National University
of computer and emerging sciences

PROJECT TITLE:

Resume Recommender

GROUP MEMBERS:

Eisha Tir Raazia K17-3730
Maham Tariq K17-3658

TEACHER:

Ms. Rubab Jaffer.

Resume Recommender

1. Introduction

1.1Purpose

The purpose of this document is to describe the requirements for designing an automated system to extract information from resumes in unstructured form and transforming it into a structured format. Also we would be ranking those resumes based on the information provided by the job seeker like their skill set and experience etc. and the job description of the company.

1.2 Scope

The major objective of our system is to take the current resume ranking system to another level and makes it more flexible for both the entity.
1) Candidates, who are to be hired.
2) Client company, who is hiring the candidates.

Candidates, who are to be hired:
Candidates who are searching for jobs after graduation. Out of those, major number of candidates are so desperate that they are ready to work on any post irrelevant to their skill set and ability.
The main reason behind this unemployment is like a cancer to our society, if a person is not got place after been passed out for 1yr, society include relatives starting blaming that guy. In spite of this reason the candidate are ready to work in any conditions, on any post. So, they don't have to face those situations.
Where our system helps such candidates to get hired by a company or an organization who really worth their ability and their skill sets. Where our algorithm will work in such a way that with the help of the previous result and previous ranking constraints, it will try to optimize the current result, which we called it Machine Learning.
This will make sure that the relevant candidate has been hired for that particular vacancy. You can say the best possible candidate.

Client company, who is hiring the candidates:
Like I am the owner of a particular organization, obviously my aim would be to create such a team which is the best team in the world. It is like, if there is a vacancy of a java developer in my organization. So, I won't prefer to hire a python developer and then make him learn Java. That will be pretty useless and time consuming for both that candidate and for the organization too.
Where our system helps the organization to make out the best possible candidates list according to their given constraints and requirement for that particular vacancy.
This kind of approach will help our hiring sector to improve like anything and make it more efficient as the relevant person is getting a relevant job. So, there would be no regrets for both the entities, client company and that hired candidate. Hence satisfaction will be achieved.

1.3 Definitions, Acronym, and Abbreviations

API– Application programming interface. It refers to a set of rules that determine how pieces of software interact with each other.

UI – User interface. The visual aspect of a tool that a person uses for controlling it.

URL – Uniform resource locator. The web address used for identifying a website or page.

TF-IDF – Term Frequency-Inverse Document Frequency. The web address used for identifying a website or page.

1.4 References

React documentation
https://reactjs.org/docs/getting-started.html
Mongo DB documentation
https://docs.mongodb.com/
Python Documentation
https://docs.python.org/
Genism documentation
https://radimrehurek.com/gensim/apiref.html
ANTD react library documentation
https://ant.design/docs/react/introduce
Special mention
https://stackoverflow.com/
https://www.w3schools.com/
Elmasri and Navathe, "Fundamentals of Database Systems", 5th Edition, Addison- Wesley, 2007.

1.5 Developer's Responsibilities: An Overview

The developer is responsible for: developing the system, installing the software on the client's hardware, conducting any user training that might be needed for using the system through soft copy of technical manual/report and presentation slides, and maintaining the system for a period of one year after the installation, and provide any useful update.

2. General Description

2.1 Product Function Perspective

Following scenarios were collected after we visited many online job portals and analyzed many CVs:
All CVs have some common sections in them i.e. experience, education, skillset etc.

- Users are categorized based on their type. There are two types of users: Job seeker and job provider
- In job seeker scenario we have taken his general information and his CV as input and then we parse this information and fill the table of CV.
- In job provider scenario we directly input the job description and his general information like name, company etc.
- We created different tables for both types of user profiles.

2.2 User Characteristics

The main user of this system will be Job Provider and Job Seeker, and they can provide job by giving job description and other one can get job by submitting his/her CV.

2.3 General Constraints

The supported Operating Systems for client include:

- Windows 7 onwards
- Linux any flavor.

The supported browsers include:

- Mozilla Firefox
- Internet Explorer
- Microsoft Edge
- Google Chrome.

2.4 General Assumptions and Dependencies

Not applicable

3. Specific Requirements

3.1 Inputs and Outputs

- Job provider will input his name, contact, email, company name, use rid, password and job description.
- Job seeker will provide his name, contact, email, company name, use rid, password and his CV.
- In Output, Job Provider will receive the recommended CV's of the applicants according to their ranks.

3.2 Functional Interface Requirements

JOB Seeker

F1: It can create an account, login, post CV and maybe get hired by job provider.

JOB provider

F2:Job provider can create an account, login, post a Job and get recommendation.

3.3 Performance Constraints

NF1: updated information on the frontend should be reflected in the database and vise-versa. This should not take more than a few seconds.

NF2: A simple query will respond in a few seconds.

3.4 Design Constraints

3.4.1. Software Constraints

The system is to run under the windows 7 or above operating system, front end is React Native and back end on Python and the DBMS used is mongo DB.

3.4.2 Hardware Constraints

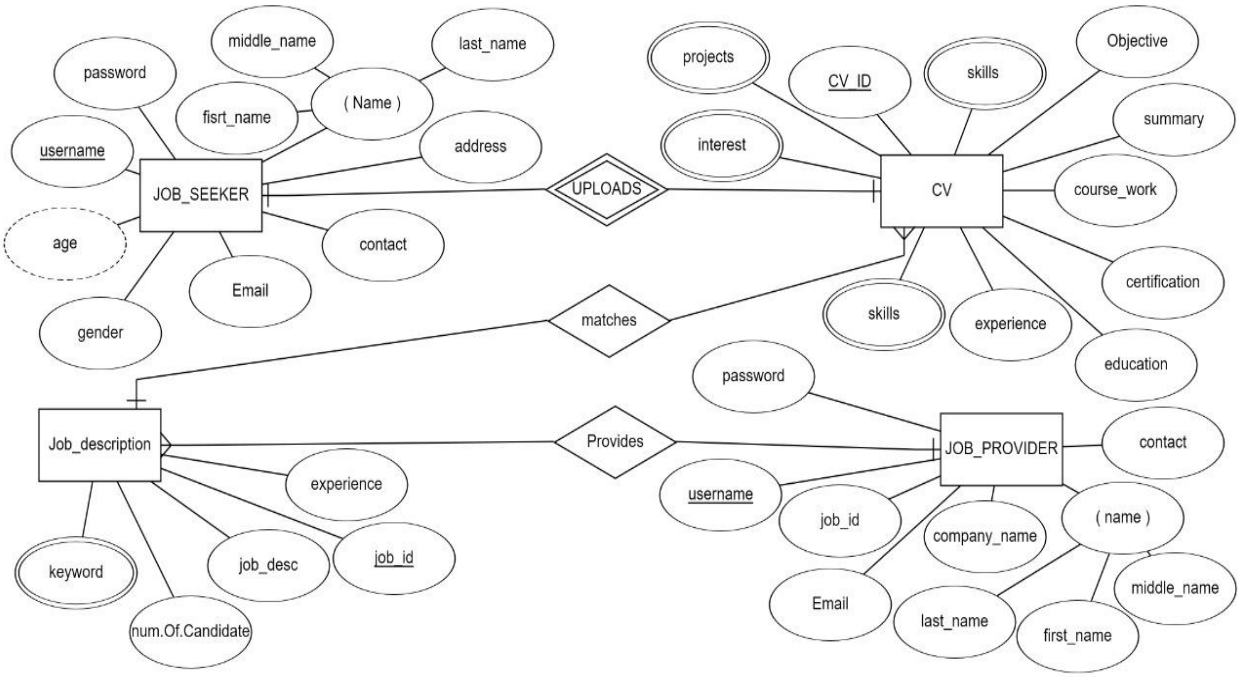
The system will run on core any workstation with i3/i5/i7core along with 512 or greater RAM, running Windows 7 and onwards.

