Department of Computer Science and Software Engineering

Model Driven Software Engineering
COEN-6312

Deliverable - 3

Submitted by:

TEAM-BLEED BLUE

Supervised by:

Dr. Abdelwahab Hamou-Lhadj

Table of Contents

1. Introduction

This section of the project includes the Class diagram and OCL constraints for Real Estate System. Class diagram is prepared in Papyrus which can later be used to generate java code. The subsequence section includes the list of OCL constraints along with the OCL descriptions.

2. Purpose

The purpose of class diagram is to model the static view of Real Estate System. This will be later mapped to generate java code with the help of papyrus java plugin. OCL Constraints impose and validated the restrictions on the model or system if any. Its plays an important role to analyze the life cycle of the software system.

3. Class Diagram:

The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling and the classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

- The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses.

Members of the class Diagram for Real Estate System are as follows:

Visibility: To specify the visibility of a class member (i.e. any attribute or method), these notations must be placed before the member's name.

+ Public
- Private

Relationships: A relationship is a general term covering the specific types of logical connections found between the classes. It shows the following relationships:

- **Association:** An *association* represents a family of links. An association can link any number of classes. A binary association (with two ends) is normally represented as a line. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties.

- **Aggregation:** Aggregation is a variant of the "has a" association relationship, it is more specific than association. It is an association that represents a part-whole or part-of relationship. However, an aggregation may not involve more than two classes; it must be a binary association. Furthermore, there is hardly a difference between aggregations and associations during implementation, and the diagram may skip aggregation relations altogether.

- **Generalization:** It indicates that one of the two related classes (the *subclass*) is considered to be a specialized form of the other (the *super type*) and the super class is considered a Generalization of the subclass. In practice, this means that any instance of the subtype is also an instance of the super class. The generalization relationship is also known as the *inheritance* or *"is a"* relationship.

Multiplicity: This association relationship indicates that (at least) one of the two related classes makes reference to the other. This relationship is usually described as "A has a B"

