BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

Topic : **Recruitment Company System**

Group no : **MLB_CSNE_01.02_01**

Campus : **Malabe**

Submission Date: **19/05/2024**

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
IT23366336	A.M.N.S. WEERARATHNE	078 688 0725
IT23363670	E.D.O. SAMARAKOON	076 194 1032
IT23363052	K.H.S. DAMSILUNI	070 624 7244
IT23362994	H.M.C.M.B. HEARATH	078 556 1199

Contents

1. Description of the requirements

- Any company that wants to find new candidates can register with our system by including their necessary information like company name, industry, location, email and contact information.
- Registered client companies will be able to log in to the website by their IDs.
- Then, the registered clients send required job vacancies to the admin for approval.
- Admin approves the job vacancies from registered client companies by checking whether the necessary information like job title, salary range, requirements like facts are mentioned.
- Then, the admin does the job posting process on the website.
- Any candidate can register to the system and search for job vacancies on the websites through their IDs.
- Then, the registered candidates can submit their applications to the given link under the job posts by considering their qualifications.
- Next, the admin collects all the applications properly.
- Then, the admin forwards applications to the relevant clients.
- The client checks the applications of all the candidates and selects the qualified ones.
- Then, they contact the qualified candidates directly to inform them about the interviews.
- Qualified candidates will face both the physical and online interviews administered by the client companies.
- Client companies select the most qualified candidates from the interviews.
- Then, the client offers placements/job offerings for the most qualified candidates.

- After that, the most qualified candidates receive those job offerings.
- Finally, any registered candidate who was selected or not and all the client companies can give feedback to our recruitment company.
- The admin reads all the feedback and uses it for the improvement of our company.
- Admin generates reports on recruitment matrices.

2. Classes Identified

- User
- Admin (Inherited from the User)
- Client company (Inherited from the User)
- Candidate (Inherited from the User)
- Application
- Interview
- Job post
- Job offer
- Feedback
- Report

3. CRC Card

ClientCompany	
Responsibilities	**Collaborators**
Register with the system	
Send job vacancies to the admin for approval	Admin
Check pending applications for their job posts	Application
Send Sheduled interviews Details	Interview
Receive sheduled job interview	Interview
Send feedback	Feedback

JobPost	
Responsibilities	**Collaborators**
Store job vacancy details	
Await approval from the admin	Admin
Display approved job vacancies to candidates	Candidate

Candidate	
Responsibilities	**Collaborators**
Register with the system	
Search for job vacancies	JobPost
Submit applications for job vacancies	Application
Receive job interview schedules	Interview
Receive job offers	JobOffer
Send feedback	Feedback

Application	
Responsibilities	**Collaborators**
Store submitted applications	
Forward applications to relevant ClientCompany	ClientCompany
Notify candidates of application status	Candidate

Admin	
Responsibilities	**Collaborators**
Approve job vacancies from client companies	ClientCompany
Manage job posts	JobPost
Schedule interviews between companies and candidates	Candidate
Generate reports on recruitment matrics	Report
Manage applications	Application
Manage feedbacks	Feedback

Interview	
Responsibilities	**Collaborators**
Schedule interviews between candidates and companies	Admin
Notify candidates and companies about interview schedules	Candidate, ClientCompany
Record interview outcomes	

