Topic : **Online Job Portal**

Group no : **MLB 01.02_07**

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.

Registration No	**Name**	**Contact Number**
IT21197246	A.M.Y.V.B.Abeykoon	0763319667
IT21198168	Weerasekara N.P	0741083333
IT21196256	Sudasinghe S.M.V	071 354 0088
IT21193804	Weerasekara D.D.R.R	0776507661
IT21193354	Prabhashwara W.U.H	70 0 7700

Part 01

01)System Requirements

- The System should function 24/7
- System should be user friendly
- Guest users can overview the system, to use the system, they must register with the system by providing details such as Name, Address, NIC, Email, contact.
- Registered users are of two types called Jobseeker and company where they can log into the system by entering the correct username and password
- Jobseeker can search and apply for jobs in the system
- After applying, application id is generated by the system
- Companies can post job vacancies in the system
- After placing job vacancy, vacancy id is generated by system
- Jobseekers can enter their qualifications on their account.
- Details should be confirmed by the web operator
- Users can edit, add, delete details on their profiles.
- Web operator can delete or update the account details.
- System should generate a unique id for the vacancies
- registered users can send their reviews about the system
- after sending review, feedback id is generated by the system

02)Noun & Verb Analysis

Nouns in red

- A jobseeker/company user can visit the online job portal via URL address and browse the site.
- Registered/unregistered company/jobseeker can search for job vacancies via search form.
- In order to apply for a vacancy, an unregistered jobseeker would have to register to the job portal and become a registered job seeker by entering details such as name, age, NIC etc.

- In order to post a vacancy, an unregistered company needs would have to register and become a registered company.
- Upon making the account, if there are any mistakes with the user's details. The user can edit the program again.
- The jobseeker can apply for vacancy by submitting application for a job vacancy in the jobs page.
- The jobseeker can view the applications that has been applied for.
- The jobseeker can cancel selected application via the edit profile.
- The users of the system can send feedback about the portal and its features and users via the feedback form.
- Moderator will view and respond to the feedback.
- Administrator can view, edit and delete user account details by managing user.

Verbs in Blue

- A jobseeker/company user can visit the online job portal via URL address and browse the site.
- Registered/unregistered company/jobseeker can search for job vacancies via search form.
- In order to apply for a vacancy, an unregistered jobseeker would have to register to the job portal and become a registered job seeker by entering details such as name, age, NIC etc.
- In order to post a vacancy, an unregistered company needs would have to register and become a registered company.
- Upon making the account, if there are any mistakes with the user's details. The user can edit the program again.
- The jobseeker can apply for vacancy by submitting application for a job vacancy in the jobs page.
- The jobseeker can view the applications that has been applied for.
- The jobseeker can cancel selected application via the edit profile.
- The users of the system can send feedback about the portal and its features and users via the feedback form.
- Moderator will view and respond to the feedback.
- Administrator can view, edit and delete user account details by managing user.

Identified Nouns

1. Guest user
2. Registered user
3. Company
4. Jobseeker
5. Job Vacancies
6. Job Applications
7. Administrator Staff
8. Feedback

Part 02

CRC Cards

Guest User	
Responsibility	**Collaborators**
Register to the system	
Allow to view the job vacancies	Job Vacancies

Registered User	
Responsibility	**Collaborators**
Can view the job vacancies	Job Vacancies

Add and update user details	

Company	
Responsibility	**Collaborators**
Log in to the system	Registered User
Publication of job vacancies	Job Vacancies
Examining job applications	Job Applications

Job Seeker	
Responsibility	**Collaborators**
Log in to the system	Registered User
Search job vacancies	Job Vacancies
Apply for a job	Job Applications