1 TO 1-> Exactly one instance

1..* TO 1..*-. Many to Many instances

1 TO 1...* One to Many instances

3.1 Class Diagram for Real Estate System:

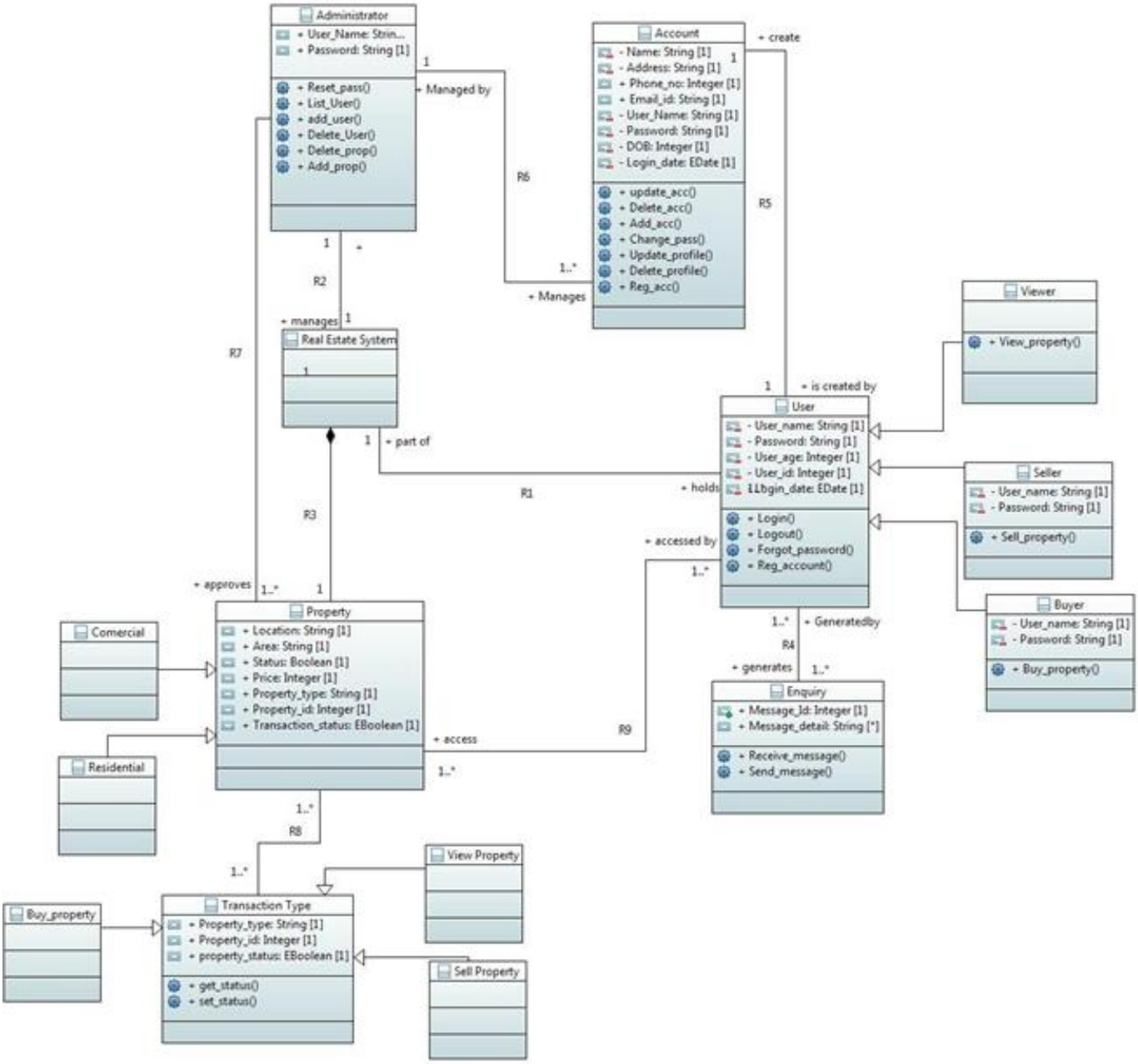

Figure 1: Real Estate System: Class Diagram

3.2 Class Descriptions:

3.2.1 User:

A user is a stakeholder to the system. They are generalized as Buyer, who can buy the property .Sellers, who can sell the property, Viewers can view the property.
The attributes and methods involved in this class are:

3.2.1.1 Attributes:

1. **User_id :** The user id is the unique id that a user will have to be able to be differentiated from the other users.
2. **User_age :** User age should be greater than 18 to be able to create an account with the system.
3. **User_name**: User name is a unique name selected by user while creating an account and which is used by users to login the system
4. **Password:** User must select a strong password while creating an account and user will have to provide password while logging in the system.

3.2.1.2 Methods:

1. **Login()** : User must login before performing any kind of transaction
2. **Logout ()**: User must logout before closing the system in order to protect their account from misuse.
3. **Forgot_pass()**: User can reset their password by answering the security question .
4. **Reg_account()**:

3.2.2 Buyer

A user is categorized as buyer. A buyer must login using the valid user name and password .Buyer can view the posting and can communicated with the seller though message.
The attributes and methods involved in this class are:

3.2.2.1 Attributes:

1. **User_name:** Buyer must login before performing any kind of transaction and in order to login the user must enter correct username.
2. **Password**: User must enter a valid password in order to login and there after perform any kind of transaction.

3.2.2.2 Method:

1. **Buy_property :** user will call the buy property method which will generate a message to the seller with the buyers details.

3.2.3 Seller

A user is categorized as seller, in order to sell the property user must login using valid user name and password.
The attributes and methods involved in this class are:

3.2.3.1 Attributes:

1. **User_name:** Seller Buyer must login before performing any kind of transaction and in order to login the user must enter correct username.

2. **Password**: User must enter a valid password in order to login and there after perform any kind of transaction

3.2.3.2 Methods:

1. **sell_property :** user will call the Sell property method which will redirect to property class where seller will create a post.

3.2.4 Viewer

User who is not willing to buy or sell property is considered as viewer and need not to create an account.

3.2.4.1 Attributes;

This class is going to inherent all the attributes of base class User.