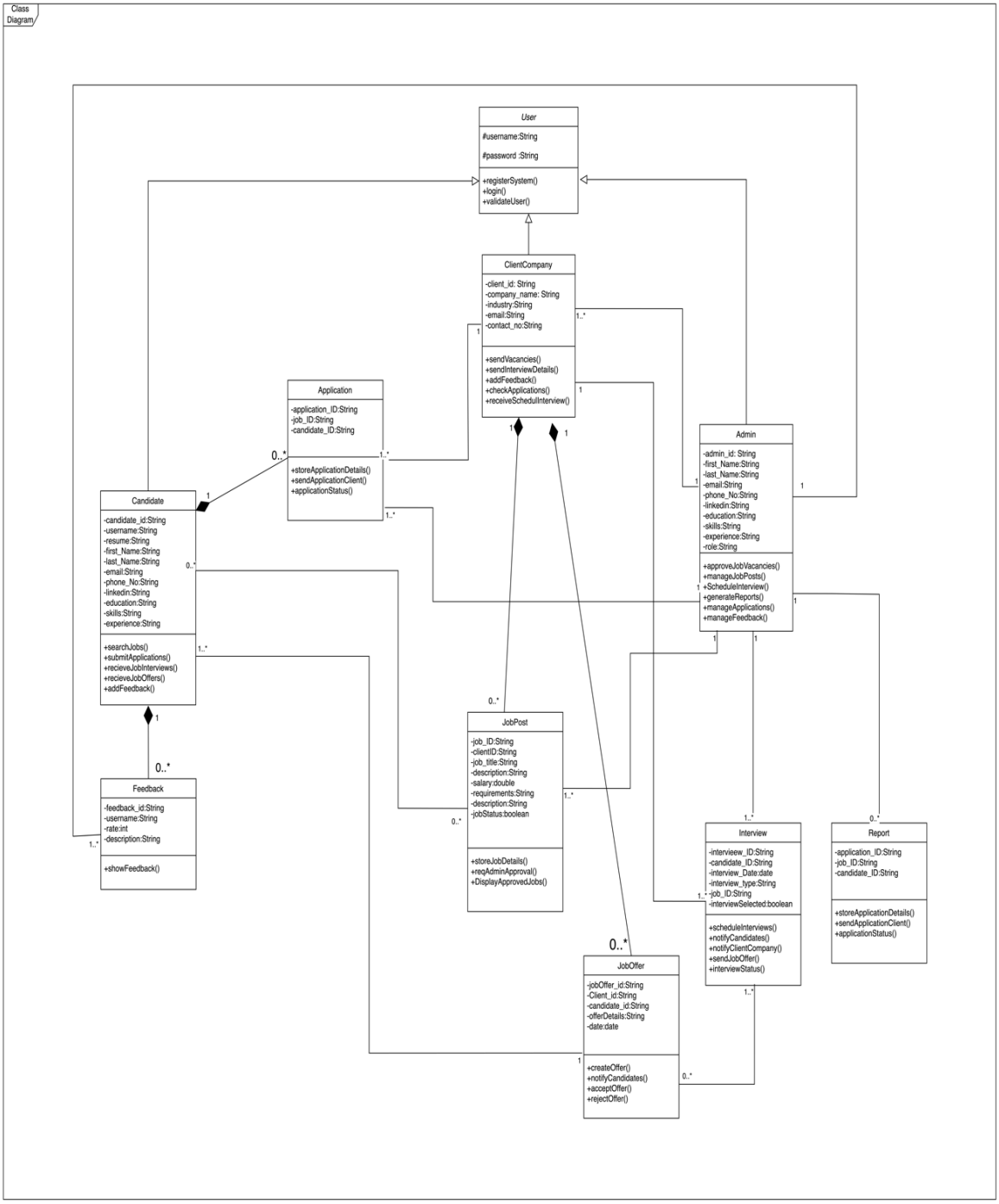
JobOffer	
Responsibilities	**Collaborators**
Create job offers for selected candidates	ClientCompany , Interview
Notify candidates of job offers	
Manage acceptance or rejection of job offers	Candidate

Report	
Responsibilities	**Collaborators**
Collect data on job postings, candidates, and interview outcomes	JobPost, Interview, Candidate
Generate reports on recruitment matrics	
Provide insights into the recruitment process	Admin

User	
Responsibilities	**Collaborators**
Login to the System	
Validate User	

Feedback	
Responsibilities	**Collaborators**
Show feedback details	
Show approved feedback	