3.5 Acceptance Criteria

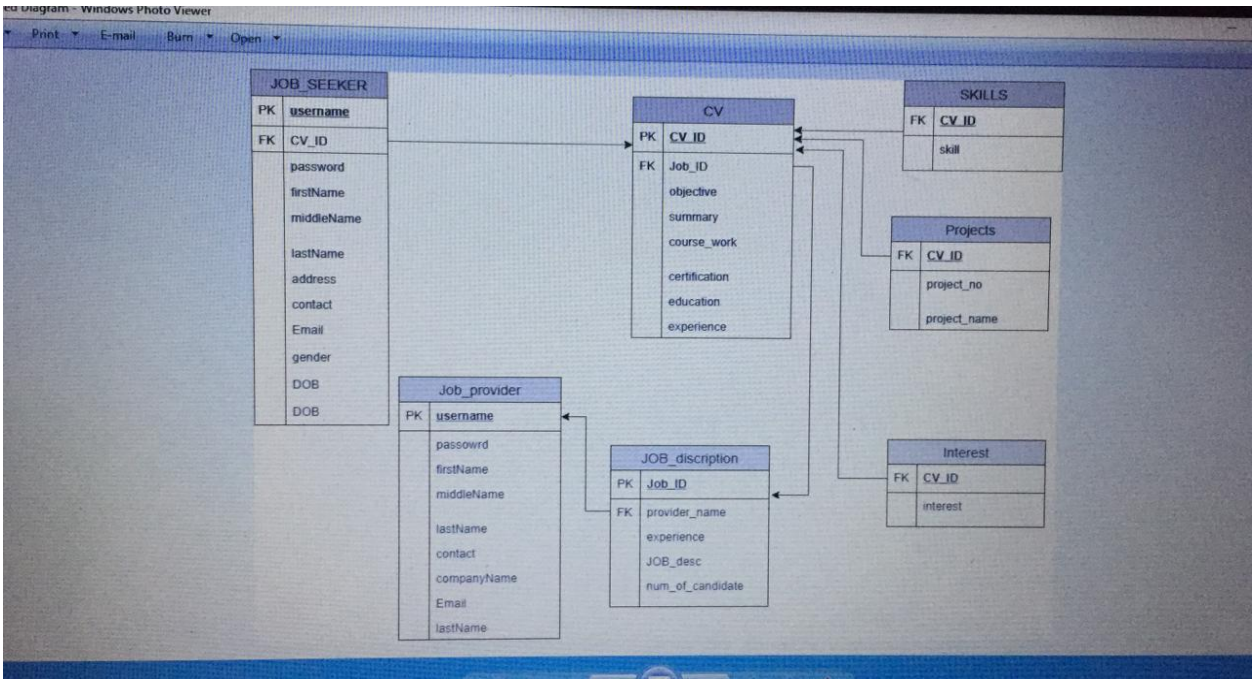
Before accepting the system, the developer must demonstrate that the system works on the data provided by both the job seeker and job provider, and that when the Job provider presses the Recommend Button after uploading the job description, they will be shown the top candidates ranked according to their scores given by the algorithm to their CVs.

4.0 System Design

4.1 ER Model



4.2 Schema Description



4.3 Tables Description

Already shown in relational schema.

4.4 System Flow

The system flow of our project is that firstly Sign up page will be displayed where both Users (Job seeker and Job Provider) will create an account and enter their respective details. Job Seeker will upload his details and CV/Resume which we will store in the MongoDB collection. Then by scraping data from the CV, we will identify his attributes like experience and his skillset etc. which will also be stored in another collection with username as a foreign key. Job provider will upload his details like company name etc. along with job description for which they are looking to hire a candidate. Both the things will be stored in different collections with JPId as foreign key in the Job description collection. Then the Job provider will click on the recommend button and he/she will be shown the top ranked job seekers along with their CV's.

4.5 User Interface

SignUp Page:
SignUp details are entered by user to create an account.
Login Page:
Form: where user enters a username and password.
Login Button: When clicked, verifies the details and login the user in.
Job_Seeker Profile page:
User is displayed his profile and a post CV button is placed which is used by the user to upload his CV.
Job_Provider Profile page:
User is displayed his profile and an upload Job button is placed which is used by the job provider to upload a new job description.
Job_Upload Page:
Form: where user will enter job description details and along with this a submit button is provided to save the job description provided. All the jobs uploaded by the job provider is displayed along with a get recommendation button which will displayed job provider all the top ranked applicants.
LogOut Button:
Log-out button is displayed on all the pages.

4.6 Error Messages / Alerts
No job seeker is shown if any of the applicant does not match with the job description provided.

4.7 Test Cases
4.7.1

TC ID#	Scenario / Condition	Username	Password	EXPECTED RESULT
1	F1: Successful Login.	V xyz	V abc	User is directed to his profile
2	F2: Successful Signup.	NA	NA	User is directed to his profile

	Scenario	username	password	message
3	S3: Incorrect Username	I XXX	NA	error message: userID or password incorrect
4	S4: Incorrect Password	V XXX	I NA	error message: userID or password incorrect
5	S5:Any field is empty in signup	NA	NA	error message enter the field
6	S6:The delete query not run successfully	NA	NA	error message:Delete can not be performed

Note: V = Valid, I = Invalid data, NA = Not applicable.

