BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

Year 1, Semester 2 **2024- Feb**

Topic : Online Land Sales Management System

Group no : KNDUNI_04

Campus : Kandy Uni

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
IT23255142	HANA .S.M.F	0778789282
IT23151710	AAZAF RITHA .J	0760737173
IT23262768	RESHMA .M.R.F	0769065052
IT23187078	AMAN MOHAMED .M.A	0744050889
IT23238480	AYMAN .M.R	0770847864

BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

Year 1, Semester 2 **2024- Feb**

Content

BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

Year 1, Semester 2 **2024- Feb**

01. Description of the requirements.

1. There are four types of users.
2. Those are admin, agent, financial officer, and users (Registered or Unregistered).
3. All the users can view the websites, check availability of the lands, check price ranges, check the FAQ sections, make buy requests, make requests for sell the lands and, contact with the admin.
4. Any guest can register for the website by providing a first name, last name, address, password, date of birth, phone number, and email address.
5. After registering, the system registered users can login using their email address and password.
6. Registered users can check land packages and make requests to buy the lands.
7. Registered users can make requests to sell land and contact with agents.
8. Registered users can give feedback about the website.
9. Registered users can view, delete, or edit their profiles.
10. Registered users can view, delete, or edit their land sell requests.
11. Agent can login with their email address and password.
12. Agents can generate progress and customer service reports.
13. Agents can contact the seller and check the given land details.
14. If given details verified by agents can request to admin to approve the land sell proposal.

15. Agents can contact buyer and negotiate with them and sell the land.
16. Admins can login with their email address and password.
17. Admin can approve or decline the user land sell requests.
18. The admin can activate and delete the user accounts.
19. The admin can edit, delete, and view the land details on database.
20. The admin can activate and delete the employee accounts.
21. The admin can activate and delete the financial officers' accounts.
22. Admin can approve or decline transactions contact with financial officers.
23. Financial officers can generate finance reports.
24. Financial officers should communicate with buyers and decide payment methods.

02. Classes Identified

- User
- Registered User
- Agent
- Admin
- Financial Officer
- Land
- Profile
- Transaction
- Report
- Feedback

03. CRC Card

Class: Agent	
Responsibilities	**Collaborations**
Login to account	
Approve Requests	
Verify land details	
Requests for land sale approval	Admin
Contacts buyer	Registered User
Contacts seller	Registered User
Generate reports	Reports

Class: Inquiry	
Responsibilities	**Collaborations**
Show details about inquiry	
Show inquiry	

Class: User	
Responsibilities	Collaborations
Register to account	
Search Land ads	Land
Contact websites	
Check FAQs	
Check feedbacks	Feedback
View about us	
Bookmark land interested	Land

Class: Land	
Responsibilities	Collaborations
Stores land details	
Manage land details	

Class: Registered User	
Responsibilities	Collaborations
Login to the system	
Send inquiry	Admin
Post Land Ad	
Enter land details	Land
Edit land details	Land
Request for property buying	Agent
Request for property selling	Agent
Cancel request	
Post feedback	Feedback

Class: Feedback	
Responsibilities	Collaborations
Details about feedback	
Approved feedback	Admin

Class: Report	
Responsibility	**Collaborators**
Generate customer service reports	Agent
Generate finance reports	Financial Officer
Generate progress reports	Agent

