Cairo University
Faculty of Computers and Information
Department of Computer Sciences

CV Shortlisting

Supervised by

Dr. Hanaa Baiomy

TA. Amira Hozaifa

Implemented by

20170134	Tarek Alaa Ali
20170152	Abdelrahman Hesham Shaker
20170094	Hussin Tarek
20170160	Abdallah Mohamed Naguib
20170212	Mohamed Ahmed Saad Abouzaid

Graduation Project

Academic Year 2020-2021

Final Documentation

List of Contents

Introduction 5
Motivation 5
Problem definition 5
Project Objective (suggested solution) 6
Gantt chart of project time plan 6
Project development methodology 11
The used tools in the project (SW and HW) 11
Related work 15
Data envelopment analysis 15
Logistic regression 16
Analytic hierarchy process 17
Previous students 17
System Analysis 18
Project specifications: 18
System Architecture 18
Functional Requirements: 19
Non-functional Requirements: 19
Brief Use Cases Description: 19
Use Case Diagram 21
System Design 22
System Class Diagrams 22
Backend class diagram 22
Frontend class diagram 23
Sequence Diagrams 24
Project ERD 26
System Component Diagram 26
System GUI Design 27
Implementation and Testing 33
Word2Vec 33
CBOW (Continuous Bag of words) 33
Skip – Gram model 34
Doc2Vec 34
PV-DM 35
DBOW 36

SBERT 37
Background 38
API Document 41
1. register 41
2. login 42
3. Get Jobs 44
4. Post a job 47
5. Delete all jobs posted by a specific user 48
6. Get a Job by it's ID 49
7. Update a job title or description 51
8. Delete a job by ID 52
9. Add resumes to an existing job 54
10. Get specific resume 56
11. Delete specific resume 58
Conventions 60
Status Codes 60
Model Results 62
References 67

List of Figures

Figure 1 : Gantt chart part 1 6
Figure 2 : Gantt chart part 2 6
Figure 3 : Machine learning (learning to rank) 16
Figure 4 : System Architecture 18
Figure 5 : use case diagram 21
Figure 6 : Backend class diagram 22
Figure 7 : Frontend class diagram 23
Figure 8 : System sequence diagram part 1 24
Figure 9 : System sequence diagram part 2 25
Figure 10 : System ERD 26
Figure 11 : System component diagram 26
Figure 12 : Register UI 27
Figure 13 : Login UI 28
Figure 14 : Main Screen 29
Figure 15 : job create 30
Figure 16 : job edit 31

Figure 17 :job details 32
Figure 18 : PV-DM example 35
Figure 19 :DBOW example 36

List of Tables

Table 1 : Work plan 7
Table 2 : Brief Use Cases Description 19
Table 3 : register request 41
Table 4 : register parameters 41
Table 5 : register response 41
Table 6 : login request 42
Table 7 : login parameters 43
Table 8 : login response 43
Table 9 : Get jobs request 44
Table 10 : Get jobs parameters 44
Table 11 : Get jobs response 44
Table 12 : Post a job request 47
Table 13 : Post a job parameters 47
Table 14 : Post a job response 48
Table 15 : Delete all jobs posted by a specific user request 48
Table 16 : Delete all jobs posted by a specific user parameters 49
Table 17 : Delete all jobs posted by a specific user response 49
Table 18 : Get a Job by it's ID request 49
Table 19 : Get a Job by it's ID parameters 49
Table 20 : Get a Job by it's ID response 50
Table 21 : Update a job title or description request 51
Table 22 : Update a job title or description parameters 52
Table 23 : Update a job title or description reponse 52
Table 24 : Delete a job by ID request 52
Table 25 : Delete a job by ID parameters 53
Table 26 : Delete a job by ID response 53
Table 27 : Add resumes to an existing job request 54
Table 28 : Add resumes to an existing job parameters 54
Table 29 : Add resumes to an existing job response 54
Table 30 : Get specific resume request 56
Table 31 : Get specific resume parameters 57
Table 32 : Get specific resume reponse 57
Table 33 : Delete specific resume request 58
Table 34 : Delete specific resume parameters 58
Table 35 : Delete specific resume response 59
Table 36 : Status Codes decription 60
Table 37 : Model Results 64

Introduction

Motivation

The process of screening resumes manually is a time consuming and challenging activity to any hiring staff, as resumes are populated with irrelevant information and constructed in variable structures and formats, which makes the hiring process very expensive. What we are trying to accomplish is to automate the time-consuming process of screening resumes or at the very least make it easier by ranking the resumes in terms of their relevance towards the job description. To accomplish that we are going to use state of the art NLP techniques. We are going to try multiple word embedding techniques such as Word2Vec, Doc2Vec and SBERT to vectorize the documents and job description, then check similarity between each resume and job description using cosine similarity

Problem definition

The problem is that any job opening can receive a huge number of resumes, including irrelevant resumes, and the hiring manager will waste a lot of time trying scrutinize and ranking them according to the job description, so our aim is to automate this process by automatically scrutinizing and ranking resumes based on job description

Project Objective (suggested solution)

using state of the art SBERT models, we vectorize all of the given resumes and the job description then we calculate the cosine similarity between each resume and the job description then we rank the resumes according to their similarity to the job description

Gantt chart of project time plan

Figure 1 : Gantt chart part 1

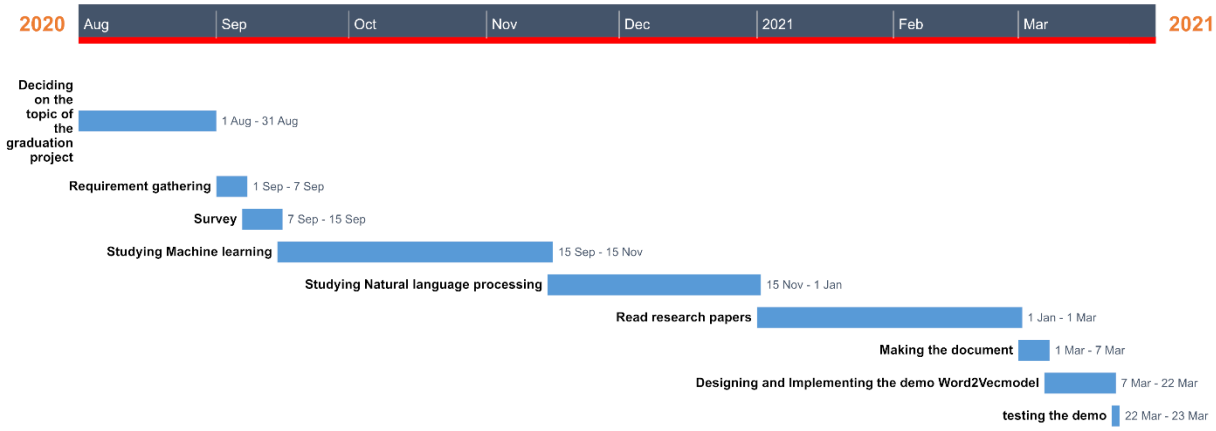


Figure 2: Gantt chart part 2

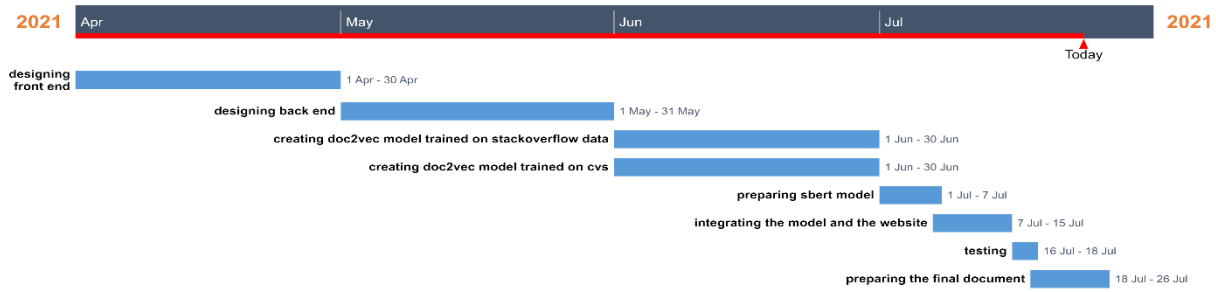


Table 1 : Work plan

Activity	Time Period	Comment
Deciding on the topic of the graduation project	30 days (Aug 2020)	After many meeting, and meeting with D. Elramly, we decide this topic,
Requirement gathering	7 days (1- 7 sept)	Requirement gathering was done by searching the internet and sharing views among group members, until we realize that we need to use NLP and machine learning techniques
Survey	7 days (7- 15 sept)	Ask the companies HR about the process of hiring and ranking resumes
Studying Machine learning	60 days (15 sept – 15 nov)	We decide to study Coursera Andrew NG course, as we
Studying Natural language processing	45 days (15 Nov – 1 Jan)	We collect 4 different courses, and each one study one of it, to collect a big scale of knowledge about NLP
Read research papers	60 days (1 Jan – 1 Mar)	We read 7 different research papers and many articles about

		different NLP techniques and methods other than NLP techniques
Meeting to decide the final scope of our project	1 hour	Final meeting to decide on final functionalities of the graduation project (we decide to make user able to use our project with and without login, depends on whether he need to save his job applicationos)
Making the document	7 days (1 Mar – 7 Mar)	Writing the project specification (usecases, usecase diagram, functional and non functional)...etc
Designing and implementing the demo Word2Vecmodel	16 days (7 Mar – 22 Mar)	Implementing a demo that takes a bunch of resumes and returns the cosine similarity between according to a specific job description

designing front end	22 days (April 2021)	Studying reactjs and after looking at multiple templates we decided on a general design and then we implement it in reactJS framework
designing back end	21 days (May 2021)	Studying NodeJs after deciding on the main functions of the website with respect to the functions requirements and we implemented it in NodeJS
creating doc2vec model trained on stackoverflow data	22 days (June 2021)	after gathering the data and preprocessing it by removing the html tags, extracting the code withing each post, removing stop words and stemming. using the gensim framework we trained a doc2vec model on stackoverflow data, with the final result being bad.
creating doc2vec model trained on CVs	22 days (June 2021)	after gathering the data and preprocessing it by remove stop words, toknization with genism simple

		processing and stemming. using the gensim framework we trained a doc2vec model on the data. the model preformed better than the previously trained model. it was shown that the more cvs used during its training the better the model preforms.
preparing SBERT model	5 days (1 July- 7 July)	found pretrained SBERT model and tested it, performed a lot better than previously trained models
integrating the model and the website	7 days (7 July- 15 July)	fter deciding on which model to use and finishing up the website we integrated the chosen model into our website
testing	1 day (16 July- 18 July)	used many different test cases to evaluate the previous models and compare them for the final time
preparing the final document	6 days (18 July- 26 July)	spent some time finalizing the documentation

Project development methodology

we used a variation of the agile iterative methodology where we work on everything in every iteration with focus on specific objectives that have strict deadlines.

The agile iterative approach focuses on delivering value as fast as possible in increments, rather than all at once. This approach is especially useful in software development and product development. An iterative approach means the software or product development process is split into multiple explicit iterations or versions, each delivering some valuable improvements or additional features.

Iterative methodology allows software developers to adjust, refine, and review software development processes constantly to improve their performance incrementally. The agile iterative approach creates opportunities for constant evaluation and improvement in development processes. The design of an iterative approach is simple and easy to implement, regardless of the context.

The used tools in the project (SW and HW)

1- ReactJS is a free and open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

2- NodeJS is primarily used for non-blocking, event-driven servers, due to its single-threaded nature. It's used for

traditional web sites and back-end API services, but was designed with real-time, push-based architectures in mind.

3- Python is a general-purpose programming language, so it can be used for many things. Python is used for web development, AI, machine learning, operating systems, mobile application development, and video games. Python plays a vital role in AI coding language by providing it with good frameworks.

We used the following Python frameworks:

1- Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

2- Gensim is an open-source library for unsupervised topic modeling and natural language processing, using modern statistical machine learning.