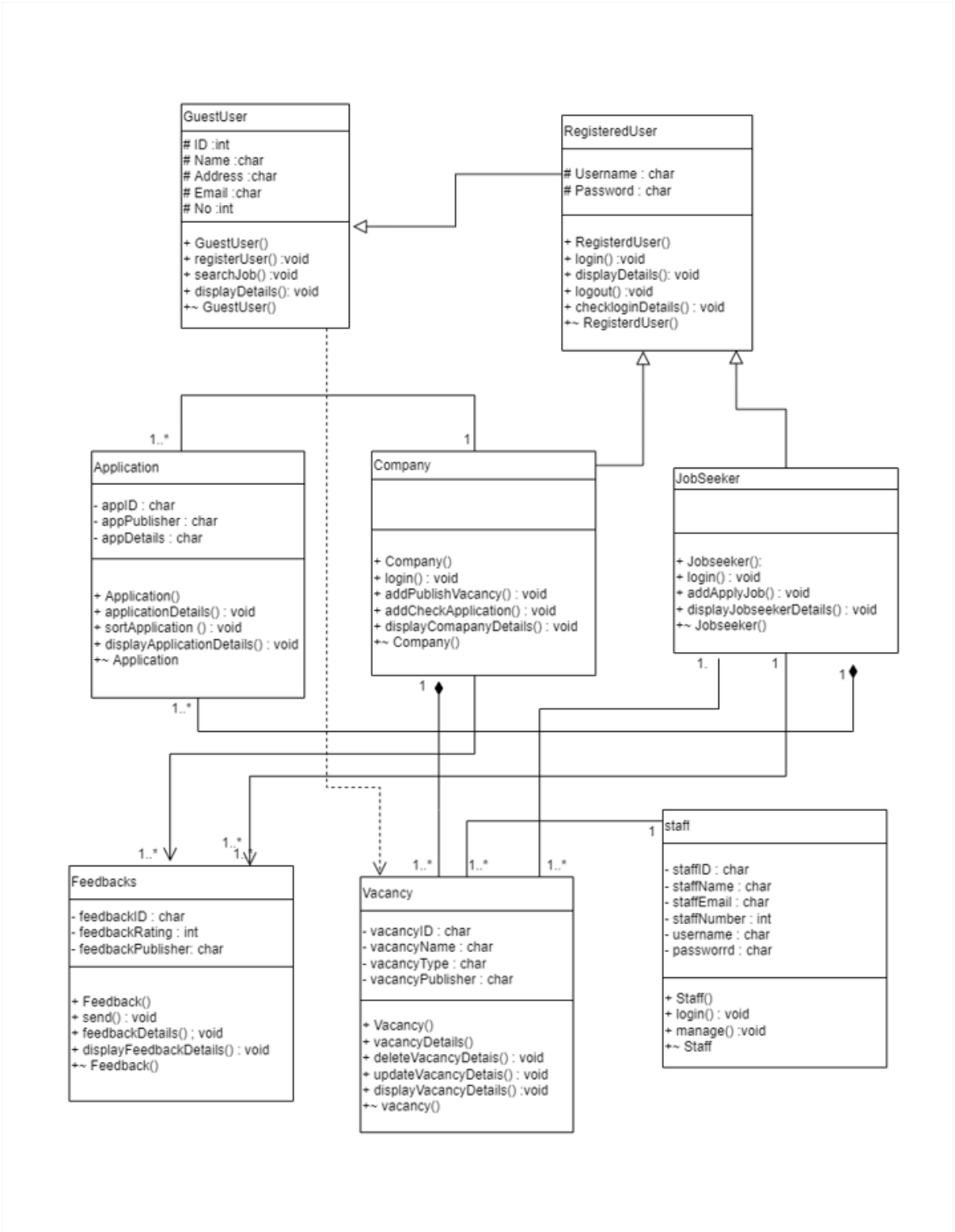
Job Vacancies	
Responsibility	**Collaborators**
Add job vacancy details	Company
Delete job vacancy details	Administrator staff
Update job vacancy details	Company , Staff

Job Applications	
Responsibility	**Collaborators**
Sort Applications	Company

Administrator Staff	
Responsibility	**Collaborators**
Login to the system	
Conform Vacancy details	Job Vacancies

