Topic : Online Bill And Event Reminder

Group no : MLB_WD_CSNE_13_10

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**
IT21275388	Fernando U. S. K	0704932161
IT21207990	Pieris M. H. N	0760999477
IT21230592	Dassanayake D. R. I. P	0741519829
IT21260360	Alwis M. A. M. D. S	0774726014
IT21297472	Rajapaksha M. D	0710377201

Table of Contents

System requirements

- New users can register to the system and become registered users and users can not set reminders without a user account.
- A new user should register to the system proving details username, email and contact number
- Registered users must login to the system by providing login credentials which are the username and the password.
- System validates the login credentials and allow the user to login to the system.
- System generates a user id for each user.
- Mainly there are two reminders as bill reminders and event reminders.
- The system allows users to set reminders for bills.
- Title of the bill, the date that should be reminded, time that should be reminded, due date of the bill, company name of the bill, amount to be paid, additional charges, additional notes, and repeating frequency can be store in the bill reminder.
- System calculates the fines of the bills of each user.
- System allows users to set reminders for events
- Title of the event, time and date of the event, venue, duration, description, date that should be reminded, the time that should be reminded must be stored in the event reminder.
- User can view the upcoming reminders and the completed reminders.
- User can edit the reminders in the upcoming reminder list.
- User can delete the completed reminders and upcoming reminders
- User can receive reminders via emails and messages
- Receiving reminders are included with all the details that the user stored when setting the reminder.
- User can give feedback as a comment.
- User will be notified continuously if the user set a repeating frequency.
- User can edit username, passwords, email, contact numbers in their profiles according to their desire.

Identified classes

- Reminder
- User
- loginCredentials
- BillReminder
- EventReminder
- UpcomingReminders
- CompletedReminders
- Feedback
- Account
- Report

CRC cards

User	
Responsibility	**Collaborators**
Register to the system	Account
Set reminder details	Reminder
View scheduled reminders	Completed reminders, upcoming reminders
Give feedbacks	Feedback

Account	
Responsibility	**Collaborators**
Edit account details	
Login to the account	Login credentials
Delete the account	User

Login credentials	
Responsibility	**Collaborators**
Store login credentials	
Validate the login credentials	Account
Edit login credentials	

Reminders	
Responsibility	**Collaborators**
Store reminder details	User, EventReminder, BillReminder
Display reminder details	

BillReminder	
Responsibility	**Collaborators**
Store the details of the bill reminders	Reminder
Edit bill reminders	
Receive bill reminders	
Calculate fines	

EventReminder	
Responsibility	**Collaborators**
Store the details of the event reminders	Reminder
Edit event reminders	
Receive event reminders	

