(doc_path)  
    text = [line.replace('\t', ' ') for line in temp.split('\n') if line]  
    return ' '.join(text)
```

Figure 35: Extract Text from Doc files

Here, I have defined a function to extract the plain text from .docx or .doc files. The doc_path parameter gives path to .doc, .docx files that is to be extracted. It returns the string of extracted text.

c) Detect the file extension

```
def extract_text(file_path, extension):  
    text = ''  
    if extension == '.pdf':  
        for page in extract_text_from_pdf(file_path):  
            text += ' ' + page  
    elif extension == '.docx' or extension == '.doc':  
        text = extract_text_from_doc(file_path)  
    return text
```

Figure 36: Detect the file extension

Here, I have defined a function to detect the file extension and call the text extraction function accordingly. If the file is on .pdf or .docx or .doc then it returns the text. The

file_path parameter gives path of the file of which text is to be extracted. After that the param_extension is the extension of the file 'file_name'.

d) Extract the entities

```
def extract_entity_sections(text):
    text_split = [i.strip() for i in text.split('\n')]

    entities = {}
    key = False
    for phrase in text_split:
        if len(phrase) == 1:
            p_key = phrase
        else:
            p_key = set(phrase.lower().split()) & set(cs.RESUME_SECTIONS)
        try:
            p_key = list(p_key)[0]
        except IndexError:
            pass
        if p_key in cs.RESUME_SECTIONS:
            entities[p_key] = []
            key = p_key
        elif key and phrase.strip():
            entities[key].append(phrase)
```

Figure 37: Extract the entities

Here, I have defined a function to extract all the raw text from the sections of resume. I have already created the list for sections of resume in constant.py file. The split () function divides a string by the separator supplied and produces a list object containing string elements. Whitespace character, such as, \t, \n, are used as the separator by default. The text parameter gives the raw text of the resume. It returns the dictionary of the entities.


```
RESUME_SECTIONS = [
    'accomplishments',
    'experience',
    'education',
    'interests',
    'projects',
    'professional experience',
    'publications',
    'skills',
]
```

e) Extract the email

```
def extract_email(text):
    email = re.findall("([^\s|@]+@[^\s]+\.[^\s|@]+)", str(text))
    if email:
        try:
            return email[0].split()[0].strip(';')
        except IndexError:
            return None
```

Figure 38: Extract the email

Here, I have defined a function to extract the email id from the text. The text parameter gives the plain text extracted from the resume files. As we know, email addresses have a set format, which includes an alphanumeric string, a @ symbol, another string, a. (dot), and a string at the last. We can extract such expressions from text using regular expressions.

f) Extract Name from Resume


```

def extract_name(text, matcher):
    email = extract_email(text)
    email = email.rsplit('@', 1)[0]
    email = re.sub(r'([a-zA-Z ]+?)', ' ', email)
    email = segment(email)
    first_name = string.capwords(email[0])
    last_name = string.capwords(email[-1])
    pattern = [cs.NAME_PATTERN]
    matcher.add('NAME', None, *pattern)
    matches = matcher(text)
    for match_id, start, end in matches:
        span = text[start:end]
        if first_name in span.text:
            return span.text
        elif last_name in span.text:
            return span.text
        elif first_name.upper() in span.text:
            return span.text
        elif last_name.upper() in span.text:
            return span.text

```

Figure 39: Extract Name from Resume

Here, I have defined a function to extract the name from the text. The text parameter gives the plain text extracted from the resume files. Whereas the matcher parameter is the object of spacy.matcher. Spacy is a text and language processing module. It includes pre-trained models for tagging, parsing, and entity recognition. The primary goal here is to extract names using Entity Recognition. One of the phases in extracting information from unstructured text is rule-based matching. It recognizes and extracts tokens and words based on patterns. Regular expressions can be used in rule-based matching to extract entities. We can extract a first and last name, which are generally proper nouns, using rule-based matching. A pattern is a collection of items that define the set of tokens to be matched. It has two PROPN POS tags in it (proper noun). As a result, the pattern is made up of two objects, each with PROPN POS tags. The NAME and the match id are then used to add the pattern to Matcher. Finally, matches are found using their starting and ending indexes.

g) Extract Mobile Number