Feedback	
Responsibility	**Collaborators**
Send feedbacks	Company, Jobseeker

UML Class Diagram


GuestUser
ID :int
Name :char
Address :char
Email :char
No :int
+ GuestUser()
+ registerUser() :void
+ searchJob() :void
+ displayDetails(): void
+~ GuestUser()
RegisteredUser
Username : char
Password : char
+ RegisterdUser()
+ login() :void
+ displayDetails(): void
+ logout() :void
+ checkloginDetails() : void
+~ RegisterdUser()
Application
- appID : char
- appPublisher : char
- appDetails : char
+ Application()
+ applicationDetails() : void
+ sortApplication () : void
+ displayApplicationDetails() : void
+~ Application
Company
+ Company()
+ login() : void
+ addPublishVacancy() : void
+ addCheckApplication() : void
+ displayComapanyDetails() : void
+~ Company()
JobSeeker
+ Jobseeker():
+ login() : void
+ addApplyJob() : void
+ displayJobseekerDetails() : void
+~ Jobseeker()
Feedbacks
- feedbackID : char
- feedbackRating : int
- feedbackPublisher: char
+ Feedback()
+ send() : void
+ feedbackDetails() ; void
+ displayFeedbackDetails() : void
+~ Feedback()
Vacancy
- vacancyID : char
- vacancyName : char
- vacancyType : char
- vacancyPublisher : char
+ Vacancy()
+ vacancyDetails()
+ deleteVacancyDetais() : void
+ updateVacancyDetais() : void
+ displayVacancyDetails() :void
+~ vacancy()
staff
- staffID : char
- staffName : char
- staffEmail : char
- staffNumber : int
- username : char
- passworrd : char
+ Staff()
+ login() : void
+ manage() :void
+~ Staff

Class Coding(C++)

//Declaration of Classes//

//Declaration of Guest user class (GuestUser.h)//

```
#include "Vacancy.h"
class GuestUser
{
protected:
	int userID;
	char userName[20];
	char userAddress[30];
	char userEmail[30];
	char userphoneNumber[10];
public:
	GuestUser();
	GuestUser(int puserid, const char puserName[], const char
		puserAddress[], const char puserEmail[], const char puserPHno[]);
	void searchVacancies(Vacancy* pVac);
	void registerUser();
	virtual void displayDetails();
	~GuestUser();
};
```

//Declaration of Registered user class (RegisteredUser.h)//

```
#include"GuestUser.h"
class RegisteredUser :public GuestUser
{
protected:
	char userUsername[10];
	char userPassword[10];
public:
	RegisteredUser();
	RegisteredUser(const char puserUsername[], const char
		puserPassword[], int puserid, const char puserName[], const char
		puserAddress[], const char puserEmail[], const char pusertNo[]);
	virtual void displayDetails();
	void login();
	void logout();
	char checkLoginDetails();
	~RegisteredUser();
```

//Declaration of Jobseeker class (Jobseeker.h)//

```
#include "Application.h"
#include "Vacancy.h"
#include "RegisteredUser.h"
#define SIZE 2
#define SIZE 5
class Jobseeker :public RegisteredUser
{
private:
	int noOfVacancies;
	Vacancy* applyVac[SIZE];
	application* apply[SIZE];
public:
	Jobseeker();
	Jobseeker(const char usName[], const char usPwd[], int id, const
		char name[], const char address[], const char email[], const char telno[],
int pnoOfVacancies, int apply1, int apply2);
	void addApplyingJob(Vacancy* papplyVac);
	void login();
	void displayDetails();
	~Jobseeker();
};
```

//Declaration of Company class (Company.h)//

```
#include "Registerduser.h"
#include "Application.h"
#include "vacancy.h"
class Company
{
private:
	int noOfVacancy
	Vacancy* cPubishVacancy[];


public:
	Company();
	Company(const char ID, const char Name, const char Address, const char
Email, const char No, int pnoOfVacancy );
	void login();
	void addPublishVacancy(Vacancy*cPubishVacancy);
	void displaycomplanyDetails();
	void addCheckApplication();
	~Company();

};
```

