Group No.	MLB_03.02_01
Topic	Online Recipe Management system
Campus	Malabe Campus
Submission Date	19/05/2022

We declare that this is our 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 Number	**Student Name**	**Contact Number**
1	IT21561566	S.R.N.M. Fernando	077 033 3510
2	IT21786464	Nisal V.P.T.	076 565 7680
3	IT21247804	Baddewithana P.	070 637 0573
4	IT21469046	A.M.N.H. Adhikari	077 940 9038
5	IT21559136	Wijethunge W.L.D.N.N.	076 805 5942

Ex-1

System Requirements

1. Genius Recipe online management system provides various kinds of unique recipes.
2. Every recipe has three main parts including paid version (with all steps and pro tips).
3. Both registered & unregistered customers can view any recipe on our website as he/she wishes.
4. The ingredients can be calculated according to the number of people needed to serve.
5. Registered & unregistered customers can read our nutritionist and chef's articles and can find the best recipes and previous pro tips via our web.
6. Registered customers can buy our unique recipes with pro tips.
7. A registered customer can choose a payment method (Credit card, Debit card, PayPal) for each order.
8. Product managers approve the payment request is valid.
9. After successful payment, registered customers receive a recipe and payment Successful message via email.
10. Editor and Product manager can add new Recipes, delete recipes, and manage recipes.
11. Manager can generate income & financial reports.
12. Registered customers give feedbacks
13. Registered customers can check order details.
14. System Administrator (Editor, Product Manager) maintain and updates the database of the system.
15. Furthermore, registered customers can contact our staff members to solve their issues.

Noun Verb Analysis

Nouns – Red Color

Verbs – Underlined with Blue

1. Genius Recipe's online management system provides various kinds of unique recipes.
2. Every recipe has three main parts including paid version (with all steps and pro tips).
3. Both registered & unregistered customers can view any recipe on our website as he/she wishes.
4. The ingredients can be calculated according to the number of people needed to serve.
5. Registered & unregistered customers can read our nutritionist and chef's articles and can find the best recipes and previous pro tips via our web.
6. Registered customers can buy our unique recipes with pro tips.
7. A registered customer can choose a payment method (Credit card, Debit card, PayPal) for each order.
8. Product managers approve the payment request is valid.
9. After successful payment, registered customers receive a recipe and payment Successful message via email.
10. Editor and Product manager can add new Recipes, delete recipes, and manage recipes.
11. Manager can generate income & financial reports.
12. Registered customers give feedbacks.
13. Registered customers can check order details.
14. System Administrator (Editor, Product Manager) maintain and updates the database of the system.
15. Furthermore, registered customers can contact our staff members to solve their issues

Identified Nouns

Genius Recipe Online Management - Out of scope

Recipes - Class

Ingredients - attribute

People - Redundant

Nutritionists - Attribute

Chef’s articles - Out of scope

Registered customer – Class

Un-registered customer - Out of scope

Tips - Out of scope

Credit card, Debit card, PayPal - Attributes

Order - Class

Product managers -Attribute

Payment - Class

Message - Out of scope

Email - Attribute

Manager - Attribute

Reports – Out of scope

Feedback – Class

System Administrator - Attribute

Database – Out of scope

System - Out of scope

Staff members – Class

Identified Classes

User

Registered customer

Staff members

Recipes

Order

Payment

Feedback

CRC Cards

Recipes	
Responsibilities	Collaborations
Keep Recipe Details	
Update Recipes	Staff members
Add Recipes	Staff members

Registered customer	
Responsibilities	Collaborations
Login to Website	
Keep customer Details	
Make payment	Payment
Search recipes	Recipes
Purchase recipes	Payment, Recipe, Order
Give feedbacks	Feedback
Contact staff member	Staff Members

