

**Topic : Online bill and Event Reminder**

**Group no : MLB\_08.01\_05**

**Campus : Malabe**

**Submission Date: 2022/05/12**

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** |
| IT21165016 | Kawya R M S | 078 289 4539 |
| IT21164408 | Warunika H P R | 077 561 4439 |
| IT21165498 | Perera W H T H | 077 866 2408 |
| IT21166310 | Kumara W H T S | 077 894 4327 |
| IT21164262 | Gamage T G S N | 078 992 1868 |

**Requirements Specification**

1. In an online bill and event reminder system it has two types of users, one is new user, and the other is registered user.
2. A registered user or new user both can visit online bill and event reminder system by using our website.
3. A new user needs to register to the system by providing details such as name, address, Email, phone number, DOB, country, username, and the password.
4. A registered user can add event reminders for their special occasions after providing date, time, email, and description.
5. They can also add bill reminders for their bill payment by adding date, bill type, bill amount, email, and description.
6. A registered user can add, edit, or delete their reminders and review added reminders.
7. User can get notification after adding reminders.
8. Our system provides 3 subscription plans for different amounts and different time durations.
9. User can purchase one subscription plan and get notification via email and SMS.
10. Users can pay for subscription using debit card, credit card and PayPal.
11. Users can rate and write feedbacks about our system.
12. Users can check for the latest and upcoming public events.
13. If users need any other help, they can refer FAQs.
14. Users have any doubt of our webpage they can contact us.
15. The system will record event reminders, bill reminders and user details.
16. The system admin can update the upcoming public events, validate payment, maintain, and update the system.

Identified classes

1. Guest user
2. Registered user
3. Event reminder
4. Bill reminder
5. Public event



1. Payment
2. Feedback
3. Subscription
4. Report

**EXERCISE 2**

CRC Cards

|  |  |
| --- | --- |
| **Class name:** Guest user | |
| **Responsibilities:** | **Collaborations:** |
| Register details |  |
| Search event | Public Event |

|  |  |
| --- | --- |
| **Class name:** Registereduser | |
| **Responsibilities:** | **Collaborations:** |
| Store User Details |  |
| Modify details |  |
| Subscribe to the system | Subscription |
| Make payment | Payment |
| Send feedback | Feedback |
| Set reminder | Event reminder, bill reminder |
| Delete reminder | Event reminder, bill reminder |
| Review added reminders | Report |
| Search event | Event reminder, bill reminder, public event |

|  |  |
| --- | --- |
| **Class name:** Event Reminder | |
| **Responsibilities:** | **Collaborations:** |
| Add event reminder |  |
| Update event reminder |  |
| Store reminder details |  |

|  |  |
| --- | --- |
| **Class name:** Bill Reminder | |
| **Responsibilities:** | **Collaborations:** |
| Add bill details |  |
| Update bill details |  |
| Store reminder details |  |

|  |  |
| --- | --- |
| **Class name:** Payment | |
| **Responsibilities:** | **Collaborations:** |
| Store payment details |  |
| Validate |  |