```
def extract_mobile_number(text):
    phone = re.findall(re.compile(r'[\+\\(]?[1-9][0-9] .\\-\\(\\){8,}[0-9]'), text)

    if phone:
        number = ''.join(phone[0])

        if text.find(number) >= 0 and len(number) < 20:
            return number
    return None
```

Figure 40: Extract Mobile Number

Here, I have defined a function to extract the mobile number from the text. Regular expressions will be used to retrieve phone numbers. Phone numbers come in a variety of formats, including (+91) 1234567890, +911234567890, +91 123 456 7890, or +91 1234567890. As a result, I created a generic regular expression which can match all identical phone number patterns.

h) Extract Skills from the resume

```
def extract_skills(nlp_text, noun_chunks):
    tokens = [token.text for token in nlp_text if not token.is_stop]
    data = pd.read_csv(os.path.join(os.path.dirname(__file__), './staticfiles/skills.csv'))
    skills = list(data.columns.values)
    skillset = []
    # check for one-grams
    for token in tokens:
        if token.lower() in skills:
            skillset.append(token)

    # check for bi-grams and tri-grams
    for token in noun_chunks:
        token = token.text.lower().strip()
        if token in skills:
            skillset.append(token)
    return [i.capitalize() for i in set([i.lower() for i in skillset])]
```

Figure 41: Extract Skills from the resume

After gathering some basic information about the individual, let's focus on what matters most to recruiters: skills. For that, I've created a function that extracts skills from spacy text. The object of spacy.tokens is the NLP text parameter. Noun chunks are extracted from the text using the noun chunks parameter.

A method known as tokenization can be used to extract skills. In order to perform tokenization, we need to establish a sample dataset against which we can compare the abilities in a given résumé. For this, I have created a comma separated values (.csv) file with the necessary skill sets. For instance, if I'm in HR and need someone with NLP, ML, and AI abilities, I can create a csv file with the following keywords: machine learning, artificial intelligence, natural language processing. Following that, we renamed the file to skills.csv

Then, we can proceed to tokenize our extracted text and compare the skills to those in skills.csv. The panda's module will be used to read a csv file. After reviewing the material, we will remove all of the stop words from our resume. At last, it returns the list of skills extracted.

i) Extract Technical Skills

```
def extract_non_technical_skills(nlp_text, noun_chunks):  
  
    tokens = [token.text for token in nlp_text if not token.is_stop]  
    data = pd.read_csv(os.path.join(os.path.dirname(__file__), './staticfiles/non_technical_skills.csv'))  
    skills = list(data.columns.values)  
    skillset = []  
    # check for one-grams  
    for token in tokens:  
        if token.lower() in skills:  
            skillset.append(token)  
  
    # check for bi-grams and tri-grams  
    for token in noun_chunks:  
        token = token.text.lower().strip()  
        if token in skills:  
            skillset.append(token)  
    return [i.capitalize() for i in set([i.lower() for i in skillset])]
```

Figure 42: Extract Technical Skills

Non-technical skills are extracted using the same procedure as technical skills.

j) Extract Education and Year

```

def extract_education(nlp_text):

    edu = {}
    # Extract education degree
    for index, text in enumerate(nlp_text):
        for tex in text.split():
            tex = re.sub(r'[?|$.|!|,]', r'', tex)
            if tex.upper() in cs.EDUCATION and tex not in cs.STOPWORDS:
                edu[tex] = text + nlp_text[index + 1]

    # Extract year
    education = []
    for key in edu.keys():
        year = re.search(re.compile(cs.YEAR), edu[key])
        if year:
            education.append((key, ''.join(year.group(0))))
        else:
            education.append([key])

    return education

```

Figure 43: Extract Education and Year

The degree and the year of graduation will be the details we will extract. If XYZ finished BS in 2022, for example, we will extract a tuple like ('BS', '2022'). We'll need to get rid of all the stop words for this. We'll use the nltk module to load a long list of stopwords, which we'll then remove from our resume text. Hr teams are quite specific about the level of education or degree necessary for a given job. As a result, we'll be putting together an EDUCATION list that will include all of the similar degrees that meet the standards.

k) Extracting address