4. Class diagram



5. Coding for the classes

Main.cpp

```
#include "Admin.h"
#include "Application.h"
#include "Candidate.h"
#include "ClientCompany.h"
#include "Feedback.h"
#include "Interview.h"
#include "JobOffer.h"
#include "JobPost.h"
#include "iostream"

int main() {
  // Create an Admin
  Admin admin("admin001", "adminpass001", "A123", "Senumi",
"Damsiluni",
              "SenumiDamsiluni@gmail.com", "0712233945",
              "Admin Sliit,Kaduwela Rd, Malabe");
  // Create a Candidate
  Candidate candidate("C001", "Amara", "Amara123",
"AmaraCV.pdf", "Amara",
                      "Senavirathne", "AmaraSena@gmail.com",
"0784838222",
                      "3rd Lane,Gamunupura rd,Malabe,
Kothalawala",
                      "linkedin.com/Amara", "Programming,
Communication",
                      "1 year");
  // Create a ClientCompany
  ClientCompany company("virtusaSL", "VirtusaSL123", "Cl001",
"Virtusa Pvt.LTD",
```

```
                        "IT Services",
"VirtusaSLCMB@virtusa.com",
                        "0112238384");

  // Create an Application
  Application application("APP001", "JOB001", "C001");

  // Create Feedback
  Feedback feedback("F001", "Amara", "5 stars", "Excellent
service!");

  // Create an Interview
  Interview interview("INT001", "C001", "JOB001", "2024-05-20",
"10:00 AM",
                      true);

  // Create a JobOffer
  JobOffer jobOffer("JOB001", "C001", "C001", "Offer details
here");

  JobPost jobPost(
      "JOB001", "Cl001", "Software Developer",
      "Manual Debugging Drudgery: Forget the old ways of
debugging. You're "
      "equipped with AI to spot and solve issues effortlessly.",
      "Skills required...Programming/Communication", "OrianCity
Colombo 08",
      "$80,000 - $100,000", true);

  // Display Admin details
  std::cout << "Admin Details:" << std::endl;
  admin.display();
  std::cout << std::endl;
  // Display Candidate details
```

```
std::cout << "Candidate Details:" << std::endl;
candidate.display();
std::cout << std::endl;
// Display ClientCompany details
std::cout << "Client Company Details:" << std::endl;
company.display();
std::cout << std::endl;
// Display Application details
std::cout << "Application Details:" << std::endl;
application.display();
std::cout << std::endl;
// Show Feedback details
std::cout << "Feedback Details:" << std::endl;
feedback.showFeedback();
std::cout << std::endl;
// Display Interview details
std::cout << "Interview Details:" << std::endl;
interview.display();
std::cout << std::endl;
// Display Job Offer details
std::cout << "Job Offer Details:" << std::endl;
jobOffer.display();
std::cout << std::endl;
// Display JobPost details
std::cout << "Job Post Details:" << std::endl;
jobPost.display();
std::cout << std::endl;

return 0;
```

User.h

```
#include <string>

class User
{
private:
  std::string username;
  std::string password;

public:
  User(const std::string &username, const std::string
&password);
  void registerSystem();
  void login();
  void validateUser();
  void display();
};
```

User.cpp

```
#include "User.h"

User::User(const std::string &username, const std::string
&password)
    : username(username), password(password) {}

void User::registerSystem() {}
void User::login() {}
void User::validateUser() {}
void User::display() {}
```

Admin.h

```
#include "User.h"
#include <string>

class Admin : public User
{
private:
  std::string admin_id;
  std::string first_Name;
  std::string last_Name;
  std::string email;
  std::string phone;
  std::string address;

public:
  Admin(const std::string &username, const std::string
&password,
        const std::string &adminID, const std::string &fname,
        const std::string &lname, const std::string &email,
        const std::string &phone, const std::string &address);
  void approveJobVacancies();
  void manageJobPosts();
  void scheduleInterview();
  void generateReport();
  void manageApplication();
  void manageFeedback();
  void display();
};
```