4.7.2 Descriptive Test Cases

ID	Test Case Description	Expected Results	Actual Results	Pass/Fail
1	Conditional Rendering as 'Recruiter'.	The welcome page should render with the 'Get Recommendation' button.	The welcome page rendered with the 'Get Recommendation' button.	Pass
2	Conditional Rendering as 'Applicant'.	The welcome page should render with the 'Post CV'' button.	The welcome page rendered with the 'Post CV'' button.	Pass

3	Email as a user's unique ID, with one 'user type'(Applicant/Recruiter) against one email.	Two user types cannot be assigned to a single email/ User cannot signup again with different 'user type'.	The user couldn't signup again with different 'user type', using same email.	Pass
4	The user clicks the submit button on the signup form without filling all the fields.	POST request shouldn't be made and an error message should appear on screen to identify unfilled entities.	Left/unfilled fields are highlighted with an error message and ask user to fill the fields in order to submit signup form.	Pass
5	The user cannot log in without signing up.	Error msg is shown and the user is redirected to the signup page.	Error msg is shown and the user is redirected to the signup page.	Pass

4.8 Risks Analysis

Risk Summary	Risk Category	Probability	Impact (1-4)	RMMM
Inappropriate/graphical representation of skills resulting in low score of a strong resume.	Process definition	0.25	3	Before uploading the resume applicant is asked to make sure that his/her resume doesn't contain algorithm unfriendly representations or graphics.
A single user signing up with two different emails and user names (violation of privacy of account features and functionalities according to different user types)	Development environment	0.8	2	System can ask for user's NIC number and can make it primary key instead of email to avoid the risk of single user violating the rule of not having two user types at the same time

Impact – (1) catastrophic (2) critical (3) marginal (4) negligible

5. System Implementation

5.1 Hardware and Software Platform description

Our UI is made on React, while we are entering data in mongo dB database using python. Connectivity is done using FLASK between REACT and python. All the algorithm behind the recommendation system is written in python.

5.2 Tools used

We will work on web-based environment with the help of REACT. For data storage we will use MongoDB. And for the implementation of algorithms we will use python and its various libraries.

5.3 System Verification and Testing (Test Case Execution)

5.4 Future work / Extension

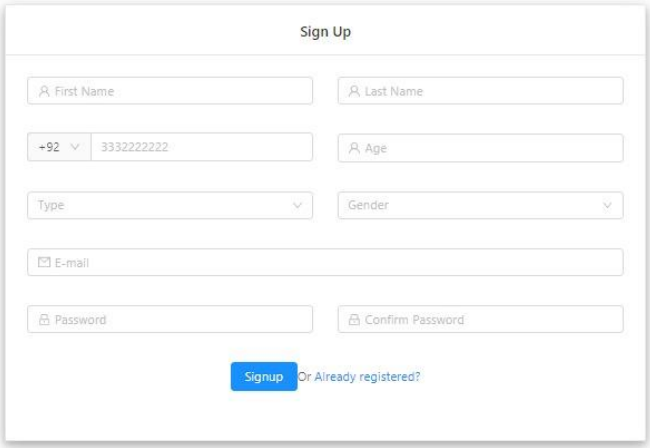
Finding a job is a complex process, affected by both explicit and implicit factors. Our work establishes the validity of using information extraction techniques to create a more personalized job matching system, with ample potential for improvement in the future.
First we can introduce a more complex job and resume model for improving performance of the system. In the resume model, we can consider hiring history and project experience of the job seekers. To improve the job description model, job responsibilities and company characteristics (size, dress code, etc.) should be considered as well.
Second, to improve searching speed of our system, we can reduce the number of comparison by filtering out jobs that are clearly not related to resumes. The system can classify the jobs into subsets, the system only need to calculate the similarity between the resume and according subset of jobs. CV/Resume recommender system is a content based recommendation system that is mostly focused on comparing the similarities between the resume and a relevant job description.
In future work, we could introduce a hybrid recommendation system that would take advantage of other recommendation algorithms such as Collaborative Filtering. Future work on this system would place greater consideration on job seeker's personal preference like job location, career development plan, and company background
The systems designed so far extracts all the information about the candidate only through his/her resume and after extraction it stores the information in a centralized database, finally ranking them and giving the top results to the HR recruiter according to their specifications. Future advancements in this system can be as follows:
1. The profiles of the candidates can be tracked on social media sites as this will help in analyzing the personality of the candidate and whether he/she is a perfect fit for the post can be judged.
2. Analysis can be done over the past records of the candidate which will help us determine his expected tenure of work in the organization.

5.5 Conclusion

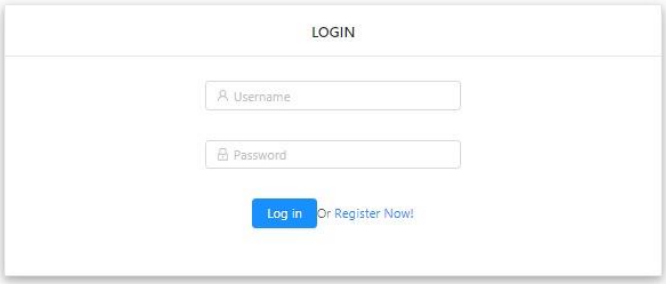
This work has made an extensive effort to provide a system through which CV's can be ranked with maximum efficiency. By implementing this system, the task of obtaining the most relevant resumes can be achieved which will save the recruiter time to manually select appropriate resumes, which even after processing may not be a complete fit for the profile. Since, there are multiple levels of screening involved in order to find the most relevant resumes; the accuracy of the system also improves. Thus, using this ranking technique, we can obtain the best results for obtaining the ideal resumes. This approach will automate as well as speedup the process of the HR recruiters.
In the system, job descriptions and resumes are parsed into job models and resume models by the information extraction module.
We did the following:
1. We developed a resume - job matching system.
2. We used a TF-IDF based matching tool to extract information from unstructured data source.

