Job Consultancy System

Algorithm and Problem Solving Lab

23/05/16

Submitted By:

14103118	Vaibhav Garg
14103170	Anchit Goyal
14103174	Archit Vishnoi
14103168	Aman Singh

Submitted To :

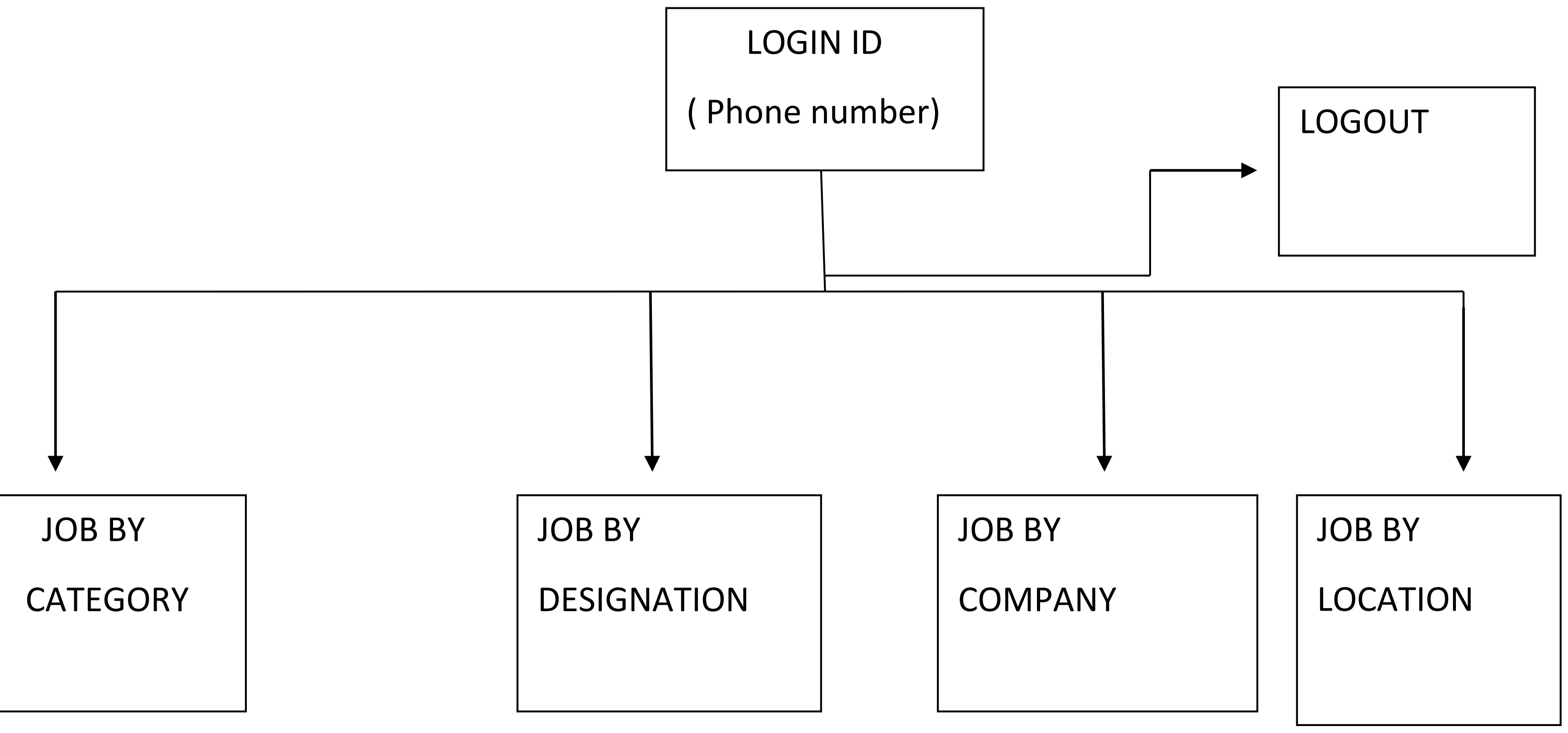
Ms. Sakshi Agrawal

Contents

PROBLEM DESCRIPTION -

This project is C++ application in which user can find his/her favourite job by sitting at his home. User will enter all the details and according to the experience and the priority of the companies selected, he/she will be given the job. Also this application will give the user the shortest route to choose to reach to his/her office. It is the perfect application to get ahead of the others and it is also an ability to perform and measure the success among different companies by their respective CVs.

Schematic Diagram



Functions –

JOB BY CATEGORY :-
This option consists of different variety of jobs for the candidate.

There are 5 categories of Jobs under which 2 posts of each categories are there and each one of them have their score according to which the job will be given.

Company have their own criteria (score criteria).
if candidate score > company criteria score
he will be eligible for the interview.

JOB BY COMPANY:-

Firstly, there will be a short quiz consisting of 5 questions and according to that, score will be allotted.

Also there will be a prealloted score of the company.
Then the function consists of two options in which one can be eligible for the company.

1) Search the company.
2) Select from the list.

In the 1) option, some initials of the company will be given and then **KMP** search will give the company name if present and also if the score of the quiz is >= company score , then the person will be eligible for the interview.

In the 2) option, list of all the company will be shown and the user will select one of the company. Again if the score of the quiz >= company score, then he will be called for the interview otherwise he will be rejected.

JOB BY LOCATION :-

We have 10 locations and one head centre i.e. KAROL BAGH. From head centre he will apply for different destination to find the job.

Haversine Formula is applied to calculate the distance between two coordinates which is Longitude & Latitude .
After calculating several distances we applied **DJIKSTRA ALGORITHM** which show the minimum distance between source point and destination point .

JOB BY DESIGNATION :-

Job will be searched according to the designation provided by the user. If the designations are empty, user will set his/her preference for the different companies.

Company which are having the given designation empty will ask the user for their resume.

Then the company will calculate the score according to the resume provided by the user and will set the preference for the different users.

STABLE MARRIAGE PROBLEM will be applied for assigning the job to different user according to the highest preference set by the company.

Note :

dos.h header file in the code is used for the **Sleep** function which will delay the printing of the output.

time.h header file in the code is used for the **random** function which will generate the random numbers.

windows.h header file in the code is used for the **system** function which will clear the screen.

Structure Used in the code-

1) struct **category**

```
{
string cname;
string cjob[100];
double cpoints[100];
};
struct category cat[5];
```

2) struct **record_number**

```
{
 char nmbr[100];
int id;
};
struct record_number *r,*r1;
```

3) struct **desgn**

```
 {
 char id[10];
string name,post[100];
};
```

```
struct desgn designation[10];
```

4) struct **company**

```
{
char cmp_name[100];
int cmp_id;
int status;
int prefer;
int companysetpreferencearray[4];
int b[4];
int score;
};
struct company cmp[4];
```

5) struct **employee**

```
 {
string emp_name;
string email;
int status;
int experience;
string achieve[100];
int id;

int a[4];

};  struct employee emp[4];
```

6) struct **edge**

```
{

   int eno,p1,p2;

   float p1_lat,p1_lng;

   float p2_lat,p2_lng;

};
```

7) struct **node**

```
      {
         char place[30];

    int num;

          float lat,lng;
      }node[10];
```