Sri Lanka Institute of Information Technology

Assignment 1

MLB_09.01_08

Recruitment Company System

Object-Oriented Concepts - IT1050

B.Sc. (Hons) in Information Technology

1. Cover Page

Topic: Recruitment Company System

Group no: MLB_09.01_08

Campus: Malabe

Submission Date: 20.05.2022

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Group Details:

Registration No	**Name**	**Contact Number**
IT21175152	E.A.H.A. Peiris	0714586272
IT21173318	A.M.S.S. Adhikari	0766122273
IT21176838	E.D.T.V. Elvitigala	0754027930
IT21176524	I.T.U. Sampath	0755574289
IT21172878	D.P.Liyanagama	0712413634

Table of Contents:

2. Identification of the system requirements.

2.1 Introduction

This recruitment system performs the recruitment process automatically. The main parties involved in this process are job seekers and client companies. This system has introduced several special features to facilitate the recruitment process.

Client companies as well as job seekers can register with the system. This allows client companies to post job ads, and applicants have the opportunity to apply for them. All companies and applicants are automatically controlled by the system. The candidate selection process includes the ability to conduct online exams and conduct online interviews. Communication between applicants, companies and the system takes place via email notifications. Client companies can automatically obtain information about applicants through reports.

2.2 Main features of the system

Administrative requirements

1. **Account Management** - The admin can review user accounts before adding them to the system. Also, the admin can edit account details and delete accounts if required.

2. **Vacancy Management** - The admin can review job advertisements posted by client companies before adding them to the system. Also, the admin can edit vacancy details and delete advertisements if required. If such a deletion occurs, the company will be notified via email.

3. **Shortlist Applicants -** The admin can view all the details of the applicant. If an applicant meets the basic requirements requested by the company, the admin can select them for the selection exam and/ or interview. Also, the admin can request further details from the applicant if required.

4. **Send Notifications** - Admin can send various types of notification emails such as payment reminder notifications for clients, invite selection tests and interviews for the applicants, rejection notification emails to the applicants, selection notifications for the selected applicants, and inform the selected applicant details to the client company etc.

5. **Schedule Tests -** Admin can upload selection exam questions to the database, delete the question from the database and add exams for the database. Also, the admin can view these questions and edit questions if required.

6. **Schedule Interviews** - The administrator can schedule interviews for the selected applicants. Also, admin can add, edit, delete and postpone interviews.'

7. **Make Reports** - Administrators can generate various Reports such as applicant details, client details, number of vacancy details, selected applicants etc.

8. **Database Management** - Administrator can get backups of the database, update data and delete backups from the database.

Client-side requirements

1. **Register to the website** - Client companies and employees can register to the system by filling out the registration forms.

2. **User login** - Existing users have to login to the system in order to use the system facilities.

3. **View vacancies and job categories** - Registered users and unregistered users can view vacancies and job categories posted on the website.

4. **Search vacancies and job categories** - Registered users and unregistered users can search vacancies and job categories posted on the website.

5. **Clients can get a paid membership** - Client companies should get a paid membership in order to get services from the website. They can do payments through online banking.

6. **Clients can add vacancies** - Client companies can add their vacancy details to the system.

7. **Employees can apply for vacancies** - Employees can apply for the vacancies posted by client companies.

8. **Users can manage accounts** - Clients and employees can edit, update and delete their accounts.

9. **Attend for exams and interviews** – If employee selected for a vacancy, he/she can attend for exams and interviews.

3. Noun Verb Analysis

3.1 Introduction

This recruitment system performs the recruitment process automatically. The main parties involved in this process are **job seekers** and **client companies**. This system has introduced several special features to facilitate the recruitment process.

Client companies as well as **job seekers** can **register** with the **system**. This allows client companies to **post job ads**, and **applicants** have the opportunity to **apply** for them. All companies and applicants are automatically **controlled** by the system. The **candidate selection** process includes the ability to **conduct online exams** and conduct **online interviews**. **Communication** between applicants, companies and the system takes place via **email notifications**. Client companies can automatically **obtain information** about applicants through **reports**.

3.2 Main features of the system

Administrative requirements

1. **Account Management** - The admin can review user accounts before adding them to the system. Also, the admin can edit account details and delete accounts if required.

2. **Vacancy Management** - The admin can review job advertisements posted by client companies before adding them to the system. Also, the admin can edit vacancy details and delete advertisements if required. If such a deletion occurs, the company will be notified via email.

3. **Shortlist Applicants** - The admin can view all the details of the applicant. If an applicant meets the basic requirements requested by the company, the admin can select them for the selection exam and/ or interview. Also, the admin can request further details from the applicant if required.

4. **Send Notifications** - Admin can send various types of notification emails such as payment reminder notifications for clients, invite selection tests and interviews for the applicants, rejection notification emails to the applicants, selection notifications for the selected applicants, and inform the selected applicant details to the client company etc.

