VACATION HOUSE EXCHANGE APPLICATION

Software Engineering Design EEET2482
Advanced Programming Techniques

Lecturer: Tran Duc Linh
Email: linh.tranduc@gmail.com

Nguyen Tuong Khang   s3927112
Le Duy Quang   s3912105
Do Phuong Linh   s3926823
Nguyen Phan Duc Anh   s3915181

Group: Beautiful Coders

RMIT
UNIVERSITY

Introduction

This report aims to provide information on our group collective effort to design an application for vacation house exchange which allows registered members to use houses of other members when they go on vacation. Our application has a numerous feature that enables registered users to list their house or to rent others, and new users to register.

Our group's application has simple mechanism and logic with four classes house, member, admin, and system. To register, non-member user has to pay an initial fee of $500 and he can earn 500 credit points into his newly made account. He also needs to provide basic personal information (e.g., full name, phone number) and the house's information he wants to list on our app. The app can receive comments, feedbacks from users and calculate house-rating score, occupier-rating score in order to provide comprehensive reviews on members and on their houses.

Since this is an initial version of the app, we decide to set a few limitations. First, each member can only add one house to his/her account. Second, each member can only occupy one house at a time. Third, members are not allowed to cancel if the request is already accepted. Finally, the application is early available to users in several cities: Hanoi, Saigon, Da Nang.

Application Design and Development

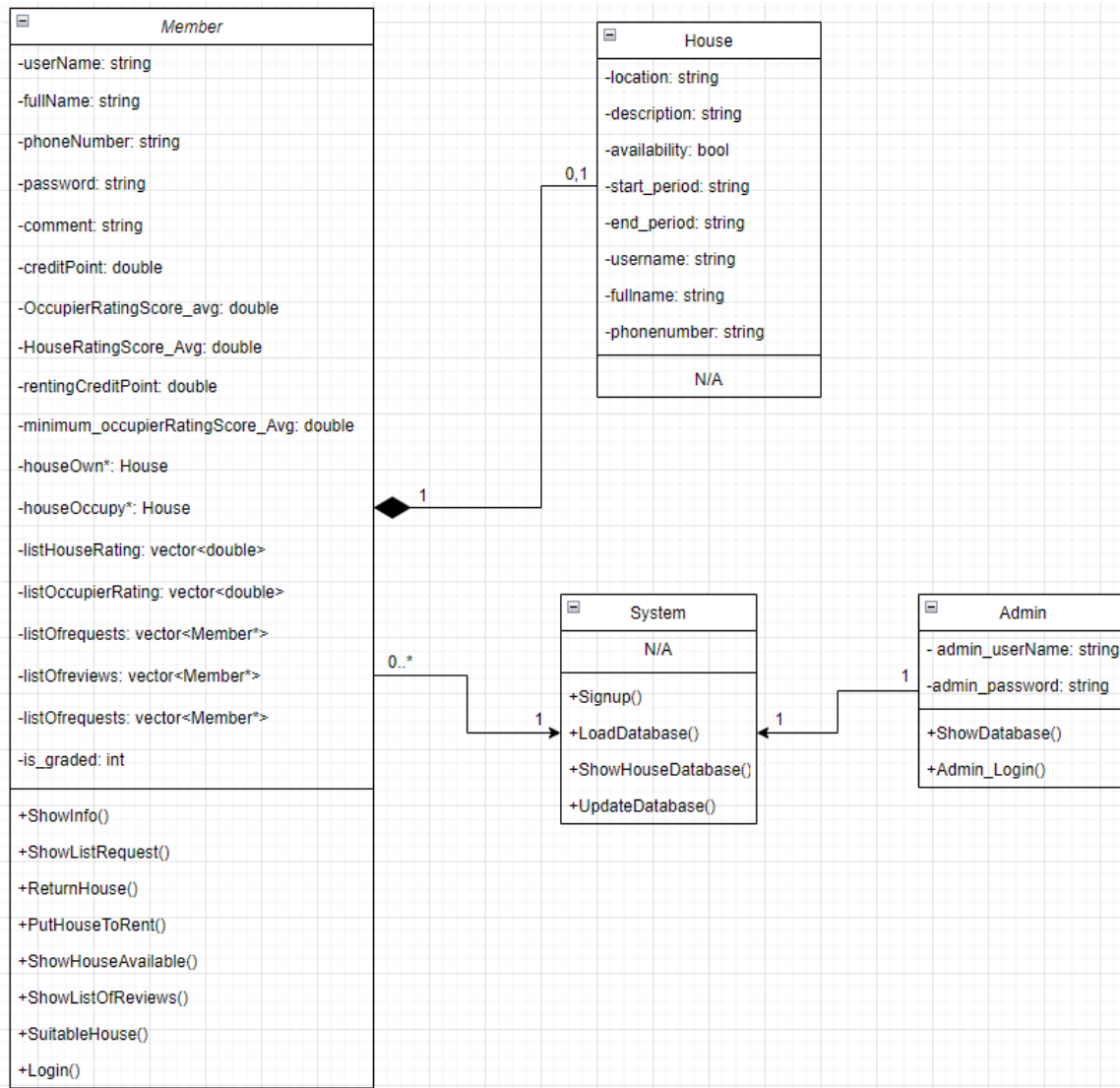## SOFTWARE DESIGN (CLASS DIAGRAM)

