Topic : Online Examination System for Employees

Group no : MLB_01.02_05

Campus : Malabe

Submission Date:

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**
IT21196638	Perera G.M.T	0773319179
IT21192500	Kandepola H.M.H.B	0762758021
IT21196324	Dilshan N.M	0765269557
IT21196010	Madushanka W.A.T	0773158596
IT21197796	Wasana K.H.I.M	0703636127

Exercise 1 – Requirements

- Employee can register to the system providing details such as Name, Nic, email, and contact number.
- Registered Employee can login to the website by providing valid login details by entering username and password.
- Registered Employee can search and enrol exam by paying exam fee
- The employees specify a payment method (credit card, debit card, PayPal) for each exam registration.
- User should be able to cancel an exam.
- Employee should be able to attempt an exam.
- Employees can view exam results and get certificates.
- Employees can view exam resources and attempt practice quizzes.
- Users should be able to provide feedback using employee email.
- System admins can add, remove, update employee details
- System admins can add, remove, update examiners and details
- Admins can add, remove, update resources
- Admins can review feedback update the site.
- Examiner manages an Exam and examiner should be able to upload questions and answers. An Exam has multiple questions, and each question has one or more answers.
- Examiners can add, remove, and update Exams and exam information.
- Examiners should login to the system using examiner ID and password
- Examiners can generate and publish results.
- Examiners can generate certificates for those who have passed the exam.
- Examiners can add Instructions, timetables.
- System should generate and store a Candidate id after confirming user registration.

Noun Verb analysis

- Employee can register to the website providing details such as Name, Nic, email, contact number.
- Registered Employee can login to the website by providing valid login details by entering username and password.
- Registered Employee can search and view exam information.
- Registered Employee can enrol to an exam by providing employee email, exam date/time and making exam payment.
- The employees specify a payment method (credit card, debit card, paypal) for each exam registration.
- Employee should be able to cancel an exam.
- Employee should be able to attempt an exam.
- Employees can view exam results and get certificates.
- Employees can view exam resources and attempt practice quizzes.
- User should be able to provide feedback using employee email.
- System admins can add, remove, update employee details
- System admins can add, remove, update examiners and details
- Admins can add, remove, update resources
- Admins can review feedback update the website.
- Examiners can add, remove, and update Exams.
- Examiners should login to the system using examiner ID and password
- Examiners can generate and publish results.
- Examiners can generate certificates for those who have passed the exam.
- Examiners can add Instructions, timetables.
- System should generate and store a Candidate id after confirming user registration.

Nouns

1. Employee
2. website
3. name
4. NIC
5. Email
6. Contact number
7. Username
8. password
9. Employee email
10. Exam date/time
11. Exam payment
12. exam
13. Exam results
14. certificates
15. Exam resources
16. Practice quizzes
17. feedback
18. Employee mail
19. System admins
20. employee details
21. examiners
22. details
23. Resources
24. website
25. instructions
26. timetables
27. Candidate ID

Classes

1. Employee
2. Admins
3. Examiner
4. Exams
5. Results
6. Payment
7. Feedback
1. Resources

Exercise 2 - CRC Cards

Class Name: Employee	
Responsibilities	**Collaborations**
Search exam	Exam
Enrol Exam	Exam
Pay exam fee	Payment
View results	Result
Access resources	resource

Class Name: Administrator	
Responsibilities	**Collaborations**
Add and remove resource	
Add and remove users	
Review feedback	Feedback
Add, remove and modify sample quiz	

Class Name: Examiner	
Responsibilities	**Collaborations**
Add and remove papers	
Add and remove marking scheme	
Add exam details	
Published results	Result

Class Name: Enrolment	
Responsibilities	**Collaborations**
Store enrolment details	Exam/employee
Add Marks	
Update enrolment details	

Class Name: Exam	
Responsibilities	**Collaborations**
Store exam papers	
Store marketing Scheme	

Class Name: Question	
Responsibilities	**Collaborations**
Add question	
Store question	

Class Name: Answer	
Responsibilities	**Collaborations**
Add answers	
Store answers	

Class Name: Payment	
Responsibilities	**Collaborations**
Store payment details	Employee
Display payment details	
Generate transaction ID	
Validate	
Store transaction details	

Class Name: Resource	
Responsibilities	**Collaborations**
Provide sample quiz	Administrator
Store exam resource	Administrator