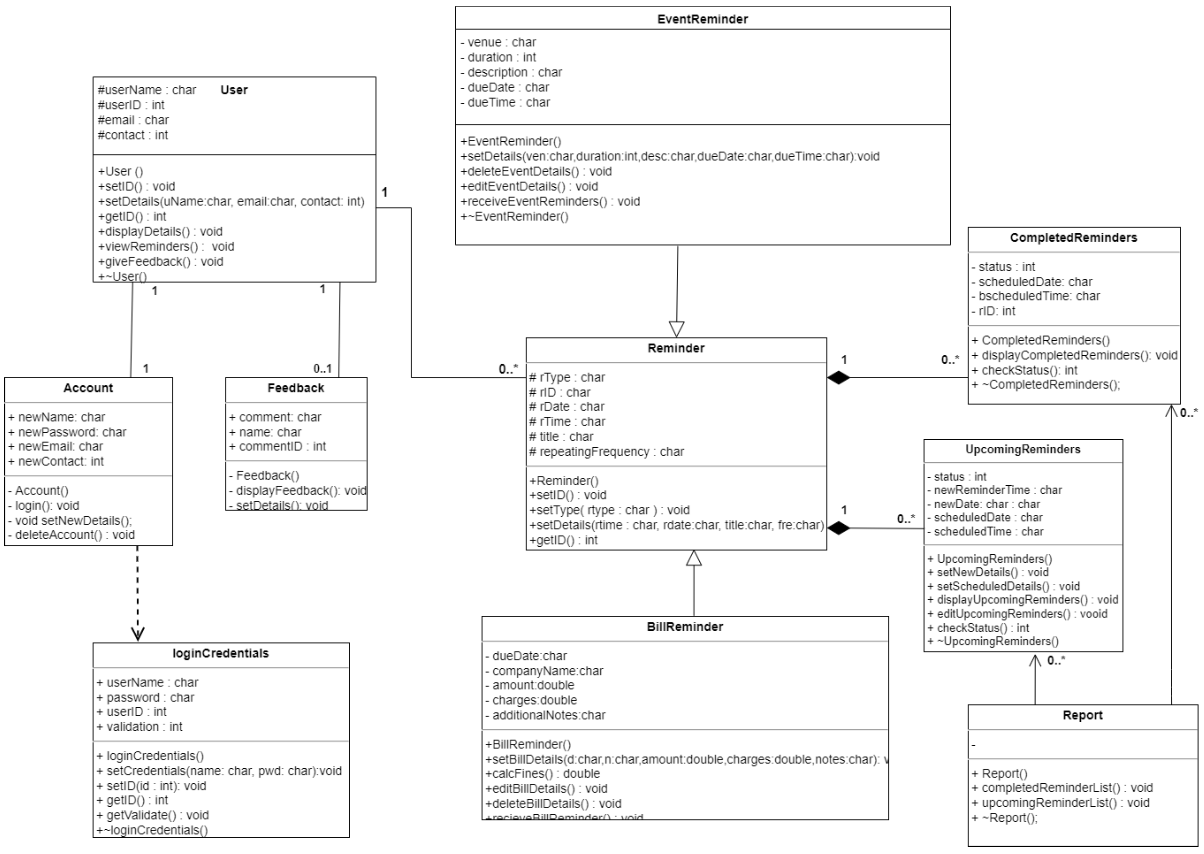
CompletedReminders	
Responsibility	**Collaborators**
Store the reminders which are completed	Reminder
Display completed reminders	
check status	

UpcomingReminders	
Responsibility	**Collaborators**
Store the reminder details which are not completed yet	Reminder
Display upcoming reminders	
Check status	

Report	
Responsibility	**Collaborators**
List of completed reminders	Completed reminders
List of upcoming reminders	Upcoming reminders

Feedback	
Responsibility	**Collaborators**
Store the comments	user
Display the comments	
edit comments	User

Class Diagram (UML Notation)



Header files of the classes

Reminder.h (IT21207990 - Pieris M. H. N)

```
#include "User.h"
#include "UpcomingReminders.h"
#include "CompletedReminders.h"
#define SIZE1 5
#define SIZE2 5

class Reminder {
protected:
	char rType[10];
	int rID;
	char rDate[10];
	char rTime[10];
	char title[50];
	int repeatingFrequency;

	User *user1;
	CompletedReminders *CompR[SIZE1];
	UpcomingReminders *UpcoR[SIZE2];

public:
	Reminder();
	Reminder(char type[], int id, char time[], char date[], char title[], int fre,
User *puser);
	void setID();
	void setType(char type);
	void setDetails(char time, char date, char title, char fre);
	int getID();
	void displayDetails();
	~Reminder();
};
```

EventReminder.h (IT21275388 - Fernando U. S. K)

```
#include "Reminder.h"
#include "User.h"

class EventReminder : public Reminder
{
private:
	char venue[20];
	int duration;
	char description[100];
	char dueDate[10];
	char dueTime[10];

public:
	EventReminder();
	EventReminder(char ven[], int dur, char desc[], char dDate[], char dTime[],
		char type[], int id, char time[], char date[], char title[],
		int fre, User *puser);
	void setDetails(char ven[], int dur, char desc[], char dDate[], char dTime);
	void deleteEventDetails();
	void receiveEventReminders();
	~EventReminder();
}
```

BillReminder.h (IT21207990 - Pieris M. H. N)

```
#include "Reminder.h"
#include "User.h"

// the child class of the reminder class
class BillReminder : public Reminder {
private:
	char dueDate[10];
	char companyName[20];
	double amount;
	double charges;
	char additionalNotes[100];

public:
	BillReminder();
	BillReminder(char dDate[], char compName[], double amnt, double chrg,
		char type[], int id, char time[], char date[], char title[],
		int fre);
	void setBillDetails(char dt, char amnt, double chr, char nts);
	double calcFines();
	void editBillDetails();
	void deleteBillDetails();
	void recieveBillReminder();
	~BillReminder();
};
```