Figure 1. Class diagram (UML)

CLASS DETAILS

Tables below will describe all our classes and their attributes and methods. There are a total 4 main classes namely System, Admin, Member, House.

Class System	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method).
Attribute(s)	Class system accesses other classes attributes and methods as a friend class.		

Method(s)	Void Signup()	Sign up option for non-member to be become member	To store the information of new members so that members can login later
	Void LoadDatabase()	Load data from the database	Input data to the program from a saved file
	Void ShowHouseDatabase()	Show house information for non-members	Print out the information about the house excluding the availability and comments
	Void UpdateDatabase()	Update data from the database	Output the data from the program to update/create a saved file

Figure 2. Class System.

Class Member	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method).
Attribute(s)	String username	Member's username	Store username of member when initiated
	String fullName	Member's full name	Store full name of member when initiated
	String phoneNumber	Member's phone number	Store phone number of member when initiated
	String password	Member's password	Store password of member when initiated
	String comment	Member's comment	Store comment of member which later be used to add to comment list
	Double creditPoint	Member's credit point	Store credit point of member when signed up
	Double OccupierRatingScore_Avg	Occupier average rating scores	Store the average rating of the occupier member and update when new scores are

		given by the house owner
Double HouseRatingScore_Avg	House average rating scores	Store the average rating of the house and update when new scores are given by the occupiers
Double rentingCreditPoint	Credit Point required to rent the house	The member can set a Credit Point for the house to be rented.
Double minimum_occupierRatingScore_Avg	Minumun occupier score to rent the house	The member can set a minimum occupier rating score for the house to be rented.
House *houseown	House owned by this member	Store the information of the house owned by this member
House * houseOccupy	House occupies by this member	Store the information of the house occupied by this member
Vector<double> listHouseRating	List of house rating scores	Store the list of house ratings to use for calculating the average house rating score
Vector<double> listOccupierRating	List of occupier rating score	Store the list of occupier ratings to use for calculating the average occupier rating score
vector<Member*> listOfrequests	List of requests from members	Store member to list if they request to occupy the house
Vector<Member*>listOfreviews	List of reviews from members who were occupiers	Store the reviews from the occupiers after they return to the house
House houseData	Data of the House	Database needs to be immediately updated

		owned by this member	after user's input and action.
	Int graded	Is the house owner rated occupier	Check if the house owner has rated the occupier
Method(s)	Void ShowInfo()	Show information of the member	Show all the information about the house including availability and comments
	Void ShowListRequest()	Show list of requests	Show all the requests to rent the member's house for occupier to choose
	Void ReturnHouse()	Return house from occupier to owner	Return the house after occupying period
	Void PutHouseToRent()	Put house to rent	Put the house on the renting list for other members to view, it can set the credit point and occupier score to specific renters.
	Void showHouseAvailable()	Show the available houses	Show all the houses that are available to rent
	Void ShowListOfReviews()	Show the reviews of the house	Show the reviews of the house to renters
	Void SuitableHouse()	Filter the information to choose the suitable house for the occupier	Ask the user for a particular city and period of stay to list out all the suitable houses for that occupier
	Void Login()	Login as member	Login as member to rent and occupy house.

Figure 3. Class Member.

