UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SECJ2013 DATA STRUCTURE AND ALGORITHM

SESSION 2022/2023

MINI PROJECT

"E-RECRUITMENT SYSTEM"

GROUP 07

GROUP MEMBERS:

KHALED MAHMUD SUJON (A20EC4082)
MAHMUDUL HASAN (A20EC4082)
AMMAR MOSTAFA (A21EC0250)

SECTION: 08

LECTURER'S NAME:

DR. Bander Ali

Table of Contents

Contents **Pages**

1.0. INTRODUCTION

1.1. Synopsis of the Mini Project

We had to create a medium-sized to a sophisticated application using many data structure ideas in the C++ programming language as part of the course's final project this semester. We have decided to use the topics of data structures to create a program for an electronic recruitment system.

We have digitized the applicant recruiting process for this project. The hiring panel receives the applications once the applicant submits them via a job portal. They review the applications and supporting documentation before choosing which candidates the HR manager should contact.

This project is based on a system in which a Hiring Panel can add Applicants' information into an Applicant list that is created by the system and can also display and delete Applicants' information from the list. The HR manager can access and check the list of all applicants, then add applicants they want to hire into another list that is for hired applicants.

Stack is used to adding applicants' information to the Applicant file and/or to delete them in the Last in First Out movement and also to display the list of Applicants. while queue data structure is used to create a new list for shortlisted applicants in queue once the HR manager chooses whether and who to hire or not and also to display the full list of applicants if the HR manager wants to see the list.

1.2. Objectives of the Project

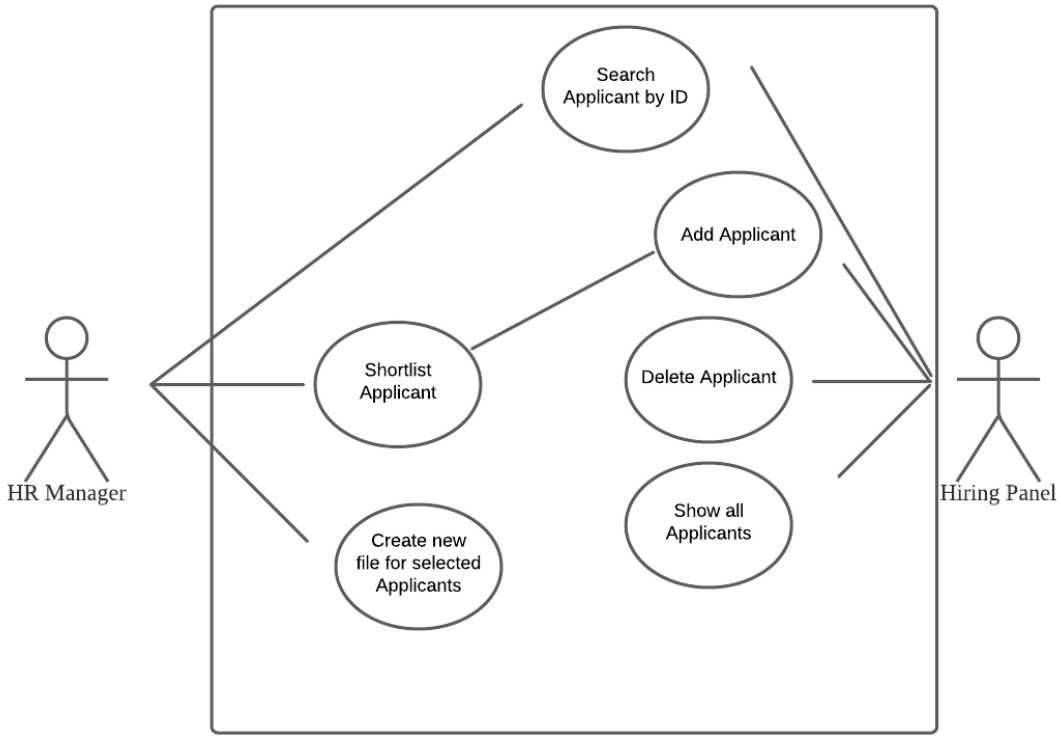
The objectives of this project are:

- Implementing the concept of Data Structure and Algorithm into a real-world situation.
- Developing a medium to complex size program on E-recruitment system by applying data structure concepts such as, stack and queue, in C++ programming language.
- Improving the skills of solving a problem in a group

2.0. SYSTEM ANALYSIS AND DESIGN (USE CASE, FLOWCHART AND CLASS DIAGRAM)

2.1. System Requirements

Use Case Diagram



Use Case Description for E-recruitment System

The system users are Hiring Panel and HR Manager.

Actor	Task
Hiring Panel	Hiring Panel can add Applicant's information into Applicant file or delete the information from the file and also can display Applicant's information and can check if new Applicants have been added.

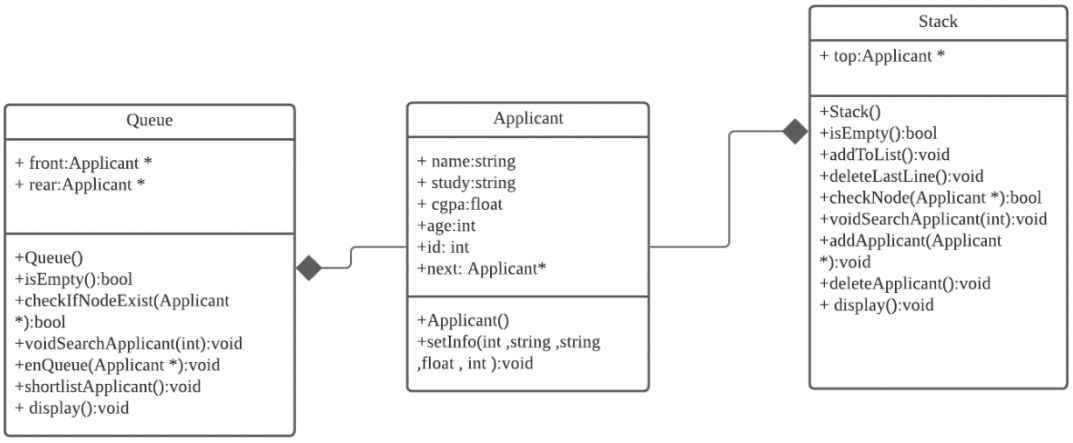
HR Manager	HR Manager will shortlist applicants and display the list of all Applicants.