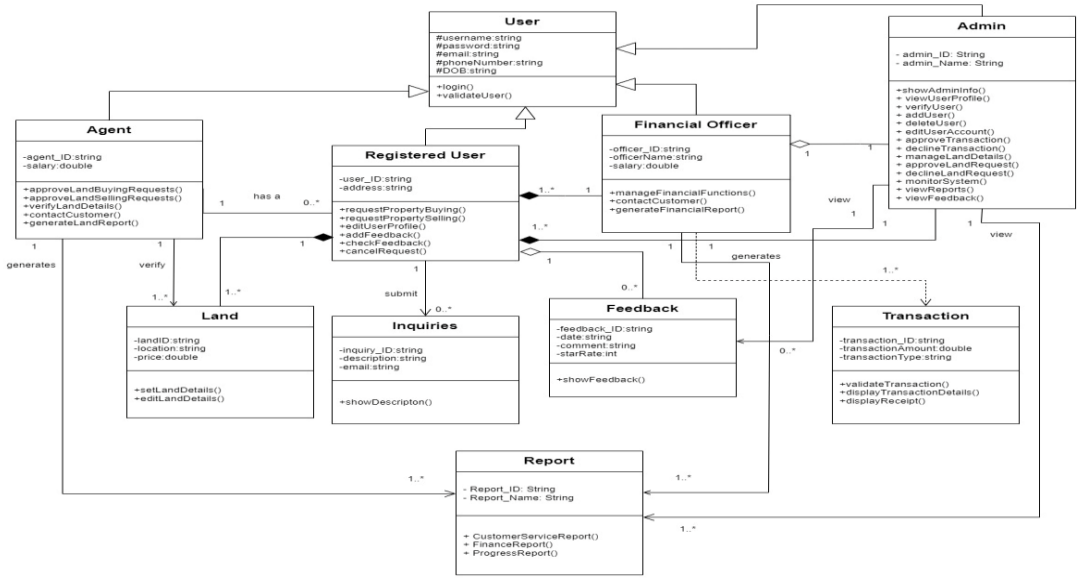
Class: Admin	
Responsibility	**Collaborators**
Login	
View user profiles and verify them	Registered User, Financial Officer, Agent
Add/ delete and edit users accounts	Registered User, Financial Officer, Agent
Reply inquiry	Registered User
Approve or decline transactions	Financial Officer, Transaction
Manage land details	
Approve or decline land requests	Registered User, Agent

Monitor system	
View reports	Reports
View feedback	Feedback

Class: Financial Officer	
Responsibilities	Collaborations
Manage risk	
Manage expense	
Manage treasury	
Generate financial reports	Reports
Contacts buyer	Registered User
Contacts Seller	Registered User

Class: Transaction	
Responsibilities	Collaborations
Store land sales transaction details	
Store land purchase transaction details	
Maintain transaction history	
Validate transaction data	Financial Officer

04.Class diagram



05.Coding for the classes

1.Main.cpp

...\Desktop\ooc\OOC Assignmment\OOC Assignmment\Main.cpp 1

```
#include <iostream>
#include "Registered_User.h"
#include "Agent.h"
#include "Financial_Officer.h"
#include "Admin.h"
#include "Feedback.h"
#include "Inquiry.h"
#include "Transaction.h"
#include "Land.h"

int main()
{
    //data insert to Registered User
    RegisteredUser*regUser= new RegisteredUser
      ("Reshma","hello","reshma@gmail.com","0767856924","01.01.2001","U0001
      ","123,Kandy rd,Kandy");
    cout<<"Details of User:"<< endl;
    regUser->display();
    cout<<endl;

    //data insert to Agent
    Agent*Ag= new Agent
      ("Hana","Blue","hana@gamil.com","0778596124","05.06.2007","A003","100
      0000");
    cout<<"Details of Agent"<< endl;
    Agent->display();
    cout<<endl;

    //data insert to Financial Officer
    financialOfficer*fin= new financialOfficer
      ("Ayman","1234","ayman@gmail.com","06.07.2002","O005","89000");
    cout<<"Details of Financial Officer"<< endl;
    financialOfficer->display();
    cout<<endl;

    //data insert to  Admin
    Admin* admin = new Admin
      ("Aazaf01","4536","aazaf@gmail.com","0752384956","07.05.2001","R003",
      "Aazaf");
    cout << "Details of Admin: " << endl;
    admin->showAdminInfo();
    cout << endl;

    //data insert to  Feedback
    Feedback*fb= new FeedbacK("F0001","06.08.2024","Nice Location","4");
    cout << "Details of Feedback:" << endl;
    admin->showFeedback();
    cout << endl;

```

```
...\Desktop\ooc\OOC Assignmment\OOC Assignmment\Main.cpp                    2
43      //data insert to Inquiry
44      Inquiry*inquire= new Inquiry("I001", "How can I purchase a Land?",
          "123user@gmail.com");
45      cout<<"Details of Inquiry:"<< endl;
46      inquire->display();
47      cout<<endl;
48
49      //data insert to Transaction
50      Transaction*transact= new Transaction("T001","1200000","Online
          Transaction");
51      cout<<"Details of Transaction:"<< endl;
52      transact->display();
53      cout<<endl;
54
55      //data insert to Land
56      Land*land= new land("L001","49/A, King Street, Matale");
57      cout<<"Details of Land:"<< endl;
58      land->display();
59      cout<<endl;
60
61      return 0;
62
63  }
64
65
66
```