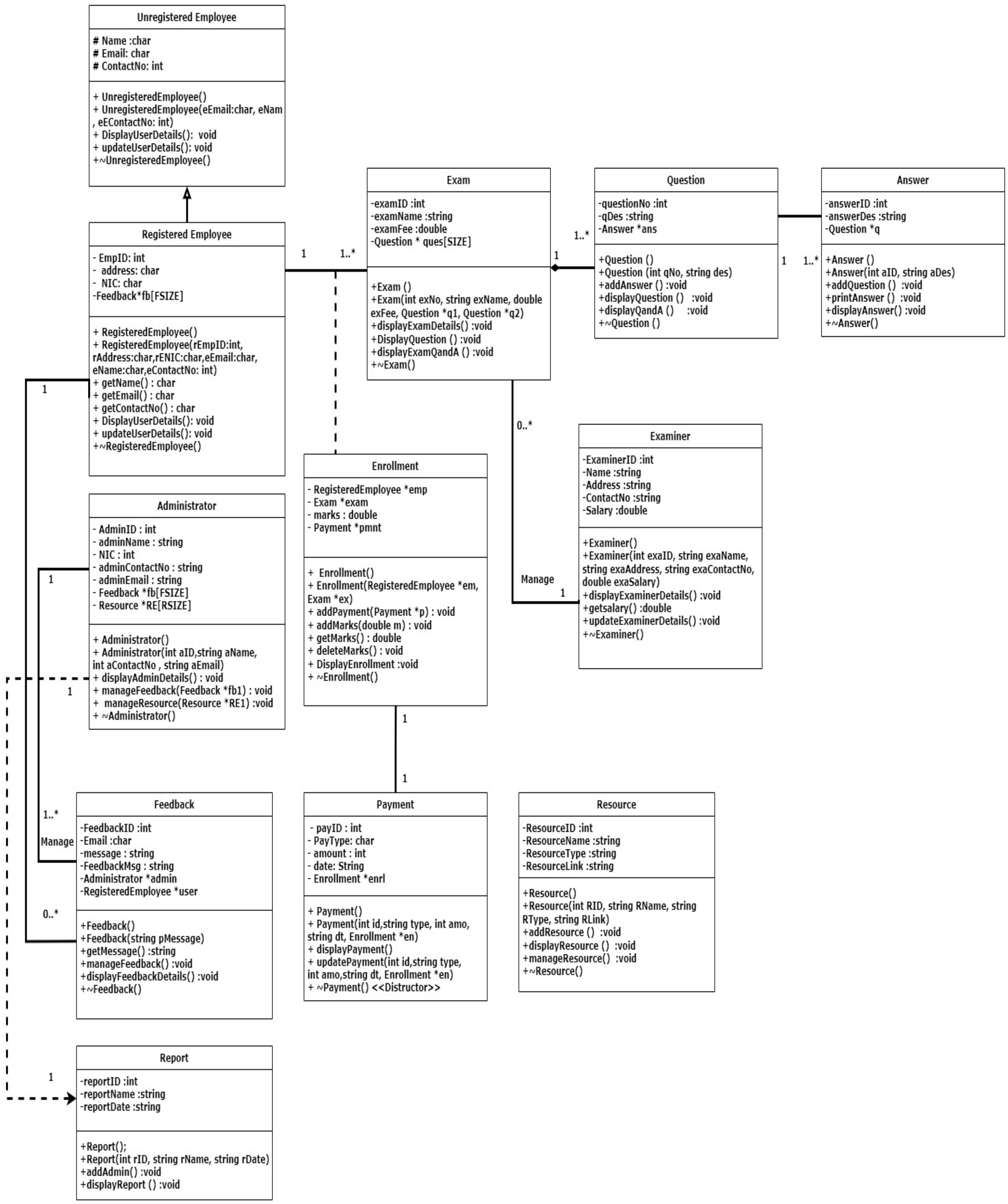
Class Name: Feedback	
Responsibilities	**Collaborations**
Store feedback details	
Display feedback	

Class Name: Report	
Responsibilities	**Collaborations**
Store Report details	Administrator
Update report details	

CONTRIBUTIONS

IT number	IT21196638	IT21192500	IT21196324	IT21196010	IT21197796
UML contribution	Answer Exam	Feedback Examiner	Administrator Enrolment report	Payment Resources	Registered employee Unregistered employee Question

