Topic : Online Help Desk for University Students

Group no : MLB_10.02_01

Campus : Malabe

Submission Date : 17/05/2022

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**
IT21377594	Keshala G.P.	0740477828
IT21182532	Lakshan G.N.	0702151345
IT21307126	Nawarathne N.S.N.	0772068109
IT21379574	Dilshani H.T.D.P.	0767615996
IT21189180	Janagan K.	0766251694

1. User requirements

An unregistered student in an online help desk system needs to the first register providing details such as name, NIC number, and email address. Then the registered student can log in to the system using his/her credentials. If there are some mistakes in the login details the registered student can edit user details. A registered student can browse the module code, module name, and the number of credits from the module catalog. A registered student can submit their problems and complaints. And also, a registered student can search books in the online library. Coordinator logs in to the system and communicate with students, lecturers and management. Furthermore, the coordinator can upload lecture materials to the system. The faculty Head provides solutions for students' problems. The university administrator can provide users access to the system. Administrator can also check user accounts and remove user accounts. The university administrator gets a list of user accounts already registered and updates the system.

2. Noun & verb Analysis
(Noun are in red and Verb are in blue)

An unregistered student in an online help desk system needs to the first register providing details such as name, NIC number, and email address. Then the registered student can log in to the system using his/her credentials. If there are some mistakes in the login details the registered student can edit user details. A registered student can browse the module code, module name, and the number of credits from the module catalog. A registered student can submit their problems and complaints. And also, a registered student can search books in the online library. Coordinator logs in to the system and communicate with students, lecturers and management. Furthermore, the coordinator can upload lecture materials to the system. The faculty Head provides solutions for students' problems. The university administrator can provide users access to the system. Administrator can also check user accounts and remove user accounts. The university administrator gets a list of user accounts already registered and updates the system.

NOUN	VERB
Redundant Registered student Unregistered students Students **Attributes** Name, NIC Number, e-mail address Module code, Module name, NoOfCredits Login details User details **Outside the scope** System University administrator Management Lecturer **Classes** User Module Problems Books Online library Coordinator Faculty Head Lecture materials Solution Report	**User** Register Log in Edit Browse Submit Search **Coordinator** Communicate Upload **Faculty Head** Provide **Administrator** Check Remove Get Update

3. CRC Cards

User	
Responsibility	**Collaborators**
Register details	
Log in	
Edit user details	
Update user profile	

Module	
Responsibility	**Collaborators**
Store module details	
Update Module details	

Books	
Responsibility	**Collaborators**
Store book details	
Provide book details	

Online library	
Responsibility	**Collaborators**
Show online library facilities	
Store library members' details	User

Coordinator	
Responsibility	**Collaborators**
Communicate with users, lecturers, and faculty head	User, lecturer, faculty head
Upload lecture materials	Lecture materials

Faculty Head	
Responsibility	**Collaborators**
Provides solutions to users’ problems	Solutions, Problems

Lecture materials	
Responsibility	**Collaborators**
Add lecture materials	Coordinator
Store lecture materials	