Detail Description for Each Use Cases

The system has 6 main use cases.

Use Case	Task
Add Applicant	Hiring Panel inserts information of Applicant that is then added into the Applicant file created by the system.
Delete Applicant	Hiring Panel deletes information of Applicant from Applicant file.
Show all Applicants	The system displays the list of Applicants from Applicant file to Hiring Panel
Search Applicant by ID	Both Hiring Panel and HR Manager can search for an Applicant from the Applicant list by inputting the specific Applicant's ID.
Shortlist Applicant	Once Hiring Panel adds Applicant information to the list of Applicants, HR Manager decides whether to shortlist them or not.
Check new file for selected Applicants	Once an Applicant is shortlisted, the system creates a new file for those selected Applicants.

Class Diagram

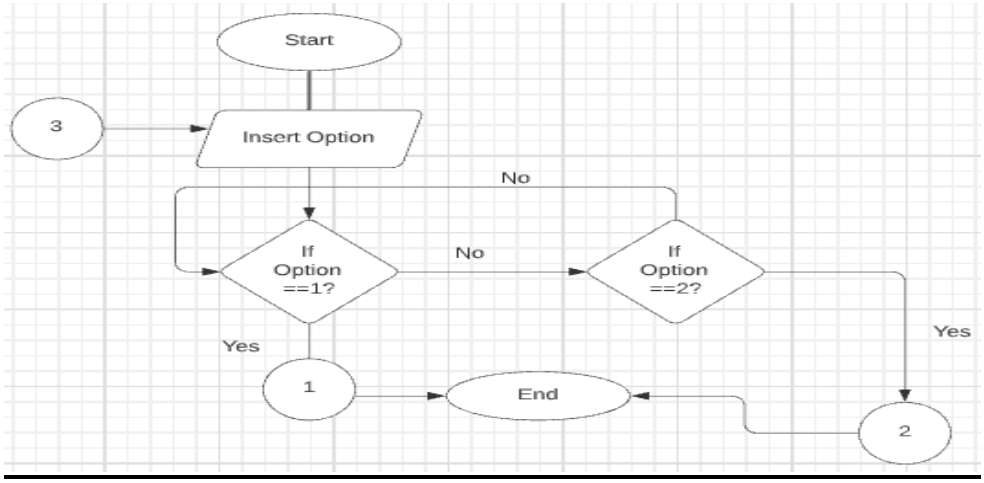


There are 3 classes in the program:

- Queue - Checking if there are any applicants in the system for their resume to be checked and adding selected applicant into the file by HR manager
- Applicant – Applicant's name and other information is inserted into system my hiring panel for HR manager to check them
- Stack – Hiring panel adds applicants into the file and can also delete file from the system. Can display the inputted applicant data

