Topic : Online Land Sales System

Group no : MLB_PG.03.02_09

Campus : Malabe

Submission Date : 18.10.2020

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

Exercise 1

System Requirements for Online Land Sales System

1. All the users can search properties sorting by Lands, Apartments, and Houses.
2. Any visitor can register as a member.
3. All the members can login to the website by providing email and password.
4. Once user registered, he/she can view his profile, delete profile, or edit profile details.
5. A member can request for a tour to a selected property.
6. A member can buy or rent a property categorizing in Lands, Apartments or Houses.
7. If a member needs to sell or rent a property first, they must fill the request form.
8. Once a member filled the form, they can add their own properties to the website for rent or sell in preferred category.
9. A member can make payments online through our website via using credit cards, online money transfer or EFT.
10. Admin also needs to login to the system before do the activities.
11. Admin can generate reports such as available property reports, cash flow of the company.
12. Admin can view requests for new properties and decide whether it accepts or not.
13. After a member request for a tour admin checks available date slots and send an email to the member through the system.
14. Admin also can add or remove properties own by the company.

Noun & Verb Analysis

- Nouns in Red color
- Verbs in Blue color

All the users can search properties sorting by Lands, Apartments, and Houses.

Any visitor can register as a member.

All the members can login to the website by providing email and password.

Once user registered, they can view their profile, delete profile, or edit profile details.

A member can request for a tour to a selected property.

A member can buy or rent a property categorizing in Lands, Apartments or Houses.

If a member needs to sell or rent a property first, they must fill the request form.

Once a member filled the form, they can add their own properties to the website for rent or sell in preferred category.

A member can make payments online through our website via using credit cards, online money transfer or EFT.

Admin also needs to login to the system before do the activities.

Admin can generate reports such as available property reports, cash flow of the company.

Admin can view requests for new properties and decide whether it accepts or not.

After a member request for a tour, admin checks available date slots and send an email to the member through the system.

Admin also can add or remove properties own by the company.

Identified Classes using Noun Verb Analysis

Identified Classes:

- Member
- Admin
- Property
- Tour
- Report
- Payment

Nouns

- User
- Property
- Land
- Apartment
- House
- Visitor
- Member
- Website
- Email
- Password
- Tour
- Form
- Credit card
- Online transfer
- EFT
- System
- Reports
- Admin
- Cashflow
- Company
- Date slots
- Payment

- **Reasons for Rejecting Other Nouns**

1. Redundant:
    - In a land sales system user, visitor, member refers to the same person as “Member”.

2. Outside scope of system:
    - Website, company, system, user profile, request form are outside scope of the land sale system.

3. Meta - Language:
    - In this system lands, apartments, houses can call as “properties”.

4. An attribute:
    - Email
    - Password
    - Credit card
    - EFT
    - Money transfer
    - Date slots

Exercise 2

CRC Cards for Online Land Sales System

Member	
Responsibilities:	**Collaborations:**
Login to the system	
Search Property	Property
View profile	
Delete profile	
Edit profile details	
Request for a tour	Admin, tour
Apply for a loan	Admin
Make payment	Payment
Request to add property	Admin
Rent or buy property	Property

Admin	
Responsibilities:	**Collaborations:**
Login to the system	
Accept or decline the request Property	Member
Add or remove property	Property
Update tour schedule	Tour
Generate report	Report

Property	
Responsibilities:	**Collaborations:**
Display property details	
Update property details	Admin