2.User.cpp and User.h

...esktop\ooc\OOC Assignmment\OOC Assignmment\User.h.cpp 1

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

class User
{
    protected:
        char username [20];  // Username of the user
        char password [12];  // password of the user
        char email [50];       // E-mail of the user
        char phoneNumber [12];   // Phone Number of the user
        char DOB [10];     // Date of Birth of the user

    public:
        User(); // Default constructor
        User(char u_name, char pass_word, char e_mail, char phone_Number,
          char D_O_B);   // Parameterized constructor
        void displayDetails();
        void login();
        void validateUser();
        ~User(); // Destructor
};

User::User()  // Default constructor
{
    strcpy(username,"");
    strcpy(password,"");
    strcpy(email,"");
    strcpy(phoneNumber,"");
    strcpy(DOB,"");
}

User::User(char u_name, char pass_word, char e_mail, char phone_Number,
  char D_O_B)  // Parameterized constructor
{
    strcpy(userName,u_name);
    strcpy(password,pass_word)
    strcpy(email,e_mail)
    strcpy(phoneNumber,phone_Number)
    strcpy(DOB,D_O_B)
}

void User::displayDetails()
{
    cout<<username<<endl<<
    password<<endl<<
    email<<endl<<
    phoneNumber<<endl<<
    DOB<<endl;
}
```

...esktop\ooc\OOC Assignmment\OOC Assignmment\User.h.cpp 2

```
48
49  void User::login()
50  {
51
52  }
53
54  void User::validateUser()
55  {
56
57  }
58
59  User::~User()   // Destructor
60  {
61
62  }
63
64
65
66
67
```

3.Registered User.h

...ign 2\Registered user\OOC assignment\RegisteredUser.h 1

```
1  #pragma once
2  #include "admin.h"
3  #include "land.h"
4  #include "agent.h"
5  #include "Feedback.h"
6  #include "User.h"
7  #define SIZE 2
8
9  class RegisteredUser : public User {
10 private:
11     char user_ID[10];
12     char address[100];
13     Feedback* feedback[SIZE];
14
15 public:
16     RegisteredUser(); // default constructor
17     RegisteredUser(char ru_username[], char ru_password[], char ru_email[],
          char ru_phoneNumber[],
18         char ru_DOB[], char ru_id[], char ru_address[]);
19     // overloaded constructor
20
21     void display();
22     void addFeedback();
23     void editUserProfile();
24     void requestPropertyBuying();
25     void requestPropertySelling();
26     void cancelRequest();
27
28     void addFeed(char Fb_ID[], char Fb_Date[], char Fb_comment[], int
         Fb_starRate);
29     void displayFeedback();
30
31     ~RegisteredUser(); // destructor
32 };
33
```

4.Registered user.cpp

...n 2\Registered user\OOC assignment\RegisteredUser.cpp 1

```cpp
#include "RegisteredUser.h"
#include <iostream>
#include <cstring>

using namespace std;

// Default constructor
RegisteredUser::RegisteredUser() {
    strcpy(user_ID, "");
    strcpy(address, "");
    for (int i = 0; i < SIZE; ++i) {
        feedback[i] = nullptr;
    }
}

// Overloaded constructor
RegisteredUser::RegisteredUser(char ru_username[], char ru_password[], char ru_email[],
    char ru_phoneNumber[], char ru_DOB[], char ru_id[], char ru_address[])
    : User(ru_username, ru_password, ru_email, ru_phoneNumber, ru_DOB) {
    strcpy(user_ID, ru_id);
    strcpy(address, ru_address);
    for (int i = 0; i < SIZE; ++i) {
        feedback[i] = nullptr;
    }
}

void RegisteredUser::display() {
    User::display();
    cout << "User ID: " << user_ID << endl;
    cout << "Address: " << address << endl;
}

void RegisteredUser::addFeedback() {
    // Implement feedback addition here
}

void RegisteredUser::editUserProfile() {
    // Implement user profile editing here
}

void RegisteredUser::requestPropertyBuying() {
    // Implement property buying request here
}

void RegisteredUser::requestPropertySelling() {
    // Implement property selling request here
}

```

...n 2\Registered user\OOC assignment\RegisteredUser.cpp 2