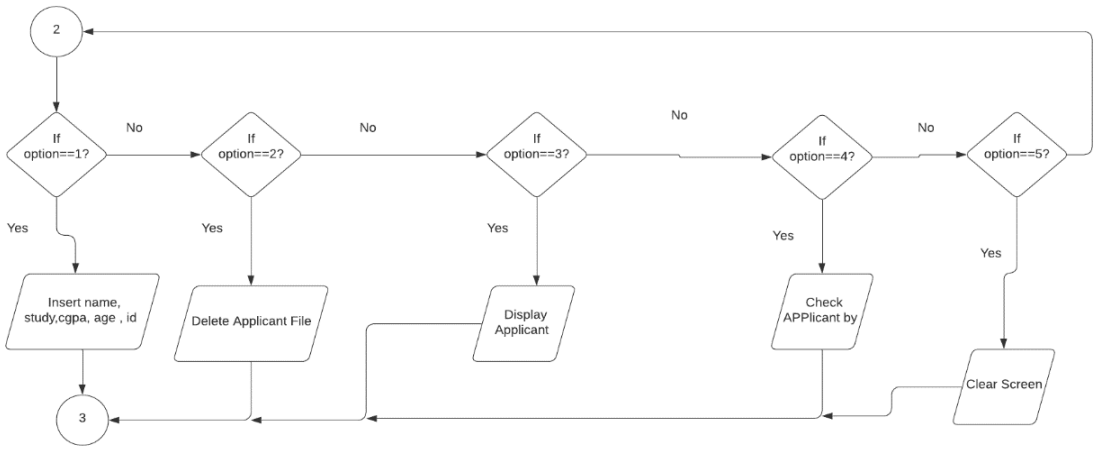
2.2. System Design

Overall Flowchart



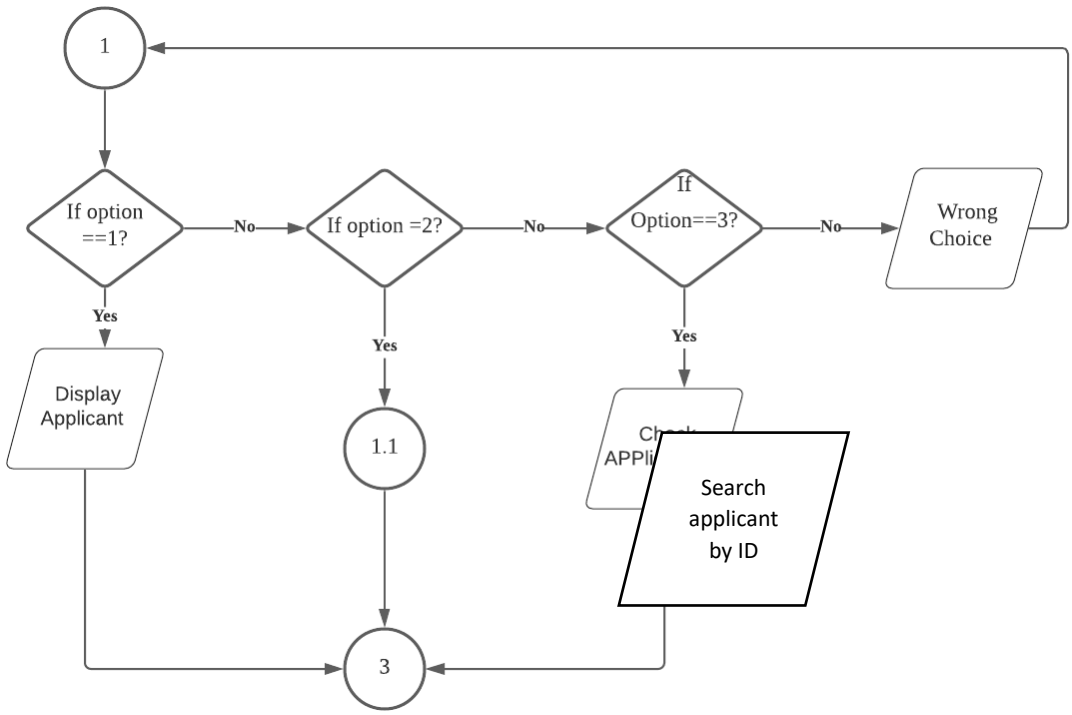
The menu appears. If option 1 is selected it goes through the process 1 and if option 2 is pressed , the process 2 functionality is showcased in the code.

Flowchart 2



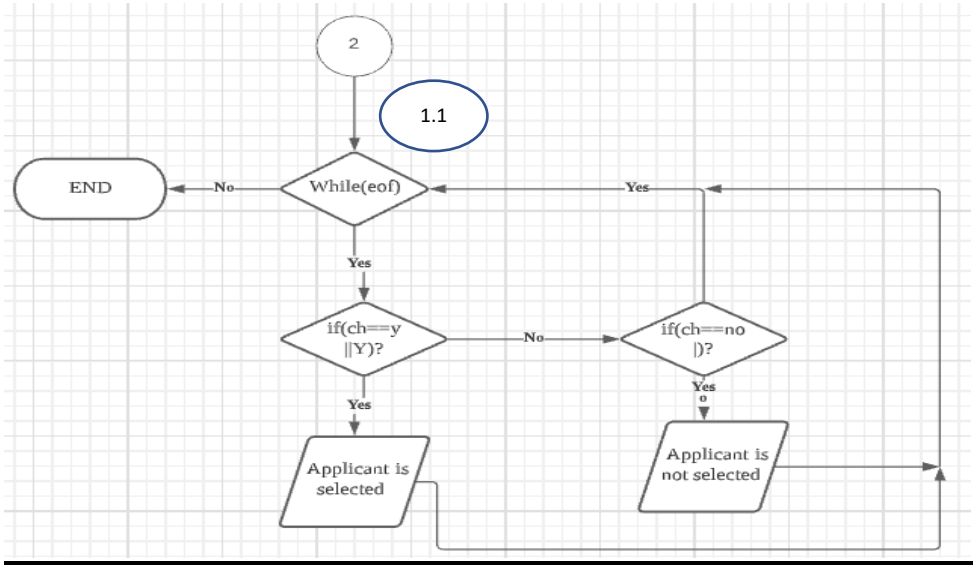
Process 2 is described here. If the user presses option 1 , it lets the users key in the applicant names and other information. If option 2 is pressed , the user can delete the latest keyed in Applicant file. If option 3 is pressed , user can check all applicant information . If option 4 is pressed , applicant can be searched by their ID, option 5 lets the user clear the screen.

Flowchart 1



In flowchart 1, If aoption 1 is selected , all the applicants he hiring panel added is showcased, if option 2 is pressed it goes through process 1.1 , which is described later and if option 3 is pressed , applicant information can be search by their Id, if any other key is pressed , it shows wrong choice , and every process after finishing returns back to the process 3.

Flowchart 1.1



Process 1.1 involves HR manager selecting Applicants one by one, first they review on applicant, select or reject them, then move on to the next applicant until the list of Applicant is over in the file. If user presses Y, the applicant is selected, if user presses no, the user is not selected. The selected users' information is saved in a new file.

3.0. SYSTEM PROTOTYPE

```
#################Welcome to the company#################
Enter your choice below.. Enter 0 to exit
1. HR manager
2. Hiring panel
3. Clear Screen

Enter your choice:
```