```

def extract_address(text):
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(text)
    for ent in doc.ents:
        if ent.label_ in ['LOC', 'GPE']:
            return ent.text

```

Figure 44: Extracting address

Here, I have defined a function to extract the address from the text using Spacy. GPE often refers to geopolitical entities such as cities, states, nations, continents,

and so on. In addition, LOCATION also can symbolize well-known mountains, rivers, and other natural features. The act of identifying named entities in unstructured text and then categorizing them into pre-defined categories is known as **Named entity recognition (NER)**.

I) Extract Experience

```
def extract_experience(resume_text):
    wordnet_lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))

    # word tokenization
    word_tokens = nltk.word_tokenize(resume_text)

    # remove stop words and lemmatize
    filtered_sentence = [w for w in word_tokens if not w in stop_words and wordnet_lemmatizer.lemmatize(w) not in stop_words]
    sent = nltk.pos_tag(filtered_sentence)

    # parse regex
    cp = nltk.RegexpParser('P: {<NNP>+}')
    cs = cp.parse(sent)

    # for i in cs.subtrees(filter=lambda x: x.label() == 'P'):
    #     print(i)

    test = []

    for vp in list(cs.subtrees(filter=lambda x: x.label()=='P')):
        test.append(" ".join([i[0] for i in vp.leaves() if len(vp.leaves()) >= 2]))

    # Search the word 'experience' in the chunk and then print out the text after it
    x = [x[x.lower().index('experience') + 10:] for i, x in enumerate(test) if x and 'experience' in x.lower()]
    return x
```

Figure 45: Extract Experience

Here, I have defined a function to extract the experience from the resume text. The resume_text parameter gives the plain text extracted from the resume files. After doing different preprocessing part. It searches the word 'experience' in the chunk and then print outs the text after it. At last, it returns the list of experience.

6.11.5. Output