```
49 void RegisteredUser::cancelRequest() {
50     // Implement request cancellation here
51 }
52
53 void RegisteredUser::addFeed(char Fb_ID[], char Fb_Date[], char Fb_comment
     [], int Fb_starRate) {
54     feedback[0] = new Feedback(Fb_ID, Fb_Date, Fb_comment, Fb_starRate);
55 }
56
57 void RegisteredUser::displayFeedback() {
58     if (feedback[0] != nullptr) {
59         feedback[0]->showFeedback();
60     }
61     else {
62         cout << "No feedback available" << endl;
63     }
64 }
65
66 RegisteredUser::~RegisteredUser() {
67     for (int i = 0; i < SIZE; ++i) {
68         if (feedback[i] != nullptr) {
69             delete feedback[i];
70         }
71     }
72 }
73
```

5.Admin.cpp

...Desktop\ooc\OOC Assignmment\OOC Assignmment\Admin.cpp 1

```cpp
// Admin.cpp
#include "Admin.h"
#include <iostream>
#include <cstring>  // For strcpy
using namespace std;

// Default constructor
Admin::Admin() {
    strcpy(Admin_ID, ""); // Initialize with empty string
    strcpy(Admin_Name, ""); // Initialize with empty string
}

// Parameterized constructor
Admin::Admin(char a_username[], char a_password[],char a_email[], char a_phoneNum[],char a_DOB[],char id[],char name[])
    :User(a_username,a_password,a_email,a_phoneNum,a_DOB){
    strcpy(Admin_ID, id);
    strcpy(Admin_Name, name);
}

// Methods for Admin class functionalities

void Admin::showAdminInfo() {
    cout << "Admin ID: " << Admin_ID << endl;
    cout << "Admin Name: " << Admin_Name << endl;
}
void Admin::viewUserProfile() {

}

void Admin::verifyUser() {

}

void Admin::addUser() {

}

void Admin::deleteUser() {

}

void Admin::editUserAccount() {

}

void Admin::approveTransaction() {

}
```

...Desktop\ooc\OOC Assignmment\OOC Assignmment\Admin.cpp 2

```
49
50 void Admin::declineTransaction() {
51
52 }
53
54 void Admin::manageLandDetails() {
55
56 }
57
58 void Admin::approveLandRequest() {
59
60 }
61
62 void Admin::declineLandRequest() {
63
64 }
65
66 void Admin::monitorSystem() {
67
68 }
69
70 void Admin::viewReports() {
71
72 }
73
74 void Admin::viewFeedback() {
75
76 }
77
78
79
80 // Destructor
81 Admin::~Admin() {
82
83 }
84
```

6.Admin.h

...d\Desktop\ooc\OOC Assignmment\OOC Assignmment\Admin.h 1

```
1  // Admin.h
2  #pragma once
3  #include <string>
4  #include "RegisteredUser.h"
5  #include "Transaction.h"
6  #include "financial officer.h"
7  #include "Agent.h"
8  #include "Report.h"
9  #include "Feedback.h"
10
11
12
13 class Admin:public User {
14 private:
15     char Admin_ID[10];  // Identifier for the admin
16     char Admin_Name[100];  // Admin's name
17     char Admin_Email[100];  // Admin's email
18
19 public:
20     Admin();  // Default constructor
21     Admin(char a_username[], char a_password[],char a_email[], char
         a_phoneNum[],char a_DOB[],char id[],char name[]);  // Parameterized
         constructor
22
23     void showAdminInfo();  // Method to display admin information
24     void viewUserProfile();
25     void verifyUser();
26     void addUser();
27     void deleteUser();
28     void editUserAccount();
29     void approveTransaction();
30     void declineTransaction();
31     void manageLandDetails();
32     void approveLandRequest();
33     void declineLandRequest();
34     void monitorSystem();
35     void viewReports();
36     void viewFeedback();
37
38
39
40     ~Admin();  // Destructor
41 };
42
```

7.Agent.cpp

...Desktop\ooc\OOC Assignmment\OOC Assignmment\Agent.cpp 1