3- Amazon Textract is a machine learning service that automatically extracts text, handwriting and data from scanned documents that goes beyond simple optical character recognition (OCR) to identify, understand, and extract data from forms and tables.

4- Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such

as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

5- The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

6- Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

7- SentenceTransformers is a Python framework for state-of-the-art sentence, text and image embeddings. ... You can use this framework to compute sentence / text embeddings for more than 100 languages. These embeddings can then be compared e.g. with cosine-similarity to find sentences with a similar meaning.

1.7 Report Organization (summary of the rest of the report)

in terms of related works. Our alumni have also made a similar project to the project we proposed. the main difference between us is that they rank the resumes based on the similarity between a given set of technical skills and the technical skills in each resume. while we rank the resumes based on the similarity between a given job description and each resume. we also used a newer and better technology which is SBERT while they used Doc2vec.

there are also other methods to screen resumes that aren't based on matching for example Data envelopment analysis, machine learning (Learning to rank), Logistic regression and Analytic Hierarchy process.

Project specifications:

Our projects takes in resumes and extracts all the information within the resumes through Texract python module then it vectorizes both the given resumes and the job description using a certain model(SBERT) then calculate the cosine similarity and rank the resumes according to said similarity.

Related work

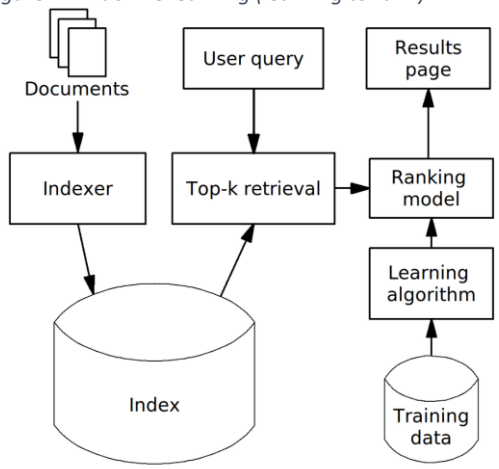
There are other methods to screen resumes that is not based on matching

Data envelopment analysis

(DEA) can be used as a fair screening and sorting tool to support the candidate selection and decision-making process. Each applicant is viewed as an entity with multiple achievements. Without any a priori preference or information on the multiple achievements, DEA identifies the non-dominated solutions, which, in our case, represent the ‘‘best’’ candidates. A DEA-aided recruiting process was developed that (1) determines the performance levels of the ‘‘best’’ candidates relative to other applicants; (2) evaluates the degree of excellence of ‘‘best’’ candidates’ performance; (3) forms consistent tradeoff information on multiple recruiting criteria among search committee members, and, then, (4) clusters the Applicants.[4]

Machine learning (Learning to rank)

Figure 2 : Machine learning (learning to rank)



Learning to rank refers to machine learning techniques for training the model in a ranking task.

Training data consists of queries and documents matching them together with the relevance degree of each match. It is not feasible to check the relevance of all documents, and so typically a technique called pooling is used — only the top few documents, retrieved by some existing ranking models are checked.

Training data is used by a learning algorithm to produce a ranking model which computes the relevance of documents for actual queries.[5][6]

Logistic regression

We have seen that “Logistic Regression” forecasts the probability based on weightages for various attributes learned from which resumes were shortlisted or rejected in the past. This probability — in our case — would indicate if the candidate is suitable or not. We would use this number to rank candidates in descending order of suitability.[7]

Analytic hierarchy process

The first step in the analytic hierarchy process is to define the ranking criteria and the criteria's

Our work differs from all of these in which it's not based on machine learning models to learn how to rank learned from labeled resumes, it's based on text matching taking into consideration semantic relationship among words, also the only data needed to train our model is text corpus including technical terms, no resumes needed as the model is based on matching text. However, it's useful to train on resumed also.

Previous students

Our alumni have also made a similar project to the project we proposed. the main difference between us is that they rank the resumes based on the similarity between a given set of technical skills and the technical skills in each resume. while we rank the resumes based on the similarity between a given job description and each resume. we also used a newer and better technology which is SBERT while they used Doc2Vec.

we also trained two Doc2Vec models one using stackoverflow data and the other trained on full resumes with the one trained on resumes giving a better result on our ranking problem but in the end we decided to use an SBERT

model as it gave even better results and is a newer and better technology

.

System Analysis

Project specifications:

- Our projects take in resumes
- then extracts all the information within the resumes through Texract python module
- then it vectorizes both the given resumes and the job description using a certain model(SBERT)
- then calculate the cosine similarity
- then rank the resumes according to said similarity.

System Architecture

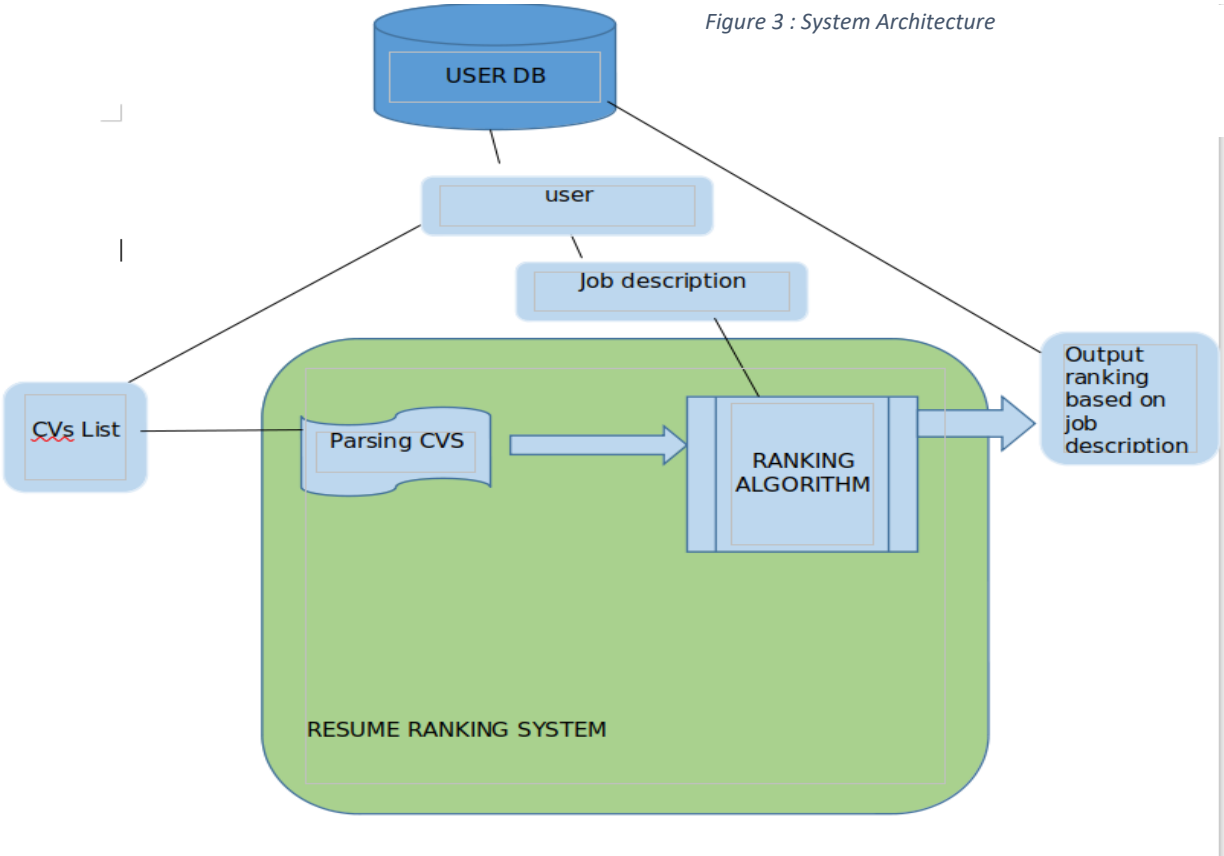


Figure 3 : System Architecture

Functional Requirements:

1- User can add a job and its job description

2- User can sort the CVs according to job requirements

3- User can list(view) his previous jobs and ranked CVs for them

4- User can upload new CVs for a job

5- User can view sorted CVs for a job

Non-functional Requirements:

Usability: The user should be able to sort the resumes with almost 3 click

- upload CVs
- Upload job description
- sort

so User can use the system without login, if he wants to save this job application, then he must login to save it with his job applications inserted.

Brief Use Cases Description:

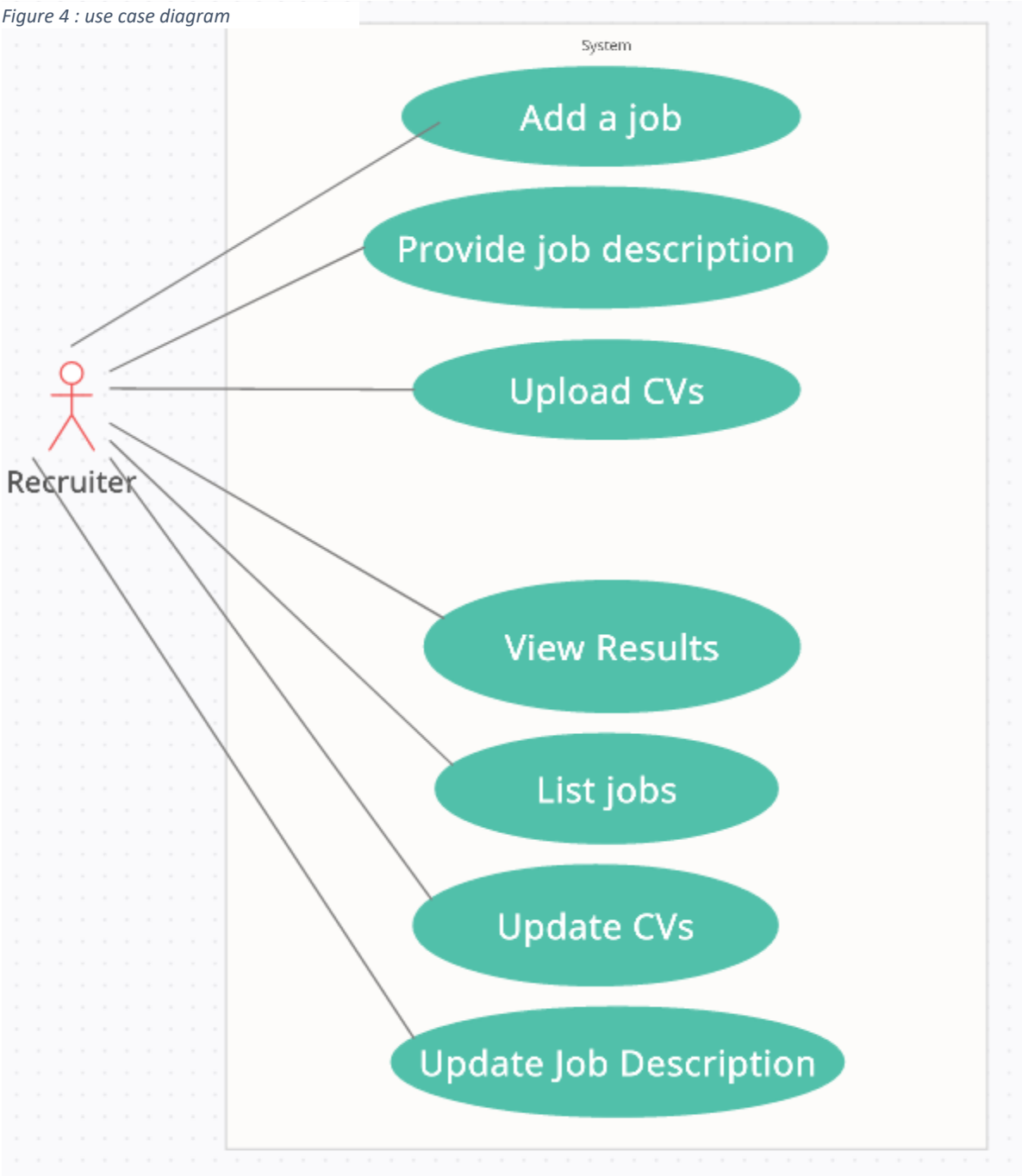
Table 2 : Brief Use Cases Description

Use case	Actor	description
Add a job	Recruiter	Recruiter request to add a job and fill its description form