```
[{'address': 'Brooklyn',
  'education': [],
  'email': 'karlasantos2@gmail.com',
  'experience': [' Senior Front End Developer Xero January'],
  'languages': [],
  'link': ['Github.com/karla-santos'],
  'mobile_number': '(123) 456-7890',
  'name': 'KARLA SANTOS',
  'nationality': [],
  'soft_skills': [],
  'technical_skills': ['Django',
                       'Python',
                       'Css',
                       'Mysql',
                       'Postgresql',
                       'Javascript']}]
```


Figure 46: Output

.json format is used to display the retrieved data. This allows the HR staff to quickly identify candidates that meet the organization's needs. This technique also benefits the HR department by saving time by eliminating the need to manually review each resume.

7. Conclusion

A normal resume is a compilation of information about a person's work experience, academic background, qualifications, and personal details. These elements might be present in a variety of ways or not at all. It's difficult to keep up with the jargon used in resumes. A resume is made up of corporate names, institutions, degrees, and other information that can be written in a variety of ways. It will take time to review all the resume by an individual.

Machine works faster than human and their accuracy to do any task was also good. Therefore, I have made a system which includes machine learning that extract the important information from resumes within a minute or less than a minute. The hiring individual can use this system for hiring any individual.

8. Critical Evaluation

8.1. Final Report

While working on my report, I experienced several issues. The primary issue was that I got tiny bit confused throughout the testing part. The diagrams and grouping of some topics into subtopics were the next difficult phase. However, after discussing with the teacher, I was able to resolve this issue. Aside from the flaws, the positive aspect of the report is that it is based on facts, researched information, and legitimate proofs.

8.2. Finding and process

There was some uncertainty when it came to select the right libraries and modules for data extraction. After conducting research, I was able to select a library. Following that, obtaining the appropriate dataset was difficult. I looked for a number of resumes dataset

but couldn't find one in the correct format. After that, I went online and found several templates and started using them for training data.

8.3. System

While developing the system, there were a number of technical difficulties. Following the development of the module, the next step is to integrate it into the system. I was having trouble integrating the module while working on the flask. I tried everything but couldn't get the component to work. As a result, I switched to Django as my framework. I was able to integrate after reading through several python and Django documentation. The integration part took up more time than expected.

Despite the time constraints, the majority of the features proposed in the proposal were implemented. However, the Tika parser library, which I specified in the proposal for the information extraction section, has not been implemented. It is used to detect the kind of document. As I began working on the project, I found that Spacy and Nltk were more suitable to my requirements, so I chose them over Tika.

8.4. Future Work