ANNEXURE-A

CURRICULUM VITAE RECOMMENDATION SYSTEM

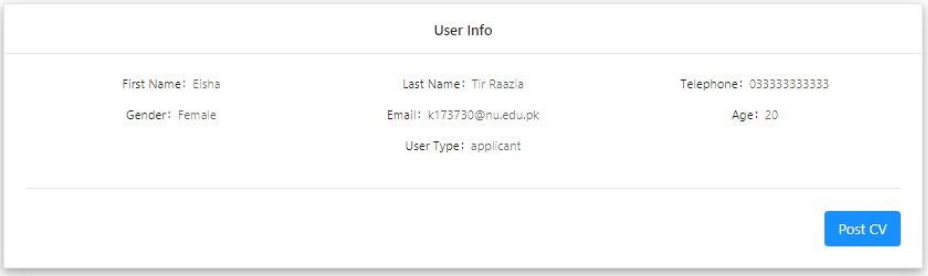


CURRICULUM VITAE RECOMMENDATION SYSTEM



CURRICULUM VITAE RECOMMENDER

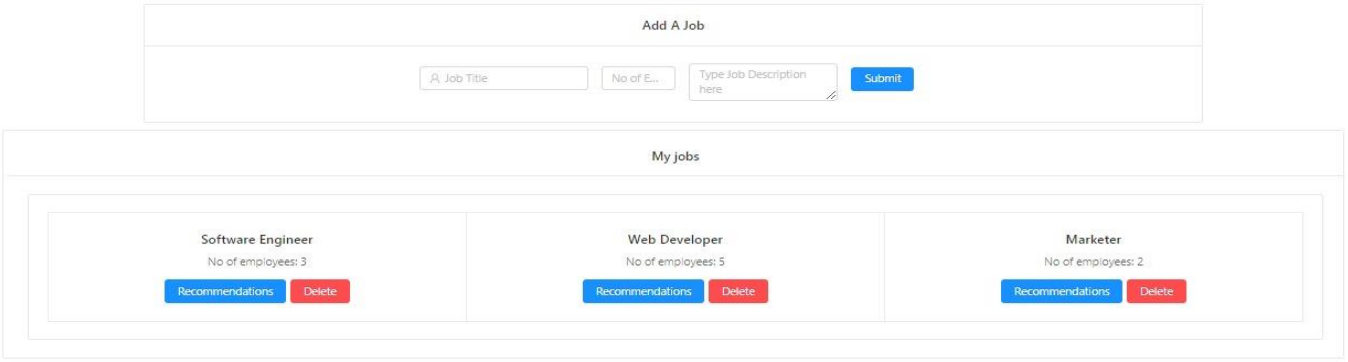
Logout



CURRICULUM VITAE RECOMMENDER

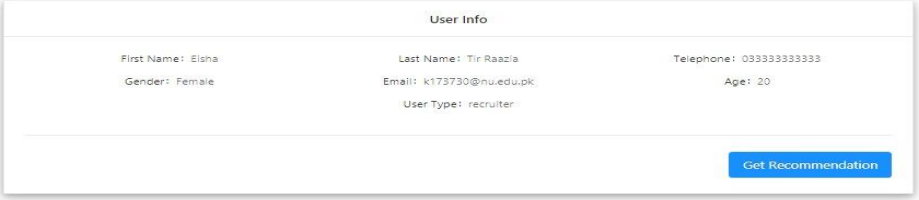
Logout

Recruiter's Page



CURRICULUM VITAE RECOMMENDER

Logout



ANNEXURE-B

Create Collection queries:
These queries are firstly run on mongo dB server to create collections explicitly.

```
db.CreateCollection("Job_Seeker")
db.CreateCollection("Job_Provider")
db.CreateCollection("Job_Description")
db.CreateCollection("CV_att")
```

Python Implementation:

Connection with the server:
```
import pymongo as pym
client = pym.MongoClient('mongodb://localhost:27017/')
db = client.test
```

Getting the collections from the database:
```
Job_Description=db["Job_Description"]
Job_Provider=db["Job_Provider"]
Job_Seeker=db["Job_Seeker"]
resume=db["CV_att"]
```

Insert data in Job Description collection:
```
insert ={  "JobProvider_id" : jpid,
           "job_title" : job_title,
           "desc" : job_desc,
           "cand" : no_cand
        }
Job_Description.insert_one(insert)

x = Job_Description.find({"JobProvider_id" : jpid})
```

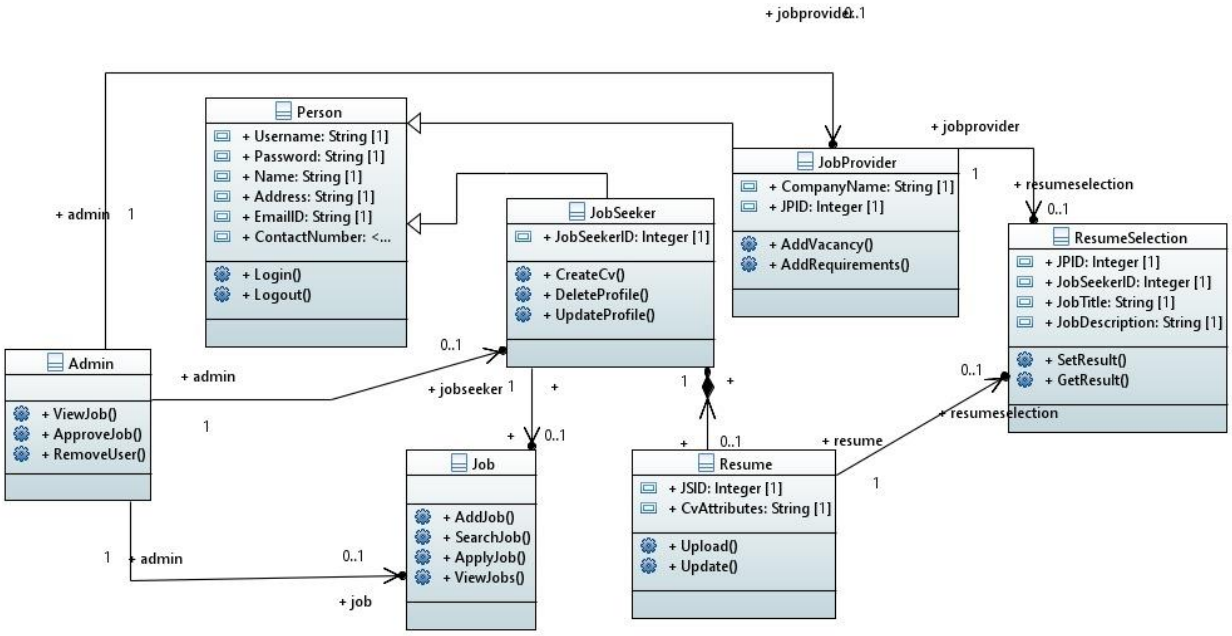
Insert data in Job Seeker collection
```
Job_Seeker.insertOne(
        {   fname : fname,
            lname : lname,
            number : number,
            gender : gender,
            email : email,
            age : age,
            password : password
        })
```