3.2.4.2 Methods:

1. **View property**: Viewer can view all the property postings using this method.

3.2.5 Enquiry:

User can communicate with the other user through the means of message. Message includes negotiation, deals, offers etc.
The attributes and methods involved in this class are:

3.2.5.1 Attributes:

1. **Message_id:**
Every message that is being generated will have a unique message_id. This will make communication easier if the user wants to refer an old message just by providing, message_id for Message_detail:

3.2.5.2 Methods:

1. **Send_message():** The user sends a message to a registered users in order to make deal or price negotiation.
2. **Receive_message():** The Registered users can receive message from another user .

3.2.6 Accounts

Once the duly filled form is submitted by the user the account will be created. The user can update their profile, change their password and can delete their account whereas admin can add account as well as delete account if he observes some malicious activity.

The attributes and methods involved in this class are:

3.2.6.1 Attributes:

1. **Name:** User must provide their name while creating an account
2. **Address :** In order to increase authenticity of the account user must provide a valid address
3. · **Phone_no:** It's an optional field, If the user wants to share the details with the others users to make communication easy **.**
4. **Email_id:** User must enter a valid email address while creating an account. The same email address will be used to to send the new password if the user forgets their password.
5. **User_name :** User name is a unique name selected by user while creating an account and which is used by users to login the system
6. **Password:** User must select a strong password while creating an account and user will have to provide password while logging in the system.
7. **DOB :** User must enter their date of birth so that system can verify that user is eligible to create an account (age>18)

3.2.6.2 Methods

1. **Update_acc() :** User can update their account details such as address , contact number **.**
2. **Delete_acc() :** Admin can delete account if they see malicious activities performed by the user.
3. **Add_acc():** Admin can create an account for the user if the user requests the admin to do so.
4. **Change_pass() :** User can change their password as it is recommended to change the password in every 90 days in order to protect the account from intruders
5. **Update_profile()** : user can change their profile details if they want certain changes to the details that were provided prior while creating the account **.**
6. **Delete_profile ()** : User can delete their profile they no longer want to use the services provided by the system
7. **Reg_acc()** : Once the user submits the duly filled form the user account gets created .

3.2.7 Administrator

This class exhibits the privileges of an admin. The admin can administer the property postings in the system, also the admin monitors the whole system, in order to do so the admin must login using username as password.

The attributes and methods involved in this class are:

3.2.7.1 Attributes

1. **User_name:** This is the mandatory field for the admin while logging in the system.
2. **Password:** Admin must enter a valid password while logging in the system.

3.2.7.2 Methods

1. **Reset_pass :** If the admin forgets or wants to change the password as a security precaution **.**
2. **List_user :** If the admin wants to see the list of the users in the system
3. **Add_user** : If the user wants the admin to create an account on behalf of user . Admin can do that using this method**.**
4. **Delete_user :** If the admin observes a malicious activity by the user . Then admin will delete the account in order to maintain authenticity of the system.

5. **Delete_prop**: If the admin observes that the property details are not valid or looks fake then admin can delete the property posting.
6. **Add_prop**: Admin can post if he himself wants to provide some deal or offer.

3.2.8 Property:

Property class holds all the property details, both User and admin can post property details which comprises Location of the property, Area where the property is located, Status of the deal, Price demanded and type of property
The attributes and methods involved in this class are:

3.2.8.1 Attributes

1. **Location:** while creating the advertisement User/Admin must provide the location.
2. **Area:** It's a mandatory field while creating a property posting**.**
3. **Status:** There might be some kind communication between the users which may lead to status change of the property to Sold (yes/no).
4. **Price: The** Seller/admin must add the desired price for the property so that the buyer can see the price and can accept or negotiate deal.
5. **Property_type**: Property can be broadly categorized as commercial and residential.
6. **Property_id**: property id acts as a primary key to select the particular property that user wants to buy.

3.2.9 Commercial:

The property is categorized as commercial, it is generalized to Property Class so it can inherit all the details related to commercial property.
Commercial Property types:

- Office
- Retail
- Hotels
- Land

3.2.10 Residential:

Property is broadly categorized as residential which is inherited from property class and will hold the details such as Location, Address, Status, Price for the residential property.