Provide job description	Recruiter	Recruiter can add job description
Upload CVs	Recruiter	Recruiter uploads CVs to a specific job
View Results	Recruiter	Recruiter views results of a certain job
List jobs	Recruiter	Recruiter lists all jobs added
Update CVs	Recruiter	Recruiter can upload more CVs or delete from them
Update job description	Recruiter	Recruiter can edit the job description he entered before

Use Case Diagram

Figure 4 : use case diagram

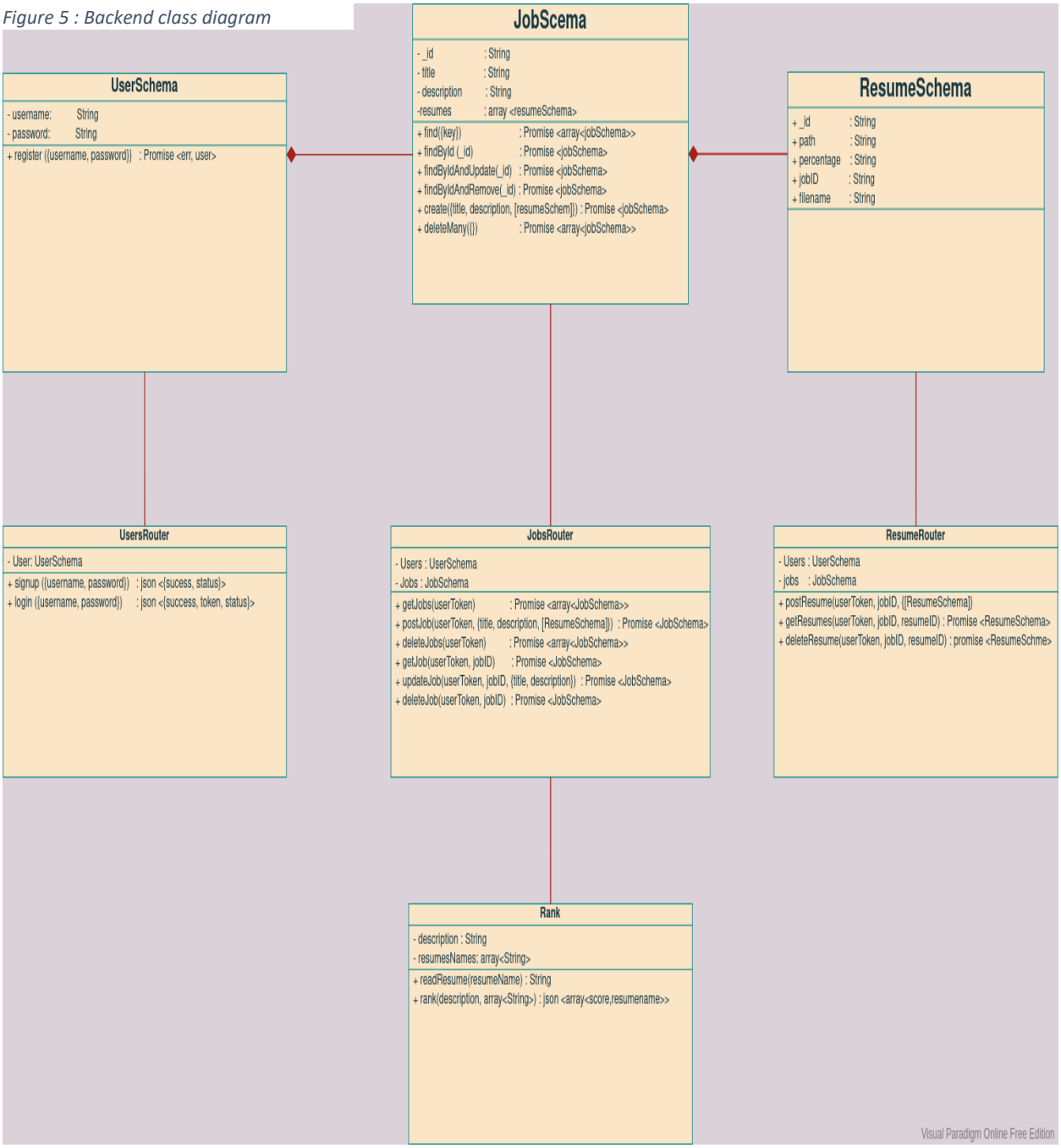


System Design

System Class Diagrams

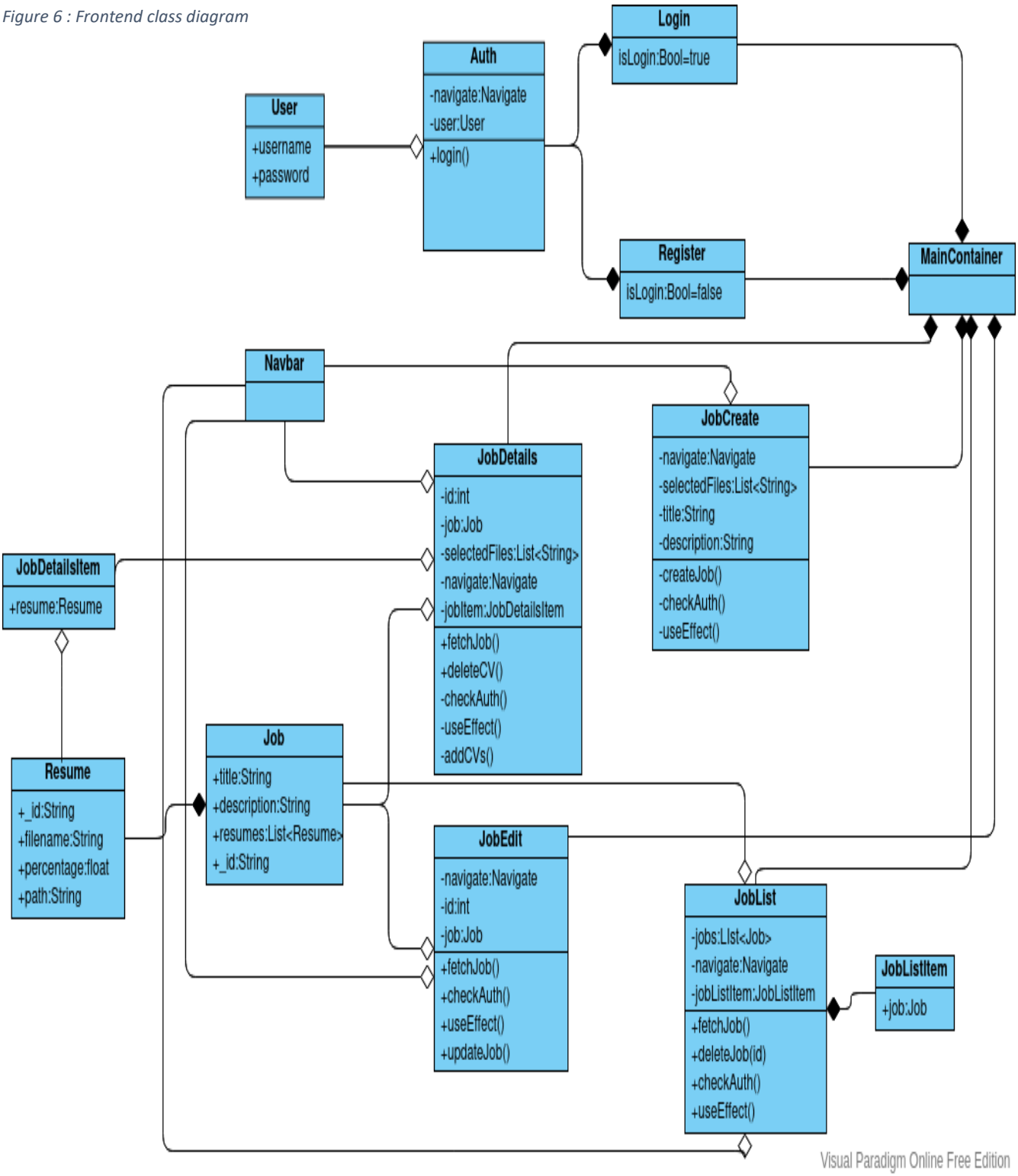
Backend class diagram

Figure 5 : Backend class diagram



Frontend class diagram

Figure 6 : Frontend class diagram



Sequence Diagrams

Figure 7 : System sequence diagram part 1

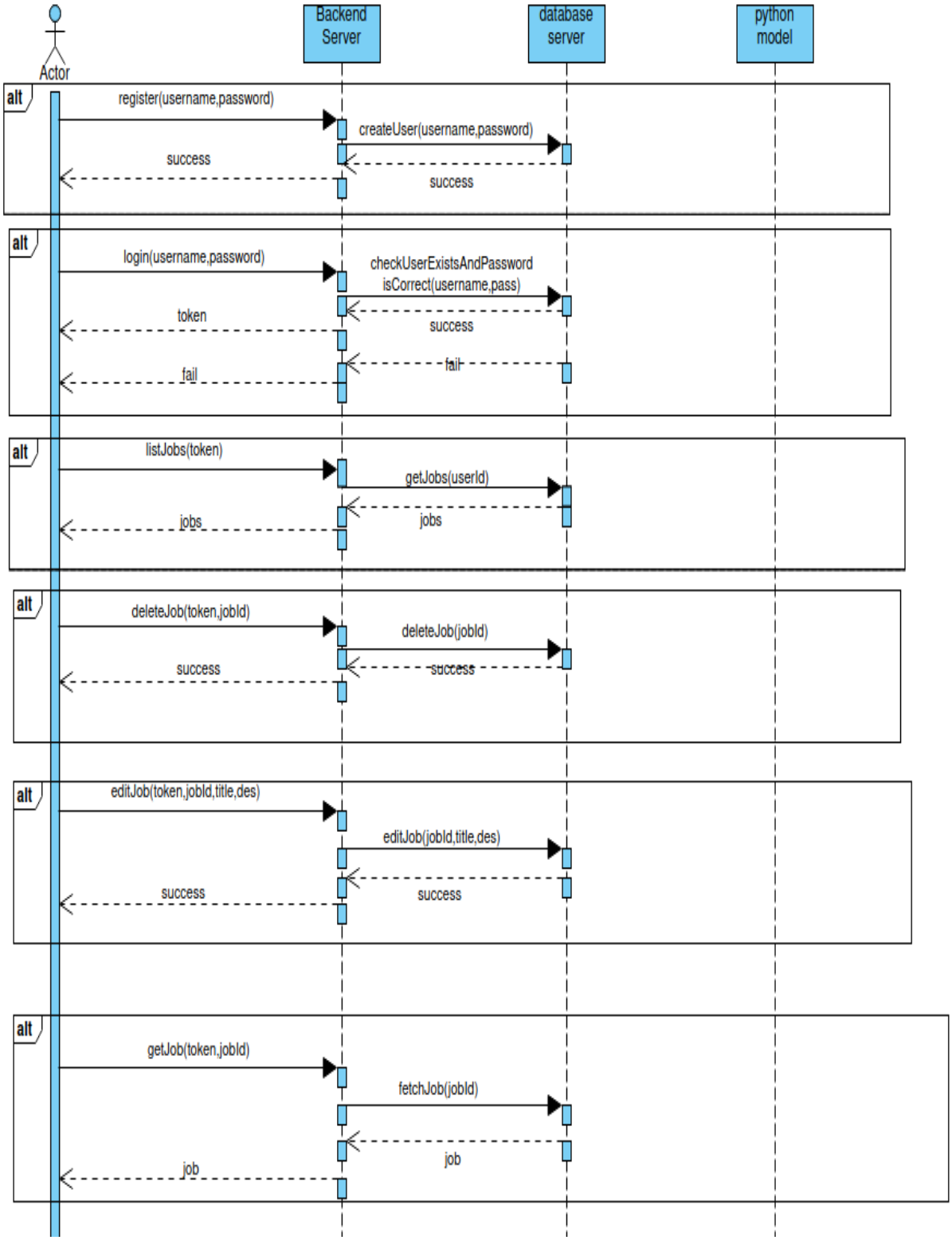
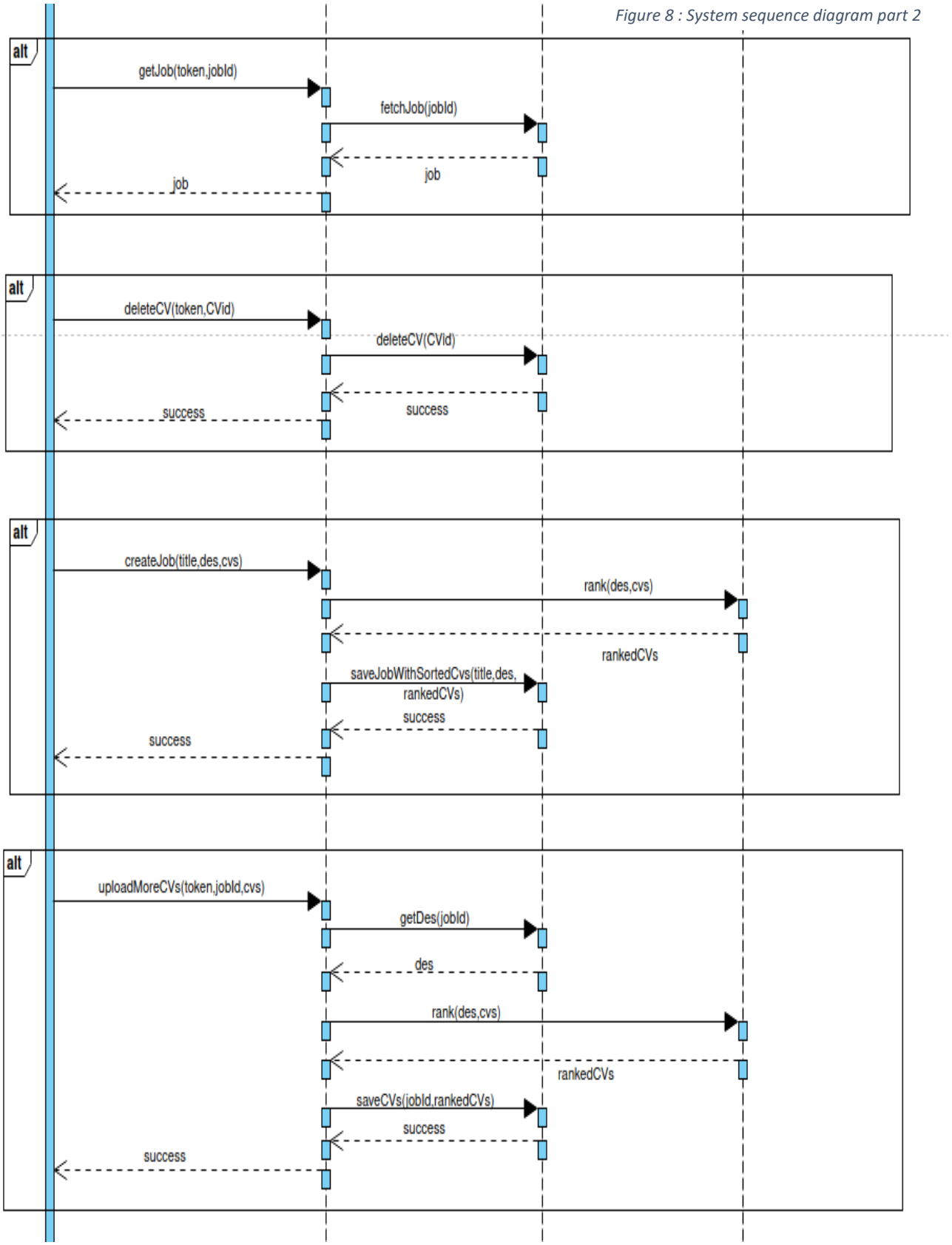
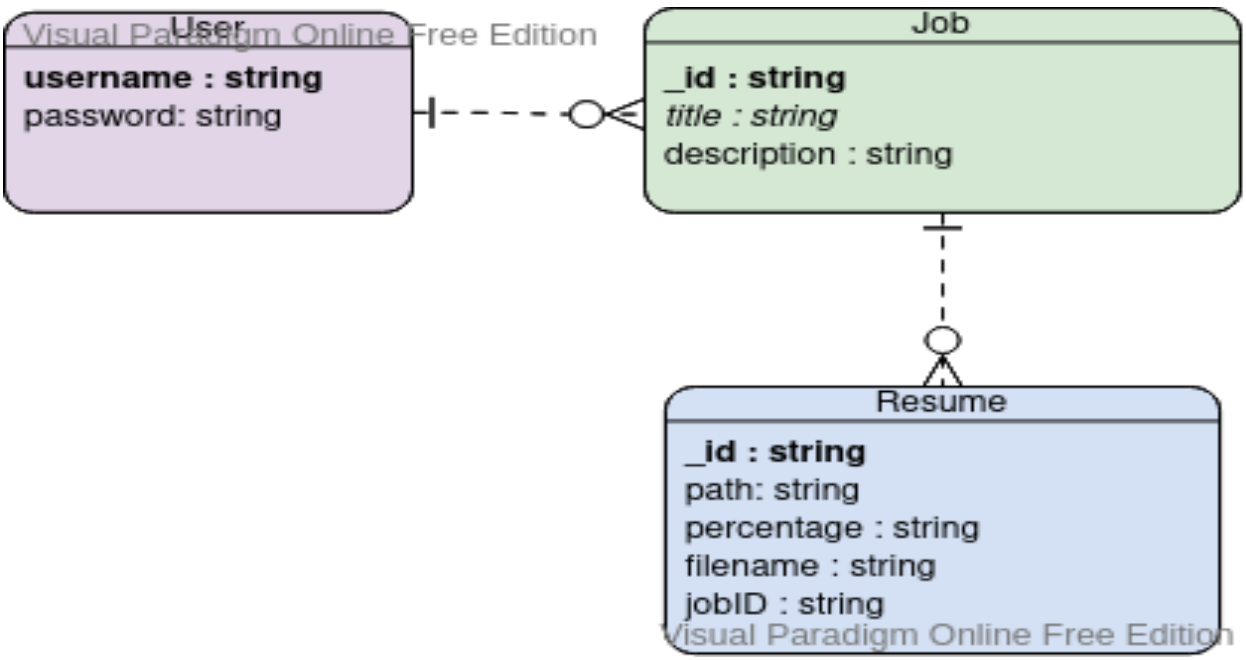