Insert data in Job Provider collection
```
Job_Provider.insertOne(
        {   fname : fname,
            lname : lname,
            number : number,
            gender : gender,
            email : email,
            age : age,
            password : password
        })
```

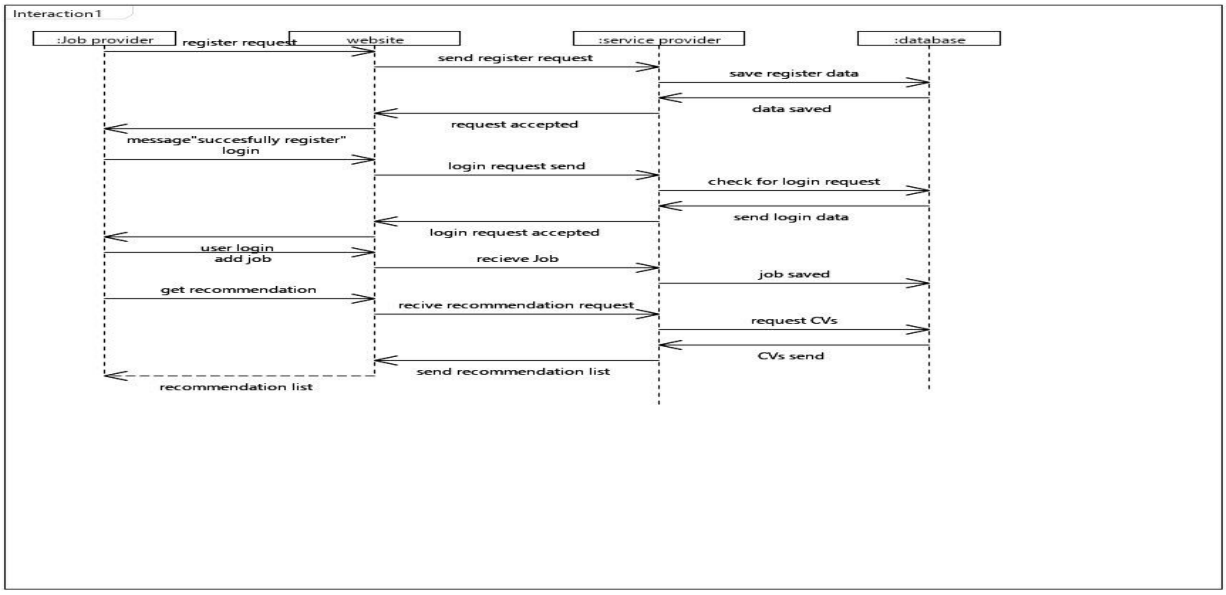
```
Job_Provider.find({ username : uname, password : password })
Job_Seeker.find({ username : uname, password : password })



insert ={       "uid" : uid,
                "cv" : resume,
                "name" : name,
                "email" : email,
                "skills" : skills,
                "education" : edu,
                "entities" : entities
        }
resume_collection.insert_one(insert)
```