Exercise 3 – Class Diagram



Exercise 4 – Coding for the classes in Class diagram

```
#include <iostream>
#include <cstring>
using namespace std;
#define FSIZE 3
#define ESIZE 3
#define RSIZE 5
#define SIZE 30
class Feedback;
class Payment;
class Question;
class Answer;
class Resource;
class Exam;

// UnregisteredEmployee.h
class UnregisteredEmployee
 {
    protected:
        char name[30];
        char email[30];
        int contactNO;

    public:
        UnregisteredEmployee();
        UnregisteredEmployee(const char eName[], const char
eEmail[], int eContactNo);
        void displayUserDetails();
        void updateUserDetails();
        ~UnregisteredEmployee();
  };
```

```
//RegisteredEmployee.h
class RegisteredEmployee : public UnregisteredEmployee
 {
    protected:
        int empID;
        string address;
        char NIC[10];
        Feedback*fb[FSIZE];
    public:
        RegisteredEmployee();
        RegisteredEmployee(const char eName[],int
rEmpID,const char rENIC[], const char eEmail[], int
eContactNo, string rAddress);
        char getName();
        char getEmail();
        char getContactNo();
        void displayUserDetails();
        void updateUserDetails();
        ~RegisteredEmployee();
 };

//Administrator.h
class Administrator
 {
    private:
        int adminID;
        string adminName;
        int adminContactNo;
        string adminEmail;
        Feedback *fb[FSIZE];
        Resource *RE[RSIZE];

    public:
        Administrator();
        Administrator(int aID,string aName,int aContactNo ,
string aEmail);
        void displayAdminDetails();
        void manageFeedback(Feedback *fb1);
        void manageResource(Resource *RE1);
        ~Administrator();
 };

//Examiner.h
class Examiner
 {
    private:
        int ExaminerID;
        string Name;
        string Address;
        string ContactNo;
        double Salary;

    public:
        Examiner();
        Examiner(int exaID, string exaName, string
exaAddress, string exaContactNo, double exaSalary);
        void displayExaminerDetails();
        double getsalary();
        void updateExaminerDetails();
        ~Examiner();
 };

//Enrollment.h
class Enrollment
{
    private:
        RegisteredEmployee *emp;
        Exam *exam;
        double marks;
        Payment *pmnt;


    public:
        Enrollment();
        Enrollment(RegisteredEmployee *em, Exam *ex);
        void addPayment(Payment *p);
        void addMarks(double m);
        double getMarks();
        void deleteMarks();
        void displayEnrollment();
        ~Enrollment();
};

 //Payment.h
class Payment
 {
    private:
        int payID;
        string payType;
        int amount;
        string date;
        Enrollment *enrl;

    public :
        Payment();
        Payment(int id,string type, int amo,string dt,
Enrollment *en);
        void displayPayment();
        void updatePayment();
        ~Payment();
 };

 //Question.h
class Question
{
    private:
        int questionNo;
        string qDes;
        Answer *ans;


    public:
        Question ();
        Question (int qNo, string des);
        void addAnswer(Answer *a);
        void displayQuestion();
        void displayQandA();
        ~Question ();
};


 //Exam.h
class Exam
 {
    private:
        int examID;
        string examName;
        double examFee;
        Question * ques[SIZE];

    public:
        Exam ();
        Exam (int exNo, string exName, double exFee,
Question *q1, Question *q2);
        void displayExamDetails();
        void DisplayQuestion ();
        void displayExamQandA();
        ~Exam();
};

 //Answer.h
class Answer
{
    private:
        int answerID;
        string answerDes;
        Question *q;

    public:
        Answer();
        Answer( int aID, string aDes);
        void addQuestion(Question *q1);
        void printAnswer();
        void displayAnswer();