5. **Schedule Tests** - Admin can upload selection exam questions to the database, delete the question from the database and add exams for the database. Also, the admin can view these questions and edit questions if required.

6. **Schedule Interviews** - The administrator can schedule interviews for the selected applicants. Also, admin can add, edit, delete and postpone interviews.'

7. **Make Reports** - Administrators can generate various Reports such as applicant details, client details, number of vacancy details, selected applicants etc.

8. **Database Management** - Administrator can get backups of the database, update data and delete backups from the database.

Client-side requirements

1. **Register to the website** - Client companies and employees can register to the system by filling out the registration forms.

2. **User login** - Existing users have to login to the system in order to use the system facilities.

3. **View vacancies and job categories** - Registered users and unregistered users can view vacancies and job categories posted on the website.

4. **Search vacancies and job categories** - Registered users and unregistered users can search vacancies and job categories posted on the website.

5. **Clients can get a paid membership** - Client companies should get a paid membership in order to get services from the website. They can do payments through online banking.

6. **Clients can add vacancies** - Client companies can add their vacancy details to the system.

7. **Employees can apply for vacancies** - Employees can apply for the vacancies posted by client companies.

8. **Users can manage accounts** - Clients and employees can edit, update and delete their accounts.

9. **Attend for exams and interviews** – If employee selected for a vacancy, he/she can attend for exams and interviews.

4. Identification of the classes

4.1 Noun Analysis

Nouns	Classes	Attributes	Out of Scope
Applicant	Applicant	Vacancy details	System
Client company	Client company	Basic requirements	User account
System	Admin	Job categories	Database
Online exam	Online exam	Online banking	Registration form
Online interview	Online interview		Registered user
Email notification	Vacancy		Unregistered user
Reports	Email notification		Service
Admin	Reports		Account
User account	Requirements		
Vacancy details	Questions		
Basic requirements	Backup		
Exam questions	Membership		
Database	Selection		
Backups	Permission		
Registration form			
Registered user			
Unregistered user			
Job categories			
Vacancy			
Paid membership			
Service			
Online banking			
Account			

4.2 Verb Analysis

Verbs	Administrator	Applicant	Company
Register	Control	Register	Register
Post job ads	Selection	Apply	Post job ads
Apply	Communication	Update	Obtain information
Control	Review	Edit	Add
Selection	Add	Delete	Update
Communication	Update	Upload	Edit
Obtain information	Edit	Filling out	Delete
Review	Delete	Login	Upload
Add	Select	Search	Filling out
Update	Request Details	Attend	Login
Edit	Send Notifications		Search
Delete	Invite		Do payments
Notify / Inform	Reject		
Select	Upload		
Request Details	Schedule		
Send Notifications	Postpone interviews		
Invite	Generate		
Reject	Login		
Upload	Search		
Schedule			
Postpone interviews			
Generate			
Filling out			
Login			
Search			
Do payments			
Attend			

5. CRC cards for the classes

Class Name: Employee	
Responsibilities:	**Collaborations:**
Register to the system	Register
Apply for the vacancies	Vacancy

Class Name: Company	
Responsibilities:	**Collaborations:**
Register to the system	Register
Add vacancies	Vacancy
Get selected applicants report	Report
Do the interviews	Interview

Class Name: Admin	
Responsibilities:	**Collaborations:**
Manage companies	Company
Manage employees	Employee
Manage vacancies	Company
Shortlist applicants	Employee / Selection
Generate reports	Reports
Send notifications	Notification
Get backups	Backup
Schedule exams	Exam

Class Name: Exam	
Responsibilities:	**Collaborations:**
Schedule exams	Admin
Postpone exams	

Class Name: Interview	
Responsibilities:	**Collaborations:**
Schedule the interviews	Admin
Manage the interviews	Admin / Company

Class Name: Vacancy	
Responsibilities:	**Collaborations:**
Add vacancies	Company
Manage vacancies	Admin

Class Name: Notification	
Responsibilities:	**Collaborations:**
Get notification details	Admin / Company / Employee
Send notifications	Admin

Class Name: Report	
Responsibilities:	**Collaborations:**
Get report details	Selection
Manage reports	Admin
Send reports	Admin / Company

Class Name: Question	
Responsibilities:	**Collaborations:**
Add questions to the exam	Exam
Delete / update questions	Admin