Diagrams

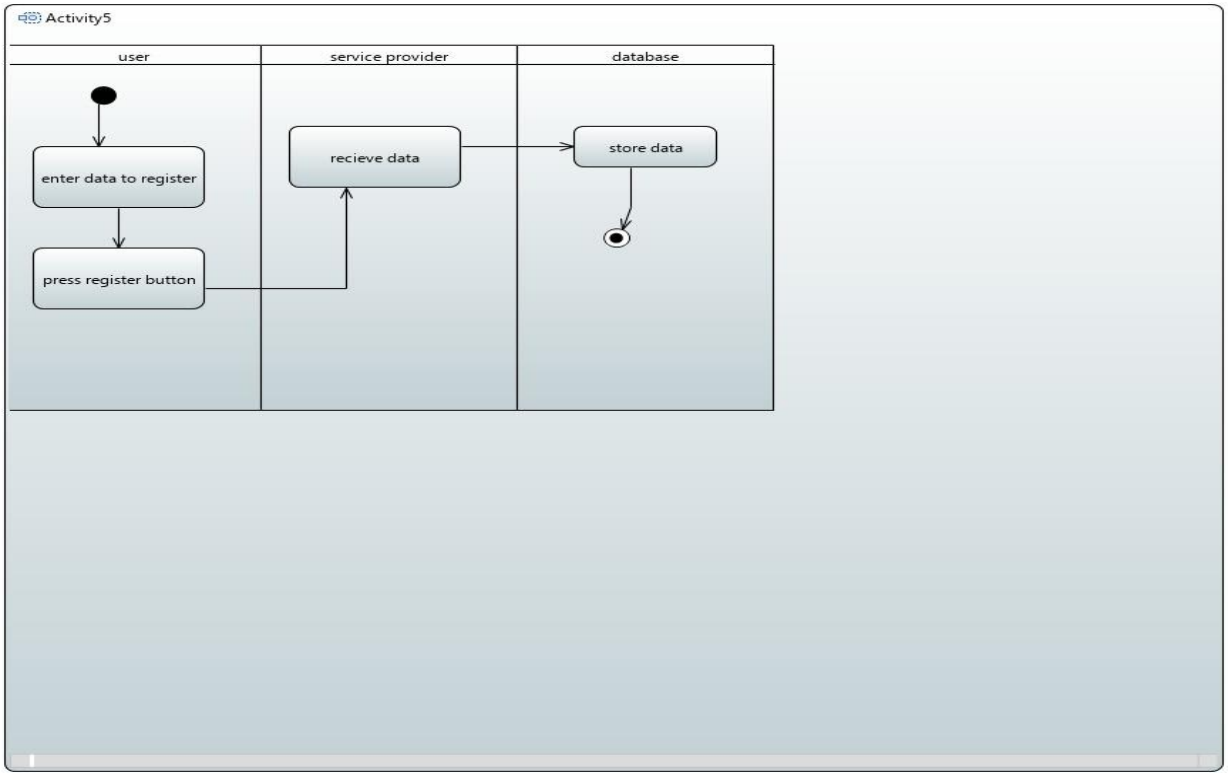
UML

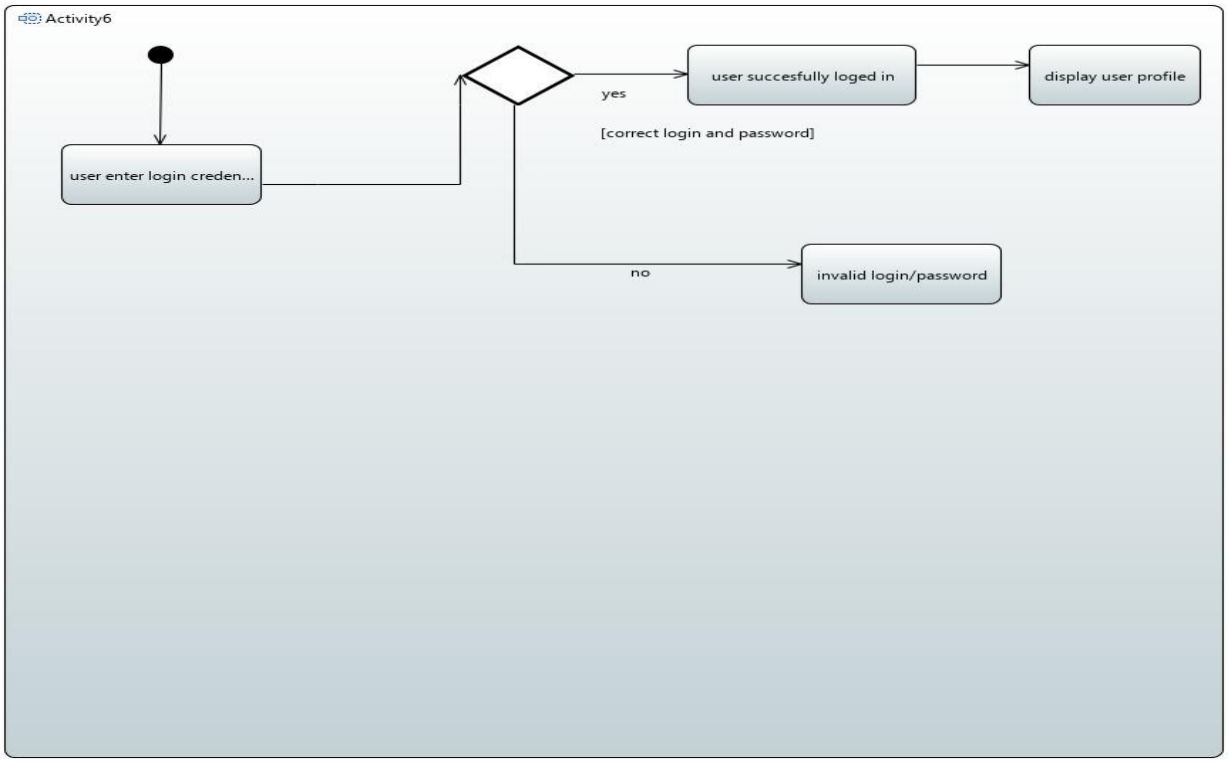


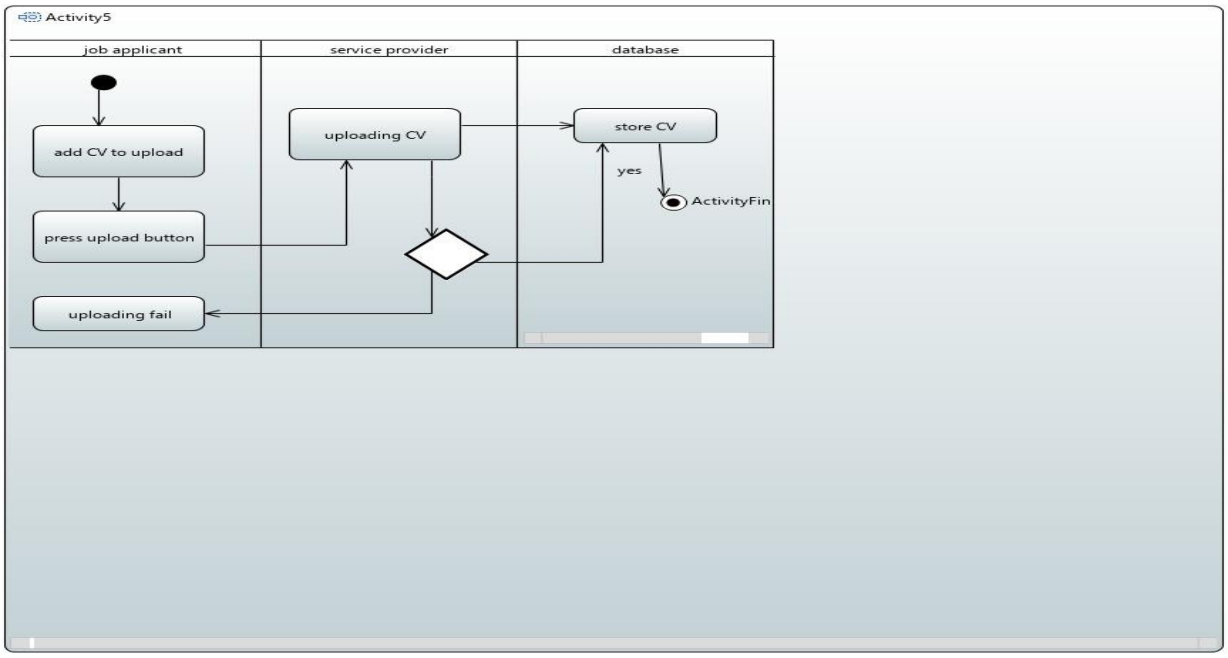
Sequence Diagram

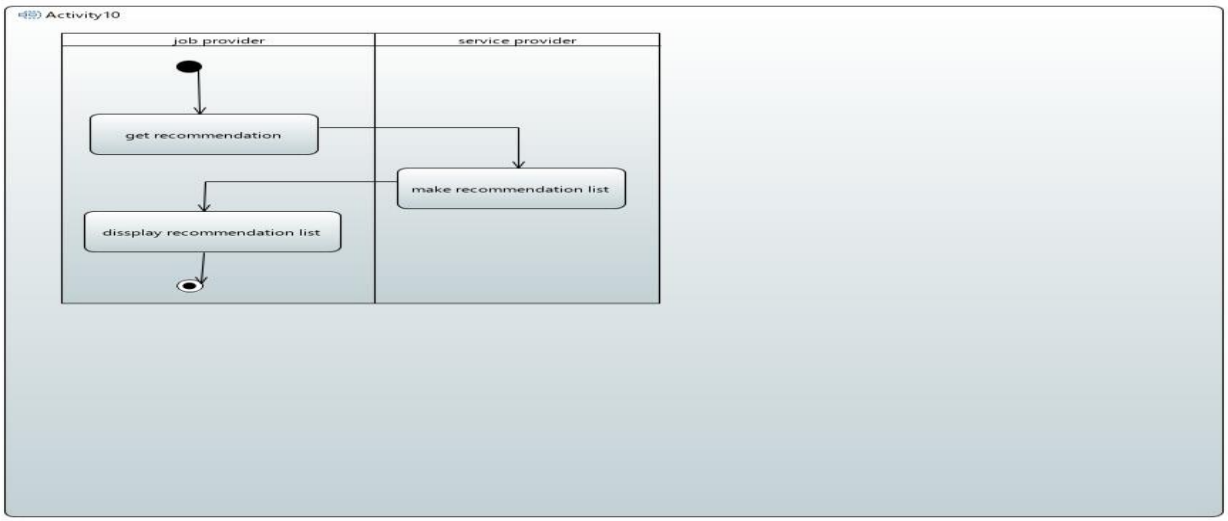


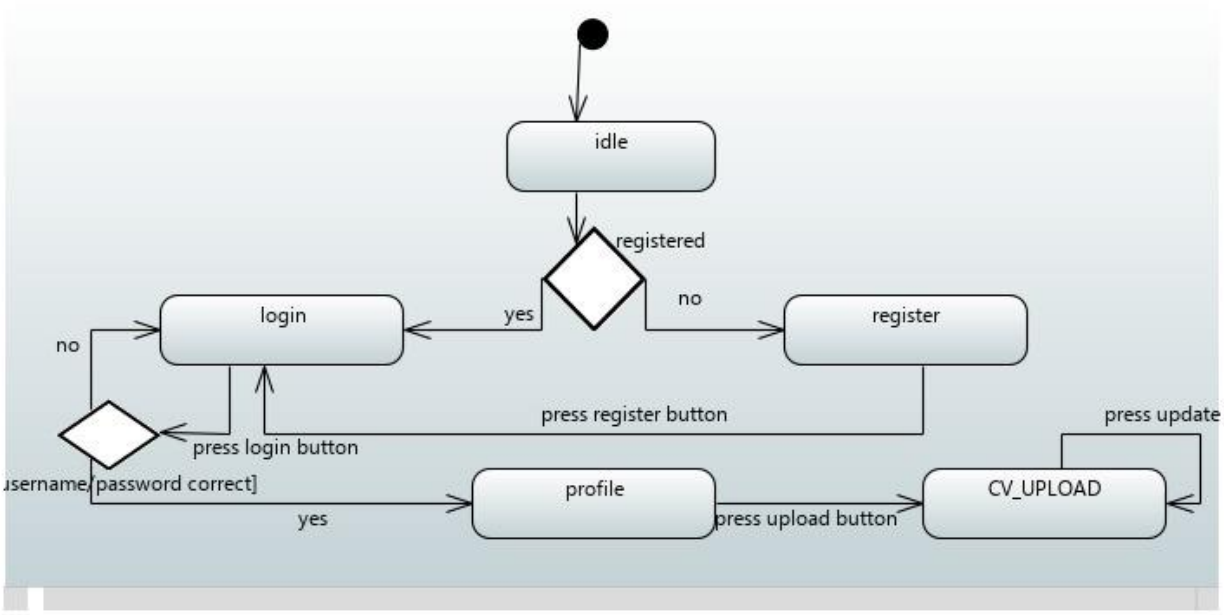