        ~Answer();
};

//Resource.h
class Resource
 {
    protected:
        int  ResourceID;
        string ResourceName;
        string ResourceType;
        string ResourceLink;

    public:
        Resource();
        Resource(int RID, string RName, string RType, string
RLink);
        void addResource();
        void displayResource();
        void manageResource(Resource *RE1);
        ~Resource();
 };

//Report.h
class Report
 {
    private:
       int reportID;
       string reportName;
       string reportDate;

    public:
        Report();
        Report(int rID, string rName, string rDate);
        void addAdmin(Administrator *A1);
        void displayReport();
        ~Report();
 };

 //Feedback.h
class Feedback
 {
    private:
        int FeedbackID;
        char Email;

        string message;
        string FeedbackMsg;
        Administrator *admin;
        RegisteredEmployee *user;

    public:
        Feedback();
        Feedback(string pMessage);
        string getMessage();
        void manageFeedback(Administrator *a);
        void displayFeedbackDetails();
 };


// methods implemetation

//UnregisteredEmployee.cpp
UnregisteredEmployee::UnregisteredEmployee()
{
}

UnregisteredEmployee::UnregisteredEmployee(const char
eName[], const char eEmail[], int eContactNo)
{
    strcpy(name, eName);
    strcpy(email, eEmail);
    contactNO = eContactNo;
}
```

```
void UnregisteredEmployee::displayUserDetails()
{
    cout << endl;
    cout << "~~~~~~~~~~~~UnregisteredEmployee~~~~~~~~~~~~"
<< endl;
    cout<< "Employee Name: " << name <<endl;
    cout<< "Employee Email: " << email <<endl;
    cout<< "Employee Contact No: " << contactNO <<endl;
}

void UnregisteredEmployee::updateUserDetails()
{
}

UnregisteredEmployee::~UnregisteredEmployee()
{
}

//RegisteredEmployee.cpp
RegisteredEmployee::RegisteredEmployee()
{
}

RegisteredEmployee::RegisteredEmployee(const char
eName[],int rEmpID,const char rENIC[], const char eEmail[],
int eContactNo, string rAddress)
{
    strcpy(name, eName);
    empID = rEmpID;
    address =rAddress;
    strcpy(NIC, rENIC);
}

char RegisteredEmployee::getName()
{
}

char RegisteredEmployee::getEmail()
```

```
{
}

char RegisteredEmployee::getContactNo()
{
}

void RegisteredEmployee::displayUserDetails()
{   cout << endl;
    cout << "~~~~~~~~~~~~RegisteredEmployee~~~~~~~~~~~" <<
endl;
    cout<< "Employee Name: " << name <<endl;
    cout<< "Employee ID: " << empID <<endl;
    cout<< "Employee Email: " << email <<endl;
    cout<< "Employee NIC: " << NIC <<endl;
    cout<< "Employee Contact No: " << contactNO <<endl;
    cout<< "Employee Address: " << address <<endl;
}

void RegisteredEmployee::updateUserDetails()
{
}

RegisteredEmployee::~RegisteredEmployee()
{
}

//Administrator.cpp
Administrator::Administrator(int aID,string aName,int
aContactNo , string aEmail)
{
    adminID = aID;
    adminName = aName;
    adminContactNo = aContactNo;
    adminEmail = aEmail;
}
void Administrator::displayAdminDetails()
{
```

```
    cout << endl;
    cout << "~~~~~~~~~~~~Administrator~~~~~~~~~~~~" << endl;
    cout << " Admin ID =" << adminID << endl;
    cout << " Admin Name =" << adminName << endl;
    cout << " Admin Contact Number = " << adminContactNo <<
endl;
    cout << " Admin Email = " << adminEmail << endl;
}
void Administrator::manageFeedback(Feedback *fb1)
{
}

void Administrator::manageResource(Resource *RE1)
{
}

//Examiner.cpp
Examiner::Examiner()
 {
    ExaminerID=0;
    Name="";
    Address="";
    ContactNo="";
    Salary=0;
 }

Examiner::Examiner(int exaID, string exaName, string
exaAddress, string exaContactNo, double exaSalary)
 {
    ExaminerID=exaID;
    Name=exaName;
    Address=exaAddress;
    ContactNo=exaContactNo;
    Salary=exaSalary;
 }

void Examiner::displayExaminerDetails()
 {
```

```
    cout << endl;
    cout << "~~~~~~~~~~~~Examiner~~~~~~~~~~~~" << endl; cout