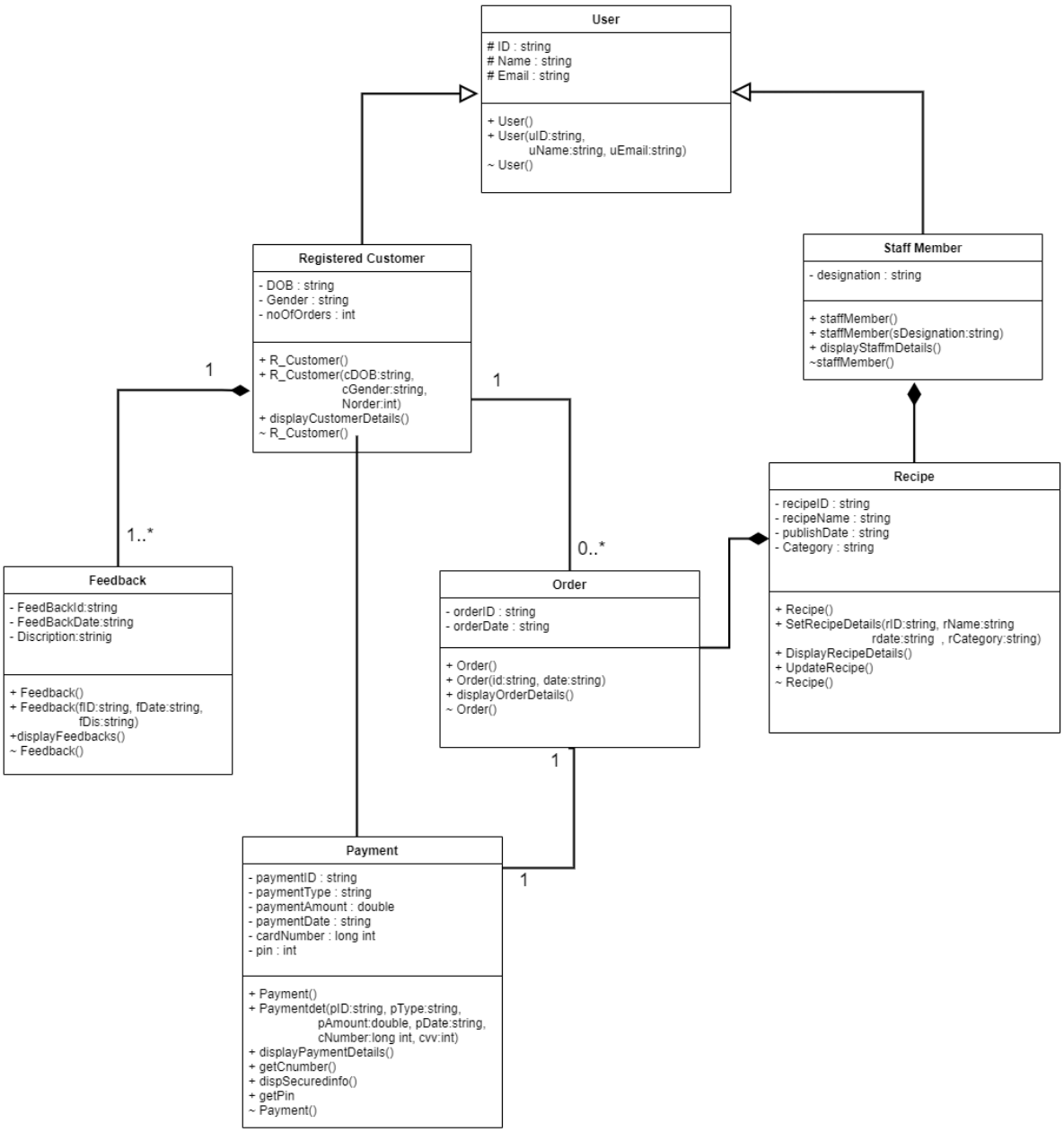
Order	
Responsibilities	Collaborations
Save order details	
Receive orders	Payment
Cancel orders	Payment
Update order details	Staff members
View order details	

Payment	
Responsibilities	Collaborations
Make Payment	
Add payment details	Order, Registered customer
Keep Payment details	
Validate Payment	
View Payment Details	