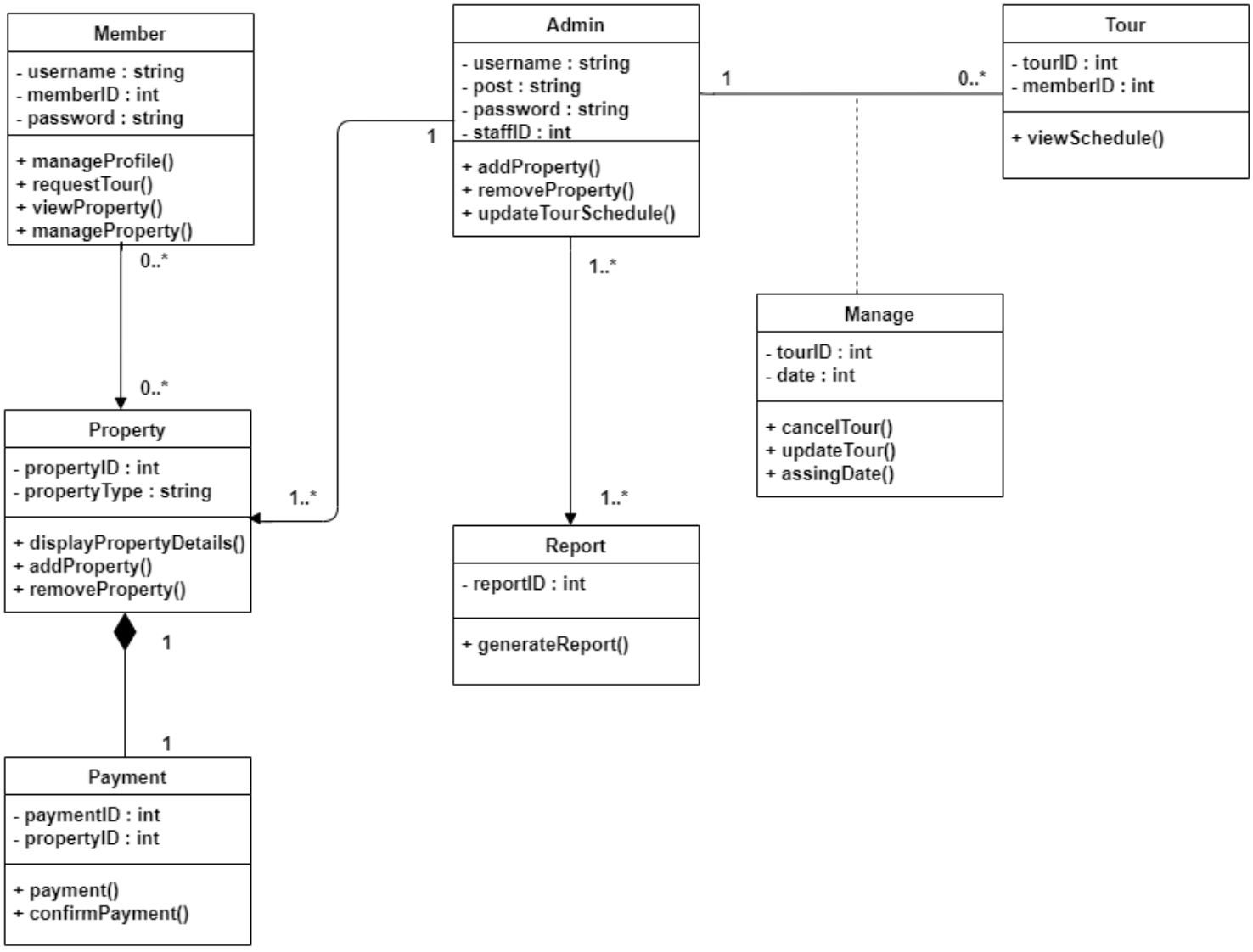
Payment	
Responsibilities:	**Collaborations:**
View payment details	Member
Validate the payment	
Save payment details	

Tour	
Responsibilities:	**Collaborations:**
Store details about available time slots	
Store member details about who requested for tour	Member

Report	
Responsibilities:	**Collaborations:**
List of available properties	Property
List of sold out properties	Property
Cash flow	Payment