<< "ID : " << ExaminerID << endl;
    cout << "Name : " << Name << endl; cout << "Address : "
<< Address << endl;
    cout << "Contact NO : " << ContactNo << endl; cout <<
"Salray : " << Salary << endl;
    cout << endl;
 }

double Examiner::getsalary()
 {
    return Salary;
 }

void Examiner::updateExaminerDetails()
 {
    cout << "Enter new contact number : ";
    cin >> ContactNo;
    cout << "Enter new salary : ";
    cin >> Salary;
 }

Examiner::~Examiner()
 {
    cout << "Examiner object " << ExaminerID << " Deleted "
<< endl;;
 }


 //Enrollment.cpp
 Enrollment::Enrollment()
{
    marks = 0;
}

Enrollment::Enrollment(RegisteredEmployee *em, Exam *ex)
{
```

```
    emp = em;
    exam = ex;
    marks = 0;
}

void Enrollment::addPayment(Payment *p)
{
    pmnt = p;
}

void Enrollment::addMarks(double m)
{
    if ((m < 100) && ( m > 0)){
        marks = m;
    }
}

double Enrollment::getMarks(){
    return marks;
}

void Enrollment::deleteMarks(){
    marks = 0;
}

void Enrollment::displayEnrollment()
{
}

Enrollment::~Enrollment()
{
}

//Payment.cpp
Payment::Payment()
{
        payID = 0;
        payType = "";
        amount = 0;
        date = "";
}

Payment::Payment(int id,string type, int amo,string dt,
Enrollment *en)
{
    payID = id;
    payType=type;
    amount = amo;
    date=dt;
    enrl = en;
};

void Payment::displayPayment()
{
        cout << endl;
        cout << "~~~~~~~~~~~~Payment~~~~~~~~~~~" << endl;
        cout << " Payment ID =" << payID << endl;
        cout << " Payment Type =" << payType << endl;
        cout << " Payment Amount = " << amount << endl;
        cout << " Payment Date = " << date << endl;
}

void Payment::updatePayment()
{
}

//Question.cpp
 Question::Question (int qNo, string des)
{
    questionNo = qNo;
    qDes = des;
};

void Question::addAnswer(Answer *a)
{
    ans = a;
```

```
}

void Question::displayQuestion()
{
    cout << endl;
    cout << "~~~~~~~~~~~~Question~~~~~~~~~~~~";
    cout<<endl<<"Question No : "<< questionNo << endl;
cout<<"Question : "<<qDes<<endl;
}

void Question::displayQandA()
{
    cout<< questionNo << ". "<<qDes<<endl;
    ans->printAnswer();
    cout<<endl;
}

Question::~Question()
{
cout<<"Deleting Question "<<questionNo<<endl;
}

//Exam.cpp
Exam::Exam ()
{
    examID = 0;
    examName = "";
    examFee = 0;
    ques [0] = new Question(0, "");
    ques [1] = new Question(0, "");
}

Exam::Exam (int exNo, string exName, double exFee, Question
*q1, Question *q2)
{
    examID = exNo;
    examName = exName;
    examFee = exFee;
```

```
    ques [0] = q1;
    ques [1] = q2;
}

void Exam::displayExamDetails()
{
    cout << endl;
    cout<< endl;
    cout << "~~~~~~~~~~~~Exam Details~~~~~~~~~~~~" << endl;
    cout<<"Exam ID : "<<examID<<endl<<"Exam Name :
"<<examName<<endl<<"Exam Fee : "<<examFee<<endl;
}

void Exam::DisplayQuestion ()
{
    for (int i=0; i<SIZE; i++){
        ques[i]->displayQuestion();
    }
}
void Exam::displayExamQandA()
{
    for (int i=0; i<SIZE; i++){
        ques[i]->displayQandA();
        cout<<endl;
    }
}

Exam::~Exam()
{
    cout<< "Deleting Exam "<<endl;
    for (int i=0; i<SIZE; i++){
        delete ques[i];
    }
}

//Answer.cpp
Answer::Answer()
{
```

```
    answerID = 0;
    answerDes = "";
}
Answer::Answer( int aID, string aDes)
{
    answerID = aID;
    answerDes = aDes;
}
void Answer::addQuestion(Question *q1)
{
    q = q1;
}
void Answer::printAnswer()
{
cout<<"Answer : "<<answerDes;
}
void Answer::displayAnswer()
{
cout << endl;
cout << "~~~~~~~~~~~~Answer~~~~~~~~~~~~" ;
cout<<endl<<"Answer No: "<<answerID<< endl; cout<<"Answer :
"<<answerDes;
}


//Resource.cpp
 Resource::Resource()
 {
    ResourceID=0;
    ResourceName="";
    ResourceType="";
    ResourceLink="";
 }

 Resource::Resource(int RID, string RName, string RType,
string RLink)
 {
    ResourceID=RID;
    ResourceName=RName;
```