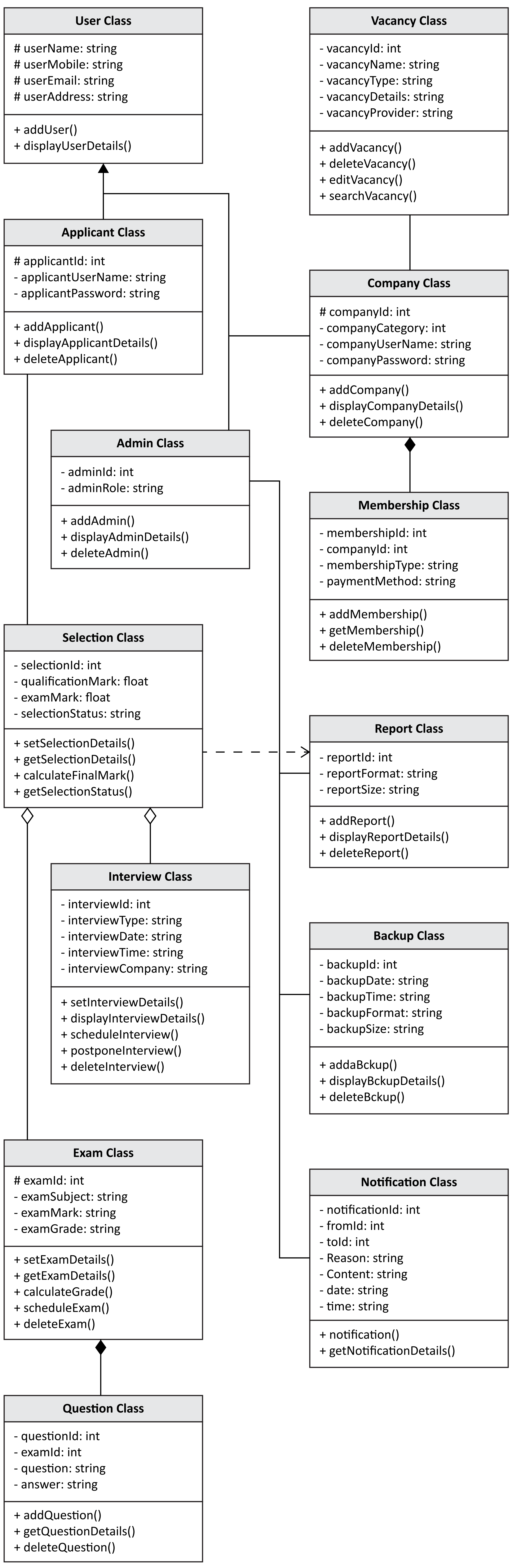
Class Name: Backup	
Responsibilities:	**Collaborations:**
Get backups of the data	Report, Employee, Admin, Company
Update/ Delete backups	Admin

Class Name: Membership	
Responsibilities:	**Collaborations:**
Get paid memberships	Company
Manage membership	

Class Name: Selection	
Responsibilities:	**Collaborations:**
Shortlist the employees	Employee
Select best employees	Employee/ Exam/ Interview
Make reports	Report

6. Class Diagram

Class Diagram



7. Coding of the classes

Contributor Name: D.P. Liyanagama

Contributor ID: IT21172878

1. Applicant Class

Applicant.h

```
#include <string>
using namespace std;

class Applicant
{
private:
	int aID;
	string aName;
	int mobileNO;
	string email;
	string address;
	string username;
	string password;

public:
	Applicant();
	int printApplicantDetails();
	~Applicant();
};
```

Applicant.cpp

```
#include <iostream>
#include <string>
#include "Applicant.h"
using namespace std;

Applicant::Applicant()
{	cout << "Enter Applicant ID: ";
	cin >> aID;
```

```
        cout << "Enter Applicant Name: ";
        cin >> aName;

        cout << "Enter Applicant mobile number: ";
        cin >> mobileNO;

        cout << "Enter Applicant Email: ";
        cin >> email;

        cout << "Enter Applicant Address: ";
        cin >> address;

        cout << "Enter Applicant Username: ";
        cin >> username;

        cout << "Enter Applicant Password: ";
        cin >> password;
        cout << "\n";
}

int Applicant::printApplicantDetails()
{
        cout << "Applicant ID: " << aID << endl;
        cout << "Applicant Name: " << aName << endl;
        cout << "Applicant mobile number: " << mobileNO << endl;
        cout << "Applicant Email: " << email << endl;
        cout << "Applicant Address: " << address << endl;
        cout << "Applicant Username: " << username << endl;
        cout << "Applicant Password: " << password << endl;
        cout << "\n";
    return 0;
}

Applicant::~Applicant()
{
        cout << "Memory Deallocation...!";
        cout << "\n";
}
```

Main.cpp

```
#include <iostream>
#include <string>
```

```
#include "Applicant.h"

int main()
{
    Applicant A1;
    A1.printApplicantDetails();

    return 0;
}
```

2. Admin Class

Admin.h

```
#pragma once
#include "string";
using namespace std;

class Admin
{

private:
      int aID;
      string aName;

public:
      void setAdminDetails(int adID, string adName);
      int displayAdminDetails();


      class user
      {
      private:
            string Uname;
            int Uid;

      public:
            void setUserDetails(string uN, int uID);
            int displayUserDetails();
            int removeUser();
      };
};
```