Admin.cpp

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

Admin::Admin(const std::string &username, const std::string
&password,
             const std::string &adminID, const std::string
&fname,
             const std::string &lname, const std::string &email,
             const std::string &phone, const std::string
&address)
    : User(username, password), admin_id(adminID),
first_Name(fname),
      last_Name(lname), email(email), phone(phone),
address(address) {}

void Admin::approveJobVacancies() {}
void Admin::manageJobPosts() {}
void Admin::scheduleInterview() {}
void Admin::generateReport() {}
void Admin::manageApplication() {}
void Admin::manageFeedback() {}
void Admin::display()
{
  std::cout << "Admin ID: " << admin_id << "\n"
            << "First Name: " << first_Name << "\n"
            << "Last Name: " << last_Name << "\n"
            << "Email: " << email << "\n"
            << "Phone: " << phone << "\n"
            << "Address: " << address << "\n";
}
```

Candidate.h

```
#include "User.h"
#include <string>

class Candidate : public User
{
private:
  std::string candidate_id;
  std::string resume;
  std::string first_Name;
  std::string last_Name;
  std::string email;
  std::string phone;
  std::string address;
  std::string linkedIn;
  std::string skills;
  std::string experience;

public:
  Candidate(const std::string &id, const std::string &username,
            const std::string &password, const std::string
&resume,
            const std::string &fname, const std::string &lname,
            const std::string &email, const std::string &phone,
            const std::string &address, const std::string
&linkedIn,
            const std::string &skills, const std::string
&experience);
  void searchJob();
  void submitApplication();
  void receiveJobInterview();
  void receiveJobOfferInfo();
  void addFeedback();
```

```
  void display();
};
```

Candidate.cpp

```
#include "Candidate.h"
#include <iostream>

// Constructor
Candidate::Candidate(const std::string &id, const std::string
&username,
                     const std::string &password, const
std::string &resume,
                     const std::string &fname, const std::string
&lname,
                     const std::string &email, const std::string
&phone,
                     const std::string &address, const
std::string &linkedIn,
                     const std::string &skills, const
std::string &experience)
    : User(username, password), candidate_id(id),
resume(resume),
      first_Name(fname), last_Name(lname), email(email),
phone(phone),
      address(address), linkedIn(linkedIn), skills(skills),
      experience(experience) {}

void Candidate::display()
{
  std::cout << "Candidate ID: " << candidate_id << "\n"
            << "Resume: " << resume << "\n"
            << "First Name: " << first_Name << "\n"
```

```
            << "Last Name: " << last_Name << "\n"
            << "Email: " << email << "\n"
            << "Phone: " << phone << "\n"
            << "Address: " << address << "\n"
            << "LinkedIn: " << linkedIn << "\n"
            << "Skills: " << skills << "\n"
            << "Experience: " << experience << "\n";
}

void Candidate::searchJob() {}

void Candidate::submitApplication() {}

void Candidate::receiveJobInterview() {}

void Candidate::receiveJobOfferInfo() {}

void Candidate::addFeedback() {}
```

ClientCompany.h

```
#include "User.h"
#include <string>

class ClientCompany : public User
{
private:
  std::string client_id;
  std::string company_name;
  std::string industry;
  std::string email;
  std::string contact_no;

public:
  ClientCompany(const std::string &username, const std::string
&password,
                const std::string &clientID, const std::string
&compName,
                const std::string &industry, const std::string
&email,
                const std::string &contact);
  void sendVacancies();
  void sendInterviewDetails();
  void checkApplication();
  void display();
  void recieveApprovedInterviewDetails();
  void addFeedback();
};
```

ClientCompany.cpp

```
#include "ClientCompany.h"
#include <iostream>

ClientCompany::ClientCompany(const std::string &username,
                             const std::string &password,
                             const std::string &clientID,
                             const std::string &compName,
                             const std::string &industry,
                             const std::string &email,
                             const std::string &contact)
    : User(username, password), client_id(clientID),
company_name(compName),
      industry(industry), email(email), contact_no(contact) {}

void ClientCompany::sendVacancies() {}
void ClientCompany::sendInterviewDetails() {}
void ClientCompany::checkApplication() {}
void ClientCompany::addFeedback() {}
void ClientCompany::recieveApprovedInterviewDetails() {}
void ClientCompany::display()
{
  std::cout << "Client ID: " << client_id << "\n"
            << "Company Name: " << company_name << "\n"
            << "Industry: " << industry << "\n"
            << "Email: " << email << "\n"
            << "Contact Number: " << contact_no << "\n";
}
```

Application.h