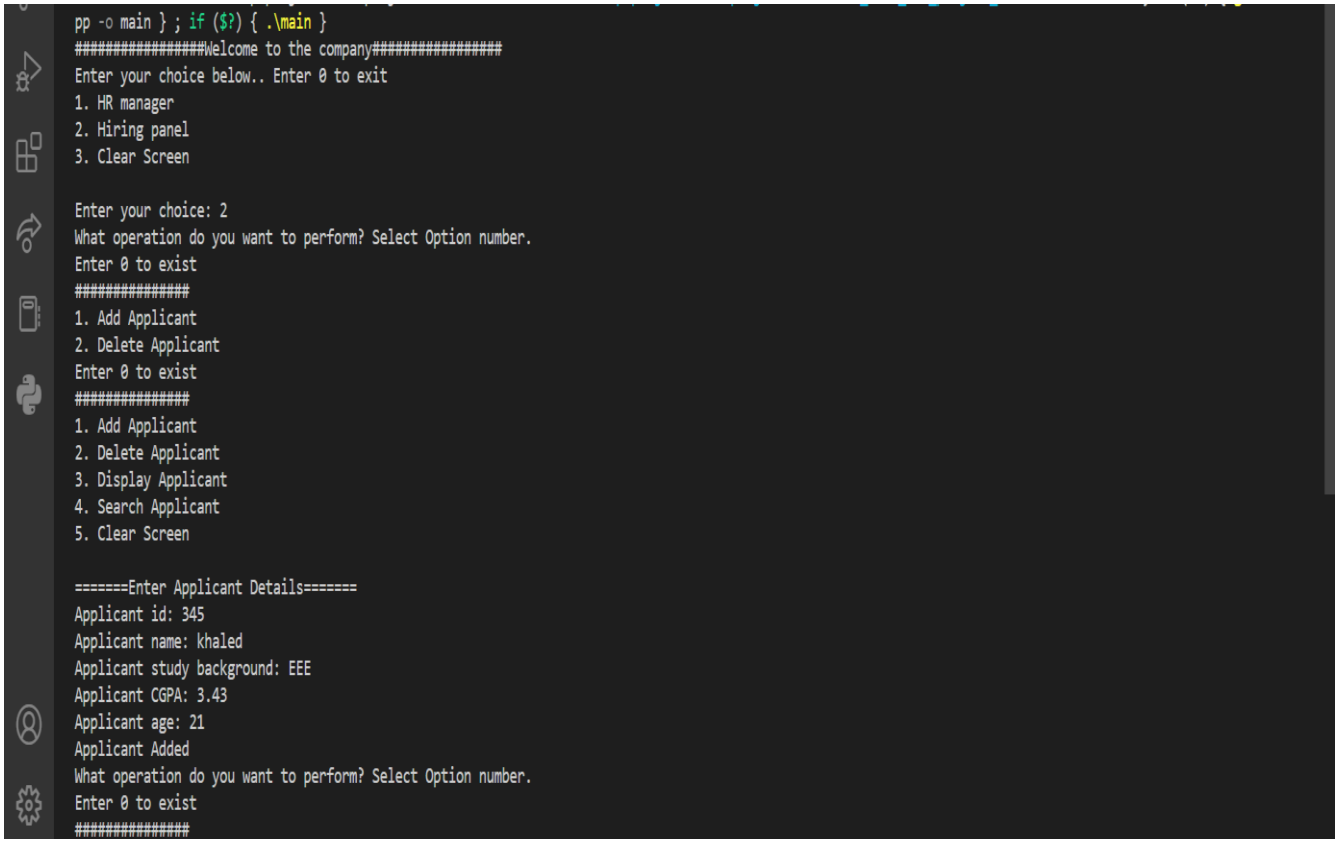
This screen shows the main menu of the system. Users must choose whether they are HR manager or a Hiring Panel. If the user is a Hiring Panel, they will have to type 2 to input or delete Applicant details and if the user is a HR Manager, they will have to type 1 for either displaying the list of applicants or shortlisting applicants into a new file created by the system. If the user enters an integer other than 0-3 , the program will display an error message and the screen will be displayed again. Then, users are allowed to enter their input again. They can also enter 0 if they want to stop running the program.

```
##################Welcome to the company#################
Enter your choice below.. Enter 0 to exit
1. HR manager
2. Hiring panel
3. Clear Screen

Enter your choice: 1


What Do you want to do
1. Display All applicant
2. Shortlist Applicant
3. Search applicant

Enter your choice: 1

queue is empty
##################Welcome to the company#################
Enter your choice below.. Enter 0 to exit
1. HR manager
2. Hiring panel
3. Clear Screen

Enter your choice: 1


What Do you want to do
1. Display All applicant
2. Shortlist Applicant
3. Search applicant

Enter your choice: 2

No data
##################Welcome to the company#################
Enter your choice below.. Enter 0 to exit
1. HR manager
```

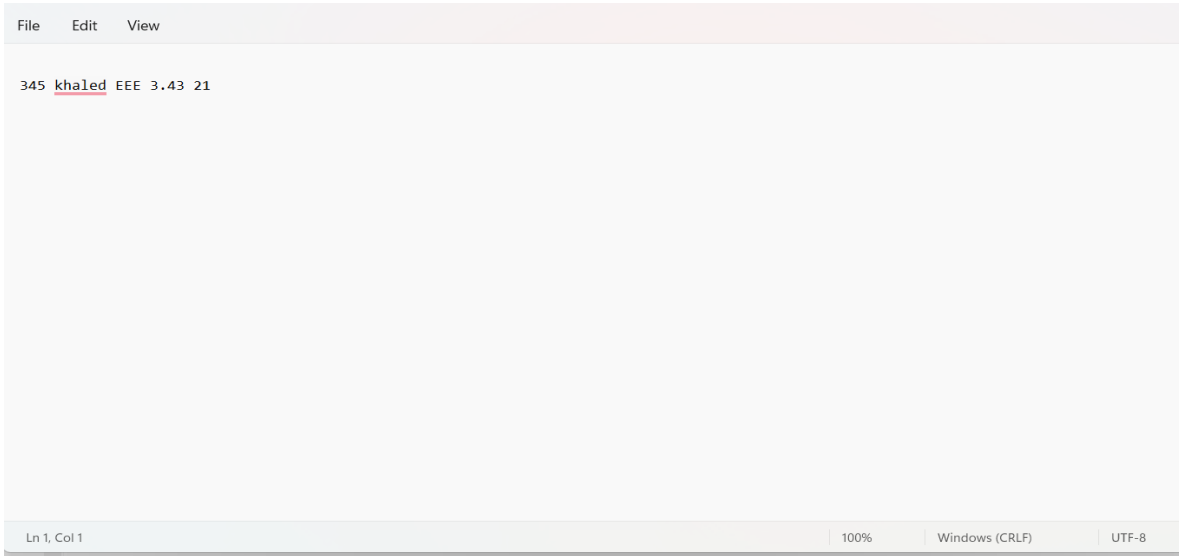
This screen shows that if there is no Applicant information in the list file then appropriate error messages will be displayed. If the HR Manager inputs 1 to display the list of all applicants when the file doesn't contain any Applicant information, the system will display the error message 'queue is empty'. If the HR Manager inputs 2 to shortlist applicant when the file doesn't contain any Applicant information, the system will display the error message 'No data'.

```
##################Welcome to the company#################
Enter your choice below.. Enter 0 to exit
1. HR manager
2. Hiring panel
3. Clear Screen

Enter your choice: 1


What Do you want to do
1. Display All applicant
2. Shortlist Applicant
3. Search applicant

Enter your choice: 3

Search the Applicant....
Enter Applicant ID: 1234
Applicant not found

##################Welcome to the company#################
Enter your choice below.. Enter 0 to exit
1. HR manager
2. Hiring panel
3. Clear Screen

Enter your choice:
```

This screen shows that if the HR Manager inputs 3 to search for Applicant information from the list of Applicants when the file doesn't contain any Applicant information or if the Applicant ID inputted is not in the file, the system will display the error message 'Applicant not found'.