Figure 8 : System sequence diagram part 2



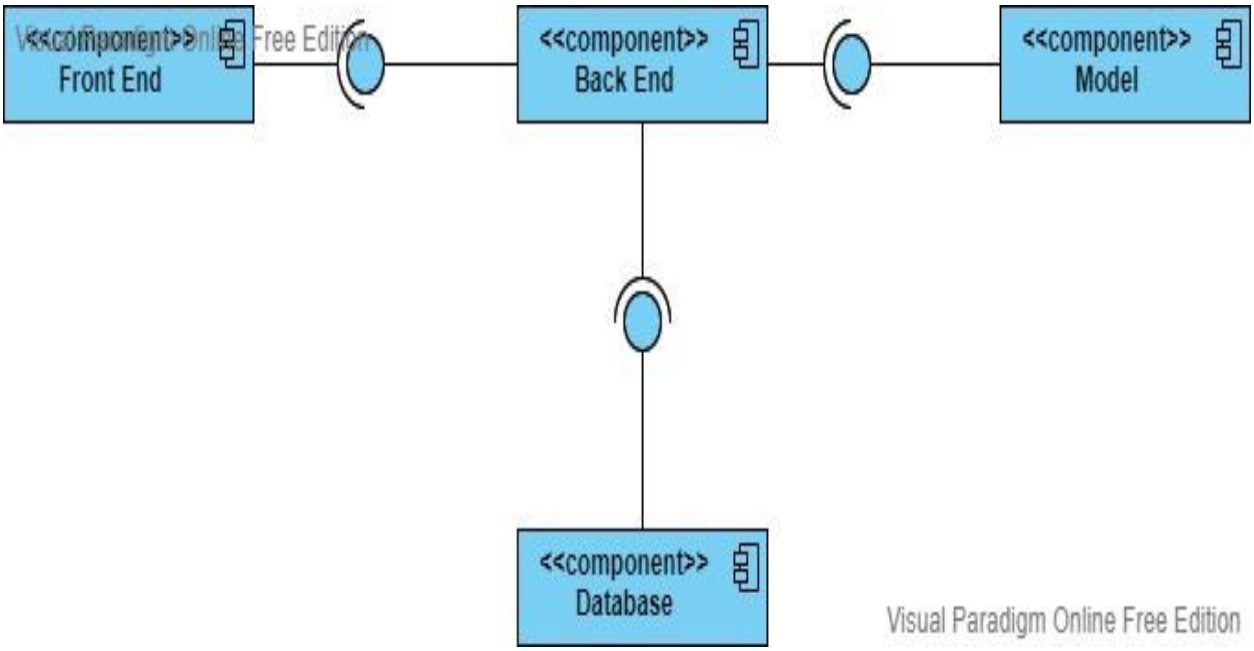
Project ERD

Figure 9 : System ERD



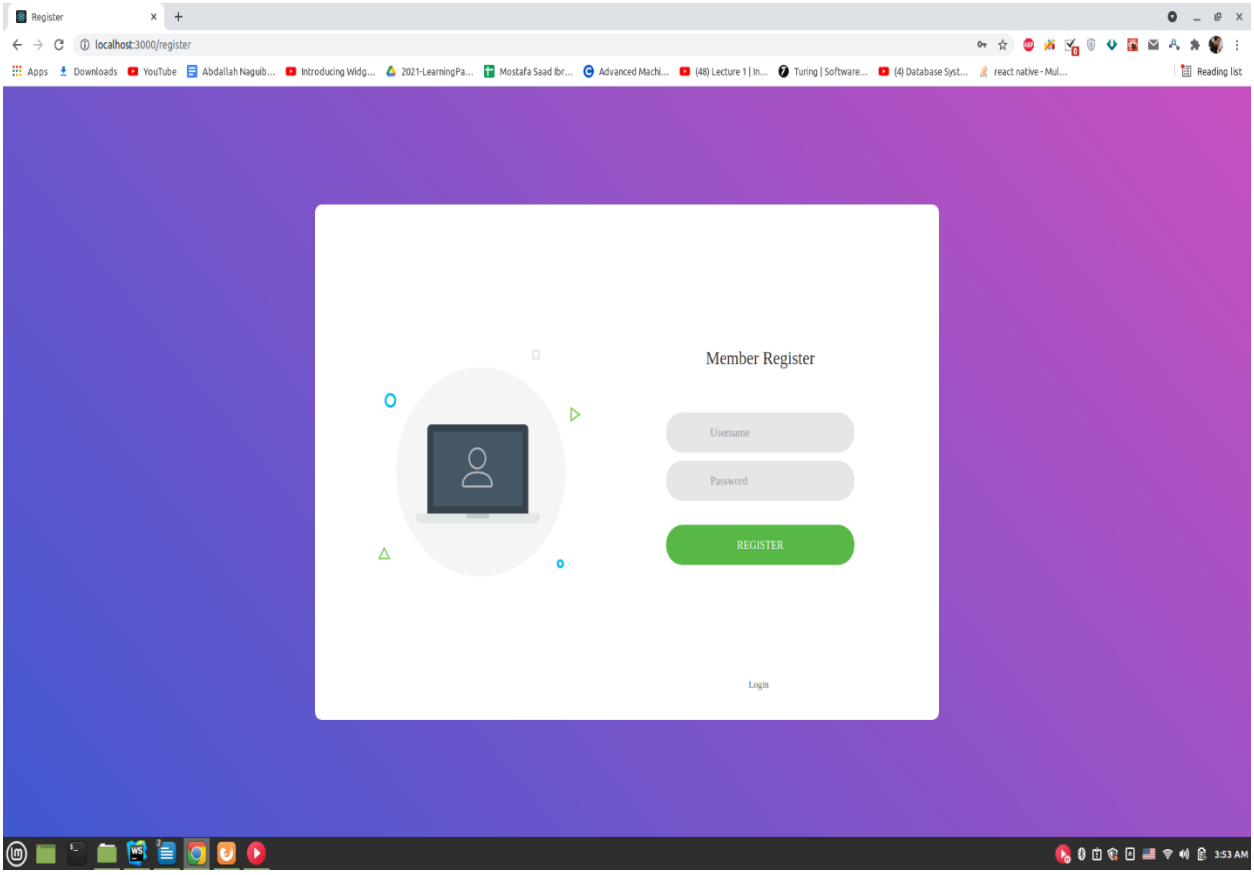
System Component Diagram

Figure 10 : System component diagram



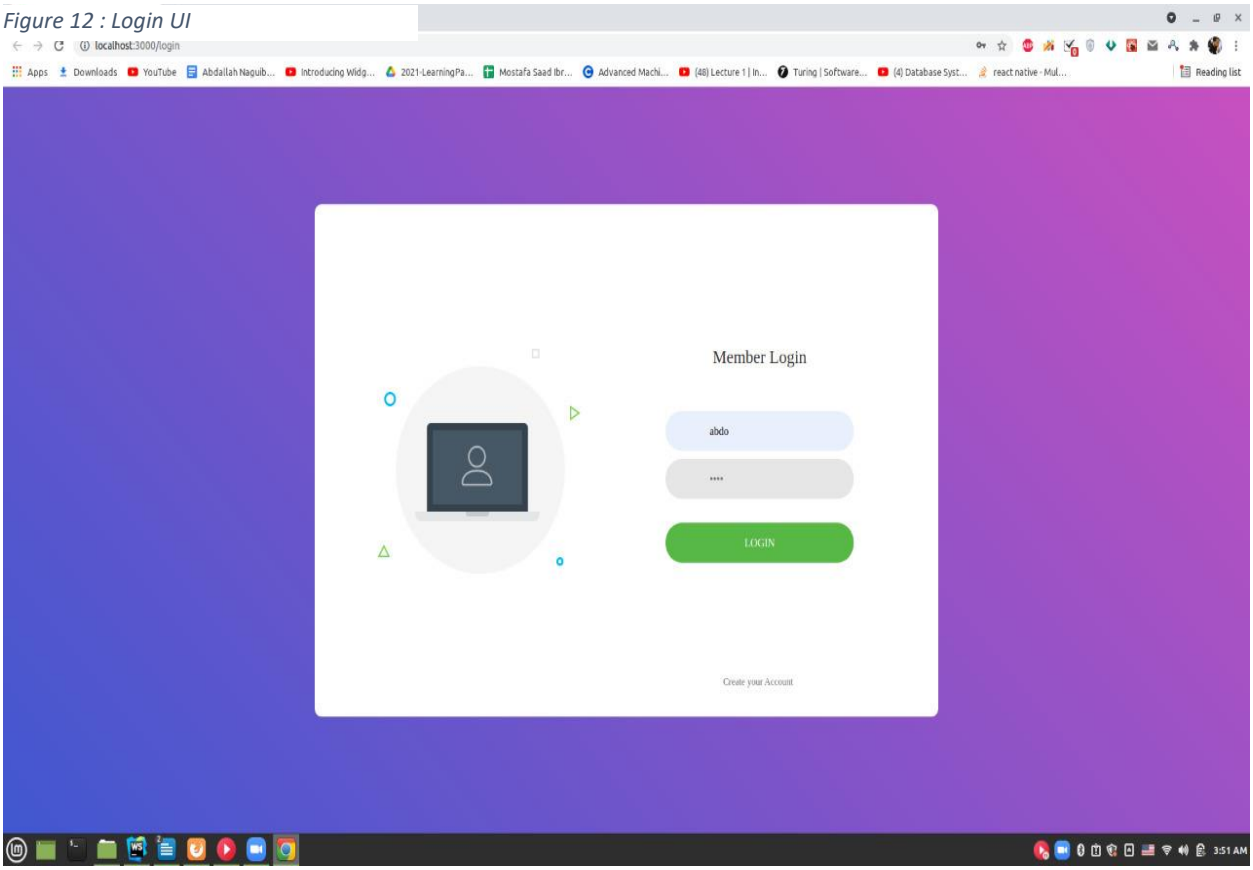
System GUI Design

Figure 11 : Register UI



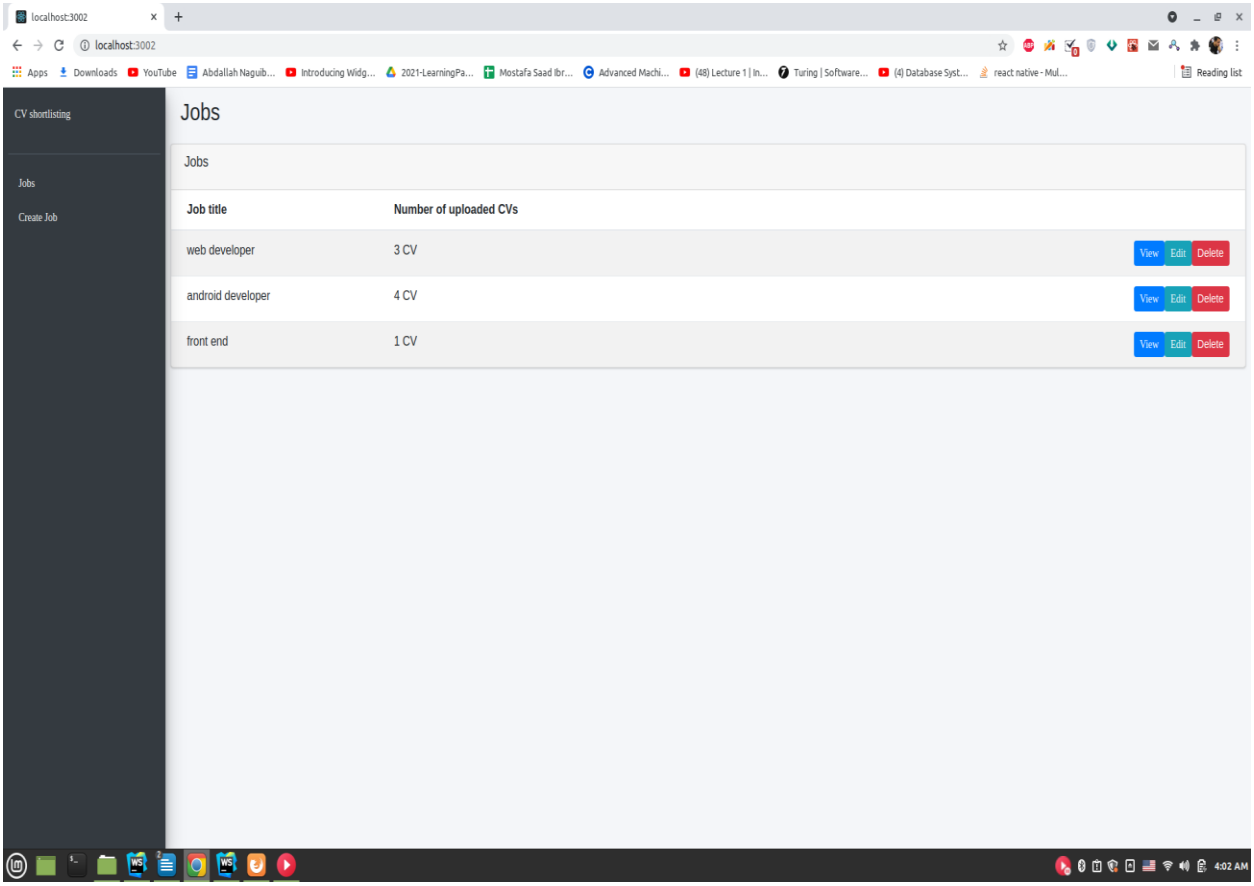
in this screen user enters his username and password, to register in the system and after he registers, he is redirected to the login screen

Figure 12 : Login UI



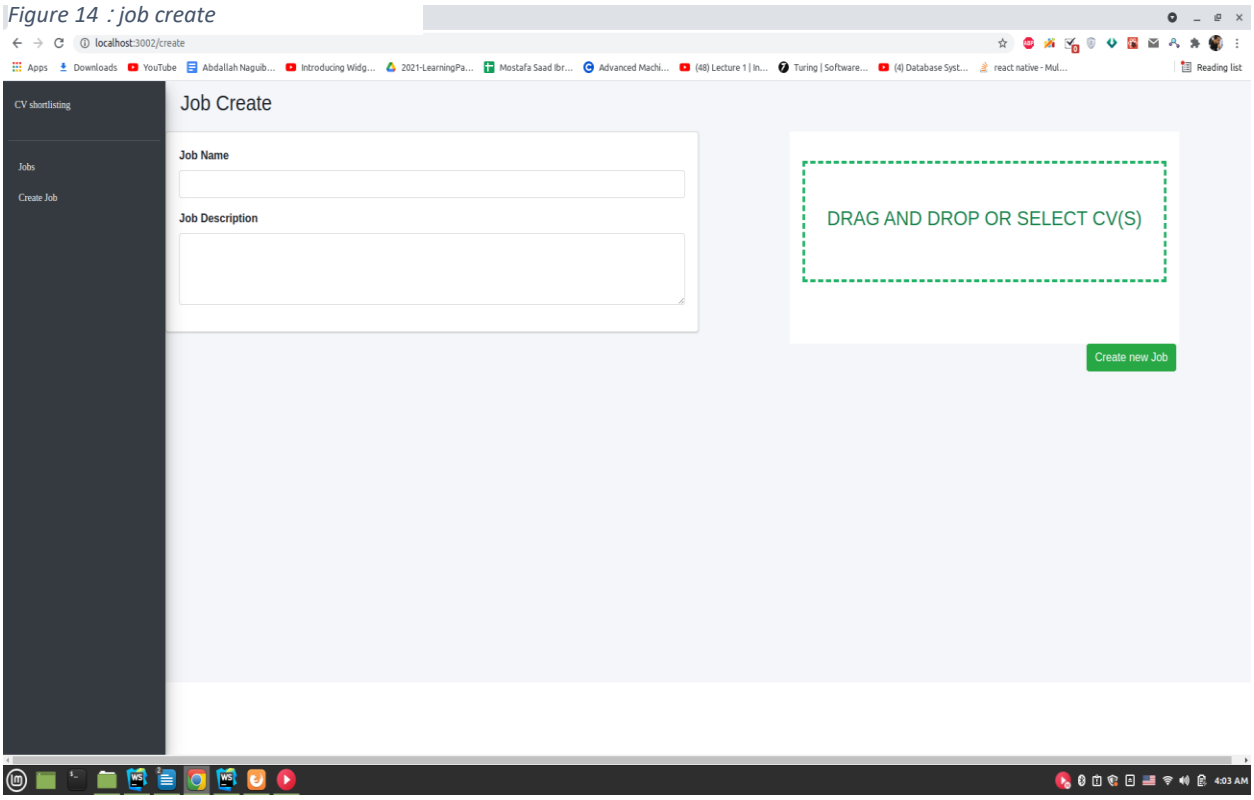
and then in this screen user enters his data to login and then he will be redirected to the main page