```cpp
#include <iostream>
#include<cstring>
#include "Agent.h"
#include "RegisteredUser.h"
#include "Reports.h"
#include "land.h"

using namespace std;
// Default constructor
Agent::Agent() {
    strcpy(agent_ID, "");
    salary = 0.0;
}
//Overloaded Constructor implementation
Agent::Agent(char ag_username[], char ag_password[],char ag_email[], char ag_phoneNum[],char ag_DOB[], char ag_ID[], double ag_salary)
    :User(ag_username, ag_password,ag_email,ag_phoneNum,ag_DOB) {
    strcpy(agent_ID, ag_ID);
    salary=ag_salary
}
//Display Agent Details
void Agent::display() {
    User::display();
    cout << "Agent ID:"<<agent_ID << endl;
    cout<<"Salary:"<<salary<<endl;
}
void Agent::approveLandBuyingRequests() {

}
void Agent::approveLandSellingRequests() {

}
void Agent::verifyLandDetails() {

}
void Agent::contactCustomer() {

}
void Agent::generateLandReport() {

}
// Destructor
Agent::~Agent() {

}

```

8.Agent.h

```
...d\Desktop\ooc\OOC Assignmment\OOC Assignmment\Admin.h                    1
 1  // Admin.h
 2  #pragma once
 3  #include <string>
 4  #include "RegisteredUser.h"
 5  #include "Transaction.h"
 6  #include "financial officer.h"
 7  #include "Agent.h"
 8  #include "Report.h"
 9  #include "Feedback.h"
10
11
12
13  class Admin:public User {
14  private:
15      char Admin_ID[10];  // Identifier for the admin
16      char Admin_Name[100];  // Admin's name
17      char Admin_Email[100];  // Admin's email
18
19  public:
20      Admin();  // Default constructor
21      Admin(char a_username[], char a_password[],char a_email[], char
          a_phoneNum[],char a_DOB[],char id[],char name[]);  // Parameterized
          constructor
22
23      void showAdminInfo();  // Method to display admin information
24      void viewUserProfile();
25      void verifyUser();
26      void addUser();
27      void deleteUser();
28      void editUserAccount();
29      void approveTransaction();
30      void declineTransaction();
31      void manageLandDetails();
32      void approveLandRequest();
33      void declineLandRequest();
34      void monitorSystem();
35      void viewReports();
36      void viewFeedback();
37
38
39
40      ~Admin();  // Destructor
41  };
42
```

9.Transection.h

...top\ooc\OOC Assignmment\OOC Assignmment\transaction.h 1

```
1  #pragma once
2
3  class Transaction {
4  private:
5      char transaction_ID[15];
6      char transactionAmount[15];
7      char transactionType[250];
8
9
10 public:
11     Transaction();
12     Transaction(char Fb_ID[], char Fb_Date[], char Fb_comment[], int
         Fb_starRate);
13
14     void showTransaction();
15     ~Transaction(); // Destructor
16 };
17
```

7.Transection.cpp

...oc\OOC Assignmment\OOC Assignmment\transaction[1].cpp 1

```cpp
#include "Transaction.h"
#include "financialOfficer.h"
#include <iostream>
#include <cstring>



using namespace std;

Transaction::Transaction() {
    strcpy(transaction_ID, "");
    strcpy(transactionAmount, "");
    strcpy(transactionType, "");
}

Transaction::Transaction(char tr_id[],char amnt[],char tr_type[]) {
    strcpy(transaction_ID, tr_id);
    strcpy(transactionAmount, tr_amnt);
    strcpy(transactionType, tr_type);
}

void Transaction::showTransaction() {
    cout << "Transaction Amount: " << transactionAmount << endl;
    cout << "Transaction Type: " << transactionType << endl;
}

Transaction::~Transaction() {
    // Destructor
}

```

8.Land.h and cpp file

...esktop\ooc\OOC Assignmment\OOC Assignmment\land.h.cpp 1

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

class Land
{
    private:
        char land_ID [50];    //to represent land ID
        char location [50];  //array to store location
        double price;       // Price of the land

    public:
        Land();  // Default constructor
        Land(char pland_ID, char plocation, double pprice);   //
          Parameterized constructor
        void display();   // function
        void setLandDetails();  // Function
        void getLandDetails(); // function
        ~Land();     // Destructor
};

Land::Land()  // Default constructor
{
    strcpy(land_ID,"");
    strcpy(location,"");
    price=0;
}

Land::Land(char pland_ID, char plocation, double pprice)   // Parameterized
   constructor
{
    strcpy(land_ID,"pland_ID");
    strcpy(location,"plocation");
    price=pprice;
}

void Land::display()
{
    cout<<land_ID<<endl<<
    location<<endl<<
    price<<endl;
}

void Land::setLandDetails()
{

}

Land::~Land()    // Destructor
```

```
...esktop\ooc\OOC Assignmment\OOC Assignmment\land.h.cpp                    2
48  {
49
50  }
51
52
53
```