This screen shows the menu for Hiring Panel and how an Applicant's information is added into the Applicant file.



This is the file with list of Applicants that is created by the system automatically once the Hiring Panel adds Applicant information and this is how the information inputted by Hiring Panel should be stored in the file by the system.

```
What operation do you want to perform? Select Option number.
Enter 0 to exist
###############
1. Add Applicant
2. Delete Applicant
3. Display Applicant
4. Search Applicant
5. Clear Screen
```

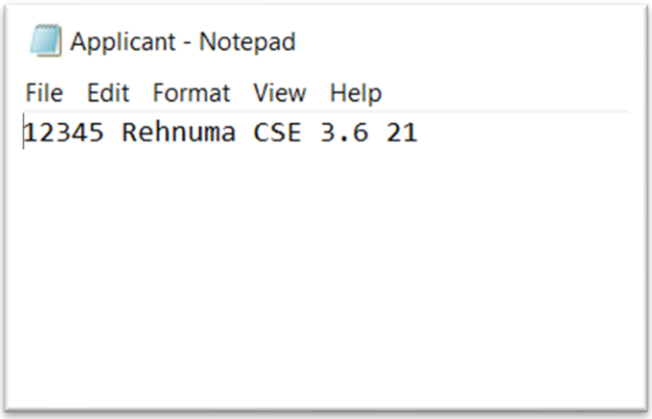
If the Hiring Panel wants to display the list of Applicants from the file then they need to input 3 and this is how the information of Applicants will be displayed.

```
###############
1. Add Applicant
2. Delete Applicant
3. Display Applicant
4. Search Applicant
5. Clear Screen

(Enter Your Option): 4
Search the Applicant....
Enter Applicant ID: 12345

||||||||||||Applicant Details||||||||||||
Applicant ID: 12345
Applicant name: Rehnuma
Applicant cgpa: 3.6
Applicant age: 21

What operation do you want to perform? Select Option number.
Enter 0 to exist
###############
1. Add Applicant
2. Delete Applicant
3. Display Applicant
4. Search Applicant
5. Clear Screen

(Enter Your Option):
```

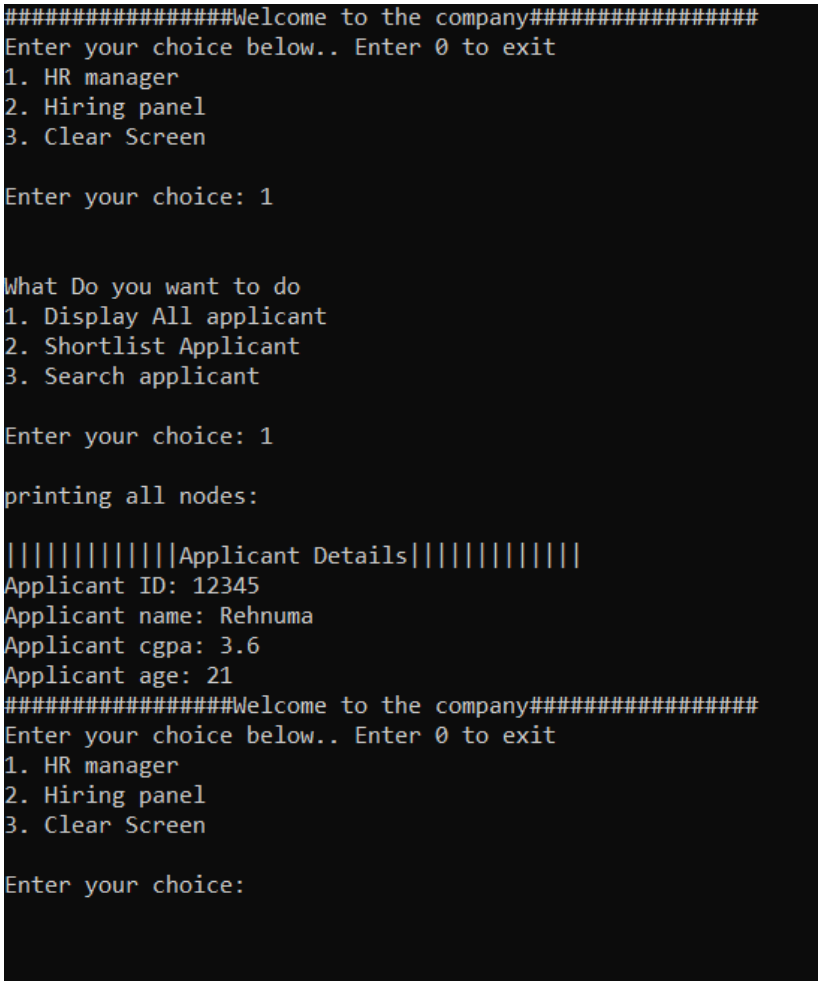
If the Hiring Panel wants to Search for a particular Applicant's information, they will have to input 4 and then input the Applicant's ID and this is how the information of that particular Applicant will be displayed.

```
What operation do you want to perform? Select Option number.
Enter 0 to exist
###############
1. Add Applicant
2. Delete Applicant
3. Display Applicant
4. Search Applicant
5. Clear Screen

(Enter Your Option): 2
Applicant data is deleted
What operation do you want to perform? Select Option number.
Enter 0 to exist
###############
1. Add Applicant
2. Delete Applicant
3. Display Applicant
4. Search Applicant
5. Clear Screen

(Enter Your Option):
```