```
#include <string>

class Application
{
private:
  std::string application_ID;
  std::string job_ID;
  std::string candidate_ID;

public:
  Application(const std::string &appID, const std::string
&jobID,
              const std::string &candID);
  void storeApplicationDetails();
  void sendApplicationClient();
  void applicationStatus();
  void display();
};
```

Application.cpp

```
#include "Application.h"
#include <iostream>
Application::Application(const std::string &appID, const
std::string &jobID,
                         const std::string &candID)
    : application_ID(appID), job_ID(jobID), candidate_ID(candID)
{}

void Application::storeApplicationDetails() {}
void Application::sendApplicationClient() {}
void Application::applicationStatus() {}
void Application::display() {
  std::cout << "Application ID: " << application_ID << "\n"
            << "Job ID: " << job_ID << "\n"
            << "Candidate ID: " << candidate_ID << "\n";
}
```

Interview.h

```
#include <string>

class Interview
{
private:
  std::string interview_id;
  std::string candidate_id;
  std::string job_ID;
  std::string interview_date;
  std::string interview_time;
  bool status;

public:
  Interview(const std::string &interviewID, const std::string
&candID,
            const std::string &jobID, const std::string &date,
            const std::string &time, bool status);
  void scheduleInterview();
  void notifyCandidate();
  void interviewStatus();
  void sendJobOffer();
  void notifyClientCompany();
  void display();
};
```

Interview.cpp

```
#include "Interview.h"
#include <iostream>

Interview::Interview(const std::string &interviewID, const
std::string &candID,
                     const std::string &jobID, const std::string
&date,
                     const std::string &time, bool status)
    : interview_id(interviewID), candidate_id(candID),
job_ID(jobID),
      interview_date(date), interview_time(time), status(status)
{}

void Interview::scheduleInterview() {}

void Interview::notifyCandidate() {}

void Interview::notifyClientCompany() {}
void Interview::interviewStatus() {}
void Interview::sendJobOffer() {}
void Interview::display()
{
  std::cout << "Interview ID: " << interview_id << "\n"
            << "Candidate ID: " << candidate_id << "\n"
            << "Job ID: " << job_ID << "\n"
            << "Interview Date: " << interview_date << "\n"
            << "Interview Time: " << interview_time << "\n"
            << "Status: " << (status ? "Selected" : "Rejected")
<< "\n";
  ;
```

JobPost.h

```
#include <string>

class JobPost
{
private:
  std::string job_ID;
  std::string client_ID;
  std::string job_title;
  std::string job_description;
  std::string requirements;
  std::string location;
  std::string salary;
  bool status;

public:
  JobPost(const std::string &jobID, const std::string &clientID,
          const std::string &jobTitle, const std::string
&jobDesc,
          const std::string &req, const std::string &loc,
          const std::string &sal, bool status);
  void storeJobDetails();
  void notifyAdminApproval();
  void displayApprovedJob();
  bool getStatus() const;
  void setStatus(bool newStatus);
  void display();
};
```

JobPost.cpp

```
#include "JobPost.h"
#include <iostream>

JobPost::JobPost(const std::string &jobID, const std::string
&clientID,
                 const std::string &jobTitle, const std::string
&jobDesc,
                 const std::string &req, const std::string &loc,
                 const std::string &sal, bool status)
    : job_ID(jobID), client_ID(clientID), job_title(jobTitle),
      job_description(jobDesc), requirements(req),
location(loc), salary(sal),
      status(status) {}

void JobPost::storeJobDetails() {}

void JobPost::notifyAdminApproval() {}

void JobPost::displayApprovedJob() {}

bool JobPost::getStatus() const { return status; }

void JobPost::setStatus(bool newStatus) { status = newStatus; }

void JobPost::display() {
  std::cout << "Job ID: " << job_ID << "\n"
            << "Client ID: " << client_ID << "\n"
            << "Job Title: " << job_title << "\n"
            << "Job Description: " << job_description
            << "\n"
               "Requirements: "
            << requirements
```