//Declaration of Application class (Application.h)//

```
#include "Company.h"
#include "Jobseeker.h"

class Application
```

```
{
private:
	char appID[6];
	char appPublisher[50];
	char appDetails[100];

public:
	Application();
	Application(const char nappID[], const char nappPublisher[], const char
nappDetails[]);
	void applicationDetais();
	void sortApplication();
	void displayApplicationDetails();
	~Application();

};



//Declaration of Feedback class (Feedback.h)//

#include "Company.h"
#include "Jobseeker.h"

class Feedback
{


	private:
		char feedbackID[6];
		int feedbackRating;
		char feedbackPublisher[20];

	public:
		Feedback();
		Feedback(const char nfeedbackID[], const char nfeedbackRating[],
const char nfeedbackPublisher[]);
		void send();
		void feedbackDetails();
		void displayfeedbackDetails();
		~Feedback();


};

//Declaration of Vacancy class (Vacancy.h)//

#pragma once
#include "Company.h"
class Vacancy
{
private:
	char vacancyID[10];
	char vacancyName[20];
	char vacancyType[20];
	char vacancyPublisher[20];

public:
	Vacancy();
	Vacancy(const char pvacancyID[], const char pvacancyName[], const char
pvacanyType[], const char pvacancyPublisher);
```

```
        void VacancyDetails();
        void deleteVacancyDetails();
        void updateVacancyDetails();
        void displayVacancyDetails();

        ~Vacancy()
};

//Declaration of Staff class (Staff.h)

class Staff
{
private:
        char staffID[10];
        char staffName[20];
        char staffEmail[20];
        char staffNumber[10];
        char username[20];
        char password[20];

public:
        Staff();
        Staff(int pstaffID, const char pstaffName[], const char pstaffEmail[],
const char pstaffNumber[], const char pstaffUsername[], const char
pstaffPassword[];

        void login();
        void manage();
        ~Staff();
};



//Implementation of GuestUser class(Guestuser.cpp)//

#include "GuestUser.h"
#include <cstring>
GuestUser::GuestUser()
{
        userID = 0;
        strcpy(userName, "");
        strcpy(userAddress, "");
        strcpy(userEmail, "");
        strcpy(userphoneNumber, "0000000000");
}
GuestUser::GuestUser(int puserid, const char puserName[], const char
        puserAddress[], const char puserEmail[], const char userPHno[])
{
        userID = puserid;
        strcpy(userName, puserName);
        strcpy(userAddress, puserAddress);
        strcpy(userEmail, puserEmail);
        strcpy(userphoneNumber, userPHno);
}
void GuestUser::searchVacancies(Vacancy* pVac)
{
```

```
}
void GuestUser::registerUser()
{
}
void GuestUser::displayDetails()
{
}
GuestUser::~GuestUser()
{
	//Destructor
}


//Implementation of Registered User class (RegisteredUser.cpp)//

#include "RegisteredUser.h"
#include <cstring>
RegisteredUser::RegisteredUser()
{
	strcpy(userUsername, "");
	strcpy(userPassword, "");
}
RegisteredUser::RegisteredUser(const char puserUsername[],
	const char puserPassword[], int puserid, const char puserName[],
	const char puserAddress[], const char puserEmail[], const char
	puserNo[]) : GuestUser(puserid, puserName, puserAddress, puserEmail,
		puserNo)
{
	strcpy(userUsername, puserUsername);
	strcpy(userPassword, puserPassword);
}
void RegisteredUser::displayDetails()
{
}
void RegisteredUser::login()
{
}
void RegisteredUser::logout()
{
}
char RegisteredUser::checkLoginDetails()
{
}
RegisteredUser::~RegisteredUser()
{
	//Destructor
}
```

```
//Implementation of Jobseeker class (Jobseeker.cpp)//

#include "Jobseeker.h"
#define SIZE 2
Jobseeker::Jobseeker()
{
	noOfVacancies = 0;
}
Jobseeker::Jobseeker(const char usName[], const char usPwd[], int id,
	const char name[], const char address[], const char email[], const
	char telno[], int pnoOfVacancies, int apply1, int apply2)
:RegisteredUser(usName,
		usPwd, id, name, address, email, telno)
{
	apply[0] = new applying(apply1);
	apply[1] = new applying(apply2);
}
{
	noOfVacancies = pnoOfVacancies;
}
void Jobseeker::addApplingVacancy(Vacancy* papplyVac)
{
	if (noOfVacancies < SIZE)
	{
		applyVac[noOfVacancies] = papplyVac;
		noOfVacancies++;
	}
}
void Jobseeker::login()
{
}
void Jobseeker::displayJobseekerDetails()
{
}
Jobseeker::~Jobseeker()
{
	//Destructor
}



//Implementation of Company class (Company.cpp)//

#include "Company.h"

Company::Company()
{
	noOfVacancy=0
}

Company::Company(const char ID, const char Name, const char Address, const char
Email, const char No, int pnoOfVacancy)
{
```

```
	noOfVacancy = pnoOfVacancy;
}

void Company::login()
{
}

void Company::addPublishVacancy(Vacancy* cPubishVacancy)
{
}

void Company::displaycomplanyDetails()
{
}

void Company::addCheckApplication()
{
}

Company::~Company()
{
}


//Implementation of Application class (Application.cpp)//

#include "Application.h"
#include <cstring>


Application::Application()
{
	strcpy(appID, "");
	strcpy(appPublisher, "");
	strcpy(appDetail, "");
}

Application::Application(const char nappID[], const char nappPublisher[], const
char nappDetails[])
{
	strcpy(appID, nappID);
	strcpy(appPublisher, nappPublisher);
	strcpy(appDetail, nappDetail);

}

void Application::applicationDetais()
{
}

void Application::sortApplication()
{
}

void Application::displayApplicationDetails()
{
}

Application::~Application()
{
}
```

```cpp
//Implementation of Feedback class (Feedback.cpp)//


#include "Feedback.h"

Feedback::Feedback()
{
	strcpy(feedbackID, "");
	feedbackRating = 0 ;
	strcpy(feedbackPublisher, "");

}

Feedback::Feedback(const char nfeedbackID[], const int nfeedbackRating, const
char nfeedbackPublisher[])
{
	strcpy(feedbackID, nfeedbackID);
	feedbackRating = nfeedbackRating;
	strcpy(feedbackPublisher, nfeedbackPublisher);
}

void Feedback::send()
{
}

void Feedback::feedbackDetails()
{
}

void Feedback::displayfeedbackDetails()
{
}

Feedback::~Feedback()
{
}


//Implementation of Vacancy Class (Vacancy.cpp)//

#include "Vacancy.h"
#include<cstring>

Vacancy::Vacancy(){
}
Vacancy::Vacancy(const char pvacancyID[], const char pvacancyName[], const char
pvacancyType[], const char pvacancyPublisher[]) {
	strcpy_s(vacancyID, pvacancyID);
	strcpy_s(vacancyName, pvacancyName);
	strcpy_s(vacancyType, pvacancyType);
	strcpy_s(vacancyPublisher, pvacancyPublisher);
}
void Vacancy::VacancyDetails() {

}
void Vacancy::deleteVacancyDetails() {

}
void Vacancy ::updateVacancyDetails() {

}
void Vacancy::displayVacancyDetails() {
```

```
}
Vacancy::~Vacancy()
{

}



//Implementation of Staff Class (Staff.cpp)//

#include "Staff.h"
#include<cstring>

Staff::Staff() {

}
Staff::Staff(const char pstaffID, const char pstaffName, const char pstaffEmail,
const char pstaffNumber, const char pusername, const char ppassword) {

}
void Staff::login() {

}
void Staff::manage() {

}
Staff::~Staff() {

}

//Implementation of main class//

#include <iostream>
#include <cstring>
#include "GuestUser.h"
#include "RegisteredUser.h"
#include "Jobseeker.h"
#include "Company.h"
#include "Application.h"
#include "Feedback.h"
#include "Vacancy.h"
#include "Staff.h"
using namespace std;

int main(){

//object creation//
GuestUser* rg = new RegisteredUser();

RegisteredUser* jobseeker = new Jobseeker();

RegisteredUser*company = new Company();

Application* application = new Application();

Staff*staff = new Staff();

Vacancy*vacancy = new Vacancy();

Feedback* feedback = new Feedback();
```