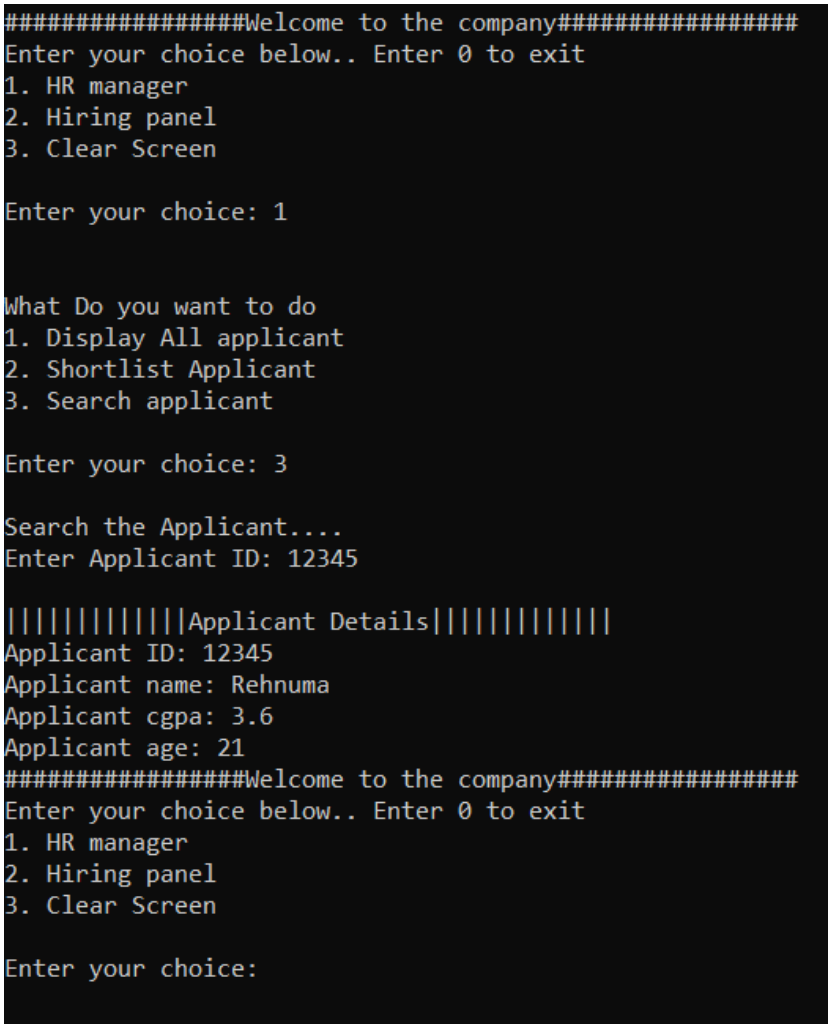
This screen shows that id the Hiring Panel inputs 2 then the system will delete the information of Applicant from the list in Last in First out movement.



As shown in the picture above, information of the last applicant in the list is deleted.



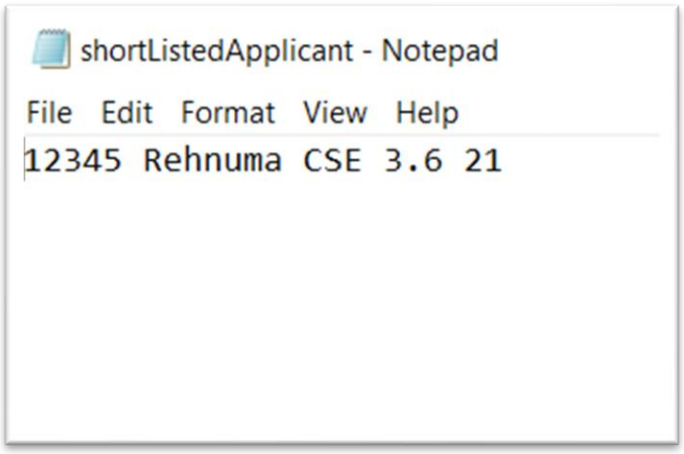
This screen shows that after the Hiring Panel adds Applicants' information and an applicant file is created by the system, the HR Manager can access the list from the main menu. The HR Manager first has to input 1 in the main menu to access the HR Manager's menu and then they can input 1 if they want to display the information of Applicants that are in the Applicant File and all the information that are in the file will be displayed.



This screen shows that a HR Manager can also search for a particular Applicant from the list to be displayed if they want to, by inputting 3 in the HR Manager menu first and then inputting that particular Applicant's ID.

```
#################Welcome to the company#################
Enter your choice below.. Enter 0 to exit
1. HR manager
2. Hiring panel
3. Clear Screen

Enter your choice: 1


What Do you want to do
1. Display All applicant
2. Shortlist Applicant
3. Search applicant

Enter your choice: 2


||||||||||||Applicant Details||||||||||||
Applicant ID: 12345
Applicant name: Rehnuma
Applicant cgpa: 3.6
Applicant age: 21
Do you want to shortlist the applicant. Press y or n: y
Applicant has been shortlisted
```

If a HR Manager wants to shortlist an applicant they will have to first input 2 in the HR Manager menu and the information of each Applicant in the list will appear in the screen and then the system will ask the HR Manager, 'Do you want to shortlist the applicant. Press y or n'. If the HR Manager inputs y to shortlist that applicant then the system will create a new file for shortlisted Applicants where all the information of shortlisted applicants will be stored. The screen shows that the HR Manager keyed in y so the applicant in from the Applicant file was shortlisted.



This is the system-created file of shortlisted Applicants that has the information of the Applicant which was shortlisted by the HR Manager.

Explanation Prepared By: KHALED MAHMUD SUJON & MAHMUDUL HASAN

Code Prepared By: KHALED MAHMUD SUJON & AMAR MoSTAFA.