Class House	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method).
Attribute(s)	String location	House's location	Store the information of the house location
	String description	House's description	Store the description of the house posted by the house owner
	String available	House's availability	Store the availability of the house and prevent other members from renting if the house is unavailable
	String start_period	Start day, which occupier can start renting	Use to compare with the information input by renter to check if the start day is suitable to be rent
	String end_period	End day, which occupier must return house	Use to compare with the information input by renter to check if the end day is suitable to be rent
	String username	Username of member house owner and occupier	Temporarily store the username of the house owner and occupier turn by turn
	String fullname	Full name of member house owner and occupier	Temporarily store the full name of the house owner and occupier turn by turn
	String phoneNumber	Phone number of member house owner and occupier	Temporarily store the phone number of the house owner and occupier turn by turn
Method(s)	No method for this class		

Figure 4. Class House.

Class Admin	Name and Data type	Description	Reason/Explanation (why we need it as an attribute/ method).
Attribute(s)	String admin_userName	Admin username	Store the data of the admin username

	String admin_password	Admin password	Store the data of the admin password
Method(s)	Void ShowDatabase()	Show information of all members	Allow admin to view all the information from members excluding their password
	Void admin_Login()	Login for admin	Allow a pre-defined username and password to login as admin

Figure 5. Class Admin.

Class Relationship

- Class House is a Composition part of Class Member:

Each member owns zero to one house and house only exist when there is a member

- Class Member associates with Class System:

Member information is stored using function in class System

- Class Admin associates with Class System:

Admin information is stored using function in class System.

Implementation Result

WELCOME SCREEN

Result Summary: We defined a function, namely WelcomeScreen(), to guide the users use the app in different roles in our application. They can use the app as guests, members, or admin. Moreover, there is always an option of Exit Program for users to quit the program. Therefore, we can use this function repeatedly to shift between different roles.

```
791     //Welcome Screen
792     void WelcomeScreen()
```

Figure 6. Function WelcomeScreen()

Screenshots of Sample Result:


```
EEET2482/COSC2082 ASSIGNMENT
VACATION HOUSE EXCHANGE APPLICATION

Instructor: Mr. Linh Tran
Group: BeautifulCoders
s3927112, Nguyen Tuong Khang
s3912105, Le Duy Quang
s3926823, Do Phuong Linh
s3915181, Nguyen Phan Duc Anh


Use the app as 1. Guest    2. Member   3. Admin  4. Exit Program
Enter your choice: 1

This is your menu:
0. Exit
1. View all house details
2. Register                          WellcomeScreen()
Enter your choice: 0

Use the app as 1. Guest    2. Member   3. Admin  4. Exit Program
Enter your choice: 4

Thank You for using Trading House App!
```

Figure 7. Sample result.

BASIC FEATURES
Result Summary

Feature Name	Feature Description	Status (*Implemented/ Not Implemented*)	Bugs/Limitations if it has
1. Non-member does registration	A non-member can register to become a member (information is recorded).	Implemented in method of class System	**Limitations:** 1) When sign in successfully the program doesn't move the user to the menu view of the member, instead of requiring user to enter again the role they want to use the app (WelcomeScreen function is called).
2. Non-member view all house details	A non-member can view all house details (but not their reviews and availability).	Implemented in method of class System	There are no **bugs** and **limitations**
3. A member log in feature	A member can login with their username and password	Implemented in method of class Member	**Limitations:** 1) If they entered wrong username or password, WelcomeScreen function is called
4. A member can view their information	Allow member to view their information includes personal information, house information, occupied house information, and occupier user.	Implemented in method of class Member	There are no **bugs** and **limitations**
5. Admin log in and view all members' information	After logging in, admin can view all member's information (not include password) and member's house details	Implemented in method of class Admin	There are no **bugs** and **limitations**