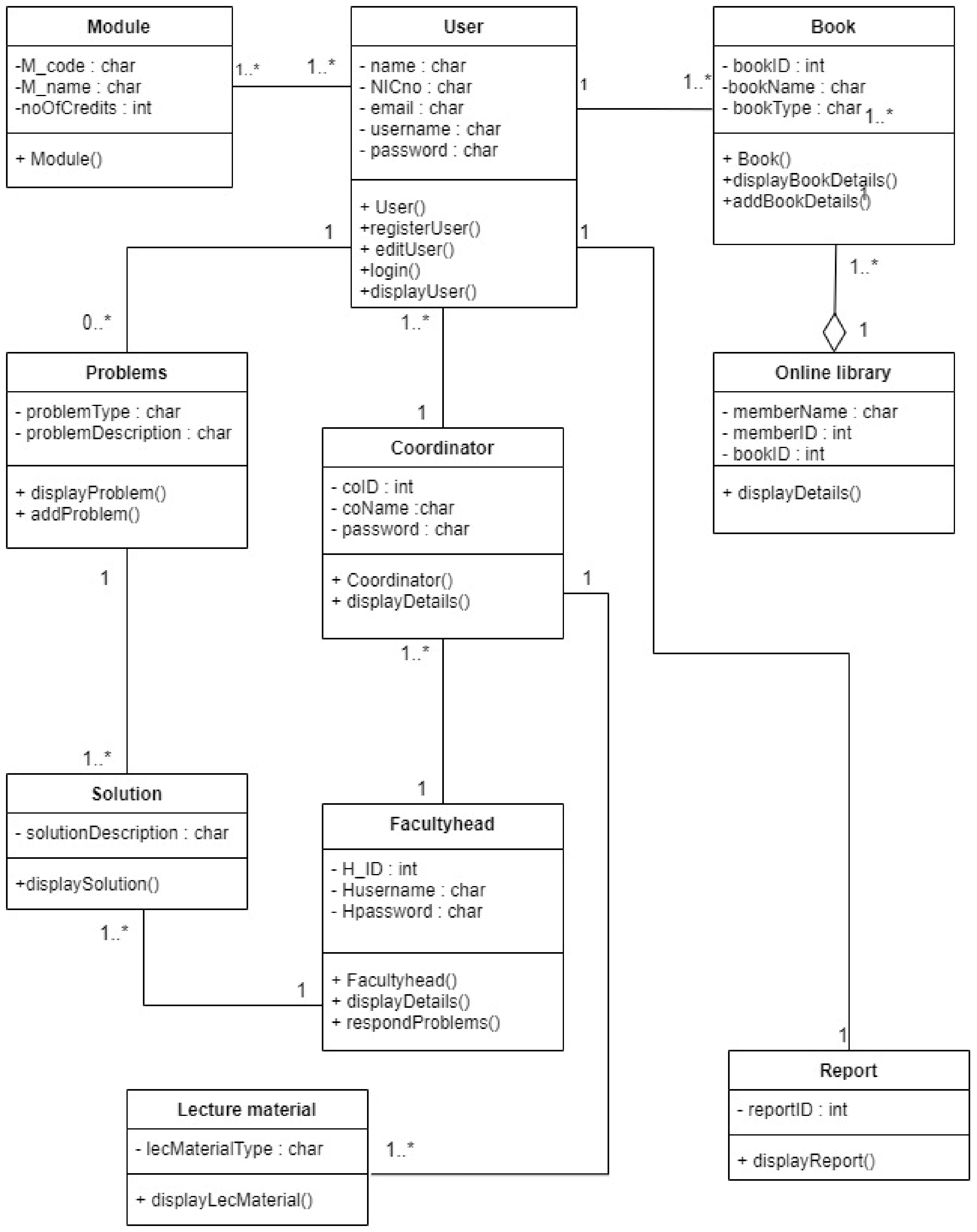
Problems	
Responsibility	**Collaborators**
Show problems	
Keep a record of all problems	User

Solution	
Responsibility	**Collaborators**
Show solution	
Store solution	

Report	
Responsibility	**Collaborators**
List of user accounts	User account

Exercise 1

Class Diagram



Exercise 2

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

// User Class.
class User {
  private :
      char Name [20] ;
      char NICno[12] ;
      char Email [30];
      char username[15] ;
      char password [8];

  public :
      User ();
      User (char Name[] ,char NICno[] ,char Email[] ,char username[] ,char password[] );
      void RegisterUser ();
      void EditUser ();
      void Login ();
      void DisplayUser ();
};

// Module Class
class Module {
  private :
      char M_code [8];
      char M_name [15];
      int NoOfCredit ;
  public :
      Module();
      Module(char M_code [],char M_name [],int NoOfCredit);
};