4.0. DEVELOPMENT ACTIVITIES

Online Meeting Date	**Members Participated in the meeting**	**Activity**	**Task for each member**	**Task Achieved (Yes/No)**
12/1/2023 10:00 PM	khaled mahmudul mustafa	-Brainstorming about the project and deciding what to be done -Trying to decide on how to implement queue and stack in the program.	**khaled** -Initiate the meeting and suggest few concepts. **mahmudul** -Relating things from previous assignments with the project **Mostafa** -Planning how to implement the concept.	Yes
15/1/2023 10:00 PM	khaled mahmudul mustafa	-Implementing the concept in Dev C++	**khaled** -Initiate the meeting and	Yes

			start writing the code. **mahmudul** - Suggest ways of implementation and write some of the code **mostafa** - Suggest ways of implementation and write some of the code	
17/1/2023	khaled mahmudul mustafa	-Final touch up on project report documentation, presentation slides, and source code.	**khaled** - Initiate the meeting, create class diagram and flow charts for the report **khaled** -Finalize code and create Use Case diagram. **Mahmudul & mostafa** -Finishing up the rest of the	Yes

			report and compile. - Create presentation slide	

5.0. APPENDIX

Source Code

```
// MINI PROJECT : "E-RECRUITMENT SYSTEM" //

// SECJ2013 DATA STRUCTURE AND ALGORITHM //

 // LECTURER: Dr Bander Ali


// GROUP - 07 //

/* MEMBERS:

 KHALED MAHMUD SUJON (A20EC4082)

 MAHMUDUL HASAN (A20EC4083)

    AMMAR MOSTAFA()c   */


#include<iostream>

#include<fstream>

#include<vector>

using namespace std;


class Applicant{

   public:
```

```
        string name;
        string study;
        float cgpa;
        int age;
        int id;
        Applicant* next;

    Applicant(){
        name=" ";
        study=" ";
        cgpa=0.0;
        age=0;
        next=NULL;
    }
    void setInfo(int ID,string n,string s,float c, int a){
     name=n;
     study=s;
     cgpa=c;
     age=a;
     id=ID;
 }
};

class Stack{
```

```
  public:
     Applicant* top;
     Stack(){
        top=NULL;
     }
     //checking the stack is empty or not
     bool isEmpty(){
        if(top==NULL){
           return true;
        }
        else{
           return false;
        }
     }

     //add to file
     void addToList(Applicant *s){
        ofstream file_out("Applicant.txt",ios::app);
        file_out<<s->id<<" "<<s->name<<" "<<s->study<<" "<<s->cgpa<<" "<<s-
>age<<endl;
    }

     //delete the last line of the file
     void deleteLastLine(){
```

```
      string line;   // To read each line from code
int count=0;    // Variable to keep count of each line


ifstream mFile ("Applicant.txt");
//get the number of line in a file
if(mFile.is_open())
{
 while(mFile.peek()!=EOF)
 {
  getline(mFile, line);
  count++;
 }
 mFile.close();
 // cout<<"Number of lines in the file are: "<<count<<endl;
}
else
 cout<<"Couldn't open the file\n";



//remove the last line of the line
string line2;
vector<string> lines;
ifstream inputStream("Applicant.txt");
```

```
while (getline(inputStream,line2)) {
    lines.push_back(line2);
}
inputStream.close();


fstream outputStream("example.txt", ios::out | ios::trunc);
if (outputStream.is_open())
{
    for (int i=0; i < lines.size()-1; i++)
    {
        outputStream << lines[i] << "\n";
    }
    outputStream.close();
}
remove("Applicant.txt");
rename("example.txt", "Applicant.txt");
}


     //check the node
     bool checkNode(Applicant* s){
         Applicant* temp=top;
         bool exist = false;
         while(temp!=NULL){
             if(temp->id==s->id){
```

```
                exist=true;
                break;
            }
            temp=temp->next;
        }
        return exist;
    }

//Search an applicant
    void searchApplicant(int searchID){

 string name;
 string study;
 float cgpa;
 int age;
 int id;

 int found=0;

 fstream infile("Applicant.txt",ios::in);
 infile>>id>>name>>study>>cgpa>>age;
 while(!infile.eof()){
  if(id==searchID){
   cout<<endl<<"||||||||||||Applicant Details||||||||||||"<<endl;
```

```
 cout<<"Applicant ID: "<<id<<endl;
 cout<<"Applicant name: "<<name<<endl;
 cout<<"Applicant cgpa: "<<cgpa<<endl;
 cout<<"Applicant age: "<<age<<endl;
 cout<<endl;
 found=1;
 }
 infile>>id>>name>>study>>cgpa>>age;
}
if(found==0){
        cout<<"Applicant not found"<<endl<<endl;
}
    }

    //push the node
    void addApplicant(Applicant *s) {

        if(top==NULL){
         addToList(s);
            top=s;
        }
        else if(checkNode(s)){
            cout<<"xxxxxxx(ERROR : Same ID exists)xxxxxxx"<<endl;
        }
```

```
        else{
         addToList(s);
            Applicant* temp=top;
            top=s;
            s->next=temp;

            cout<<"Applicant Added"<<endl;
        }
    }

    //pop operation
    void deleteApplicant(){
        Applicant *temp=NULL;
        if(isEmpty()){
            cout<<"Stack is empty"<<endl;
        }
        else{
         deleteLastLine();
            temp=top;
top=top->next;
            delete temp;
            cout<<"Applicant data is deleted"<<endl;
        }
    }
```

```
    //print the all elements
    void display(){
        Applicant *temp;
        temp=top;
        if(isEmpty()){
            cout<<"Stack is empty"<<endl;
        }
        else{
            while(temp!=NULL){
                cout<<"Student ID: "<<temp->id<<endl;
                cout<<"Student Name: "<<temp->name<<endl;
                cout<<"Student CGPA: "<<temp->cgpa<<endl;