Staff Members	
Responsibilities	Collaborations
Add recipes	Recipe
Update Recipes	Recipe
View Feedbacks	Feedback

Feedback	
Responsibilities	Collaborations
Keep Feedbacks	
Add Feedbacks	Registered customer

UML Diagram


User
ID : string
Name : string
Email : string
+ User()
+ User(uID:string, uName:string, uEmail:string)
~ User()
Registered Customer
- DOB : string
- Gender : string
- noOfOrders : int
+ R_Customer()
+ R_Customer(cDOB:string, cGender:string, Norder:int)
+ displayCustomerDetails()
~ R_Customer()
Staff Member
- designation : string
+ staffMember()
+ staffMember(sDesignation:string)
+ displayStaffmDetails()
~staffMember()
Recipe
- recipeID : string
- recipeName : string
- publishDate : string
- Category : string
+ Recipe()
+ SetRecipeDetails(rID:string, rName:string rdate:string , rCategory:string)
+ DisplayRecipeDetails()
+ UpdateRecipe()
~ Recipe()
Feedback
- FeedBackId:string
- FeedBackDate:string
- Discription:strinig
+ Feedback()
+ Feedback(fID:string, fDate:string, fDis:string)
+displayFeedbacks()
~ Feedback()
Order
- orderID : string
- orderDate : string
+ Order()
+ Order(id:string, date:string)
+ displayOrderDetails()
~ Order()
Payment
- paymentID : string
- paymentType : string
- paymentAmount : double
- paymentDate : string
- cardNumber : long int
- pin : int
+ Payment()
+ Paymentdet(pID:string, pType:string, pAmount:double, pDate:string, cNumber:long int, cvv:int)
+ displayPaymentDetails()
+ getCnumber()
+ dispSecuredinfo()
+ getPin
~ Payment()
1
1
1..*
0..*
1
1

Codes

User.h

```
#pragma once
#include <iostream>
#include<string>

using namespace std;

//class User
class user
{
protected:
  string ID;
  string Name;
  string Email;

public:
  user();
  user(string uID, string uName,string uEmail);
  ~user();

};
```

staffMember.h

```
#pragma once
#include"user.h"
#include <iostream>
#include<string>

//class staffMember
class staffMember : public user    //inheritance relationship
{
private:
  string Designation;
  Recipe* staffM;
public:
  staffMember();
  staffMember(string cName, string cID, string cEmail,string sDesignation);
  void displaystaffmDetails();
  ~staffMember();
};
```

R_Customer.h

```
#pragma once
#include"user.h"
#include <iostream>
#include<string>

//class R_Customer
class R_Customer : public user    //inheritance relationship
{
private:
  string DOB;
  string Gender;
  int noOfOrders;
public:
  R_Customer();
  R_Customer(string cName, string cID, string cEmail, string cDOB, string cGender
, int Noder);
  void displayCustomerDetails();
  ~R_Customer();
};
```

recipe.h

```
#pragma once
#include<iostream>
#include<string>
#include<cstring>
using namespace std;
#include"staffMember.h"

//class recipe
class Recipe {
private:
    char recipeId[10];
    char recipeName[50];
    char publishDate[20];
    char category[20];
public:
    Recipe();
    Recipe(const char rId[], const char rName[], const char rdate[], const char
rCategory[]);
    void DisplayRecipe();
    void UpdateRecipe();
    ~Recipe();
};
```

order.h

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

//class Order
class Order
{
private:
    string OrderID;
    string OrderDate;
    R_Customer *cus;
public:
    Order();
    Order(string id, string date, R_Customer *CUS);
    void displayOrderDetails();
    ~Order();
};
```

payment.h

```
#include "Payment.h"
#include <iostream>
#include <cstring>
#include <cstdlib>
using namespace std;

class Payment
{
private:
    char paymentID[10];
    char paymentType[10];
    double paymentAmount;
    char paymentDate[20];
    long int cardNumber;
    int pin;
public:
    Payment();
    void Paymentdet(const char pID[], const char pType[], double pAmount, const
char pDate[], long int cNumber, int cvv);
    void displayPaymentdet();
    long int getCnumber();
    int getPin();
  void dispSecuredinfo();
```