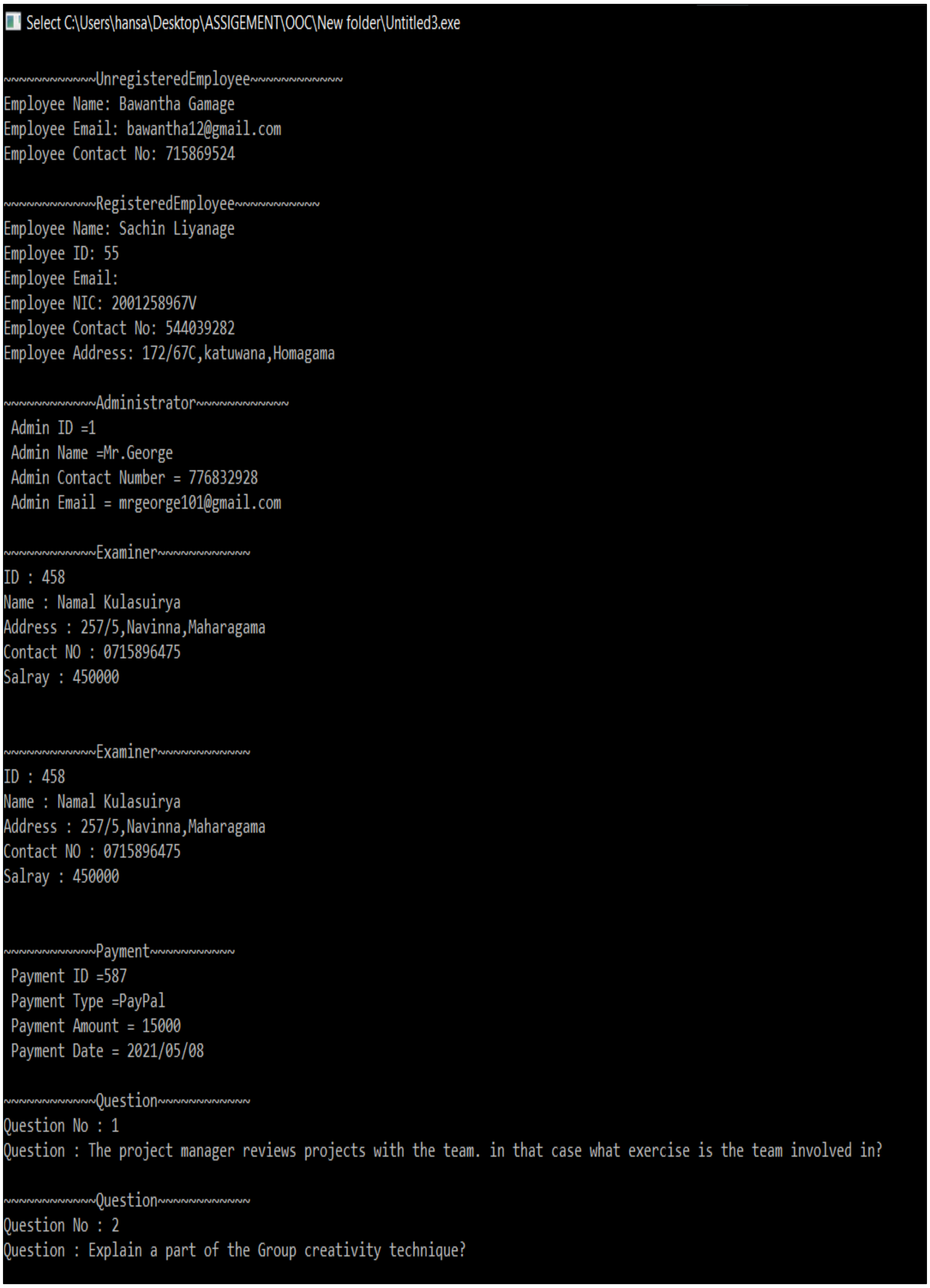
```
    ResourceType=RType;
    ResourceLink=RLink;
 }

void Resource::displayResource()
{
    cout << endl;
    cout << "~~~~~~~~~~~~Resource~~~~~~~~~~~~~" << endl;
    cout << " Resource ID =" << ResourceID << endl;
    cout << " Resource Name =" << ResourceName << endl;
    cout << " Resource Type = " << ResourceType << endl;
    cout << " Resource Link = " << ResourceLink << endl;
}

void manageResource(Resource *RE1)
{
}

void addResource()
{
}


//Feedback.cpp
Feedback::Feedback()
{
message = "";
}

Feedback::Feedback(string pMessage)
{
message = pMessage;
}

string Feedback::getMessage()
{
return FeedbackMsg;
}
```

```
void Feedback::manageFeedback(Administrator *a )
{
}

void Feedback::displayFeedbackDetails()
{
    cout << endl;
    cout << "~~~~~~~~~~~~Feedback's Details~~~~~~~~~~~" <<
endl;; cout << "Message : " << message ;
    cout << endl;
}

//Report.cpp
Report::Report(int rID, string rName, string rDate)
{
    reportID = rID;
    reportName = rName;
    reportDate = rDate;
}

void Report::addAdmin(Administrator *A1)
{
     A1->displayAdminDetails();
}

void Report::displayReport()
{
    cout << endl;
    cout << "~~~~~~~~~~~~Report~~~~~~~~~~~~" << endl;
    cout << " Report ID =" << reportID << endl;
    cout << " Report Name =" << reportName << endl;
    cout << " Report Date = " << reportDate << endl;
}
```