Admin.cpp

```cpp
#include "Admin.h"
#include <iostream>
#include "string";
using namespace std;

void Admin::setAdminDetails(int adID, string adName)
{
    aID = adID;
    aName = adName;
}

int Admin::displayAdminDetails()
{
    cout << "Admin ID: " << aID << endl;
    cout << "Admin Name: " << aName << endl;
    return 0;
}

void Admin::user::setUserDetails(string uN, int uID)
{

    Uname = uN;
    Uid = uID;
}

int Admin::user::displayUserDetails()
{
    cout << "User ID: " << Uid << endl;
    cout << "User Name: " << Uname << endl;
    return 0;
}

int Admin::user::removeUser()
{
    string result;
    cout << "You want delete this user (yes/no): ";
    cin >> result;

    if (result == "YES" || result == "Yes" || result == "yes")
    {
        int id;
        cout << "Enter the user ID that needs to be removed: ";
```

```
            cin >> id;
            cout << "\n";
            if (id == Uid)
            {
                cout << "COMPLETED USER DELETE..!";
                cout << "\n";
            }
            else
            {
                cout << "INVALID USER ID..!";
                cout << "\n";
            }
        }
        else
        {
            cout << "\n";
            cout << "Thank You..!" << endl;
            cout << "\n";
        }
        return 0;
    }
```

main.cpp

```
#include <iostream>
#include "Admin.h";

int main()
{
    Admin A1;
    A1.setAdminDetails(101, "joe");
    A1.displayAdminDetails();
    cout << "\n";

    for (int x = 0; x<100; x++)
    {
    string UNAME;
    int UID;
    cout << "Enter user ID: ";
    cin >> UID;
    cout << "Enter user Name: ";
    cin >> UNAME;
    cout << "\n";
    if (UID == NULL)
    {
        cout << "Opps..! PLEASE ENTER INTEGER VALUE" <<endl;
    }
    else
    {
        cout << "SUCCESSFULLY USER ADDED..!" << endl;
        cout << "\n";
        Admin::user U1;
        U1.setUserDetails(UNAME,UID);
        U1.displayUserDetails();
        U1.removeUser();
    }
    cout << "\n";
    }

    system("pause>0");
    return 0;
}
```

3. Exam Class

Exam.h

```
#pragma once
#include <string>
using namespace std;

class Exam
{
private:
      int Eid;
      string sub;
      float mark;
      char grade;

public:
      void setExamDetails(int exID, string exSub, float exMark, char
exGrade);
      int displayExamDetails();
      int calGrade();
};
```

Exam.cpp

```
#include <iostream>
#include "Exam.h"
using namespace std;

void Exam::setExamDetails(int exID, string exSub, float exMark, char
exGrade)
{
    Eid = exID;
    sub = exSub;
    mark = exMark;
    grade = exGrade;

    cout << "Enter Exam ID: ";
    cin >> Eid;

    cout << "Enter Exam Subject: ";
    cin >> sub;

    cout << "Enter Exam Mark: ";
    cin >> mark;
}

int Exam::displayExamDetails()
{
    cout << "\n";
    cout << "Exam ID: " << Eid << endl;
    cout << "Exam Subject: " << sub << endl;
    cout << "Exam Mark: " << mark << endl;

    return 0;
}

int Exam::calGrade()
{
    if (mark < 35)
    {
        cout << "Exam Grade: S" << endl;
    }
    else if (mark < 45)
    {
        cout << "Exam Grade: C" << endl;
    }
    else if (mark < 75)
    {
        cout << "Exam Grade: B" << endl;
    }
    else if (mark <= 100)
    {
        cout << "Exam Grade: A" << endl;
    }
    else
    {
        cout << "Invalid mark..!" << endl;;
    }
    return 0;
}
```

main.cpp

```
#include <iostream>
#include "Exam.h";
using namespace std;

int main()
{
    Exam e1;
    e1.setExamDetails(001, "SPM", 91, 'A');
    e1.displayExamDetails();
    e1.calGrade();

    system("pause>0");
    return 0;
}
```

Contributor Name: E.D.T.V. Elvitigala

Contributor ID: IT21176838

4. Interview Class

Interview.h

```
#include<iostream>
#include<string>
using namespace std;

class Interview
{
private:
	int interviewId;
	string interviewType;
	string interviewDate;
	string interviewTime;
	string interviewCompany;
public:
	void setInterviewId(int interId);
	void setinterviewType(string interType);
	void setinterviewDate(string interDate);
	void setinterviewTime(string interTime);
	void setinterviewCompany(string interCom);

	int getdisplayID();
	int getdisplayTYPED();
	int getdisplayDATE();
	int getdisplayTIME();
	int getdisplayCOMPANY();
};
```