```
    ~Payment();
};
```

feedback.h

```
#pragma once
#include"R_Customer.h"
#include <iostream>
#include<string>

//class Feedback
class FeedBack {
private:
    char FeedBackId[10];
    char FeedBackDate[20];
    char Discription[50];
public:
    FeedBack();
    FeedBack(const char FId[],const char Fdate[], const char Fdis[]);
    void DisplayFeedBack();
    ~FeedBack();
};
```

user.cpp

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

user::user()
{

}
user::user(string uID, string uName,string uEmail)
{
  ID = uID;
  Name = uName;
  Email = uEmail;
}
user::~user()
{
  cout << "User deleted" << endl;
}
```

staffMember.cpp

```
#pragma once
#include"staffMember.h"
#include"user.h"
#include"recipe.h"
#include <iostream>
#include<string>

using namespace std;

staffMember::staffMember()
{
    staffM = new Recipe();
}

staffMember::staffMember(string cName, string cID, string cEmail,string
sDesignation)
{
    Name = cName;
    ID = cID;
    Email = cEmail;
    Designation=sDesignation;
}
void staffMember:: displaystaffmDetails(){
    cout << " staff Name  = " << Name << endl;
    cout << " staff ID  = " << ID << endl;
    cout << " staff Email  = " << Email << endl;
    cout << " Staff Member Designation  = " << Designation << endl;

}
staffMember::~staffMember()
{
    cout << "Staff memeber deleted " << endl;
}
```

R_Customer.cpp

```
#pragma once
#include"R_Customer.h"
#include"user.h"
#include"feedback.h>"
#include <iostream>
#include<string>
using namespace std;

//class R_Customer Implementation
R_Customer::R_Customer()
{
    Rc = new FeedBack();
}
R_Customer::R_Customer(string cName, string cID, string cEmail, string cDOB,
string cGender ,int Noder)
{
    Name = cName;
    ID = cID;
    Email = cEmail;
    DOB = cDOB;
    Gender = cGender;
    noOfOrders=Noder;
}
void R_Customer::displayCustomerDetails()
{
    cout << " Customer Name  = " << Name << endl;
    cout << " Customer ID  = " << ID << endl;
    cout << " Customer Email  = " << Email << endl;
    cout << " Customer DOB  = " << DOB << endl;
    cout << " Customer Gender  = " << Gender << endl;
    cout << " NO. of Oders  = " << noOfOrders << endl;
}

R_Customer::~R_Customer()
{
    cout << "Registerd customer Deleted " << endl;
}
```

Recipe.cpp

```
#pragma once
#include"staffMember.h"
#include"user.h"
#include"recipe.h>"
#include <iostream>
#include<string>

Recipe::Recipe()
{
    strcpy(recipeId, "");
    strcpy(recipeName, "");
    strcpy(publishDate, "");
    strcpy(category, "");
}

Recipe::Recipe(const char rId[], const char rName[], const char rdate[], const
char rCategory[])
{
    strcpy(recipeId, rId);
    strcpy(recipeName, rName);
    strcpy(publishDate, rdate);
    strcpy(category, rCategory);
}

void Recipe::DisplayRecipe()
{
    cout << "Recipe Id:" << recipeId << endl;
    cout << "Recipe Name:" << recipeName << endl;
    cout << "Date:" << publishDate << endl;
    cout << "Category:" << category << endl;
}

void Recipe::UpdateRecipe()
{
}

Recipe::~Recipe()
{
    cout << "Everthing is deleted" << endl;
}
```

Order.cpp

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

//class Order Implimentations
Order::Order()
{

}
Order::Order(string id, string date, R_Customer *CUS)
{
    OrderID = id;
    OrderDate = date;
    cus = CUS;

}
void Order::displayOrderDetails()
{
    cout << " orderID : " << OrderID << endl;
    cus->displayCustomerDetails();
}