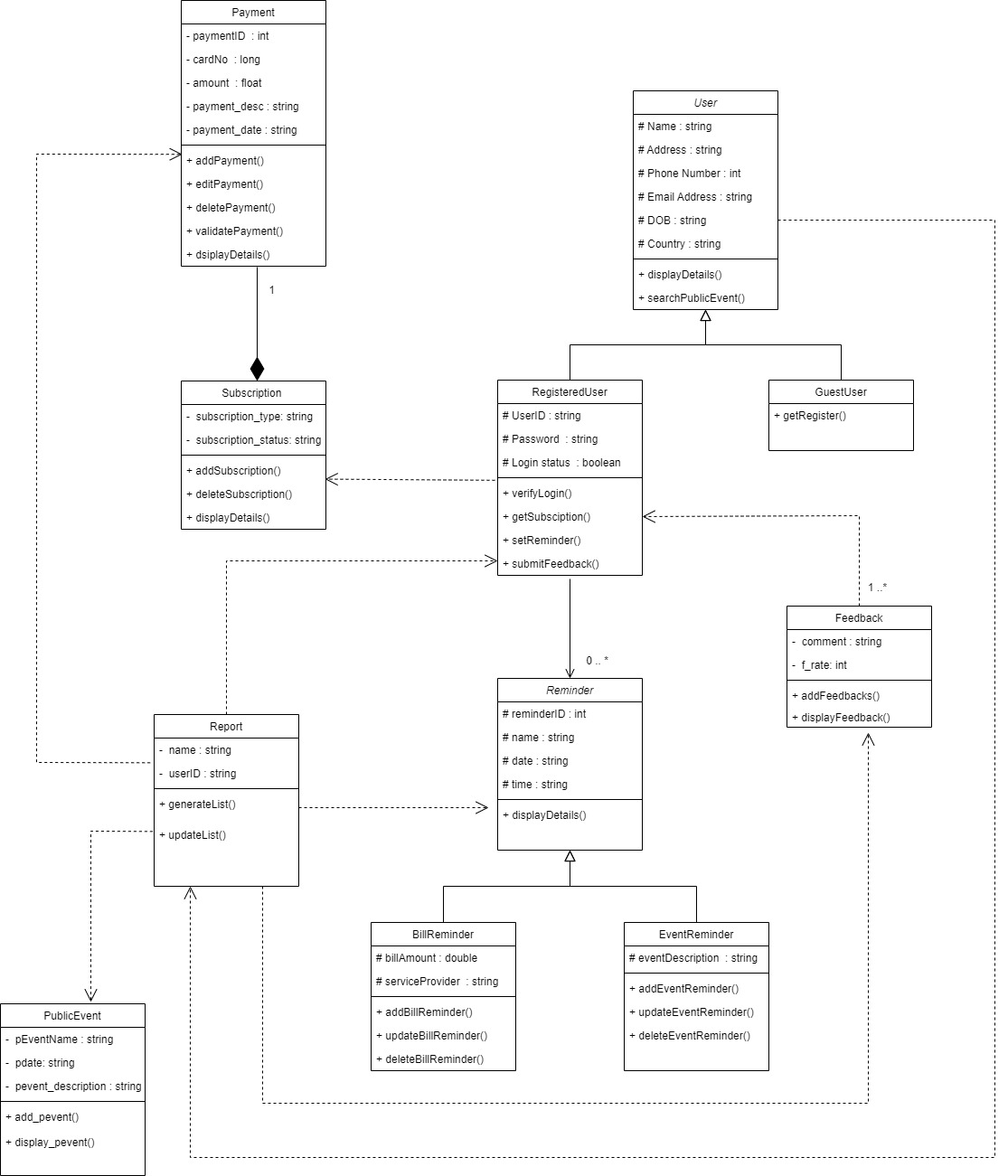
|  |  |
| --- | --- |
| **Class name:** Feedback | |
| **Responsibilities:** | **Collaborations:** |
| Send feedbacks | Registered user |
| Store feedback details |  |

|  |  |
| --- | --- |
| **Class name:** Public Event | |
| **Responsibilities:** | **Collaborations:** |
| Store public event details |  |
| Add public event |  |
| Update public event |  |
| Delete public event |  |

|  |  |
| --- | --- |
| **Class name:** Subscription | |
| **Responsibilities:** | **Collaborations:** |
| Add subscription |  |
| Confirm subscription | Payment |
| Expire subscription |  |

|  |  |
| --- | --- |
| **Class name:** Report | |
| **Responsibilities:** | **Collaborations:** |
| List of added reminders | Event reminder, bill reminder, registered user |
| List of public events | Public event |
| List of previous subscriptions | Subscription |
| List of feedbacks | Feedback |
| List of payments | Payment |

**EXERCISE 3**

****Class diagram

**EXERCISE 4**

Coding

/\*-----------------User Class------------\*/

using namespace std;