Exercise 3

Class Diagram for the Online Land Sales System



Exercise 4

Coding for the Classes in Class Diagram

```
//Composition relationship between Property Class and Payment Class
//When the Property Class deleted, Payment Class will be deleted

class Payment //Payment class  //Part class for Property Class
{
  private :

      int paymentID;
      int propertyID;

  public :

      Payment();
      Payment(int ppaymentID, int ppropertyID);
      void confirmPayment();
      ~Payment();
};

Payment::Payment() {

}

Payment::Payment(int ppaymentID, int ppropertyID) {

    paymentID = ppaymentID;
    propertyID = ppropertyID;
}

void Payment::confirmPayment() {

}

Payment::~Payment() {

}
```

```
class Property  //Property class //Whole class for Payment Class
{
  private :

       string propertyType;
       int propertyID;
       Payment *pay1;
       int payID;


  public :

       Property();
       Property(string ppropertyType, int pID);
       Property(int payID, int pID);
       void displayPropertyDetails();
       void addProperty();
       void removeProperty();
       ~Property();
};

Property::Property() {

}

Property::Property(string ppropertyType, int pID) {

    propertyType=ppropertyType;
    propertyID=pID;

     //Call the constructor "Payment"

    pay1 = new Payment(payID,pID);  //payID=paymentID & pID=propertyID

}


void Property::displayPropertyDetails() {

}

void Property::addProperty() {

}

void Property::removeProperty() {

}
```

```
Property::~Property() {

 }

//Uni-directional association between Member Class and the Property Class

class Member  //Member Class
{
  private :

      string username;
      string password;
      int memberID;
      Property *prpty; //An object of Property as attribute

  public:

      Member();
      Member(string pUsername, string pPassword, int ID);
      Member(string pUsername, string pPassword, int ID, Property *p);
      void manageProfile();
      void requestTour();
      void viewProperty();
      void manageProperty();
      ~Member();
};

Member::Member() {

}

Member::Member(string pUsername, string pPassword, int ID, Property *p) {

    username = pUsername;
    password = pPassword;
    memberID=ID;
    prpty = p;
}

void Member::manageProfile() {

}

void Member::requestTour() {

}
```

```
void viewProperty() {

}

void manageProperty() {

 }

Member::~Member() {

}


class Report //Report Class
{
  private :

      int reportId;

  public :

      Report();
      Report(int pReportId);
      void generateReport();
      ~Report();
};

Report::Report() {

}

Report::Report(int pReportId) {

    reportId=pReportId;
}

void Report::generateReport() {

}

Report::~Report() {

}
```

```
class Tour //Tour Class
{
  private :

      int tourId;
      int memberId;

  public :

      Tour();
      Tour (int tID,int pMemberId);
      void viewSchedule();
      ~Tour();
};

Tour::Tour() {

}

Tour::Tour(int tID,int pMemberId) {

    tourId=tID;
    memberId=pMemberId;
}

void Tour::viewSchedule() {

}

Tour::~Tour() {

}

//Uni-directional association between Admin Class and the Property Class
//Uni-directional association between Admin Class and the Report Class
//Manage Class is the Association class between Admin Class and the Tour Class


class Admin //Admin Class
{
  private :

      string userName;
      string  post;
      string  password;
      int staffId;
      Report*Rpt; //An object of Report as attribute
      Property *Ppt; //An object of Property as attribute
```

```
public:

      Admin();
      Admin(string pUserName,string pPost,int pStaffId,string pPassword,Report
*R, Property *P);
      Admin(string pUserName,string pPost,int pStaffId,string pPassword);
      void addProperty();
      void removeProperty();
      void updateTourSchedule();
      ~Admin();
  };

Admin::Admin() {

}

Admin::Admin (string pUserName,string pPost,int pStaffId,string
pPassword,Report *R, Property *P) {

    userName=pUserName;
    post=pPost;
    password=pPassword;
    staffId=pStaffId;
    Rpt=R;
    Ppt=P;
}

void Admin::addProperty() {

}

void Admin::removeProperty() {

}

void Admin::updateTourSchedule() {

}

Admin::~Admin() {

}
```

```
class Manage //Manage Class  //Association Class
{
  private :

      int memberId;
      string Date;
      Tour*tur; //As an object of Tour as attribute
      Admin*adm; //As an object of Admin as attribute

  public :

      Manage();
      Manage(int pMemberId, string D, Tour *T,Admin*A);
      Manage(int pMemberId, string D);
      void cancelRequest();
      void updateTour();
      void assingDate();
      ~Manage();
};

Manage::Manage() {

}

Manage::Manage(int pMemberId,string D, Tour *T, Admin*A) {

    memberId=pMemberId;
    tur=T;
    adm=A;
}

void Manage::cancelRequest() {

}

void Manage::updateTour() {

}

void Manage::assingDate() {

}

Manage::~Manage()
{

}
```

```
int main() {

  Member *mbr1 = new Member("Jeni", "Jeni@123" , 876); //Jeni = Username , 
Jeni@123 = Password & 876 = Member ID
  mbr1->viewProperty();
  mbr1->manageProfile();
  mbr1->manageProperty();
  mbr1->requestTour();

  Property prp1("Land" , 67); // Land = Property Type & 67 = Property ID
  prp1.displayPropertyDetails();
  prp1.addProperty();
  prp1.removeProperty();

  Admin adm1("Ahamed", "Manager" , 1001 , "Zee@123"); // Ahamed = Username , 
Manager = Post , 1001 = Staff ID & Zee@123 = Password
  adm1.updateTourSchedule();
  adm1.addProperty();
  adm1.removeProperty();

  Payment pay1(002, 67); // 002 = Payment ID & 67 = Property ID
  pay1.confirmPayment();

  Report *rpt1;
  rpt1->generateReport();

  Tour tr(100, 876); // 100 = Tour ID & 876 = Member ID
  tr.viewSchedule();

  Manage mng1(100, "13/10/2020"); // 100 = Tour ID & 13/10/2020 = Date
  mng1.cancelRequest();
  mng1.updateTour();
  mng1.assingDate();

  delete mbr1;
  delete rpt1;

  return 0;

}
```