Figure 13 : Main Screen



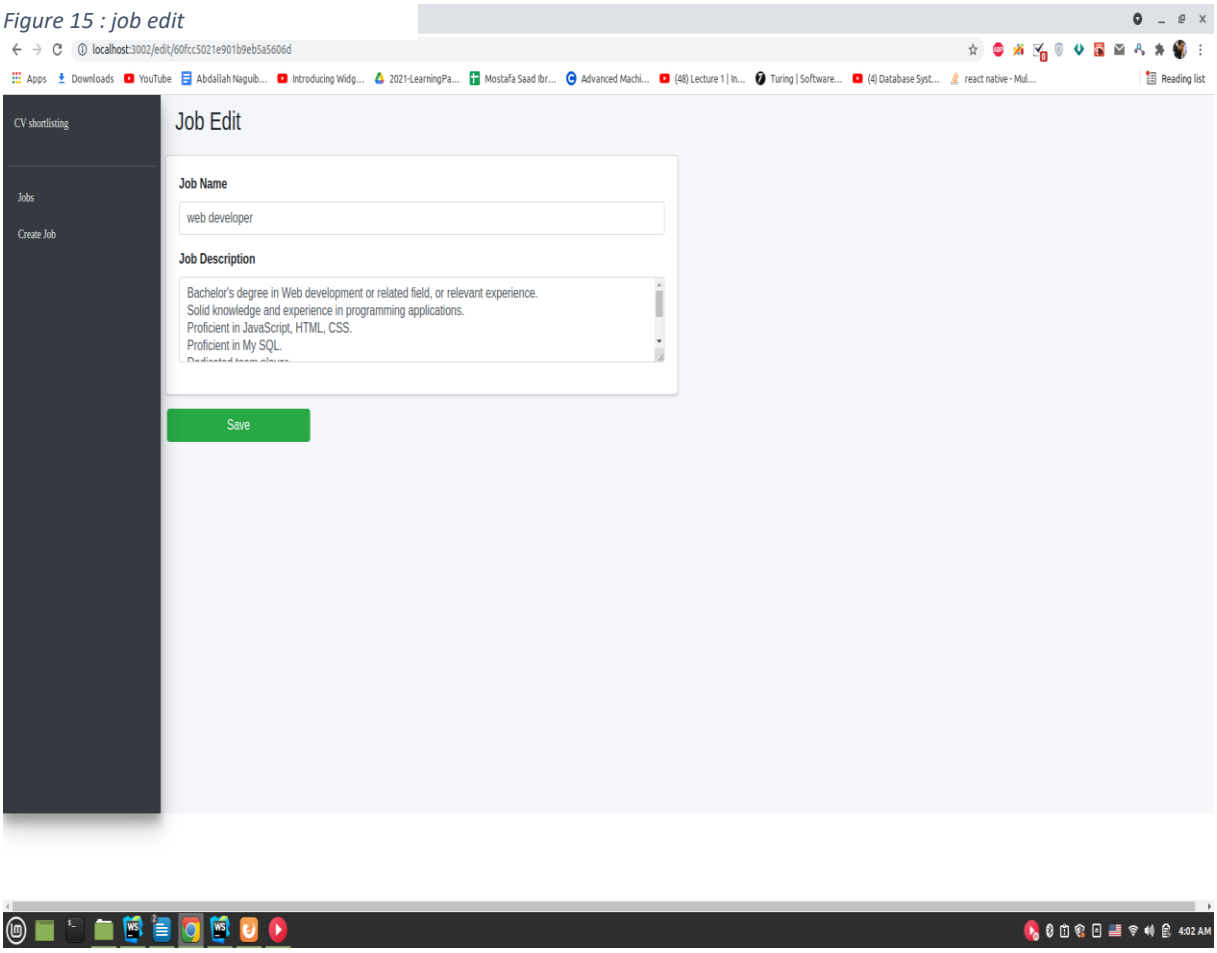
This is the main screen in the app where the user lists his jobs and he can go to create job , view a specific job , edit a job or delete a job

Figure 14 : job create



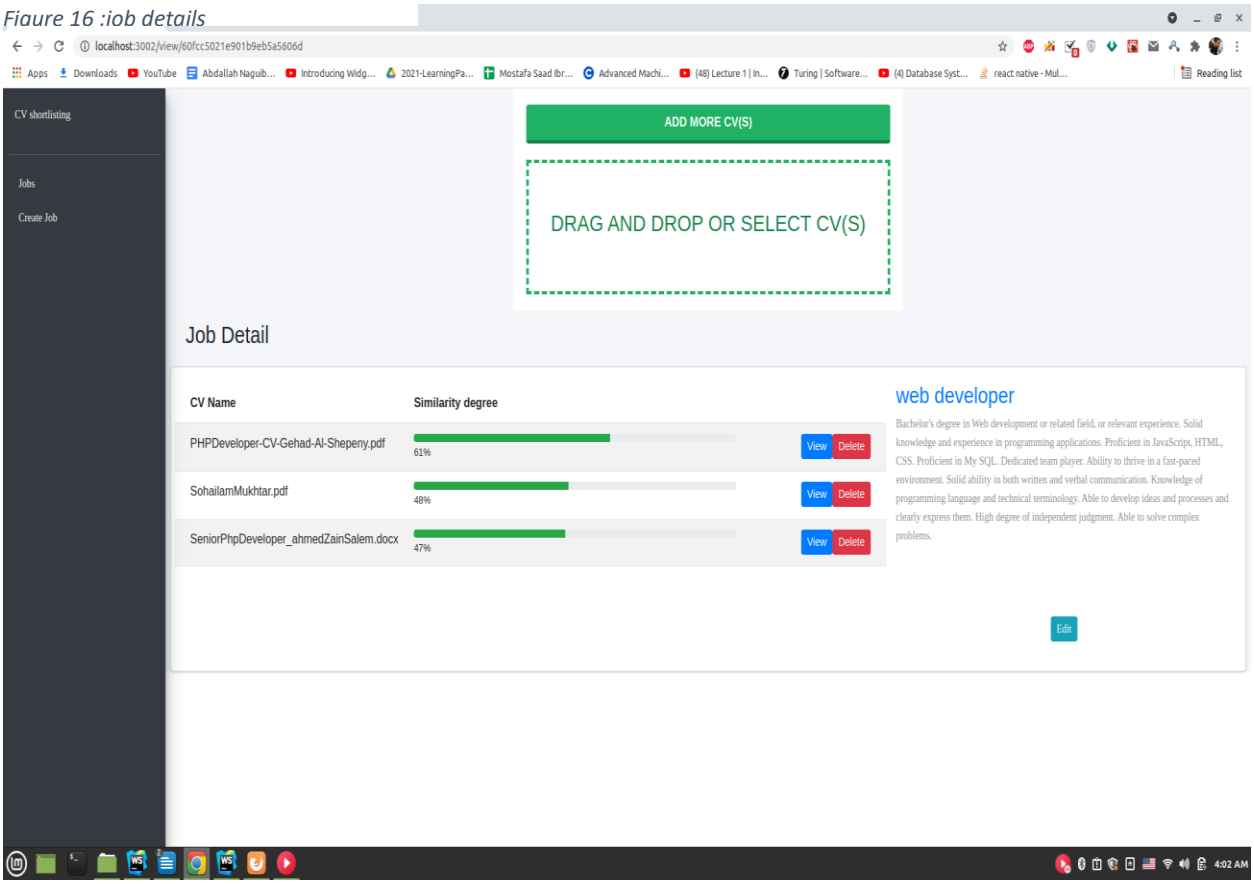
In this screen user can create a job by entering its name and its description and select CVs for this job

Figure 15 : job edit



In this screen user can edit the job , when he enters this page , he will see the current title and description already written in their respective fields

Figure 16 :job details



In this screen the user can view his job details like its title and its description and the CVs with their similarity degrees to the description and the CVs will be sorted according to this similarity , and he will be able to view any CV or delete it , and he can also add more CVs from the top button

Implementation and Testing

we trained doc2vec models and used pretrained SBERT and word2vec models. Here is a quick introduction about what each of them is:

Word2Vec

Word2vec is not a single algorithm but a combination of two techniques – CBOW (Continuous bag of words) and Skip-gram model. Both of these are shallow neural networks which map word(s) to the target variable which is also a word(s). Both of these techniques learn weights which act as word vector representations

CBOW (Continuous Bag of words)

Advantages of CBOW:

- Being probabilistic is nature, it is supposed to perform superior to deterministic methods(generally).
- It is low on memory. It does not need to have huge RAM requirements like that of co-occurrence matrix where it needs to store three huge matrices

Disadvantages of CBOW:

- CBOW takes the average of the context of a word (as seen above in calculation of hidden activation). For example, Apple can be both a fruit and a company but CBOW takes an average of both the

contexts and places it in between a cluster for fruits and companies.

- Training a CBOW from scratch can take forever if not properly optimized

Skip – Gram model

Advantages of Skip-Gram Model

- Skip-gram model can capture two semantics for a single word. i.e it will have two vector representations of Apple. One for the company and other for the fruit.

- Skip-gram with negative sub-sampling outperforms every other method generally.

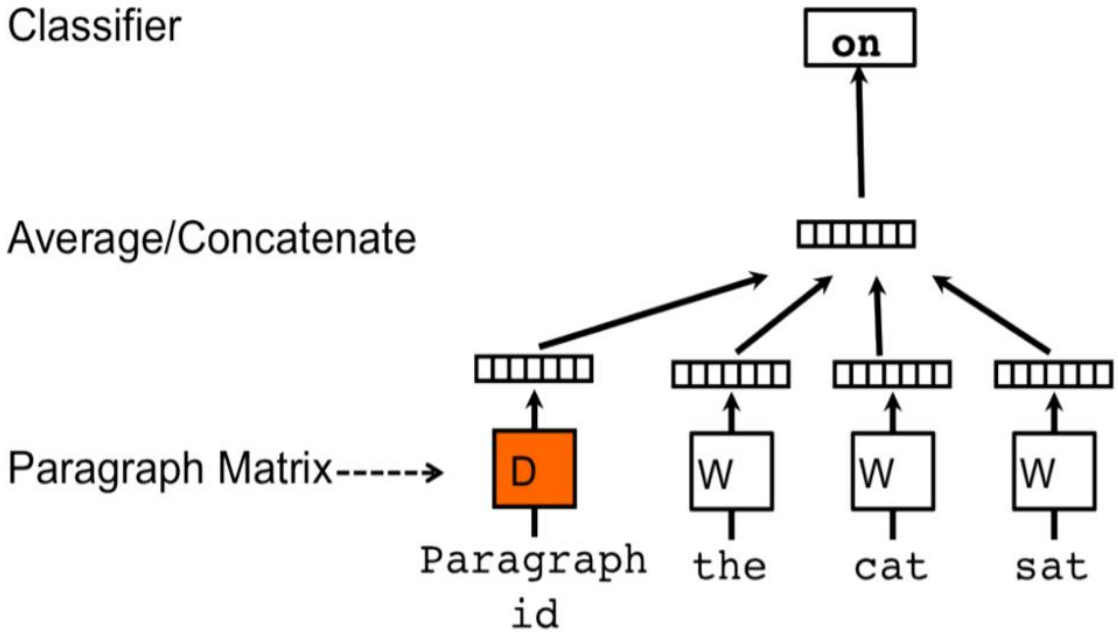
Doc2Vec

Paragraph Vector (Doc2Vec) is an extension to Word2Vec such that Word2Vec learns to project words into a latent d-dimensional space whereas Doc2Vec aims at learning how to project a document into a latent d-dimensional space. In this post, there are two document embeddings models from Paragraph Vector (more popularly known as Doc2Vec) — Distributed Memory (PV-DM) and Distributed Bag Of Words (DBOW)

PV-DM

The basic idea behind PV-DM is inspired from Word2Vec. In the CBOW model of Word2Vec, the model learns to predict a center word based on the context. For example, given a sentence “The cat sat on sofa”, CBOW model would learn to predict the word “sat” given the context words — the, cat, on and sofa. Similarly, in PV-DM, the central idea is : randomly sample consecutive words from a paragraph and predict a center word from the randomly sampled set of words by taking as input — the context words and a paragraph id.