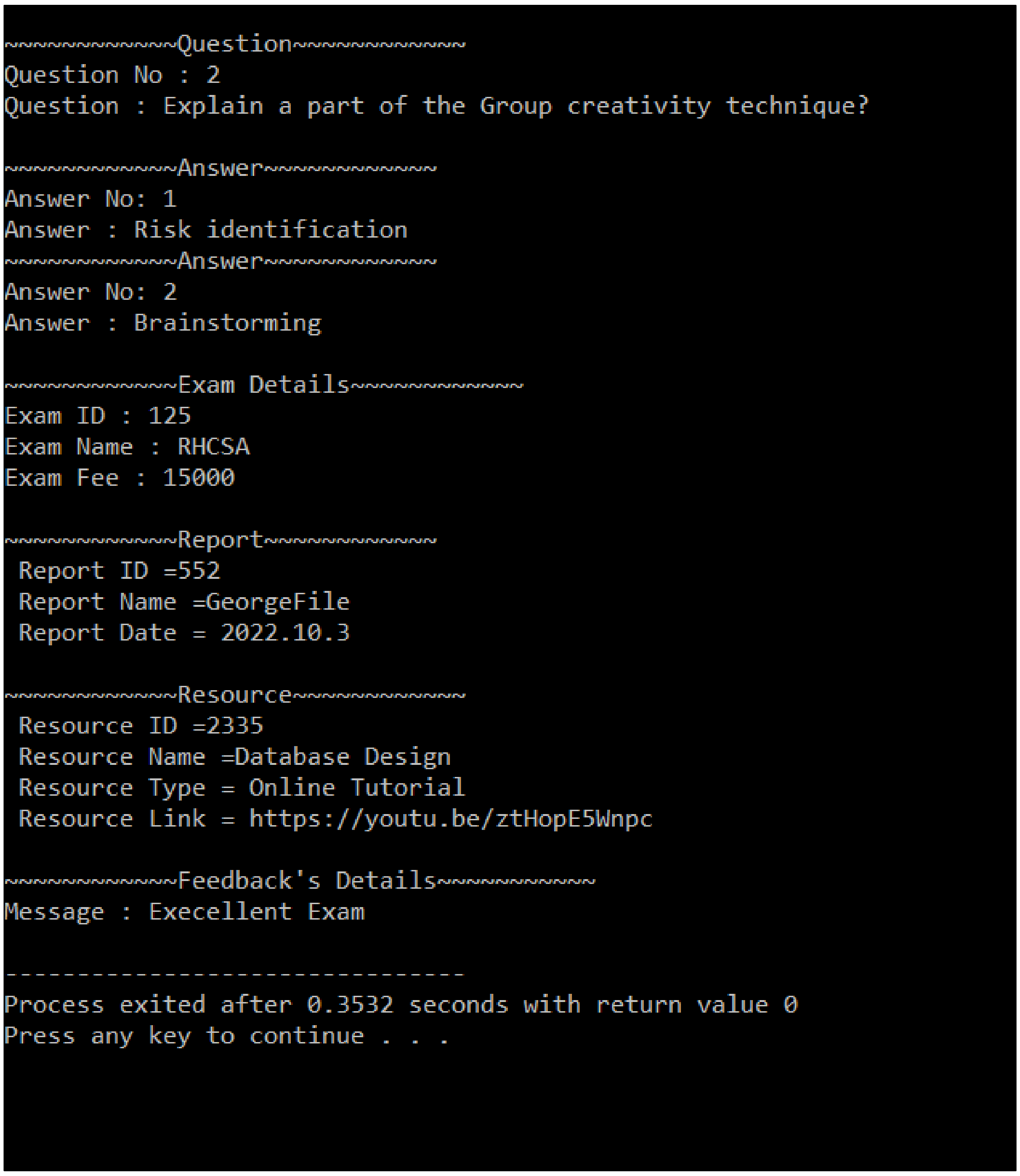
```cpp
int main(){

    Question *ques1, *ques2;
    Answer *answ1, *answ2;
    Exam *ex1;

    UnregisteredEmployee *UnE1=new
UnregisteredEmployee("Bawantha Gamage",
"bawantha12@gmail.com", 715869524);
    UnE1->displayUserDetails();

    RegisteredEmployee *RegE1=new RegisteredEmployee("Sachin
Liyanage ",55,"2001258967V", "sachin04@gmail.com",
7158963245, "172/67C,katuwana,Homagama");
    RegE1->displayUserDetails();

    Administrator *A1 = new Administrator(
0001,"Mr.George",776832928,"mrgeorge101@gmail.com");
    Report *R1 = new Report(552, "GeorgeFile","2022.10.3");
    R1->addAdmin(A1);

    ex1 = new Exam(125, "RHCSA", 15000, ques1, ques2);

    Examiner *exa1 =new Examiner(458,"Namal
Kulasuirya","257/5,Navinna,Maharagama","0715896475",450000);
    exa1->displayExaminerDetails();
    exa1->displayExaminerDetails();

    Enrollment *En1=new Enrollment(RegE1,ex1);
    Payment *p1=new
Payment(587,"PayPal",15000,"2021/05/08",En1);
    p1->displayPayment();

    ques1 = new Question(1, "The project manager reviews
projects with the team. in that case what exercise is the
team involved in?");
    ques2 = new Question(2, "Explain a part of the Group
creativity technique?");
```

```
    ques1->addAnswer(answ1);
    ques2->addAnswer(answ2);
    ques1->displayQuestion();
    ques2->displayQuestion();


    answ1 = new Answer(1, "Risk identification");
    answ2 = new Answer(2, "Brainstorming");
    answ1->addQuestion(ques1);
    answ2->addQuestion(ques2);
    answ1->displayAnswer();
    answ2->displayAnswer();

    ex1->displayExamDetails();
    R1->displayReport();

    Resource *RE1=new Resource(2335,"Database Design",
"Online Tutorial", "https://youtu.be/ztHopE5Wnpc");
    RE1->displayResource();


    Feedback*fb1=new Feedback("Execellent Exam");
    fb1->displayFeedbackDetails();
}
```