CompletedReminders.h (IT21260360 - Alwis M. A. M. D. S)

```
// A part of the reminder class
class CompletedReminders {
private:
	int status;
	char scheduledDate[10];
	char scheduledTime[10];
	int rID;

public:
	// constructor
	CompletedReminders();
	// overload constructor
	CompletedReminders(int Cstatus, char CscheduledDate[], char CscheduledTime[],
		int CrID);
	void displayCompletedReminders();
	int checkStatus(char schedDate[], char schedTime[]);
	// destructor
	~CompletedReminders();
};
```

UpcomingReminders.h (IT21260360 - Alwis M. A. M. D. S)

```
class UpcomingReminders {
private:
	int status;
	char newReminderTime[10];
	char newDate[10];
	char scheduledDate[10];
	char scheduledTime[10];

public:
	UpcomingReminders();
	UpcomingReminders(int Ustatus, char UnewReminderTime[10], char UnewDate[10],
		char UscheduledDate[10], char UscheduledTime[10]);
	void setNewDetails(char UnewReminderTime[], char UnewDate[]);
	void setScheduledDetails(char schedDate[], char schedTime[]);
	void displayUpcomingReminders();
	void editUpcomingReminders();
	int checkStatus(char schedDate[], char schedTime);
	~UpcomingReminders();
};
```

Account.h (IT21230592 - Dassanayake D. R. I. P)

```
#include "loginCredentials.h"
#include "User.h"
class Account
{
private:
	char newName[20];
	char newPassword[20];
	char newEmail[50];
	int newContact;
	User *accountUser;

public:
	Account();
	Account(char name[], char nemail[], int nCon, char npwd[], User *puser);
	void login(loginCredentials *p);//using loginCredentials class for validation
	void deleteAccount();
	void setNewDetails(char name[], char nemail[], int nCon, char npwd[]);
	void editAccount();
	~Account();
};
```

User.h (IT21275388 - Fernando U. S. K)

```cpp
#include "Account.h"
#include "Feedback.h"
#include "Reminder.h"
#define SIZE 10
class User {
private:
	char userName[20];
	int userID;
	char email[50];
	int contact;
	int noOfReminders;
	Reminder *rem[SIZE];
	Account *useraccount;
	Feedback *feed;

public:
	User();
	User(int id, char uName[], char Uemail[], int con, int pnoOfReminders);
	void addReminder(Reminder *r);
	void addAccount(Account *acc);
	void addFeedback(Feedback *userfeed);
	void setID(int id);
	void setDetails(char uName[], char Uemail[], int con);
	int getID();
	void displayDetails();
	void viewReminders();
	void giveFeedback();

	~User();
};
```

loginCredentials.h (IT21297472 – Rajapaksha M. D)

```cpp
class loginCredentials {
private:
	char userName[20];
	char password[20];
	int validation;
	int userID;

public:
	loginCredentials();
	loginCredentials(int id, char name[], char pwd[], int valid);
	void setCredentials(char name[], char pwd[], int valid);
	void setID(int id);
	void setValidation(int valid);
	int getID();
	int getValidation();
	~loginCredentials();
};
```

Feedback.h (IT21297472 – Rajapaksha M. D)

```
#include "User.h"
class Feedback {
private:
	char comments[20];
	char name[20];
	int commentID;
	User *user2;
public:
	Feedback();
	Feedback(char com[], char comname[], int comID, User *puser);
	void displayFeedback();
	void setDetails(char com[], char comname[], int comID);
	~Feedback();
};
```

Report.h (IT21230592 - Dassanayake D. R. I. P)

```
#include "CompletedReminders.h"
#include "UpcomingReminders.h"
#define SIZE1 5
#define SIZE2 5

class Report {
private:
	//One way association with CompletedReminder class
	CompletedReminders *compR[SIZE1];
	//one way association with upcomingReminder class
	UpcomingReminders *upcomR[SIZE2];
public:
	Report();
	Report(UpcomingReminders *pup[], CompletedReminders *pcom[]);
	void completedReminderList();
	void upcomingReminderList();
	~Report();
};
```