9.Inquiry.h

...Desktop\ooc\OOC Assignmment\OOC Assignmment\Inquiry.h 1

```
1  #pragma once
2  class Inquiry
3  {
4  private:
5      char inquiry_ID[10];
6      char description[500];
7      char email[100];
8  public:
9      Inquiry();//Default Constructor
10     Inquiry(char i_ID[], char i_description[], char i_email[]);
11
12     void display();//Display Inquiry
13     void showDescription();
14
15     ~Inquiry();//Destructor
16 };
17
18
```

10.Inquiry.cpp

...sktop\ooc\OOC Assignmment\OOC Assignmment\Inquiry.cpp 1

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

//Default Constructor Implementation
Inquiry::Inquiry() {
    strcpy(inquiry_ID, "");
    strcpy(description, "");
    strcpy(email, "");
}

//Overloaded Constructor Implementation
Inquiry::Inquiry(char i_ID[], char i_description[], char i_email[]) {
    strcpy(inquiry_ID, i_ID);
    strcpy(description, i_description);
    strcpy(email,i_email);
}
void Inquiry::display() {
    cout << "Inquiry ID:" << inquiry_ID << endl;
    cout << "Email:" << email << endl;
}
void Inquiry::showDescription() {
    cout << "Description" << description << endl;
}
Inquiry::~Inquiry() {

}
```

11.Feedback.h

...med\Desktop\OOC assign 2\Feedback\Feedback\Feedback.h 1

```
1  #pragma once
2
3  class Feedback {
4  private:
5      char feedback_ID[15];
6      char date[15];
7      char comment[250];
8      int starRate;
9
10 public:
11     Feedback();
12     Feedback(char Fb_ID[], char Fb_Date[], char Fb_comment[], int
         Fb_starRate);
13
14     void showFeedback();
15     ~Feedback(); // Destructor
16 };
17
```

12.Feedback.cpp

...d\Desktop\OOC assign 2\Feedback\Feedback\Feedback.cpp 1

```cpp
#include "Feedback.h"
#include <iostream>
#include <cstring>

using namespace std;

Feedback::Feedback() {
    strcpy(feedback_ID, "");
    strcpy(date, "");
    strcpy(comment, "");
    starRate = 0;
}

Feedback::Feedback(char Fb_ID[], char Fb_Date[], char Fb_comment[], int Fb_starRate) {
    strcpy(feedback_ID, Fb_ID);
    strcpy(date, Fb_Date);
    strcpy(comment, Fb_comment);
    starRate = Fb_starRate;
}

void Feedback::showFeedback() {
    cout << "Date: " << date << endl;
    cout << "Comment: " << comment << endl;
    cout << "Star Rate: " << starRate << endl;
}

Feedback::~Feedback() {
    // Destructor
}

```

13.report.cpp

...esktop\ooc\OOC Assignmment\OOC Assignmment\Report.cpp 1

```cpp
// Report.cpp
#include "Report.h"
#include <iostream>
#include <cstring>  // For strcpy

// Default constructor
Report::Report() {
    strcpy(Report_ID, ""); // Initialize with empty string
    strcpy(Report_Name, ""); // Initialize with empty string
}

// Parameterized constructor
Report::Report(const char id[10], const char name[100]) {
    strcpy(Report_ID, id);
    strcpy(Report_Name, name);
}

// Methods
void Report::CustomerServiceReport() {

}

void Report::FinanceReport() {

}

void Report::ProgressReport() {

}

// Destructor
Report::~Report() {

}

```

14.report.h

...\Desktop\ooc\OOC Assignmment\OOC Assignmment\Report.h 1

```
1  // Report.h
2  #pragma once
3  #include <string>
4  #include "financial officer.h"
5  #include "Agent.h"
6
7
8  class Report {
9  private:
10     char Report_ID[10];      // Identifier for the report
11     char Report_Name[100];   // Name of the report as a character array
12
13
14 public:
15     Report();
16     Report(const char id[10], const char name[100]);
17     void CustomerServiceReport();
18     void FinanceReport();
19     void ProgressReport();
20     ~Report();               // Destructor
21 };
22
```

15. financial officer.h

...c\OOC Assignmment\OOC Assignmment\financial officer.h 1