                cout<<"Student age: "<<temp->age<<endl;
                temp=temp->next;
            }
        }
    }

};//end of stack class

class Queue{

 public:
```

```
Applicant*front,*rear;
Queue(){
 front=NULL;
 rear=NULL;
}

//checking wheather list is empty or not
     bool isEmpty(){
        if(front==NULL&&rear==NULL){
            return true;
        }
        else{
            return false;
        }
     }

     //Search an applicant
     void searchApplicant(int id){
      int found=0;
        Applicant*temp=front;

        while(temp!=NULL){
            if(temp->id==id){
  cout<<endl<<"||||||||||||Applicant Details||||||||||||"<<endl;
```

```
   cout<<"Applicant ID: "<<temp->id<<endl;
   cout<<"Applicant name: "<<temp->name<<endl;
   cout<<"Applicant cgpa: "<<temp->cgpa<<endl;
   cout<<"Applicant age: "<<temp->age<<endl;


              found=1;


                 break;
              }
             temp=temp->next;
          }
          if(found==0){
           cout<<"Applicant not found"<<endl<<endl;
 }
      }

//enQueue function
void enQueue(Applicant*s){
 if(isEmpty()){
  front=s;
  rear=s;
 }
 else{
  rear->next=s;
```

```
  rear=s;
 }
}

//shortlist the applicant
void shortlistApplicant(){
 if(isEmpty()){
  cout<<"No data"<<endl;
 }
 else{
  ofstream sl_out("shortListedApplicant.txt",ios::app);
  string ch;
  Applicant* temp=front;
  while(temp!=NULL){
   cout<<endl<<"||||||||||||Applicant Details||||||||||||"<<endl;
   cout<<"Applicant ID: "<<temp->id<<endl;
   cout<<"Applicant name: "<<temp->name<<endl;
   cout<<"Applicant cgpa: "<<temp->cgpa<<endl;
   cout<<"Applicant age: "<<temp->age<<endl;
   cout<<"Do you want to shortlist the applicant. Press y or n: ";
   cin>>ch;
   if(ch=="y"||ch=="Y"){
            sl_out<<temp->id<<" "<<temp->name<<" "<<temp->study<<" "<<temp->cgpa<<"
"<<temp->age<<endl;
    cout<<"Applicant has been shortlisted"<<endl<<endl;
```

```
  }
  else{
   cout<<"Applicant data not added"<<endl;
  }
  temp=temp->next;
 }


}
}


    //display all nodes in the queue
    void display(){
        if(isEmpty()){
            cout<<"queue is empty"<<endl;
        }
        else{
            cout<<"printing all nodes: "<<endl;
            Applicant* temp=front;
            while(temp!=NULL){
              cout<<endl<<"||||||||||||Applicant Details||||||||||||"<<endl;
  cout<<"Applicant ID: "<<temp->id<<endl;
  cout<<"Applicant name: "<<temp->name<<endl;
  cout<<"Applicant cgpa: "<<temp->cgpa<<endl;
  cout<<"Applicant age: "<<temp->age<<endl;
```

```
                temp=temp->next;
            }
        }
    }

};

int main(){

 int choice;
 do{
  cout<<"##################Welcome to the company##################"<<endl;
  cout<<"Enter your choice below.. Enter 0 to exit"<<endl;
  cout<<"1. HR manager"<<endl;
  cout<<"2. Hiring panel"<<endl;
  cout<<"3. Clear Screen"<<endl;
  cout<<endl<<"Enter your choice: ";cin>>choice;
  switch(choice){
   case 0: break;
   case 1:{
    Queue q;
    int option, option2;
    string name;
    string study;
```

```
    float cgpa;
    int age;
    int id;

    fstream
infile("Applicant.txt",ios::in);

    int searchID;

    infile>>id>>name>>study>>cgpa>>age;
    while(!infile.eof()){
     Applicant* ss=new Applicant();

     ss->setInfo(id,name,study,cgpa,age);
     infile>>id>>name>>study>>cgpa>>age;

     q.enQueue(ss);
    }
    cout<<endl;

    cout<<endl<<"What Do you want to do"<<endl;
    cout<<"1. Display All applicant"<<endl;
    cout<<"2. Shortlist Applicant"<<endl;
    cout<<"3. Search applicant"<<endl;
```

```
 cout<<endl<<"Enter your choice: ";
 cin>>option2;
 cout<<endl;
 switch(option2){
  case 0: break;
  case 1:
   q.display();
   break;
  case 2:
   q.shortlistApplicant();
   break;
  case 3:
   cout<<"Search the Applicant...."<<endl<<"Enter Applicant ID: ";
   cin>>searchID;
   q.searchApplicant(searchID);
   break;
  default:
   cout<<"Wrong choice"<<endl;
 }

 break;
}
case 2:{
```

```
    Stack s1;
    int option, position;
    string name;
    string study;
    float cgpa;
    int age;
    int id;

    do{
     cout<<"What operation do you want to perform? Select Option number."<<endl<<"Enter 0 to exist"<<endl;
     cout<<"################"<<endl;
     cout<<"1. Add Applicant"<<endl;
     cout<<"2. Delete Applicant"<<endl;
     cout<<"3. Display Applicant"<<endl;
     cout<<"4. Search Applicant"<<endl;
     cout<<"5. Clear Screen"<<endl;

     cout<<endl<<"(Enter Your Option): ";
     cin>>option;

     Applicant* s=new Applicant();

     switch(option){
```

```
case 1:
 cout<<"=======Enter Applicant Details======="<<endl;
 cout<<"Applicant id: ";cin>>id;
 cout<<"Applicant name: ";cin>>name;
 cout<<"Applicant study background: ";cin>>study;
 cout<<"Applicant CGPA: ";cin>>cgpa;
 cout<<"Applicant age: "; cin>>age;
 s->id=id;
 s->name=name;
 s->study=study;
 s->cgpa=cgpa;
 s->age=age;
 s1.addApplicant(s);
 break;
case 2:
 s1.deleteApplicant();
 break;
case 3:
 s1.display();
 break;
case 4:
 int searchID;
 cout<<"Search the Applicant...."<<endl<<"Enter Applicant ID: ";
 cin>>searchID;
```

```
      s1.searchApplicant(searchID);
      break;
     case 5:
      system("cls");
      break;
     default:
      cout<<"Wrong operation number"<<endl;

    }
   }while(option!=0);

   break;
  }////
  case 3: system("cls");
  break;
  default:
   cout<<"Wrong choice..Please Enter correct option"<<endl;
   break;
 }
}while(choice!=0);
   return 0;
}
```

----- END OF DOCUMENTATION -----