Cpp files of the classes

Reminder.cpp (IT21207990 - Pieris M. H. N)

```
#include <iostream>
#include <cstring>
#include "User.h"
#include "Reminder.h"

using namespace std;

Reminder::Reminder()
{
	strcpy(rType, "");
	rID = 0;
	strcpy(rDate, "");
	strcpy(rTime, "");
	strcpy(title, "");
	repeatingFrequency = 0;
	user1 = NULL;

	for (int i = 0; i < SIZE1; i++) {
		CompR[i] = new CompletedReminders[0];
	}
	for (int i = 0; i < SIZE2; i++) {
		UpcoR[i] = new UpcomingReminders[0];
	}
}
Reminder::Reminder(char type[], int id, char time[], char date[], char title[], int
fre, User *puser)
{
	strcpy(rType, type);
	rID = id;
	strcpy(rDate, date);
	strcpy(rTime, time);
	strcpy(title, title);
	repeatingFrequency = fre;
	user1 = puser;

	for (int i = 0; i < SIZE1; i++) {
		CompR[i] = new CompletedReminders[id];
	}
	for (int i = 0; i < SIZE2; i++) {
		UpcoR[i] = new UpcomingReminders[id];
	}
}
void Reminder::setID() {

}
void Reminder::setType(char type) {

}
void Reminder::setDetails(char time, char date, char title, char fre) {

}
int Reminder::getID() {

}
void Reminder::displayDetails() {

}
```

```
Reminder::~Reminder()
{
	cout << "reminder shutting down!" << endl;
	for (int i = 0; i < SIZE1; i++)
	{
		delete CompR[i];
	}
	for (int i = 0; i < SIZE2; i++)
	{
		delete UpcoR[i];
	}

}
```

EventReminder.cpp (IT21275388 - Fernando U. S. K)

```
#include <iostream>
#include <cstring>
#include "EventReminder.h"
#include "BillReminder.h"
#include "User.h"
#include "Reminder.h"

using namespace std;
// defualt constructor
EventReminder::EventReminder()
{
	strcpy(venue, "");
	duration = 0;
	strcpy(description, "");
	strcpy(dueDate, "");
	strcpy(dueTime, "");
}
// overloaded constructor
EventReminder::EventReminder(char ven[], int dur, char desc[], char dDate[],
	char dTime[], char type[], int id, char time[],
	char date[], char title[], int fre, User *puser)
	: Reminder(type, id, time, date, title, fre, NULL) // attributes that inherits
from the parent
{
	strcpy(venue, ven);
	duration = dur;
	strcpy(description, desc);
	strcpy(dueDate, dDate);
	strcpy(dueTime, dTime);
}
void EventReminder::setDetails(char ven[], int dur, char desc[], char dDate[],
	char dTime)
{

}
void EventReminder::deleteEventDetails()
{

}
void EventReminder::receiveEventReminders()
{

}
// destructor
EventReminder::~EventReminder(){
	cout << "desctuctor activated" << endl;
}
```

BillReminder.cpp (IT21207990 - Pieris M. H. N)

```
#include <cstring>
#include <iostream>
#include "Reminder.h"
#include "BillReminder.h"
#include "User.h"
using namespace std;

BillReminder::BillReminder() {
	strcpy(dueDate, "");
	strcpy(companyName, "");
	amount = 0;
	charges = 0;
	strcpy(additionalNotes, "");
}

BillReminder::BillReminder(char dDate[], char compName[], double amnt,
	double chrg, char type[], int id, char time[],
	char date[], char title[], int fre)
	: Reminder(type, id, time, date, title, fre, NULL) {
	strcpy(dueDate, dDate);
	strcpy(companyName, compName);
	amount = amnt;
	charges = chrg;
}
void BillReminder::setBillDetails(char dt, char amnt, double chr, char nts) {}
double BillReminder::calcFines() {}
void BillReminder::editBillDetails() {}
void BillReminder::deleteBillDetails() {}
void BillReminder::recieveBillReminder() {}
BillReminder::~BillReminder() {
	cout << "desctuctor for the bill activated" << endl;
}
```

CompletedReminders.cpp (IT21260360 - Alwis M. A. M. D. S)