As stated in the paper, the project has a broad reach in the current context. The proposal's majority of proposed features have been implemented. So, if I continue working on this project, I intend to create a database for the system where the admin may keep the extracted data. Further, future study will include a more in-depth examination of certain techniques, further research on other libraries, and new approaches to explore different methods.

8.5. Self-reflection

After working on a project, everyone learns something new. After completing this project, I learned several new things about information extraction. I learnt how to deal with various NLP libraries, how to implement the module, and so on. Aside than that, I learnt about Django by watching videos and reading documentation. I also learned how to integrate the system from this portion.

9. Evidence of Project Management

9.1. Gantt Chart

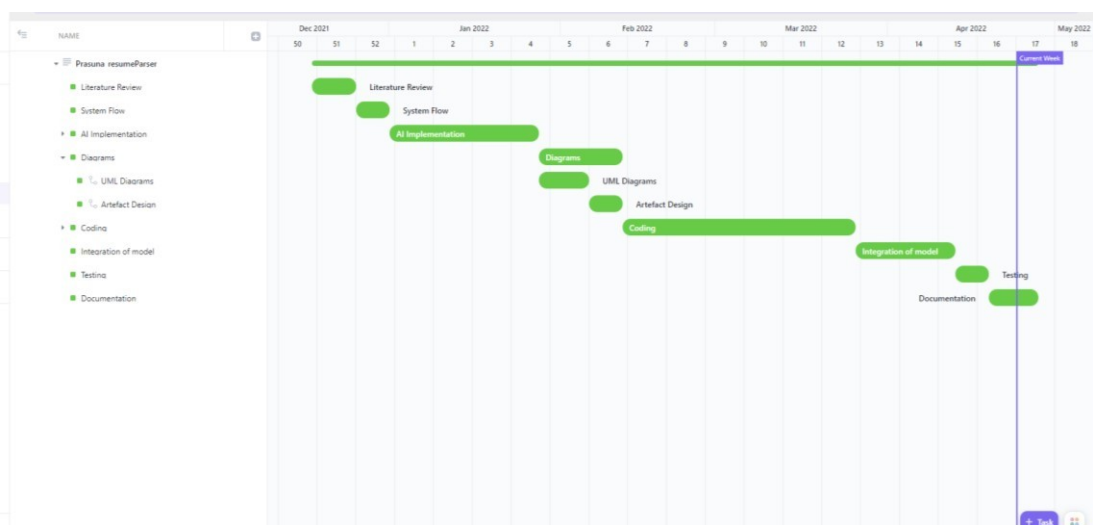


Figure 47: Final Gantt Chart

9.2. Project Management tool

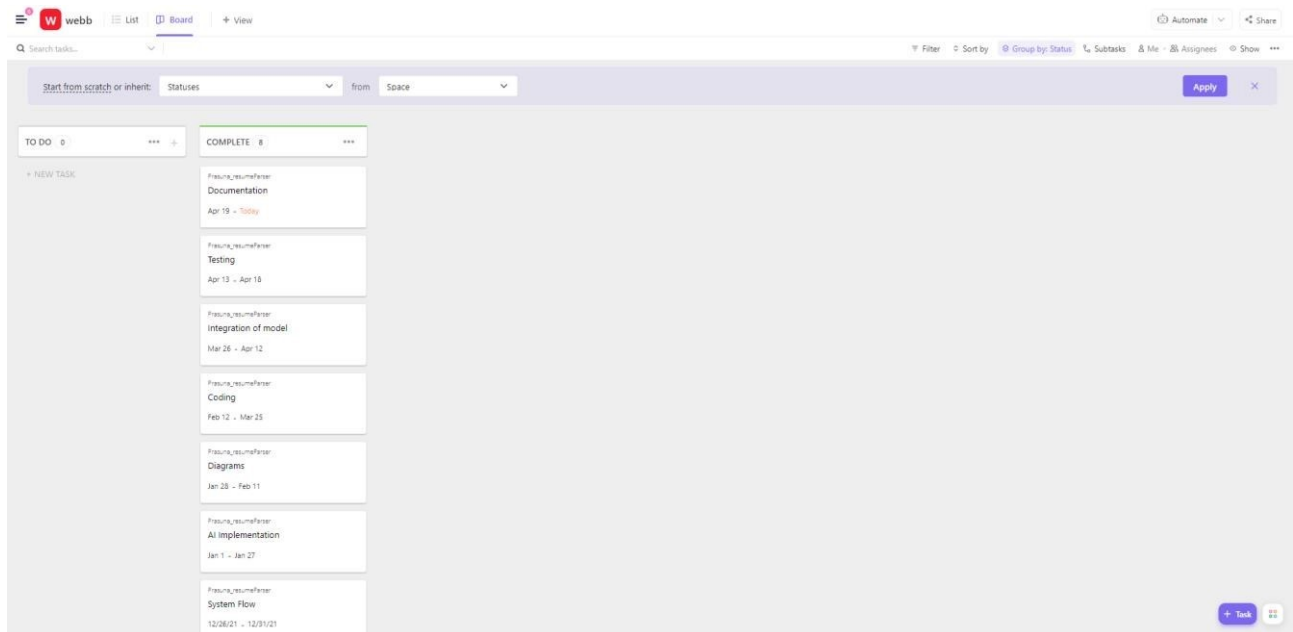


Figure 48: Project Management

10. References

- C, B. P., 2020. *towardsai*. [Online]
Available at: <https://towardsai.net/p/nlp/natural-language-processing-concepts-andworkflow-48083d2e3ce7> [Accessed 2022].
- Agrawal , R., 2021. *analyticsvidhya*. [Online]
Available at: <https://www.analyticsvidhya.com/blog/2021/06/must-knowntechniques-for-text-preprocessing-innlp/#:~:text=Text%20preprocessing%20is%20a%20method,text%20in%20a%20different%20case.> [Accessed 2022].
- Bhatia, V., Rawat, P., Kumar, A. & Shah, R. R., 2019. End-to-End Resume Parsing and Finding Candidates for a Job Description using BERT. *arxiv*.
- chavan, J., 2020. *medium*. [Online]
Available at: <https://medium.com/@jeevanchavan143/nlp-tokenization-stemminglemmatization-bag-of-words-tf-idf-pos-7650f83c60be> [Accessed 2022].
- Chen, J., Gao, . L. & Tang, Z., 2016. Information Extraction from Resume Documents in PDF. *researchgate*.

- Kopparapu, S. . K., 2015. Automatic Extraction of Usable Information from Unstructured Resumes to Aid Search. *ieeexplore*.
- Nguyen, V. V., Pham, V. . L. & Vu, N. . S., 2018. Study of Information Extraction in Resume. *semanticsscholar*.
- D., 2021. *analyticsvidhya*. [Online]
Available at: <https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-innlp-with-python-codes/> [Accessed 2022].
- geeksforgeeks, 2018. *geeksforgeeks*. [Online]
Available at: <https://www.geeksforgeeks.org/unified-modeling-language-umlsequence-diagrams/> [Accessed 2022].
- geeksforgeeks, 2018. *geeksforgeeks*. [Online]
Available at: <https://www.geeksforgeeks.org/unified-modeling-language-uml-activitydiagrams/#:~:text=An%20activity%20diagram%20is%20a,the%20activity%20is%20being%20executed.>
[Accessed 2022].
- ibm, 2021. *ibm*. [Online]
Available at: <https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagramseuse-case>
[Accessed 2022].
- journaldev, 2022. Python IO Module. *journaldev*.
- Kurama, V., 2021. *nanonets*. [Online]
Available at: <https://nanonets.com/blog/information-extraction/> [Accessed 2022].
- teamwork, 2021. *teamwork*. [Online]
Available at:
<https://www.teamwork.com/project-management-guide/projectmanagement-methodologies/> [Accessed 2022].
- VUKADIN, D., KURDIJA, A. . S., DELAČ, G. & ŠILIĆ, M., 2021. Information Extraction From Free-Form. *ieeexplore*.
- Wang, X. & Zu, S., 2019. RESUME INFORMATION EXTRACTION WITH ANOVEL TEXT BLOCK SEGMENTATION ALGORITHM. *researchgate*, Volume 8.
- Wolff, R., 2020. *monkeylearn*. [Online]
Available at: <https://monkeylearn.com/blog/nlp-benefits/> [Accessed 2022].