Interview.cpp

```
#include <iostream>
#include " Interview.h";
using namespace std;

void Interview::setInterviewId(int interId)
{
	interviewId= interId;
	cout << "Interview ID :" ;
	cin >> interviewId;
}

int Interview::getdisplayID()
{
	cout << "Interview ID :" << interviewId << endl;
	return 0;
}

void Interview::setinterviewType(string interType)
{
	interviewType= interType;
	cout << "Interview Type :";
	cin >> interviewType;
}

int Interview::getdisplayTYPED()
{
	cout << "Interview Type :" << interviewType << endl;
	return 0;
}

void Interview::setinterviewDate(string interDate)
{
	interviewDate= interDate;
	cout << "Interview Date :";
	cin >> interviewDate;
}

int Interview::getdisplayDATE()
{
	cout << "Interview Date :" << interviewDate << endl;
	return 0;
}

void Interview::setinterviewTime(string interTime)
{
	interviewTime= interTime;
	cout << "Interview Time :" ;
	cin >> interviewTime;

}

int Interview::getdisplayTIME()
```

```
{
      cout << "Interview Time :" << interviewTime << endl;
      return 0;
}

void Interview::setinterviewCompany(string interCom)
{
      interviewCompany= interCom;
      cout << "Interview Company :" ;
      cin >> interviewCompany;

}

int Interview::getdisplayCOMPANY()
{
      cout << "Interview Company :" << interviewCompany << endl;
      return 0;
}
```

main.cpp

```
#include <iostream>
#include " Interview.cpp";
using namespace std;

int main()
{

    Interview* I;
    I = new Interview();

    I->setInterviewId(23);
    I->setinterviewType("online");
    I->setinterviewDate("sdf");
    I->setinterviewTime("12:08");
    I->setinterviewCompany("Tesla");

     cout << endl;

    I->getdisplayID();
    I->getdisplayTYPED();
    I->getdisplayDATE();
    I->getdisplayTIME();
    I->getdisplayCOMPANY();
    delete I;
    return 0;
}
```

Contributor Name: I.T.U. Sampath

Contributor ID: IT21176524

5. Notification Class

Notification.h

```
#include <iostream>
using namespace std;
class Notification{
    private:
         int notificationId;
         int fromID;
         int toID;
         string notificationDate;
         string notificationTime;
         string notificationReason;
         string notificationContent;
    public:
        void setNotificationDetails(int noid,int forid,int
toid,string nodate,string notime,string noreason,string nocontent);
        void getNotificationDetails();

};
```

Notification.cpp

```
#include <iostream>
#include " Notification.h";
using namespace std;

void Notification::setNotificationDetails(int noid,int forid,int
toid,string nodate,string notime,string noreason,string nocontent)
{
```

```
         notificationId = noid;

         fromID = forid;

         toID = toid;

         notificationDate = nodate;

         notificationTime = notime;

         notificationReason = noreason;

         notificationContent = nocontent;

}

void Notification::getNotificationDetails()

{

  cout <<"NotificationId :"<<notificationId<<endl;

  cout <<"FromID :"<<fromID<<endl;

  cout <<"ToID :"<<toID<<endl;

  cout <<"NotificationDate :"<<notificationDate<<endl;

  cout <<"NotificationTime :"<<notificationTime<<endl;

  cout <<"NotificationReason :"<<notificationReason<<endl;

  cout <<"NotificationContent :"<<notificationContent<<endl;

  cout <<endl;

}
```

main.cpp

```
#include <iostream>
#include " Notification.cpp";
using namespace std;

int main()
{
  Notification n1;

n1.setNotificationDetails(001,213,333,"5/5/2022","4.00pm","login","L
ogin Successfully");
  n1.getNotificationDetails();
  cout << "\n" << endl;
  cout << 
"..............................................................." << 
endl;

  Notification n2;
  
n2.setNotificationDetails(002,313,444,"6/6/2022","2.00pm","register"
,"Register Successfully");
  n2.getNotificationDetails();

  cout << "\n" << endl;
  cout << 
"..............................................................." << 
endl;

  return 0;
}
```

Contributor Name: A.M.S.S. Adhikari

Contributor ID: IT21173318

6. Report Class

Report.h

```
#include<iostream>
#include<cstring>
using namespace std;

class Report{
	private:
		int report_id;
		string reportformat;
		string reportsize;

	public:
		Report();
		Report(int preport_id,string preportformat,string
preportsize);
		void addReport();
		void displayReportDetails();
		void deleteReport();
};
```

Report.cpp

```
#include <iostream>
#include " Report.h";
using namespace std;

Report::Report(){}

Report::Report(int preport_id,string preportformat,string
preportsize){

	report_id=preport_id;
	reportformat=preportformat;
	reportsize=preportsize;
}
void Report::addReport(){

	cout << "Report id : " ;
	cin >> report_id;

	cout << "Report Format : " ;
	cin >> reportformat;

	cout << "Report Size :" ;
	cin >> reportsize;
}
void Report::displayReportDetails(){
	cout << "Report id : " << report_id << endl;
	cout << "Report Format : " << reportformat << endl;
	cout << "Report Size : " << reportsize << endl;
}
void Report::deleteReport(){

}
```

main.cpp

```
#include <iostream>
#include " Report.cpp";
using namespace std;

int main(){

	Report R1(6556,"Interview report","1GB");
	R1.displayReportDetails();
	R1.deleteReport();

	cout <<
"...................................................................
................................................"<<endl;
	Report R2;
	R2.addReport();
	R2.displayReportDetails();
	R2.deleteReport();

	return 0;

}
```