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

// defualt constructor
CompletedReminders::CompletedReminders() {
	status = 0;
	strcpy(scheduledDate, "");
	strcpy(scheduledTime, "");
	rID = 0;
}
// overload constructor
CompletedReminders::CompletedReminders(int Cstatus, char CscheduledDate[], char
CscheduledTime[], int CrID) {
	status = Cstatus;
	strcpy(scheduledDate, CscheduledDate);
	strcpy(scheduledTime, CscheduledTime);
	rID = CrID;
}
void CompletedReminders::displayCompletedReminders() {}
int CompletedReminders::checkStatus(char schedDate[], char schedTime[]) {}
// destructor
CompletedReminders::~CompletedReminders() {
	cout << "destructor activated!" << endl;
}
```

UpcomingReminders.cpp (IT21260360 - Alwis M. A. M. D. S)

```cpp
#include "UpcomingReminders.h"
#include <cstring>
#include <iostream>
using namespace std;

UpcomingReminders::UpcomingReminders() {
	status = 0;
	strcpy(newReminderTime, "");
	strcpy(newDate, "");
	strcpy(scheduledDate, "");
	strcpy(scheduledTime, "");
}

UpcomingReminders::UpcomingReminders(int Ustatus, char UnewReminderTime[], char
UnewDate[], char UscheduledDate[], char UscheduledTime[]) {
	status = Ustatus;
	strcpy(newReminderTime, UnewReminderTime);
	strcpy(newDate, UnewDate);
	strcpy(scheduledDate, UscheduledDate);
	strcpy(scheduledTime, UscheduledTime);
}
void UpcomingReminders::setNewDetails(char UnewReminderTime[], char UnewDate[]) {}
void UpcomingReminders::setScheduledDetails(char schedDate[], char schedTime[]) {}
void UpcomingReminders::displayUpcomingReminders() {}
void UpcomingReminders::editUpcomingReminders() {}
int UpcomingReminders::checkStatus(char schedDate[], char schedTime) {}
UpcomingReminders::~UpcomingReminders() {
	cout << "destructor activated!" << endl;
}
```

Account.cpp (IT21230592 - Dassanayake D. R. I. P)

```cpp
#include <cstring>
#include <iostream>
#include "Reminder.h"
#include "BillReminder.h"
#include "User.h"
using namespace std;

BillReminder::BillReminder() {
	strcpy(dueDate, "");
	strcpy(companyName, "");
	amount = 0;
	charges = 0;
	strcpy(additionalNotes, "");
}

BillReminder::BillReminder(char dDate[], char compName[], double amnt,
	double chrg, char type[], int id, char time[],
	char date[], char title[], int fre)
	: Reminder(type, id, time, date, title, fre, NULL) {
	strcpy(dueDate, dDate);
	strcpy(companyName, compName);
	amount = amnt;
	charges = chrg;
}
void BillReminder::setBillDetails(char dt, char amnt, double chr, char nts) {}
double BillReminder::calcFines() {}
void BillReminder::editBillDetails() {}
void BillReminder::deleteBillDetails() {}
void BillReminder::recieveBillReminder() {}
BillReminder::~BillReminder() {
	cout << "desctuctor for the bill activated" << endl;
}
```

User.cpp (IT21275388 - Fernando U. S. K)

```
#include "Account.h"
#include "Reminder.h"
#include "User.h"
#include <cstring>
using namespace std;

User::User()
{
	strcpy(userName, "");
	userID = 0;
	strcpy(email, "");
	contact = 0;
	noOfReminders = 0;
}
User::User(int id, char uName[], char Uemail[], int con, int pnoOfReminders)
{
	strcpy(userName, uName);
	userID = id;
	strcpy(email, Uemail);
	contact = con;
	noOfReminders = pnoOfReminders;
}
void User::addReminder(Reminder *r)
{
	if (noOfReminders < SIZE) {
		rem[noOfReminders] = r;
		noOfReminders++;
	}
}
void User::addAccount(Account *acc)
{

}
void User::addFeedback(Feedback *userfeed)
{

}
void User::setID(int id)
{

}
void User::setDetails(char uName[], char email[], int con)
{

}
int User::getID()
{

}
void User::displayDetails()
{
}
void User::viewReminders()
{

}
void User::giveFeedback()
{

}
User::~User()
{
	// Destructor
	for (int i = 0; i < SIZE; i++)
	{
		delete rem[i];
	}
}
```

loginCrerdentials.cpp (IT21297472 – Rajapaksha M. D)