//Book Class
class Book{
  private :
    int Book_ID;
    char Book_Name [15];
    char Book_Type [5];

  public :
    Book();
    Book(int Book_ID,char Book_Name [],char Book_Type []);
    void DisplayBookDetails();
    void AddBookDetails();
};

//Online Library Class
class Online_Library{
    private :
      char MemberName [20];
       int MemberID ;
      int BookID ;
    public :
      Online_Library();
      Online_Library(char MemberName [],int MemberID,int BookID);
      void DisplayLibraryDetails();
};
```

```
//Coordinator Class
class Coordinator {
  private :
    char Co_ID [7] ;
    char CoName [20];
    char Copassword [8];

  public :
    Coordinator();
    Coordinator(char Co_ID [],char CoName [],char Copassword []);
    void DisplayCoDetails();
};

//Faculty Head Class
class Faculty_Head {
    private:
      char H_ID [10];
      char Husername [20];
      char Hpassword [8];

     public :
      Faculty_Head();
      Faculty_Head(char H_ID [],char Husername [],char Hpassword []);
      void DisplayFaculty_HeadDetails ();
      void RespondProblems();
};
```

```
// Lecture Materials
class Lecture_Materials{
    private :
      char LectureMaterialsType [10];

    public :
      Lecture_Materials();
      Lecture_Materials(char LectureMaterialsType []);
};

// Problem Class
class Problem {
    private :
      char ProblemType [10];
      char ProblemDescription [100];

    public :
      Problem ();
      Problem(char ProblemType [],char ProblemDescription []);
      void Displayproblem();
      void Addproblem ();

};

// Solution Class
class Solution {
    private :
      char SolutionDescription [100];
public :
      Solution();
      Solution(char SolutionDescription [100]);
      void DisplaySolution();
};

// Report Class
class Report {
    private :
      int ReportID;

    public :
      Report();
      Report(int ReportID);
      void DisplayReport();
};

// Methord Implementatoin

//User Class Implementatoin
User :: User (){}
User :: User(char Uname[] ,char UNIC[] ,char Uemail[] ,char Uusername[] ,char
Upassword[] ){
  strcpy(Name , Uname);
  strcpy(NICno , UNIC);
  strcpy(Email , Uemail);
  strcpy(username , Uusername);
  strcpy(password , Upassword);
}
```

```
void User :: RegisterUser (){}
void User :: EditUser (){}
void User :: Login (){}
void User :: DisplayUser (){
  cout << "Display User : "<< Name <<endl;
}
//Module Class Implementatoin
Module :: Module (){
   NoOfCredit = 0;
}
Module :: Module (char Mod_code [],char Mod_name [],int MNoOfCredit){
  strcpy (M_code , Mod_code);
  strcpy (M_name ,Mod_name);
  NoOfCredit = MNoOfCredit;
}
//Book Class Implementatoin
Book :: Book (){
  Book_ID = 0;
}
Book :: Book(int B_ID,char B_Name[],char B_Type []){
  Book_ID = B_ID;
  strcpy (Book_Name , B_Name);
  strcpy(Book_Type , B_Type);
}
void Book:: DisplayBookDetails(){
  cout << "Book ID : "<< Book_ID << endl;
  cout << "Book Name  : "<< Book_Name << endl;
  cout << "Book Type : "<< Book_Type << endl;
}
void Book:: AddBookDetails(){}
```

```
//Online Library Class Implementatoin
Online_Library :: Online_Library(){}
Online_Library :: Online_Library(char OLMemberName [],int OLMemberID,int OLBookID
){
  strcpy (MemberName,OLMemberName);
  MemberID = OLMemberID;
  BookID = OLBookID;
}
void Online_Library ::  DisplayLibraryDetails(){
  cout << "Member Name : "<< MemberName << endl;
  cout << "Member ID  : "<< MemberID << endl;
  cout << "Book ID : "<< BookID << endl;
}