	(not include the occupiers)		
6. A member list or unlist their house	Allow member to list their house to be occupied (with consuming credit points, minimum required occupier rating, and occupied time period for house availability)	Implemented in method of class Member	**Bugs:** 1) In case the users input characters to set the require credit point and minimum occupier score the program will crash due to input variable we set them as double not string. There are no **limitations**
7. A member search for suitable house and request to occupy house	Allow member to search for house they want to occupy by 2 ways: 1. show all available houses and request a house. 2. request member to input requirements (including the location and time period they want to occupy) and show list of houses that meet the requirements. Then ask if member want to occupy any house in the list and enter the index number if yes.	Implemented in method of class Member	There are no **bugs** and **limitations**
8. A member view list of requests to their house	Allow member to view list of occupiers requesting their house	Implemented in method of class Member	There are no **bugs** and **limitations**
9. A member accept request	Allow member to accept one occupier out of the list of requests	Implemented in method of class Member	There are no **bugs** and **limitations**

10. *A member occupy the requested house*	Allow member to view their occupied house in their information detail	Implemented in method of class Member	There are no **bugs** and **limitations**
11. *A member rate their occupied houses*	Allow member (occupier) to return house, give house-rating score and comment for the occupied house	Implemented in method of class Member	There are no **bugs** and **limitations**
12. *A member rate their house's occupier*	Allow member (house owner) to enter rating score for the house occupier	Implemented in method of class Member	There are no **bugs** and **limitations**
13. *Save all data into a file*	All data of members including personal information, house details, and list of requests are saved into a data file	Implemented in method of class System	There are no **bugs** and **limitations**

FEATURES PICTURES.

1. *Non-member does registration*


```
Use the app as 1. Guest    2. Member   3. Admin  4. Exit Program
Enter your choice: 1

This is your menu:
0. Exit
1. View all house details
2. Register
Enter your choice: 2
Require intial entry of $500 to become a member($500 = 500 credit points)
Balance has changed from $348490658 to  $348490158
Successfully paid initial entry fee!!


//------------------------FILLLING THE FORM----------------------------//
Please Fill the form:
    > Full name: Nguyen Van A
    > Phone number: 0123456
Adding your house information:
    > Location: Saigon
    > House Description "OPTIONAL" (press ENTER if no description): big house


//--------------------REGISTER USERNAME AND PASSWORD----------------------//
    > Username (No Space!): nguyenvana
    > Password (No space!): 123
Sign In Successfully!!
```

Figure 8. Sample result of feature 1

2. Non-Member view all house details

```
This is your menu:
0. Exit
1. View all house details
2. Register
Enter your choice: 1

House of Le Duy Quang. Location at Saigon.
Description: big house. For more contact call 021939123

House of Nguyen Tuong Khang. Location at Saigon.
Description: 4 floors. For more contact call 092139213

House of Do Phuong Linh. Location at Hanoi.
Description: 3 floors. For more contact call 0291388213

House of Duc Anh. Location at Da Nang.
Description: small house. For more contact call 023923832

House of Nguyen Van A. Location at Saigon.
Description: big house. For more contact call 0123456
```

Figure 9. Sample result of feature 2.

3. A member log-in feature.

```
Use the app as 1. Guest    2. Member   3. Admin  4. Exit Program
Enter your choice: 2
Enter username: nguyenvana
Enter password: 123
Log In Successfully!!

This is your menu:
0. Exit
1. View Informations detail
2. Put your house for occupying
3. View house for occupied menu
4. Show list of requests
5. Show your house reviews
6. Return your occupied house
Enter your choice: |
```

Figure 10. Sample result of feature 3.

4. A member can view their information

Figure 11. Sample result of feature 4.

5. Admin log-in and view all member's information

Figure 12. Sample result of feature 5.

6. A member list or unlist their house.

Figure 13. Sample result of feature 6.

7. *A member searches for suitable and requests to occupy house.*

Figure 14. Sample result of feature 7.

8. *A member views all list of requests to their house*
9. *Accept one request to occupy their house.*

Figure 15. Sample result of feature 8 and 9.

10. A member occupies the requested house.

```
Use the app as 1. Guest    2. Member   3. Admin  4. Exit Program
Enter your choice: 2
Enter username: linhdo
Enter password: abc
Log In Successfully!!

This is your menu:
0. Exit
1. View Informations detail
2. Put your house for occupying
3. View house for occupied menu
4. Show list of requests
5. Show your house reviews
6. Return your occupied house
Enter your choice: 1
Your Information:
   Fullname = Do Phuong Linh. Username = linhdo. Password = abc
   Phone number = 0291388213. Credit point = 460. Occupier Rating Score = 0

Your House Information:
   House location = Hanoi. House description = 3 floors.
   House Rating Score = 0. Availability = NULL.
   Require credit point for rent = 0.  Minimum require occupier rating = 0   .Start Period = NULL
. End Period = NULL.

Your House Occupied:
   House location = Saigon. House description = Big house
. House start period = 22/05/2022. House end period = 23/05/2022
   Information of Owner of this house:
        >Fullname = Nguyen Van A. Phone number = 0123456. Username = nguyenvana

There is no information of the member your house's occupier
```

Figure 16. Sample. result of feature 10.