Order::~Order()
{
    cout << "Order Deleted" << endl;
}
```

Payment.cpp

```
#include "Payment.h"
#include <iostream>
#include <cstring>
#include <cstdlib>
using namespace std;

Payment::Payment()
{
    strcpy(paymentID, "");
    strcpy(paymentType, "");
    paymentAmount = 0;
    strcpy(paymentDate, "");
    cardNumber = 0;
```

```
    pin = 0;
}

void Payment::Paymentdet(const char pID[], const char pType[], double pAmount,
const char pDate[], long int cNumber, int cvv)
{
    strcpy(paymentID, pID);
    strcpy(paymentType, pType);
    paymentAmount = pAmount;
    strcpy(paymentDate, pDate);
    cardNumber = cNumber;
    pin = cvv;
}

long int Payment::getCnumber()
{
    return cardNumber;
}

int Payment::getPin()
{
    return pin;
}

void Payment::displayPaymentdet()
{
    cout << "________" << endl;
    cout << "Payment Id:" << paymentID << endl;
    cout << "Payment Type:" << paymentType << endl;
    cout << "Payment Amount :" << paymentAmount << endl;
    cout << "Payment Date:" << paymentDate << endl;
    cout << "________" << endl;
}
void Payment::dispSecuredinfo()
{
cout<< "Card Number :" << getCnumber() << endl;
cout<< "CVV :" << getPin() << endl;
}
Payment::~Payment()
{
    cout << "Payment deleted" << endl;
}
```

Feedback.cpp

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include<string>
using namespace std;
#include "user.h"
#include "R_Customer.h"
#include "feedback.h"

FeedBack::FeedBack()
{
    strcpy(FeedBackId, "");
    strcpy(FeedBackDate, "");
    strcpy(Discription, "");
}

FeedBack::FeedBack(const char FId[], const char Fdate[], const char Fdis[])
{
    strcpy(FeedBackId, FId);
    strcpy(FeedBackDate, Fdate);
    strcpy(Discription, Fdis);
}

void FeedBack::DisplayFeedBack()
{
    cout<<"----------------------"<<endl;
    cout << "FeedBack Id:" << FeedBackId << endl;
    cout << "Date:" << FeedBackDate << endl;
    cout << "Discription:" << Discription << endl;
}

FeedBack::~FeedBack()
{
    cout << "Everthing is deleted" << endl;
}
```

Main.cpp

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

#include "user.h"
#include "staffMember.h"
#include "R_Customer.h"
#include "recipe.h"
#include "Order.h"
#include "Payment.h"
#include "feedback.h"

int main()
{
  cout<<"************"<<endl;

  R_Customer *C1 = new R_Customer("Nipuna", "IT21469046", "niphes1123@gmail.com",
"2000-05-01", "Male" ,5);
    C1->displayCustomerDetails();
    delete C1;

  cout<<"************"<<endl<<endl;

  staffMember*s1 = new staffMember("Mahinda R.", "22222AV"
,"mahindar22@gmail.com" ,"senior Editor");
    s1->displaystaffmDetails();
    delete s1;

  cout<<"************"<<endl;

  Order* O1 = new Order("OD2154", "2021-02-30", C1);
    O1->displayOrderDetails();
    delete O1;

    cout<<"************"<<endl;

    Payment* p1 = new Payment();
    Payment* p2 = new Payment();
    Payment* p3 = new Payment();

  p1->Paymentdet("P101256", "Debit Card", 5000, "2022-05-14", 2115566669, 556 );
  p2->Paymentdet("P101587", "Credit Card", 3500, "2022-06-20", 2115558348, 539 );
```

```
  p3->Paymentdet("P159345", "Debit Card", 4700, "2022-03-09", 2117854112, 457 );

  p1->displayPaymentdet();
  p1->dispSecuredinfo();

  p2->displayPaymentdet();
  p2->dispSecuredinfo();

  p3->displayPaymentdet();
  p3->dispSecuredinfo()
return 0;
}
```