Figure 17 : PV-DM example



Let’s look at the model diagram for some more clarity. In the given model, we see Paragraph Matrix, Average/Concatenate and Classifier sections. Paragraph matrix is the matrix where each column represents the vector of a paragraph. By Average/Concatenate, it means whether the word vectors and paragraph vector are averaged or concatenated. Lastly, the Classifier part takes

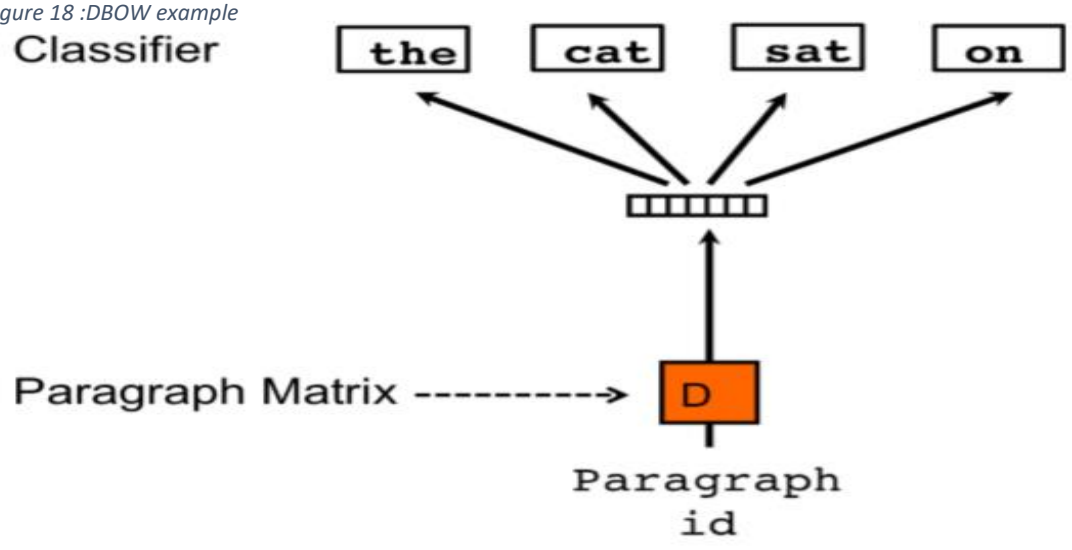
the hidden layer vector (the one that was concatenated/averaged) as input and predicts the center word.

Matrix D has the embeddings for “seen” paragraphs (i.e. arbitrary length documents), the same way Word2Vec models learns embeddings for words. For unseen paragraphs, the model is again ran through gradient descent (5 or so iterations) to infer a document vector.

DBOW

Distributed Bag Of Words (DBOW) model is slightly different from the PVDM model. The DBOW model “ignores the context words in the input, but force the model to predict words randomly sampled from the paragraph in the output.” For the above example, let’s say that the model is learning by predicting 2 sampled words. So, in order to learn the document vector, two words are sampled from { the, cat, sat, on, the, sofa}, as shown in the diagram.

Figure 18 :DBOW example



SBERT

in order to understand what SBERT is first we have to understand what BERT is, BERT which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection. (In NLP, this process is called attention.)

Historically, language models could only read text input sequentially -- either left-to-right or right-to-left -- but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once. This capability, enabled by the introduction of Transformers, is known as bidirectionality.

Using this bidirectional capability, BERT is pre-trained on two different, but related, NLP tasks: Masked Language Modeling and Next Sentence Prediction.

The objective of Masked Language Model (MLM) training is to hide a word in a sentence and then have the program predict what word has been hidden (masked) based on the hidden word's context. The objective of Next Sentence Prediction training is to have the program predict whether two given sentences have a logical, sequential connection or whether their relationship is simply random.

Background

Transformers were first introduced by Google in 2017. At the time of their introduction, language models primarily used recurrent neural networks (RNN) and convolutional neural networks (CNN) to handle NLP tasks.

Although these models are competent, the Transformer is considered a significant improvement because it doesn't require sequences of data to be processed in any fixed order, whereas RNNs and CNNs do. Because Transformers can process data in any order, they enable training on larger amounts of data than ever was possible before their existence. This, in turn, facilitated the creation of pre-trained models like BERT, which was trained on massive amounts of language data prior to its release.

In 2018, Google introduced and open-sourced BERT. In its research stages, the framework achieved groundbreaking results in 11 natural language understanding tasks, including sentiment analysis, semantic role labeling, sentence classification and the disambiguation of polysemous words, or words with multiple meanings.

Completing these tasks distinguished BERT from previous language models such as word2vec and GloVe, which are limited when interpreting context and polysemous words. BERT effectively addresses ambiguity, which is the greatest challenge to natural language understanding according to research scientists in the field. It is capable of parsing language with a relatively human-like "common sense".

BERT have achieved SOTA results in sentence pair regression tasks such as Semantic Textual Similarity. However, they both need to send two sentences into the network at the same time, which will cause huge computational overhead: it takes about 50 million to find the most similar sentence pair from 10,000 sentences (C 10000 2 = 49 , 995 , 000 C_{10000}^2=49,995,000 C100002 =49,995,000) An inference calculation, which took about 65 hours on V100GPU. This structure makes BERT not suitable for semantic similarity search, nor for unsupervised tasks, such as clustering.

A common method to solve clustering and semantic search is to map each sentence to a vector space so that the semantically similar sentences are very close. There are usually two ways to obtain sentence vectors:

Calculate the average of all token output vectors

use [CLS]Position output vector

However, UKP researchers have found in experiments that on the text similarity (STS) task, the results obtained by using the above two methods are not satisfactory. Even the Glove vector is significantly better than the plain BERT sentence embeddings

Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

introduced by Nils Reimers and Iryna Gurevych Ubiquitous Knowledge Processing Lab (UKP-TUDA) Department of Computer Science, Technische Universitat Darmstadt.

(SBERT), a modification of the pretrained

BERT network that uses Siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. This reduces the

effort for finding the most similar pair from 65

hours with BERT to about 5 seconds with SBERT, while maintaining the accuracy from BERT

SBERT adds a pooling operation to the output

of BERT to derive a fixed sized sentence embedding. there are three pooling strategies: Using the output of the CLS-token,

computing the mean of all output vectors (MEAN strategy), and computing a max-over-time of the

output vectors (MAX-strategy). The default configuration is MEAN.

And here is the API documentation

API Document

1. register

Register the user in the system

Request

Table 3 : register request

Method	URL
POST	users/signup/

Table 4 : register parameters

Type	Params	Values
POST	username	string
POST	password	string

Response

Table 5 : register response

Status	Response

200	{
	"success": "true",
	"status": "Registration Successful!",
	}

500	{ "error": {

	"name": "UserExistError" "Message": "A user with the given username is already registered"} } }

500	{
	"error":
	{
	"name": "MissingUsernameError"
	"Message": "No username was given"}
	}
	}

500	{
	"error":
	{
	"name": "MissingPasswordError"
	"Message": "No password was given"}
	}
	}

2. login

Authenticate the user with the system and obtain the token

Request

Table 6 : login request

Method	URL
POST	users/login/

Table 7 : login parameters

Type	Params	Values
POST	username	string
POST	password	string

Response

Table 8 : login response

Status	Response

200	{
	"success": "true",
	"token": <token>,
	"status": "Registration Successful!",
	}

400	Bad Request
401	unauthorized

token

Token will be used in subsequent requests in the system

3. Get Jobs

Get the posted jobs by a specific user, so this request require the token, to retrieve only posted job by the user who request

Request

Table 9 : Get jobs request

Method	URL
GET	users/jobs/

Table 10 : Get jobs parameters

Type	Params	Values
HEAD	token	string

Response

Table 11 : Get jobs response

Status	Response
200	**An array containing an object for each job** Example response:- [{ "title": "python developer", "description": "We need a Python developer to work on our payment gateway\n ", "_id": "60fd7aa7d747502aadba8c32",

```
        "userID": "60fb4f51cabc835f2a448cae",
        "resumes": [
            {
                "filename": "1627224743900$pythonic.docx",
                "path": "public/resumes/1627224743900$pythonic.docx",
                "percentage": 0.34041885,
                "_id": "60fd7aa7d747502aadba8c36",
                "jobId": "60fd7aa7d747502aadba8c32"
            },
            {
                "filename": "1627224743900$Peter cv.pdf",
                "path": "public/resumes/1627224743900$Peter cv.pdf",
                "percentage": 0.15951452,
                "_id": "60fd7aa7d747502aadba8c35",
                "jobId": "60fd7aa7d747502aadba8c32"
            },
            {
                "filename": "1627224743899$naguib.pdf",
                "path": "public/resumes/1627224743899$naguib.pdf",
                "percentage": 0.14930017,
                "_id": "60fd7aa7d747502aadba8c34",
                "jobId": "60fd7aa7d747502aadba8c32"
            }
        ],
        "__v": 1
    },
    {
```

```
        "title": "php developer",
        "description": " We are looking for a PHP Developer responsible for managing back-end services and the interchange of data between the server and the users.",
        "_id": "60fd7b3ad747502aadba8c39",
        "userID": "60fb4f51cabc835f2a448cae",
        "resumes": [
            {
                "filename": "1627224890127$CV-PHP-Developer.pdf",
                "path": "public/resumes/1627224890127$CV-PHP-Developer.pdf",
                "percentage": 0.5337484,
                "_id": "60fd7b3ad747502aadba8c3b",
                "jobId": "60fd7b3ad747502aadba8c39"
            },
            {
                "filename": "1627224890127$mean_stack.docx",
                "path": "public/resumes/1627224890127$mean_stack.docx",
                "percentage": 0.44921058,
                "_id": "60fd7b3ad747502aadba8c3c",
                "jobId": "60fd7b3ad747502aadba8c39"
            }
        ],
        "__v": 1
    }
]
```

401	Unauthorized

percentage

This is the relevance score of specific resume to the job description.

4. Post a job

Post a job request to open a job application, by uploading job title, description, and resumes to be ranked according to its relevance with the job description

Request

Table 12 : Post a job request

Method	URL
POST	users/jobs/

Table 13 : Post a job parameters

Type	Params	Values
HEAD	token	string
POST	Title	string
POST	description	string
POST	resumes	array <resumeObj>

Response

Table 14 : Post a job response

Status	Response
200	**The same response as get jobs request** It will create the job and return successfully even if you doesn't provide job description or title, as it will use the default values (job description for description and title for title)
401	Unauthorized
500	Internal Server Error **This will occur in case you didn't provide the resumes for the job**

5. Delete all jobs posted by a specific user

Delete all the jobs posted by a specific user who posted them, user is identified by the token sent in the request.

Request

Table 15 : Delete all jobs posted by a specific user request

Method	URL
DELETE	users/jobs/

Table 16 : Delete all jobs posted by a specific user parameters

Type	Params	Values
HEAD	token	string

Response

Table 17 : Delete all jobs posted by a specific user response

Status	Response

200	{
	"success": true
	}

401	Unauthorized

6. Get a Job by it's ID

Get the posted jobs by a specific user, so this request require the token, to retrieve only posted job by the user who request

Request

Table 18 : Get a Job by it's ID request

Method	URL
GET	users/jobs/<_id>/

Table 19 : Get a Job by it's ID parameters

Type	Params	Values
HEAD	token	string
URL_PARAM	_id	string

Response

Table 20 : Get a Job by it's ID response

Status	Response
200	**An array containing an object for each job** Example response:- { "title": "php developer", "description": " We are looking for a PHP Developer responsible for managing back-end services and the interchange of data between the server and the users.", "_id": "60fd7b3ad747502aadba8c39", "userID": "60fb4f51cabc835f2a448cae", "resumes": [{ "filename": "1627224890127$CV-PHP-Developer.pdf", "path": "public/resumes/1627224890127$CV-PHP-Developer.pdf", "percentage": 0.5337484, "_id": "60fd7b3ad747502aadba8c3b", "jobId": "60fd7b3ad747502aadba8c39" }, { "filename": "1627224890127$mean_stack.docx",

	"path": "public/resumes/1627224890127$mean_stack.docx", "percentage": 0.44921058, "_id": "60fd7b3ad747502aadba8c3c", "jobId": "60fd7b3ad747502aadba8c39" } }
401	Unauthorized
500	Internal Server Error **This will occur in case you sent incorrect job _id**

7. Update a job title or description

Update the job title or job description or both, if only the title is changed, no changes should be expected in the resumes relevance score, but if the job description changed, the NLP model will calculate the resumes relevance percentage to the updated job description.