State Diagrams

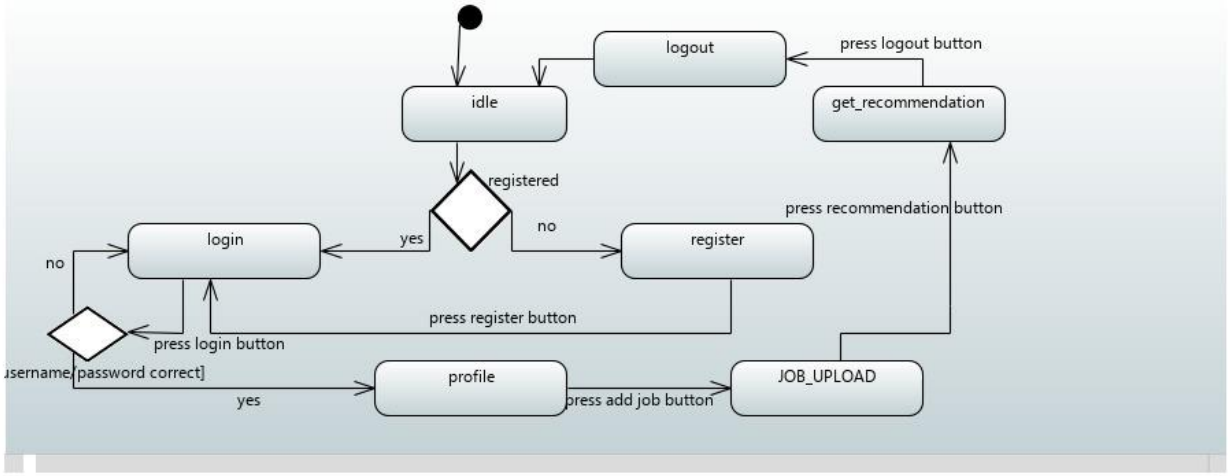




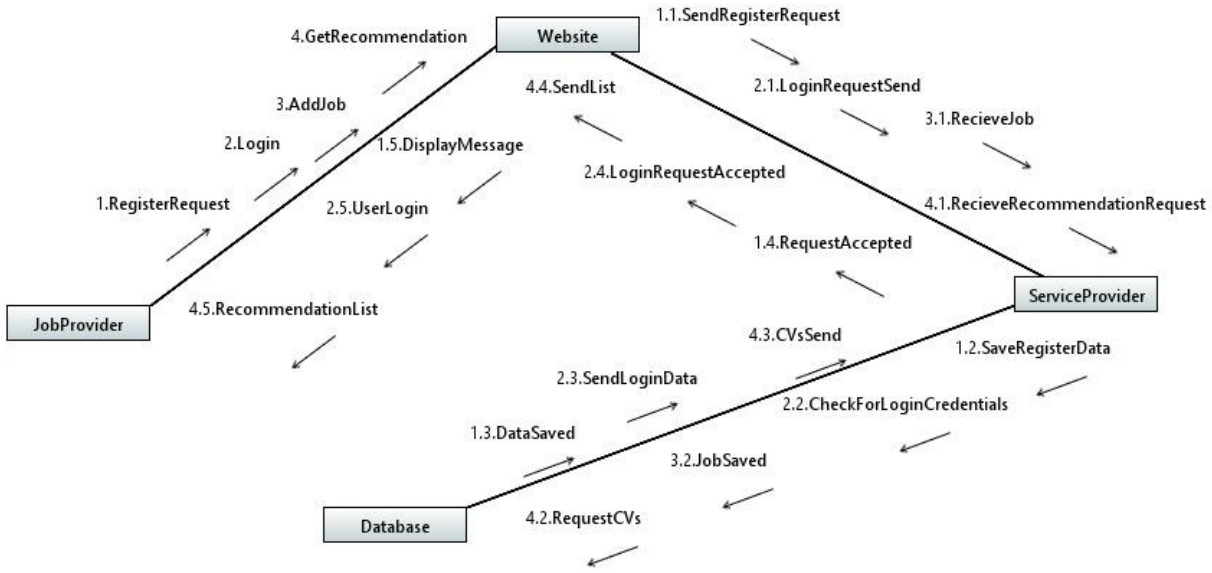

Activity5
job applicant
service provider
database
add CV to upload
press upload button
uploading fail
uploading CV
store CV
yes
ActivityFin