7. Vacancy class

Vacancy.h

```
#include<iostream>
#include<cstring>
using namespace std;
class vacancy{
	private:
		int vacancy_id;
		string vacancy_name;
		string vacancy_type;
		string vacancy_details;
		string vacancy_provider;

	public:
		vacancy();
		vacancy(int pvacancy_id,string pvacancy_name,string pvacancy_type,string vacancy_details,string vacancy_provider);
		void addvacancy();
		void searchvacancy();
		void displayvacancy();
		void deletevacancy();

};
```

Vacancy.cpp

```
#include <iostream>
#include " Vacancy.h";
using namespace std;

vacancy::vacancy(){}

vacancy::vacancy(int pvacancy_id,string pvacancy_name,string pvacancy_type,string pvacancy_details,string pvacancy_provider){

	vacancy_id = pvacancy_id;
	vacancy_name = pvacancy_name;
	vacancy_type= pvacancy_type;
	vacancy_details = pvacancy_details;
	vacancy_provider = pvacancy_provider;
}
void vacancy::addvacancy(){
	cout << "Input vacancy id : ";
	cin >> vacancy_id ;

	cout << "Input vacancy Name : ";
	cin >> vacancy_name ;

	cout << "Input vacancy Type : ";
	cin >> vacancy_type ;

	cout << "Input vacancy Details : ";
	cin >> vacancy_details ;

	cout << "Input vacancy Provider : ";
	cin >> vacancy_provider ;
}

void vacancy::searchvacancy(){}

void vacancy::displayvacancy(){
	cout << "Vacancy id : " << vacancy_id << endl;
	cout << "Vacancy Name : " << vacancy_name << endl;
	cout << "Vacancy Type : " << vacancy_type << endl;
	cout << "Vacancy Details : " << vacancy_details << endl;
	cout << "Vacancy Provider : " << vacancy_provider << endl;
}
void vacancy::deletevacancy(){}
```

main.cpp

```
#include <iostream>
#include " Vacancy.cpp";
using namespace std;

int main(){

	vacancy v1(0001,"Software Engineer","Technical","Primary
Duties and responsibility and education qualification","OrangeHRM
Inc.");
	v1.displayvacancy();
	v1.searchvacancy();
	v1.deletevacancy();
```

```
     cout <<
"..................................................................
......................................................."<<endl;

     vacancy v2;
     v2.addvacancy();
     v2.displayvacancy();
     v2.searchvacancy();
     v2.deletevacancy();

     cout <<
"..................................................................
......................................................."<<endl;

     return 0;

}
```

Contributor Name: E.A.H.A. Peiris

Contributor ID: IT21175152

8. Backup Class

Backup.h

```
#include <iostream>
#include <string>
using namespace std;

class Backup{
	private:
		int backupId;
		string backupDate;
		string backupTime;
		string backupFormat;
		string backupStorage;
		string backupLocation;

	public:
		void setBackupDetails(int bid, string bDate, string
bTime,
		string bFormat, string bStorage, string bLocation);
		void getBackupDetails();
};
```

Backup.cpp

```
#include <iostream>
#include "backup.h"
using namespace std;

void Backup::setBackupDetails(int bid, string bDate, string bTime,
	string bFormat, string bStorage, string bLocation){
	backupId = bid;
	backupDate = bDate;
	backupTime = bTime;
	backupFormat = bFormat;
	backupStorage = bStorage;
```

```
        backupLocation = bLocation;
}

void Backup::getBackupDetails(){

        cout << "Backup Details:" << endl << endl;

        cout << "1. Backup ID: " << backupId << endl
        << "2. Backup Date: " << backupDate << endl
        << "3. Backup Time: " << backupTime << endl
        << "4. Backup Format: " << backupFormat << endl
        << "5. Backup Storage: " << backupStorage << endl
        << "6. Backup Location: " << backupLocation << endl;
}
```

main.cpp

```
#include <iostream>
#include "backup.cpp"
using namespace std;

int main(){

        Backup b1;
        int bid;
        string bDate, bTime, bFormat, bStorage, bLocation;

        cout << "1. Enter Backup ID: ";
        cin >> bid;
        cout << "2. Enter Backup Date: ";
        cin >> bDate;
        cout << "3. Enter Backup Time: ";
        cin >> bTime;
        cout << "4. Backup Format (xlsx, docs, pdf): ";
        cin >> bFormat;
        cout << "5. Backup Storage (MB,KB,GB): ";
        cin >> bStorage;
        cout << "6. Backup Location (Cloud, Computer): ";
        cin >> bLocation;

        cout << endl;

        b1.setBackupDetails(bid, bDate, bTime, bFormat, bStorage,
bLocation);
```

```
	b1.getBackupDetails();

	return 0;
}
```