Exercise 4 – Output of coding



```
Select C:\Users\hansa\Desktop\ASSIGEMENT\OOC\New folder\Untitled3.exe

~~~~~~~~~~~~UnregisteredEmployee~~~~~~~~~~~~
Employee Name: Bawantha Gamage
Employee Email: bawantha12@gmail.com
Employee Contact No: 715869524

~~~~~~~~~~~~RegisteredEmployee~~~~~~~~~~~
Employee Name: Sachin Liyanage
Employee ID: 55
Employee Email:
Employee NIC: 2001258967V
Employee Contact No: 544039282
Employee Address: 172/67C,katuwana,Homagama

~~~~~~~~~~~~Administrator~~~~~~~~~~~~
 Admin ID =1
 Admin Name =Mr.George
 Admin Contact Number = 776832928
 Admin Email = mrgeorge101@gmail.com

~~~~~~~~~~~~Examiner~~~~~~~~~~~~
ID : 458
Name : Namal Kulasuirya
Address : 257/5,Navinna,Maharagama
Contact NO : 0715896475
Salray : 450000


~~~~~~~~~~~~Examiner~~~~~~~~~~~~
ID : 458
Name : Namal Kulasuirya
Address : 257/5,Navinna,Maharagama
Contact NO : 0715896475
Salray : 450000


~~~~~~~~~~~~Payment~~~~~~~~~~~
 Payment ID =587
 Payment Type =PayPal
 Payment Amount = 15000
 Payment Date = 2021/05/08

~~~~~~~~~~~~Question~~~~~~~~~~~~
Question No : 1
Question : The project manager reviews projects with the team. in that case what exercise is the team involved in?

~~~~~~~~~~~~Question~~~~~~~~~~~~
Question No : 2
Question : Explain a part of the Group creativity technique?
```



```
~~~~~~~~~~~~Question~~~~~~~~~~~~
Question No : 2
Question : Explain a part of the Group creativity technique?

~~~~~~~~~~~~Answer~~~~~~~~~~~~
Answer No: 1
Answer : Risk identification
~~~~~~~~~~~~Answer~~~~~~~~~~~~
Answer No: 2
Answer : Brainstorming

~~~~~~~~~~~~Exam Details~~~~~~~~~~~~
Exam ID : 125
Exam Name : RHCSA
Exam Fee : 15000

~~~~~~~~~~~~Report~~~~~~~~~~~~
 Report ID =552
 Report Name =GeorgeFile
 Report Date = 2022.10.3

~~~~~~~~~~~~Resource~~~~~~~~~~~~
 Resource ID =2335
 Resource Name =Database Design
 Resource Type = Online Tutorial
 Resource Link = https://youtu.be/ztHopE5Wnpc

~~~~~~~~~~~~Feedback's Details~~~~~~~~~~~~
Message : Execellent Exam

--------------------------------
Process exited after 0.3532 seconds with return value 0
Press any key to continue . . .
```