```
1  #pragma once
2  #include "reports.h"
3  #include "financialOfficer.h"
4  #define SIZE 2
5
6  class financialOfficer : public User  // Derived from User
7  {
8  private:
9      char officer_ID[10];
10     char officerName[100];
11     reports* reports[SIZE];
12
13 public:
14     financialOfficer(); // default constructor
15     financialOfficer(char fin_username[], char fin_pass[], char fin_id[],
         char fin_email[], char fin_address[]);
16     //overloaded constructor
17
18     void display();
19     void manageFinancialFunctions();
20     void contactCustomer();
21     void generateFinancialReport();
22
23     void generateFinancialReport(char rep_Date[],char rep_Progress[],char
         rep_Finance[],char rep_cusSer,char rep_content);
24     void displayReports();
25
26     ~financialOfficer(); // destructor
27 };
28
```

16.Financial officer.cpp

...OOC Assignmment\OOC Assignmment\financial officer.cpp 1

```cpp
#include "reports.h"
#include "financialOfficer.h"
#include <iostream>
#include <cstring>


using namespace std;

// Default constructor
financialOfficer::financialOfficer() {
    strcpy(officer_ID, "");
    strcpy(officerName, "");
    for (int i = 0; i < SIZE; ++i) {
        reports[i] = nullptr;
    }
}

// Overloaded constructor
financialOfficer::financialOfficer(char fin_username[], char fin_pass[],
  char fin_id[], char fin_email[], char fin_address[])
    : User(fin_username, fin_pass) {
    strcpy(officer_ID_ID, fin_id);
    strcpy(address, ru_address);
    for (int i = 0; i < SIZE; ++i) {
        reports[i] = nullptr;
    }
}

void financialOfficer::display() {
    User::display();
    cout << officer_ID << endl;
    cout << officerName << endl;
}

void financialOfficer::manageFinancialFunctions() {
    // Manage financial functions here
}

void financialOfficer::contactCustomer() {
    // Contact buyer and seller
}

void financialOfficer::generateFinancialReport() {
    // Generate Financial reports
}


void financialOfficer::generateFinancialReport(char rep_Date[],char
  rep_Progress[],char rep_Finance[],char rep_cusSer,char rep_content) {
```

```
...OOC Assignmment\OOC Assignmment\financial officer.cpp                    2
48      reports[0] = new Reports(rep_Date, rep_Progress, rep_Finance,
          rep_cusSer,rep_content);
49  }
50
51  void financialOfficer::displayReports() {
52      if (reports[0] != nullptr) {
53          reports[0]->showReports();
54      }
55      else {
56          cout << "No Reports available" << endl;
57      }
58  }
59
60  financialOfficer::~financialOfficer() {
61      for (int i = 0; i < SIZE; ++i) {
62          if (Reports[i] != nullptr) {
63              delete Reports[i];
64          }
65      }
66  }
67
```

6.Individual Contributions

IT23255142: HANA S.M.F

❖ Individual contributions

*** Created Agent and Inquiry CRC cards**

*** Worked Agent and Inquiry CRC Diagram**

*** Worked on creating Agent.h, Agent.cpp and Inquiry.h, Inquiry.cpp**

IT23262768: Reshma M.R.F

❖ Individual contributions

*** Created User and Land CRC cards**

*** Worked User and Land CRC Diagram**

*** Worked on creating User.h, User.cpp and Land.h, Land.cpp**

IT23151710: Aazaf Ritha J

❖ Individual contributions

*** Created Admin and Report CRC cards**

*** Worked Admin and Report CRC Diagram**

*** Worked on creating Admin.cpp , Admin.h hand Report.h, Report.cpp**

SLIIT
Discover Your Future

BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

Year 1, Semester 2 **2024- Feb**

IT23187078: Aman Mohamed M. A

❖ Individual contributions

*** Created Registered User and Feedback CRC cards**

*** Worked Registered User and Feedback CRC Diagram**

*** Worked on creating RegisteredUser.h, RegisteredUser.cpp and Feedback.h, Feedback.cpp**

IT23238480: Ayman M R

❖ Individual contributions

*** Created Financial Officer and Transaction CRC cards**

*** Worked Financial Officer and Transaction CRC Diagram**

*** Worked on creating Financial Officer.cpp , Financial Officer.h hand Transaction.h, Transaction.cpp**