```
            << "\n"
               "Location: "
            << location << "\n"
            << "Salary: " << salary << "\n"
            << "Status: " << (status ? "Approved" : "Pending")
<< "\n";
}
```

JobOffer.h

```
#include <string>

class JobOffer
{
private:
  std::string jobOffer_id;
  std::string client_id;
  std::string candidate_id;
  std::string offerDetails;

public:
  JobOffer(const std::string &offerID, const std::string
&clientID,
           const std::string &candID, const std::string
&details);
  void createOffer();
  void notifyCandidate();
  void acceptOffer();
  void rejectOffer();
  void display();
};
```

JobOffer.cpp

```
#include "JobOffer.h"
#include <iostream>

JobOffer::JobOffer(const std::string &offerID, const std::string
&clientID,
                   const std::string &candID, const std::string
&details)
    : jobOffer_id(offerID), client_id(clientID),
candidate_id(candID),
      offerDetails(details) {}

void JobOffer::createOffer() {}

void JobOffer::notifyCandidate() {}

void JobOffer::acceptOffer() {}

void JobOffer::rejectOffer() {}
void JobOffer::display() {
  std::cout << "Job Offer ID: " << jobOffer_id << "\n"
            << "Client ID: " << client_id << "\n"
            << "Candidate ID: " << candidate_id << "\n"
            << "Offer Details: " << offerDetails << "\n";
}
```

Feedback.h

```
#include <string>

class Feedback
{
private:
  std::string feedback_id;
  std::string username;
  std::string rating;
  std::string description;

public:
  Feedback(const std::string &feedbackID, const std::string
&username,
           const std::string &rating, const std::string &desc);
  void showFeedback();
  void display();
};
```

Feedback.cpp

```
#include "Feedback.h"
#include <iostream>

Feedback::Feedback(const std::string &feedbackID, const
std::string &username,
                   const std::string &rating, const std::string
&desc)
    : feedback_id(feedbackID), username(username),
rating(rating),
      description(desc) {}

void Feedback::showFeedback() {
  std::cout << "Feedback ID: " << feedback_id << "\n"
            << "Username: " << username << "\n"
            << "Rating: " << rating << "\n"
            << "Description: " << description << "\n";
}
```

Report.h

```
#include <string>

class Report
{
private:
  std::string application_ID;
  std::string job_ID;
  std::string candidate_ID;

public:
  Report(const std::string &appID, const std::string &jobID,
         const std::string &candID);
  void storeApplicationDetails();
  void sendApplicationClient();
  void applicationStatus();
  void display();
};
```

Report.cpp

```
#include "Report.h"
#include <iostream>

Report::Report(const std::string &appID, const std::string
&jobID,
               const std::string &candID)
    : application_ID(appID), job_ID(jobID), candidate_ID(candID)
{}

void Report::storeApplicationDetails() {}

void Report::sendApplicationClient() {}

void Report::applicationStatus() {}
void Report::display() {
  std::cout << "Application ID: " << application_ID << "\n"
            << "Job ID: " << job_ID << "\n"
            << "Candidate ID: " << candidate_ID << "\n";
}
```

6. Individual Contribution

Registration No	Name	Contributed Classes
IT23366336	A.M.N.S. WEERARATHNE	• Admin class • Report Class • Main Class
IT23363670	E.D.O. SAMARAKOON	• Job Posting Class, • Client Class, • Application Class • Main Class
IT23363052	K.H.S. DAMSILUNI	• User class • Candidate class • feedback class • Main Class
IT23362994	H.M.C.M.B. HEARATH	• Interview class • Job Offer class • Main Class