```
#include "loginCredentials.h"
#include "User.h"
#include <cstring>
#include <iostream>
using namespace std;

loginCredentials::loginCredentials() {
	strcpy(userName, "");
	strcpy(password, "");
	userID = 0;
	validation = 0;
}
loginCredentials::loginCredentials(int id, char name[], char pwd[], int valid) {
	strcpy(userName, name);
	strcpy(password, pwd);
	userID = id;
	validation = valid;
}
int loginCredentials::getValidation() { return validation; }
void loginCredentials::setCredentials(char name[], char pwd[], int valid) {}
void loginCredentials::setID(int id) {}
int loginCredentials::getID() {}
loginCredentials::~loginCredentials() {}
```

Feedback.cpp (IT21297472 – Rajapaksha M. D)

```
#include <iostream>
#include <cstring>
#include "Feedback.h"
#include "User.h"
using namespace std;

Feedback::Feedback()
{
	strcpy(comments, "");
	strcpy(name, "");
	commentID = 0;
}
Feedback::Feedback(char com[], char comname[], int comID, User *puser)
{
	strcpy(comments, com);
	strcpy(name, comname);
	commentID = comID;
	user2 = puser;
}
void Feedback::displayFeedback()
{

}
void Feedback::setDetails(char com[], char comname[], int comID)
{

}
Feedback::~Feedback()
{
	cout << " desctuctor activated " << endl;
}
```

Report.cpp (IT21230592 - Dassanayake D. R. I. P)

```
#include "Report.h"
#define SIZE1 5
#define SIZE2 5

Report::Report() {
	for (int i = 0; i < SIZE1; i++) {
		compR[i] = 0;
	}
	for (int j = 0; j < SIZE2; j++) {
		upcomR[j] = 0;
	}
}
Report::Report(UpcomingReminders *pup[], CompletedReminders *pcom[]) {
	for (int i = 0; i < SIZE1; i++) {
		compR[i] = pcom[i];
	}
	for (int j = 0; j < SIZE2; j++) {
		upcomR[j] = pup[j];
	}
}
void Report::completedReminderList()
{

}
void Report::upcomingReminderList()
{

}
Report::~Report() {
	for (int i = 0; i < SIZE1; i++) {
		delete compR[i];
	}
	for (int j = 0; j < SIZE2; j++) {
		delete upcomR[j];
	}
```

Main program

main.cpp

```
#include "Account.h"
#include "BillReminder.h"
#include "CompletedReminders.h"
#include "EventReminder.h"
#include "Feedback.h"
#include "Reminder.h"
#include "Report.h"
#include "UpcomingReminders.h"
#include "User.h"
#include "loginCredentials.h"

#include <iostream>

using namespace std;
int main() {
	// object creation
	Reminder *BR = new Reminder();    // object of BillReminder class
	Reminder *ER = new Reminder();    // object of EventReminder class
	Account *acc = new Account();     // object of Account class
	User *usr = new User();           // object of User class
	Feedback *fBack = new Feedback(); // object of Feedback class
	Reminder *upRem = new Reminder(); // object of UpcomingReminder class
	Reminder *compR = new Reminder(); // object of CompletedReminder class
	loginCredentials *lgn = new loginCredentials(); // object of loginCrerdentials 
class
	Report *rpt = new Report(); // object of Report class

	// deleting dynamic objects
	delete BR;
	delete ER;
	delete acc;
	delete usr;
	delete fBack;
	delete upRem;
	delete compR;
	delete lgn;
	delete rpt;

	return 0;
}
```

Individual contribution

IT21207990 - Pieris M. H. N – **Reminder** and **BillReminder** classes designed and created

IT21275388 - Fernando U. S. K – **User** and **EventReminder** classes designed and created.

IT21297472 - Rajapaksha M. D – **Feedback** and **loginCredentials** classes created.

IT21230592 - Dassanayake D. R. I. P - **Account** and **Report** classes designed and created.

IT21260360 - Alwis M. A. M. D. S – **Completed** and **UpcomingReminder** classes created.