9. Membership Class

Membership.h

```
#include <iostream>
#include <string>
using namespace std;

class Membership{
	private:
		int memberId;
		string memberType;
		string paymemtMethod;

	public:
		void setMembershipDetails(int memId, string memType,
string payMeth);
		void getMembershipDetails();
		string selectMembershipType(string memType);
		string selectPaymentMethod(string payMeth);
};
```

Membership.cpp

```
#include "membership.h"

void Membership::setMembershipDetails(int memId, string memType,
string payMeth){

	memberId = memId;
	memberType = memType;
	paymemtMethod = payMeth;
}

void Membership::getMembershipDetails(){
        cout << "Membership Details:" << endl << endl;

        cout << "1. Member ID: " << memberId << endl
        << "2. Member Type: " << memberType << endl
        << "3. Payment Method: " << paymemtMethod << endl;
}

string Membership::selectMembershipType(string memType){

        cout << "MEMBERSHIP PLANS: BASIC | STANDARD | PREMIUM" << endl
<< endl;

        cout << "Please select your Membership Plan (B | S | P): ";
        cin >> memType;
        cout << endl;

        return memType;
}

string Membership::selectPaymentMethod(string payMeth){

        cout << "PAYMENT METHODS: CREDIT CARD | DIPOSIT | PAYPAL" <<
endl << endl;

        cout << "Please select your Payment Method (C | D | P): ";
        cin >> payMeth;
        cout << endl;

        return payMeth;
}
```

main.cpp

```
#include "membership.cpp"

int main(){

        Membership m1;
        string become, memType, payMeth;
        int memId;

        cout << "Enter member ID: ";
        cin >> memId;
```

```
	cout << endl;

	memType = m1.selectMembershipType(memType);
	payMeth = m1.selectPaymentMethod(payMeth);
	m1.setMembershipDetails(memId, memType, payMeth);
	m1.getMembershipDetails();

	return 0;
}
```

10. Question Class

Question.h

```
#include <iostream>
#include <string>
using namespace std;

class Question{
	private:
		int questionId;
		int examId;
		string question;
		string answer;
	public:
		void addQuestion(int qId, int eId, string question,
string answer);
		void getQuestionDetails();
		void deleteQuestion();
};
```

Question.cpp

```
#include "question.h"

void Question::addQuestion(){

	cout << "1. Enter Question ID: ";
	cin >> questionId;
	cout << "2. Enter Exam ID: ";
	cin >> examId;
```

```
        cout << "3. Question: ";
        cin >> question;
        cout << "4. Answer: ";
        cin >> answer;
}

void Question::getQuestionDetails(){

        cout << endl;

        cout << "Question Details:" << endl << endl;

        cout << "1. Question ID: " << questionId << endl
        << "2. Exam ID: " << examId << endl
        << "3. Question: " << question << endl
        << "4. Answer: " << answer << endl;
}
```

main.cpp

```
#include "question.cpp"

int main(){

        Question quiz1;
        int qId, eId;
        string que, ans;

        cout << "1. Enter Question ID: ";
        cin >> qId;
        cout << "2. Enter Exam ID: ";
        cin >> eId;
        cout << "3. Question: ";
        cin >> que;
        cout << "4. Answer: ";
        cin >> ans;

        cout << endl;

        quiz1.addQuestion(qId, eId, que, ans);
        quiz1.getQuestionDetails();

        return 0;
}
```

11. Selection Class

Selection.h

```
#include <iostream>
#include <string>
using namespace std;

class Selection{
	private:
		int selectId;
		int applicantId;
		float qualificationMark;
		float examMark;
		string selectStatus;
	//	float finalMark;

	public:
		void setSelectionDetails(int sId, int appId, float qMark, 
float eMark, string sStatus);
		void getSelectionDetails();
		float calculateFinalMark(float qMark, float eMark);
		string getSelectStatus(float finalMark);
};
```

Selection.cpp

```
#include <iostream>
#include "selection.h"
using namespace std;

void Selection::setSelectionDetails(int sId, int appId, float qMark, 
float eMark, string sStatus){

	selectId = sId;
	applicantId = appId;
	qualificationMark = qMark;
	examMark = eMark;
	selectStatus = sStatus;
}

void Selection::getSelectionDetails(){
	cout << "Selection Details:" << endl << endl;

	cout << "- Selection ID: " << selectId << endl
	<< "- Applicant ID: " << applicantId << endl
	<< "- Qualification Mark: " << qualificationMark << endl
	<< "- Exam Mark: " << examMark << endl
	<< "- Select Status: " << selectStatus << endl << endl;
}

float Selection::calculateFinalMark(float qMark, float eMark){

	float finalMark = (qMark + eMark) / 2.0;
	return finalMark;
}

string Selection::getSelectStatus(float finalMark){

	string sStatus;

	if(finalMark >= 75){

		sStatus = "Selected";
	}
	else{

		sStatus = "Not Selected";
	}

	return sStatus;
}
```