Request

Table 21 : Update a job title or description request

Method	URL
PUT	users/jobs/<_id>/

Table 22 : Update a job title or description parameters

Type	Params	Values
HEAD	token	string
URL_PARAM	_id	string
POST	title	string
POST	description	string

Response

Table 23 : Update a job title or description reponse

Status	Response
200	**The same response as get job by ID request, it will return the updated job with the new updated fields and updated relevance scores for resumes in case you update job description**
401	Unauthorized
500	Internal Server Error **This will occur in case you sent incorrect job _id**

8. Delete a job by ID

Delete the job with the ID specified in the URL, which is posted by the user whose id is given in the token sent with the request

Request

Table 24 : Delete a job by ID request

Method	URL
DELETE	users/jobs/<_id>/

Table 25 : Delete a job by ID parameters

Type	Params	Values
HEAD	token	string
URL_PARAM	_id	string
POST	Title	string
POST	description	string

Response

Table 26 : Delete a job by ID response

Status	Response

200	{
	"success": "true",
	"msg": "removed successfully"
	}

401	Unauthorized
500	Internal Server Error **This will occur in case you sent incorrect job _id**

9. Add resumes to an existing job

Update the job by adding more resumes to a specific job to rank them also alongside with the existing resumes, so this request will trigger the NLP model

Request

Table 27 : Add resumes to an existing job request

Method	URL
POST	users/jobs/<_id>/resumes

Table 28 : Add resumes to an existing job parameters

Type	Params	Values
HEAD	token	string
URL_PARAM	_id	string
POST	resumes	array <resumeObj>

Response

Table 29 : Add resumes to an existing job response

Status	Response
200	**Job object with the new resumes added** Example response:- { "title": "php developer", "description": " We are looking for a PHP Developer responsible

```
for managing back-end services and the interchange of data between the server and the users.",
   "_id": "60fd7b3ad747502aadba8c39",
   "userID": "60fb4f51cabc835f2a448cae",
   "resumes": [
       {
           "filename": "1627227275576$PHPDeveloper-CV-Gehad-Al-Shepeny.pdf",
           "path": "public/resumes/1627227275576$PHPDeveloper-CV-Gehad-Al-Shepeny.pdf",
           "percentage": 0.6198988,
           "_id": "60fd848bd747502aadba8c5b",
           "jobId": "60fd7b3ad747502aadba8c39"
       },
       {
           "filename": "1627227275572$Mohamed Abdelhai Cv.pdf",
           "path": "public/resumes/1627227275572$Mohamed Abdelhai Cv.pdf",
           "percentage": 0.5726268,
           "_id": "60fd848bd747502aadba8c5a",
           "jobId": "60fd7b3ad747502aadba8c39"
       },
       {
           "filename": "1627224890127$CV-PHP-Developer.pdf",
           "path": "public/resumes/1627224890127$CV-PHP-Developer.pdf",
           "percentage": 0.5337484,
           "_id": "60fd7b3ad747502aadba8c3b",
           "jobId": "60fd7b3ad747502aadba8c39"
```

	},
	{
	"filename": "1627224890127$mean_stack.docx",
	"path": "public/resumes/1627224890127$mean_stack.docx",
	"percentage": 0.44921058,
	"_id": "60fd7b3ad747502aadba8c3c",
	"jobId": "60fd7b3ad747502aadba8c39"
	}
	],
	"__v": 2
	}

401	Unauthorized
500	Internal Server Error **This will occur in case you sent incorrect job _id or didn't pass the resumes in the body of the request**

10. Get specific resume

Get specific resume with it's ID for a specific job

Request

Table 30 : Get specific resume request

Method	URL
GET	users/jobs/<_id>/resumes/<resume_id>

Table 31 : Get specific resume parameters

Type	Params	Values
HEAD	token	string
URL_PARAM	_id	string
URL_PARAM	resume_id	resumeObj

Response

Table 32 : Get specific resume reponse

Status	Response

200	**an object of the resume**
	
	Example response:-
	
	{
	"filename": "1627227275572$Mohamed Abdelhai Cv.pdf",
	"path": "public/resumes/1627227275572$Mohamed Abdelhai Cv.pdf",
	"percentage": 0.5726268,
	"_id": "60fd848bd747502aadba8c5a",
	"jobId": "60fd7b3ad747502aadba8c39"
	}

401	Unauthorized
500	Internal Server Error **This will occur in case you sent incorrect job _id or resume_id**

11. Delete specific resume

Delete specific resume with it's ID for a specific job

Request

Table 33 : Delete specific resume request

Method	URL
DELETE	users/jobs/<_id>/resumes/<resume_id>

Table 34 : Delete specific resume parameters

Type	Params	Values
HEAD	token	string
URL_PARAM	_id	string
URL_PARAM	resume_id	resumeObj

Response

Table 35 : Delete specific resume response

Status	Response

200	**an object of the deleted resume**
	
	Example response:-
	
	{
	"filename": "1627227275572$Mohamed Abdelhai Cv.pdf",
	"path": "public/resumes/1627227275572$Mohamed Abdelhai Cv.pdf",
	"percentage": 0.5726268,
	"_id": "60fd848bd747502aadba8c5a",
	"jobId": "60fd7b3ad747502aadba8c39"
	}

401	Unauthorized
500	Internal Server Error **This will occur in case you sent incorrect job _id or resume_id**

Glossary

Conventions

- **Client** - Client application.
- **Status** - HTTP status code of response.
- All the possible responses are listed under 'Responses' for each method
- All response are in JSON format.
- All request parameters are mandatory unless explicitly marked as `[optional]`
- The type of values accepted for a *request* URL parameter are shown like this `<any>`

Status Codes

All status codes are standard HTTP status codes. The below ones are used in this API.

2XX - Success of some kind

4XX - Error occurred in client's part

5XX - Error occurred in server's part

Table 36 : Status Codes decription

Status Code	Description
200	OK
201	Created
202	Accepted (Request accepted, and queued for execution)
400	Bad request
401	Authentication failure
403	Forbidden
404	Resource not found

405	Method Not Allowed
409	Conflict
412	Precondition Failed
413	Request Entity Too Large
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

Model testing

Test Data

1- 6 Job Description

Game Developer, Data Scientist, Java Developer, Ruby On Rails, SCRUM Master & Pentester

2- 47 CVs of different Categories

a- 30 CVs Specific for the 6 Job Description (5 for each Job Descriptions)

b- 27 CVs of other fields

related to CS Field but not the selected Job Descriptions (ex: Bioinformatics)

in middle between CS Field & other fields (ex: Project Manager)

unrelated to CS Field (ex: Accounting)

this Varience in test data to make the test process discriminanting

this test data can be used for to generate many test cases if the CVs splitted into different sets & paired each set with different job Descriptions using combinatorics

So from CVs 4 CVs sets is generated:

Set 1: 35 CVs

Set 2: 36

Set 3: 36

Set 4: 47 all CVs

each contain the 30 CVs that related to the 6 Job Description & addedto it different CVs of different Categories

by assiging one of the job Descriptions to each CVs set this results 6 * 4 = 24 Test Case

Accuracy Function = 1 - ((smallest Index of relevant CVs - no. of of relevant CVs)/(no. of CVs - no. of of relevant CVs)) %

Problems:

1- doesn't consider similarities values only rank of CV relative to others

2- doesn't consider Rank of Top-ranked CVs from relative CVs only Rank of least Ranked CV

Model Results

Table 37 : Model Results

Models	Language d Model	Text Corpus Source	Hyperparameter	Training Time	Accuracy
1st	Word2Vec	(August/2008 - December/2017) SO Dump (15 GB)	Dimensions = 200 Window = 5 Epochs = 3	Pre-Trained (11hr)	95.70%
2nd	Doc2Vec	10% of StackOverFlow Posts (3GB)	Dimensions = 100 Window = 3 Min Word Count = 30 Epochs = 70 Min Alpha = 0.00025	4hr	54.75%
3rd			Dimensions = 300 Window = 5 Min Word Count = 1 Epochs = 100 Min Alpha = 0.00025	7hr, 40min (+ 7hr Vectorization)	54.2%
4th		1K Resumes DataSet (.csv 3MB)	Dimensions = 300 Window = 5 Min Word Count = 1 Epochs = 100 Min Alpha = 0.0001	5min	69.3%
5th		1K Resumes DataSet (.csv 3MB) + 2.3K MS Word CVs (.word 0.25GB)	Dimensions = 300 Window = 5 Min Word Count = 1 Epochs = 100 Min Alpha = 0.0001	10min	76.6%
6th			Dimensions = 300 Window = 5 Min Word Count = 3 Epochs = 100 Min Alpha = 0.0001	7min	75%

7th			Dimensions = 300 Window = 5 Min Word Count = 7 Epochs = 100 Min Alpha = 0.0001	5min	79.9%
8th		1K Resumes DataSet (.csv 3MB) + 2.3K MS Word CVs (.word 0.25GB) + 3.3K PDF CVs (.pdf 0.5GB) 6.5K CVs Total	Dimensions = 300 Window = 5 Min Word Count = 1 Epochs = 100 Min Alpha = 0.0001	15 min	83.5%
9th			Dimensions = 300 Window = 5 Min Word Count = 3 Epochs = 100 Min Alpha = 0.0001	10min	81.1%
10th	SBERT	base model was trained on 160gb of texual data training data:AIINLI, sentence-compression, SimpleWiki, altlex, msmarco-triplets, quora_duplicates, coco_captitions, flickr30k_captions, yahoo_answers_title_question, S2ORC_citation_pairs, stackexchange_duplicate_questions, wiki-atomic-edits.	Dimensions = 768 Base Model: microsoft/mpnet-base pooling: Mean Pooling	Pre-Trained	93%

Interpretation:

1st-

Accuracy seems High but this not actully accurate due to :

1- the accuracy function used doesn't consider all results properties as mentioned

2- for using word2vec model to represent a doc a naive averaging simple arithmetic mean for doc's word vectors is used which doesn't consider:

a- the relation among the words of the vectorized model

b- the importance of the words of the vectorized model

c- Syntatic ambguity

So this resultes that similarities of docs is high & close to each which makes similarities of docs isn't very discriminanting

2nd-

StackOverFlow posts data gives good results on single words level but doesn't give promising results on docs level due to the fact that CVs porperties (structure, vocabulary strength, non-technical termnology) highly varient from StackOverFlow Posts properties

3rd-

Propably Over Fitted Due to Large Epochs Number & Small Min Word Count

4th, 5th, 6th, 7th, 8th, 9th-

As Data Size Increase:

1- Accuracy Increase

2- Stability Increase

3- Similarity Values Increase

10th-

Best Model

References

[1] https://thedatasingh.medium.com/introduction-713b3d976323

[2] https://towardsdatascience.com/how-to-rank-text-content-by-semantic-similarity-4d2419a84c32

[3] https://en.wikipedia.org/wiki/Word_embedding

[4] https://core.ac.uk/download/pdf/55305289.pdf

[5] https://www.researchgate.net/publication/280941444_Learning_to_Rank_Resumes

[6] https://en.wikipedia.org/wiki/Learning_to_rank

[7]https://medium.com/@Synerzip/resume-ranking-using-machine-learning-implementation-47959a4e5d8e

[8] https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

[9] https://arxiv.org/pdf/1908.10084.pdf

[8] https://www.sbert.net/docs/pretrained_models.html

[9] https://www.kaggle.com/stackoverflow/stacksample

[10]https://medium.com/@amarbudhiraja/understanding-document-embeddings-of-doc2vec-bfe7237a26da#:~:text=Paragraph%20Vector%20(Doc2Vec)%20is%20supposed,a%20latent%20d-dimensional%20space.

[11] https://www.coursera.org/learn/server-side-nodejs

[12] https://www.coursera.org/learn/front-end-react