//Coordinator Class Implementatoin
Coordinator :: Coordinator(){}
Coordinator :: Coordinator(char Coo_ID [] ,char CooName [],char Coopassword [] ){
   strcpy(Co_ID , Coo_ID);
  strcpy (CoName,CooName);
  strcpy (Copassword,Coopassword);
}
void Coordinator :: DisplayCoDetails(){
  cout << "Coordinator ID : "<< Co_ID<<endl;
  cout << "Coordinator Name : "<< CoName<<endl;
}
```

```
//Faculty Head Class Implementatoin
Faculty_Head :: Faculty_Head(){}
Faculty_Head :: Faculty_Head(char FH_ID[],char FHusername [],char FHpassword [8]){
  strcpy (H_ID , FH_ID);
  strcpy (Husername , FHusername);
  strcpy (Hpassword , FHpassword);
}
void Faculty_Head :: DisplayFaculty_HeadDetails (){
  cout << "Faculty Head ID : "<<H_ID << endl;
}
void Faculty_Head :: RespondProblems(){}


// Lecture Materials Class Implementatoin
Lecture_Materials :: Lecture_Materials(){}
Lecture_Materials :: Lecture_Materials(char MaterialsType []){
  strcpy(LectureMaterialsType , MaterialsType );
}

// Problem Class Implementatoin
Problem :: Problem (){}
Problem :: Problem (char P_Type [],char P_Description []){
  strcpy(ProblemType,P_Type);
  strcpy(ProblemDescription,P_Description);
}
void Problem :: Displayproblem(){
  cout<<"Display Problem Type  : "<< ProblemType <<endl;
  cout << "Display problem : "<<ProblemDescription << endl;
}
void Problem :: Addproblem (){}
```

```
// Solution Class Implementatoin
Solution :: Solution(){}
Solution :: Solution(char S_Description []){
  strcpy (SolutionDescription ,S_Description);
}
void Solution :: DisplaySolution(){
  cout<<"Display Solution : "<< SolutionDescription << endl;
}


// Report Class Implementatoin
Report :: Report(){
  ReportID =0;
}
Report :: Report(int R_ID){
  ReportID = R_ID;
}
void Report ::  DisplayReport(){
  cout << "Display Report : "<<ReportID<<endl;
}
```

```
// MAIN PROGRAM.
int main() {
  User u1("Andrews Billy","992609796V","andrews@gmail.com","Andrews","A#567897");
  Module m1("BM3011","Marketing Strategy and Management",4);
  Book b1(1001,"Clean Code","Software");
  Online_Library OL1("Andrews",0012,1002);
  Coordinator co1("CO10001","Dr Ann","Ann@#565");
  Faculty_Head fh1("HD10345689","RayJones","RJ@#7854");
  Lecture_Materials lm1("PDF");
  Problem p1("Exam Excuses","unable to attend to the Exma...");
  Solution s1("You should e-mail the reason for absence to the examination to the relevant
module coordinator.");
  Report r1(00015);


  //cout << "Hello World!\n";
  return 0;
}
```

Individual Contributions

IT21377594 – Keshala G.P.

- Created and documented the user requirements and noun verb analysis.
- Created the CRC Card for User, Module and report classes.
- Created the class diagram for User, Module and report.
- Implemented the coding for the User class, Module class, and Report class.

IT21182532 – Lakshan G.N.

- Created the CRC Card for Books, Online library, Problems and Solution classes.
- Drawing of class diagram for Books, Online library, Problems and Solution with relationships to other classes.
- Implemented the coding for Books class, online library class, Problems class and solution class.

IT21307126 - Nawarathne N.S.N.

- Designed the CRC Cards for the Coordinator class.
- Created the class diagram for Coordinator.
- Implemented the coding for Coordinator class.

IT21379574 - Dilshani H.T.D.P.

- Designed the CRC Cards for the Faculty Head class.
- I have drawn the class diagram for Faculty Head class.
- I have implemented the codes for the Faculty Head.

IT21189180 - Janagan K.

- Created the class diagram for lecture materials.
- Created the CRC card for lecture materials.
- Create the classes and code the lecture materials.