Main.cpp

```
int main(){

	Selection s1;
	int sId, appId;
	float qMark, eMark, finalMark;
	string sStatus;

	cout << "1. Enter Selection ID: ";
	cin >> sId;
	cout << "2. Enter Applicant ID: ";
	cin >> appId;
	cout << "3. Enter Qualification Mark: ";
	cin >> qMark;
	cout << "4. Enter Exam Mark: ";
	cin >> eMark;

	cout << endl;

	finalMark = s1.calculateFinalMark(qMark, eMark);

	sStatus = s1.getSelectStatus(finalMark);

	s1.setSelectionDetails(sId, appId, qMark, eMark, sStatus);

	s1.getSelectionDetails();

	return 0;
}
```

12. User Class

```
//////////User.h

# include <iostream>
#include <string>
using namespace std;

class user{
	protected:
		string userName;
		string userMobile;
		string userEmail;
		string userAddress;
	public:
		void addUser(string usrName, string usrMob, string
usrMail, string usrAddress){
			userName = usrName;
			userMobile = usrMob;
			userEmail = usrMail;
			userAddress = usrAddress;
		}
		void displayUserDetails(){
			cout << "User Name: " << userName << endl
```

```
                << "User Mobile: " << userMobile << endl
                << "User Email: " << userEmail << endl
                << "User Address" << userAddress << endl << endl;
            }
};
```

13. Company Class

```
//////////Company.h

#include <iostream>
#include <string>
#include "user.h"
using namespace std;

class Company : public User{
      private:
            int comId;
            string comCategory;
            string comUserName;
            string comPassword;
      public:
            void addCompany(int cid, string cName, string cType,
string cMob,
            string cMail, string cAddress, string cUserName, string
cPassword)
                  : addUser(string name, string mob, string mail,
string address){
                        comId = cid;
                        comName = cName;
                        comType = cType;
                        comMobile = cMob;
                        comEmail = cMail;
                        comAddress = cAddress;
                        comUserName = cUserName;
                        comPassword = cPassword;
                  }
            void displayCompanyDetails();
            void deleteCompany();
};
```

Company.cpp

```
#include "company.h"

void Company::displayCompanyDetails(){

      cout << "Company Details:" << endl << endl;

      cout << "- Company ID: " << comId << endl
      << "- Company Name: " << name << endl
      << "- Company Type: " << comType << endl
      << "- Company Mobile: " << mob << endl
      << "- Company Email: " << mail << endl
      << "- Company Address: " << address << endl
      << "- Username: " << comUserName << endl
      << "- Password: " << comPassword << endl << endl;
}
```

main.cpp

```
int main(){

      Company c1;
      int cid;
      string cName, cType, cMob, cMail, cAddress, cUserName,
cPassword;

      cout << "1. Enter Company ID: ";
      cin >> cid;
      cout << "2. Company Name: ";
      cin >> cName;
      cout << "3. Enter Company Type: ";
      cin >> cType;
      cout << "4. Enter Company Mobile: ";
      cin >> cMob;
      cout << "5. Enter Company Email: ";
      cin >> cMail;
      cout << "6. Enter Company Address: ";
      cin >> cAddress;
      cout << "7. Enter Company Username: ";
      cin >> cUserName;
      cout << "8. Enter Company Password: ";
      cin >> cPassword;
```

```
	cout << endl;

	c1.addCompany(cid, cName, cType, cMob, cMail, cAddress, 
cUserName, cPassword);
	c1.displayCompanyDetails();

	return 0;
}
```

8. Individual Contribution

Member Name: E.A.H.A Peiris

Member ID: IT21175152

Contribution:

- Created the CRC cards for Company, Membership, Backup, Selection, Question and User classes.
- Created the class diagrams for Company, Membership, Backup, Selection, Question and User classes.
- Implemented the coding for Company, Membership, Backup, Selection, Question and User classes.
- Gathering the requirements.
- Did the noun verb analysis.
- Designed the class diagram.
- Created the final project file.

Member Name: A.M.S.S. Adhikari

Member ID: IT21173318

Contribution:

- Created the CRC cards for Report and Vacancy classes.
- Created the class diagrams for Report and Vacancy classes.
- Implemented the coding for Report and Vacancy classes.
- Gathering the requirements.

Member Name: E.D.T.V Elvitigala

Member ID: IT21176838

Contribution:

- Created the CRC card for Interview class.
- Created the class diagram for Interview class.
- Implemented the coding for Interview class.
- Gathering the requirements.

Member Name: I.T.U.Sampath

Member ID: IT21176524

Contribution:

- Created the CRC card for Notification class.
- Created the class diagram for Notification class.
- Implemented the coding for Notification class.
- Gathering the requirements.

Member Name: D.P. Liyanagama

Member ID: IT21172878

Contribution:

- Created the CRC cards for Admin, Applicant and Exam classes.
- Created the class diagrams for Admin, Applicant and Exam classes.
- Implemented the coding for Admin, Applicant and Exam classes.
- Gathering the requirements.

9. Marking Scheme