#include <string>

//base class

class User{

protected:

string name;

string address;

int mobileNo;

string email;

string dob;

string country;

public:

User();

User(string uname,string uaddress,int umobileNo,string uemail,string udob,string ucountry);

//this is a abstract class.

virtual void displayDetails() = 0;

//dependecy relationship with Report Class

void searchPublicEvent(string name, Report \*r);

};

/\*-----------------User Class Implementation------------\*/

#include "User.h"

#include <string>

User::User(){

name = "No value";

address = "No value";

mobileNo = 0000000000;

email = "No value";

dob = "No value";

country = "No value";

}

User::User(string uname,string uaddress,int umobileNo,string uemail,string udob,string ucountry){

name = uname;

address = uaddress;

mobileNo = umobileNo;

email = uemail;

dob = udob;

country = ucountry;

}

void User::searchPublicEvent(string name, Report \*r){

}

/\*-----------------Guest\_user Class------------\*/

//derived class from User class

#include "User.h"

class Guest\_user : public User{

public:

Guest\_user();

Guest\_user(string gname,string gaddress,int gno,string gmail,string gdob,string gcountry);

void displayDetails();

~Guest\_user();

};

/\*-----------------Guest\_user Class Implementation------------\*/

#include <iostream>

using namespace std;

#include "Guest\_user.h"

Guest\_user::Guest\_user(){

name = "No value";

address = "No value";

mobileNo = 0000000000;

email = "No value";

dob = "No value";

country = "No value";

}

Guest\_user::Guest\_user(string gname,string gaddress,int gno,string gmail,string gdob,string gcountry){

name = gname;

address = gaddress;

mobileNo = gno;

email = gmail;

dob = gdob;

country = gcountry;

}

//overriding displayDetails function

void displayDetails(){

cout << name <<endl << address <<endl << mobileNo <<endl << email <<endl << dob <<endl << country << endl;

}

Guest\_user::~Guest\_user(){

cout << "Destructor called for Guest\_user object" << endl;

}

/\*-----------------Registered\_user Class------------\*/

//derived class from User class

using namespace std;

#include <string>

#include "User.h"

#define SIZE 5

class Registered\_user : public User{

protected:

string userID;

string password;

bool login\_status;

Reminder \*r[SIZE]; //Uni-directional association relationship with Reminder class

public:

Registered\_user();

Registered\_user(string rname,string raddress,int rno,string rmail,string rdob,string rcountry,string id, string pw,bool logstatus);

void verify\_login(string userID,string password);

void getSubscription(Subscription \*s); //dependecy relationship with Subscription Class

void displayDetails();

~Registered\_user();

};

/\*----------------- Registered\_user Class Implementation------------\*/

#include <iostream>

using namespace std;

#include "Registered\_user.h"

#include <string>

Registered\_user::Registered\_user()

{

name = "No value";

address = "No value";

mobileNo = 0000000000;

email = "No value";

dob = "No value";

country = "No value";

userID = "No value";

password = "No value";

login\_status = true;

}

//implementation

void displayuserdetails(){};

Registered\_user::Registered\_user(string rname,string raddress,int rno,string rmail,string rdob,string rcountry,string id, string pw,bool logstatus){

name = rname;

address = raddress;

mobileNo = rno;

email = rmail;

dob = rdob;

country = rcountry;

userID = id;

password = pw;

login\_status = logstatus;

}

void Registered\_user::verify\_login(string userID,string password){

}

void Registered\_user::getSubscription(Subscription \*s){

}

//overriding displayDetails function

void Registered\_user::displayDetails(){

cout << name <<" "<< address <<" "<< mobileNo <<" "<< email <<" "<< dob <<" "<< country <<" "<< userID <<" "<< password <<" "<< login\_status << endl;

}

Registered\_user::~Registered\_user(){

cout << "Destructor called for Registered\_user object" << endl;

}