11.A member rates their occupied houses

```
This is your menu:
0. Exit
1. View Informations detail
2. Put your house for occupying
3. View house for occupied menu
4. Show list of requests
5. Show your house reviews
6. Return your occupied house
Enter your choice: 6
Do you want to return the house (Y/N)?: Y
Please enter the House-rating score for this house after occupied it from -10 to +10: 8.5
Please leave a comment for this house after occupied: I love the view!
```

Figure 17. Sample result of feature 11.

12. A member (house owner) rates their house's occupier.

```
Use the app as 1. Guest    2. Member   3. Admin  4. Exit Program
Enter your choice: 2
Enter username: nguyenvana
Enter password: 123
Log In Successfully!!

This is your menu:
0. Exit
1. View Informations detail
2. Put your house for occupying
3. View house for occupied menu
4. Show list of requests
5. Show your house reviews
6. Return your occupied house
Enter your choice: 5
1)   User: Devadavid: Beautiful house!. House Rating Score = 9. Occupied Score not graded yet!
2)   User: linhdo: I love the view!. House Rating Score = 8.5. Occupied Score not graded yet!
Choose the review to rate Occupied Score please enter the INDEX NUMBER: 2
Enter the Occcupied Score for this User: 10
```

Figure 18. Sample result of feature 12.

13. Save all data into a file.

Figure 19. Sample result of feature 13.

TIME PERIOD FEATURE

Result Summary:

When the members search for a suitable house to occupy, the app requires them to input the start and end date for occupying. After that, the app will check that house's start and end date in the database. If both user input and dates in the database match, users then successfully send a request to the house.

On the other hand, the data type of start period date and end period date is set as a string instead of using datetime. Therefore, if users don't input the correct date format as dd/mm/yyyy, the time period feature will lead to the wrong meaning.

Screenshots of sample result:

```
This is your menu:
0. Exit
1. Show list of houses are available for renting
2. Show all the suitable house for user requirements
Enter your choice: 2
Do you want to search for particulary city you want to occupy the house (Y/N)?: N
Enter your start period in format dd/mm/yyyy: 03/12/2022
Enter your end period in format dd/mm/yyyy: 04/12/2022
There is no suitable house for you
```

Figure 20. Sample results of time period feature.

Figure 21. Sample results of time period feature.

Discussion & Conclusion

In conclusion, our program runs without major errors (bugs), and this is achieved through consistent testing and collective team effort in clear communication and teamwork to reach optimal solutions.

One factor that greatly contributes to our success in the pursuit of simplistic solutions is planning UML diagram. There is a heavy correspondence between our implementation and our class diagram as we developed the code and planned our diagram in conjunction. Members of our group can all agree that the process of making UML diagram is significantly helpful in completing this project. Laying out all the classes, their attributes and methods and the relationships between them assist us to keep track and look at the big picture, giving us ideas of what should be done next. We realized that having a class diagram is crucial in such a big project with a size of approximately 1200 lines of code. Going back and forth from the design and the IDE, we constantly had to update both. Through a trial-and-error approach, we are able to successfully finish this program containing all the quintessential features of a vacation house exchange program should've had.

And, we have learnt a handful of lessons on designing and creating a functional computer program. Some of them are certainly how to work as a team. With a task that is at this level of complexity and time-consuming, we are forced to divide work as a team to achieve the best result. Moreover, we've learnt a great deal about advanced programming concepts and techniques namely linked list, vector, etc. These understanding helps us to feel more content to learn more about programming and software developing in the future.

Bibliography

[1] T. D. Linh, "Assignment Documentation," [Online]. Available:
    https://rmit.instructure.com/courses/100623/assignments/663739.

[2] T. D. Linh, "Lecture Slides," [Online]. Available:
    https://rmit.instructure.com/courses/100623/modules.