//method calling//

rg->login();
rg->displayDetails();

jobseeker->login();
jobseeker->displayDetails();

```
company->void login();
company->void addPublishVacancy(Vacancy*cPubishVacancy);
company->void displaycomplanyDetails();
company->void addCheckApplication();

application->void applicationDetais();
application->void sortApplication();
application->void displayApplicationDetails();

feedback->void send();
feedback->void feedbackDetails();
feedback->void displayfeedbackDetails();

staff->void login();
staff->void manage();

vacancy-> void VacancyDetails();
vacancy-> void deleteVacancyDetails();
vacancy-> void updateVacancyDetails();
vacancy-> void displayVacancyDetails();

//delete Dynamic objects//

delete rg;
delete jobseeker;
delete company;
delete application;
delete feedback;
delete staff;
delete vacancy;

return 0;
}
```

Individual Contribution

	Student ID	Student Name	Individual Contribution
1	IT21197246	A.M.Y.V.B.Abeykoon	• Noun Verb Analysis • Vacancy class, Staff class(C++) • Assembly and editing
2	IT21198168	Weerasekara N.P	• CRC diagrams, identify classes • Guest user class, Registered user(C++)
3	IT21196256	Sudasinghe S.M.V	• CRC diagrams • Jobseeker class(C++)
4	IT21193804	Weerasekara D.D.R.R	• UML diagram, system requirements • Company class, Application class(C++)
5	IT21193354	Prabhashwara W.U.H	• UML diagram • Feedback class(C++)