Residential Property types:

- Apartments
- Villa
- Houses
- Condos

3.2.11 Transaction Type:

The user can perform certain actions on the system, user can sell property, Buy property and Viewer can view postings.

3.2.11.1 Attributes:

1. **Property_type**: Property can be broadly categorized as commercial and residential.
2. **Property_id**: property id act as a primary key to select the particular property that user wants to buy.

3.2.12 Buy property

If the user wants to buy a certain property, I.e buyer accepts the deal .All the attributes of transaction type class will be inherited in buy property class.

3.2.13 Sell Property

User can create and post the property offer so that buyers and viewers can view the property and if they want to buy the property then they can contact the seller through message.

3.2.14 View Property

User who just wants to view the deals can view the property posting but cannot buy or sell the property without registering or logging in.

4. Constraints

The Object Constraint Language (OCL) is a declarative language for describing rules that apply to Unified Modeling Language. Constraints are the imposed conditions that must be satisfied while performing certain function to get the desired output. It is not always possible to express all the constraints in graphical language form so we use Object Constraint Language (OCL) that provides a formal language for specifying constraints which supplement the models created in terms of UML diagrams.

OCL statements are constructed in four parts:

1. a context that defines the limited situation in which the statement is valid
2. a property that represents some characteristics of the context (e.g., if the context is a class, a property might be an attribute)
3. an operation (e.g., arithmetic, set-oriented) that manipulates or qualifies a property, and
4. keywords (e.g., if, then, else, and, or, not, implies) that are used to specify conditional expressions

Following are the constraints based on our class diagrams:

1. Every user must have a unique user id.
   Context User
   Inv: self.allinstances->for all (U1, U2 ->User, U1<>U2
   Implies U1.id <> U2.id)

2. User must login once in 6 months in order keep his account active.
   Context User: get_login(a: date)
   Pre: self.logindate <= 6
   Post: login date = logindate@pre +1

3. Admin deactivate user's account if the user is not logged in for 6 months.
   Context Admin
   Inv : self .R6-> Reject (login date >=6)

4. User age must be greater than or equal to 18 years.
   Context User
   Inv: self .user_age>=18

5. No user can have same property to sell.
   Context User
   Inv: self.R9 ->for all (P1, P2->property,
   P1<> P2 implies P1. Prop_id <> P2. Prop_id)

6. In order to buy/ sell property, user must have an account
   Context User
   Inv: self**.**R9 **->** includes (self.R5)

7. The phone number of the user should be 10 digits.
   Context User
   Inv: self. Phone No .size () =10

8. Guest Viewers can only View property.
   Context User
   Inv: self.R5 -> is Empty ()
   Implies self. R9 ->Transaction _status =”View Property”

9. All the fields for account Registration should be mandatory.
   Context Account
   Inv: self.allInstances ->Not Empty ()

10. To manage property according to Status:
    Status 1: The Property is “Available to sell”
    Status 2: The Property is “Sold Out”

    Context Property
    Inv: self.status = if 1 then self.Available = true and self.soldout = false
    else if status = if 2 then self.Available = false and self.soldout = true
    endif

11. Enquiry message should not be empty and should not exceed 500 characters.
    Context User
    Inv : self.message_detail-> NotEmpty() and self.message_detail-> size() <= 500

12. User should enter valid password credential at the time of Registration.
    Context User

Inv: password = password.matches ((?=.*\d)(?=.*[a-z])(?=.* [A-Z])(?=.*[@#$%]).{6,20})

13. Administration should grant privileges to registered users.
    Context Administration :: listusers()
    pre:self.grantprivilegesto.role= ‘Buyer’ and self.grantprivilegesto.role = ‘Seller’

14. Property always should valid status: available or soldout
    Context Property
    Inv: self.status() -> NotEmpty()

5. References

[1]. Abdel Wahab Hamou Lhadj, COEN6312 Course Lecture Notes, Concordia University, 2016.
[2]. https://eclipse.org/papyrus/