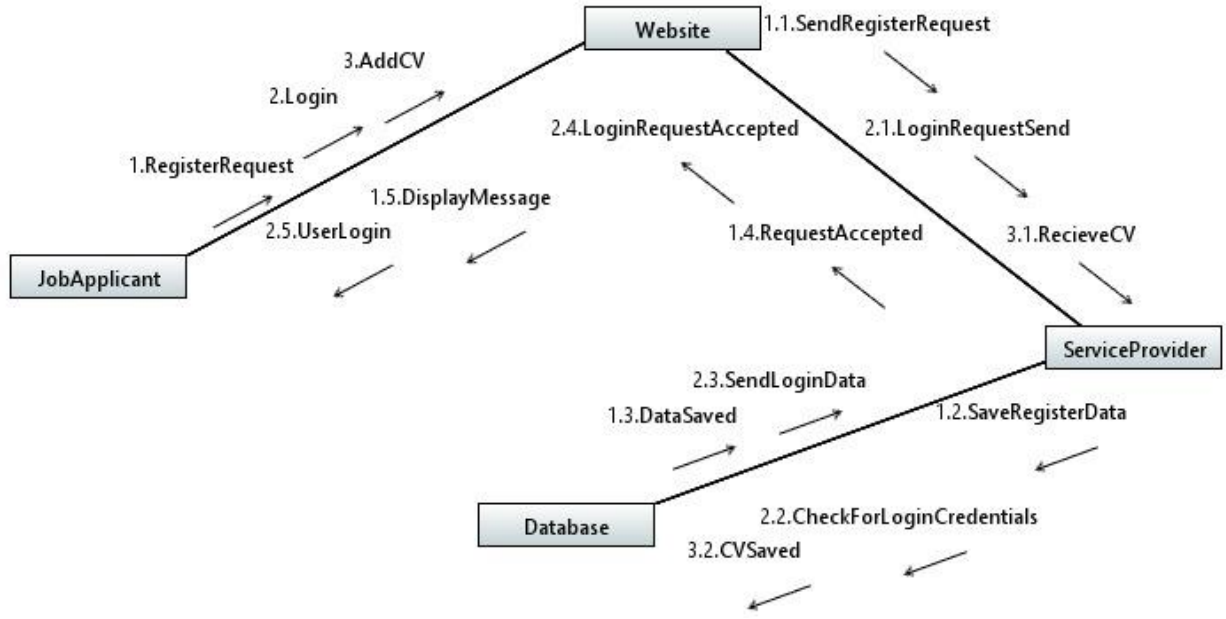

Activity10
job provider
service provider
get recommendation
make recommendation list
dissplay recommendation list


idle
registered
yes
no
login
register
no
press register button
press login button
press update C
username/password correct]
yes
profile
press upload button
CV_UPLOAD

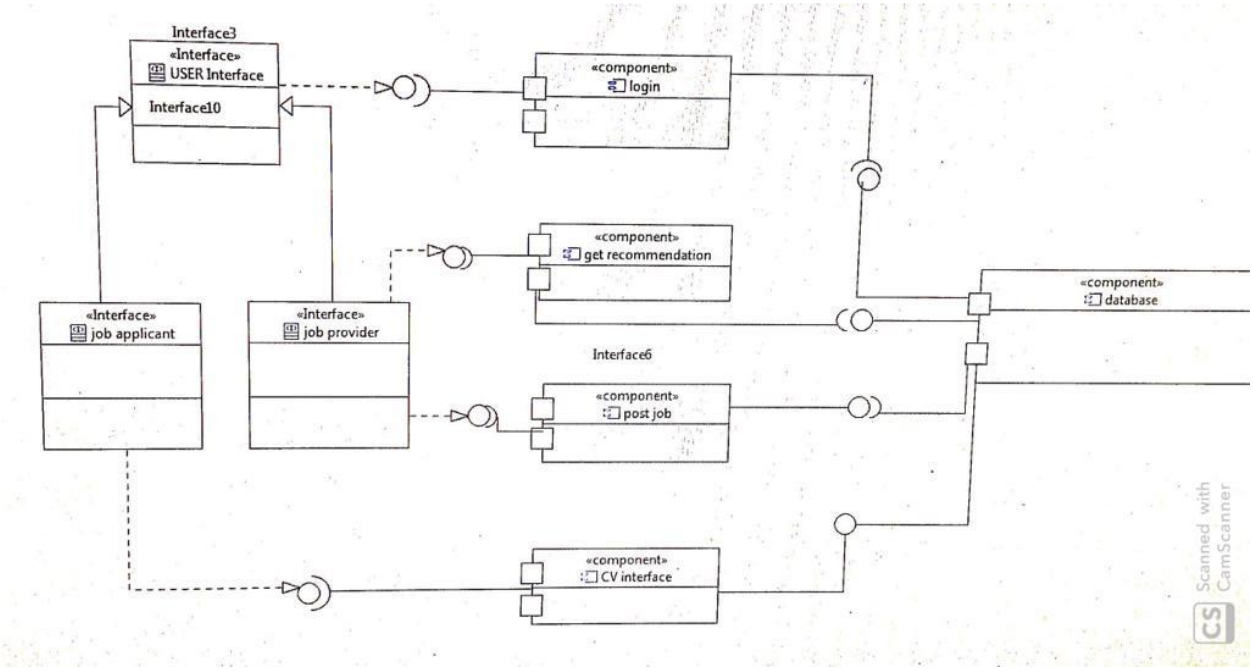

logout
press logout button
idle
get_recommendation
registered
press recommendation button
yes
no
login
register
no
press register button
press login button
username/password correct]
yes
profile
press add job button
JOB_UPLOAD

Communication Diagrams





Component Diagram



Deployment diagram