Hotel Reservation System

Object Design Document

V:1.0

23.12.2016

Cabir Sonay MERAL
Arda YAZKAN
Ecem Naz KILIÇ
Orçun TEZ

Prepared for
SE301 Software Engineering

IŞIK UNIVERSITY
COMPUTER
SCIENCE AND
ENGINEERING

Table of Contents

OBJECT DESIGN DOCUMENT

1. Introduction

1.1. Object Design Trade-offs

Firstly, our project name is Hotel Reservation System (HRS). This website is about booking. People can search any type of hotel and they can reserve using whit HRS. The purpose of this document is to describe design and architecture of the HRS site. All code in this system is own original work. It including data and user interfaces. We use the database to keep information. And the other side the objective of HRS is to be simple. But also it cover every detail and function needed by HRS. Therefore, database increase security in the system. Because of this reason people can use this site without any doubt. System keep some information like address, password, hotel name, room type… etc. securely. There is some trade-off in this project. First of all, HRS is used database. Because database increase performance in this system. System make register, log-in and search. These are main functionality. So; Database is most important thing in this project. When user want to search hotel, every user want to see quickly. Database provide fast response time, more secure than file storage and easy to use. MySQL can handle almost any amount of data, up to as much as 50 million rows or more. No need in this project. However, MySQL provide high scalability.

1.2. Interface Documentation Guidelines

In this system, object design principle had applied. There are two groups of graphical design in this project. First, graphical design for User –Who is Client, Guest or Visitor- side, and other for Admin side. User panel has also three groups which are Client, Guest and Visitor. These three designs similar to each other but every page have their own privileges.

On the user side, there are some criteria’s. For example; Users easily understand the site structure, and they can take easily actions on the site and with this they can easily make them reservations. On the other hand, HRS site easy to understand for all users (Admin, Client, Guest and Visitor). In addition, Guest can see reservations that done before, they can make a new reservation, they can search a hotel room and so on. And Client side, Client can see reservations and they can see who reserve hotel room. By the way, Client can add new hotel to system but in this step client have to get admin permissions to add new hotel into the HRS. Visitor side, Visitor can search to hotel room but when s/he try to reserve hotel room they have to be register into the HRS:

The admin panel, is more complex according to user-side. However, it also easy to useable for admin. Menus are clearly identified on the panel. Admin can check all users’ information, can approve/reject Client create hotel request, also admin can delete guest, hotel and client anytime. By the way when client try to register into the HRS, admin see their register request, if admin accept their register request Client can enter into the HRS. This step increases the confidence of the system in terms of user who is using HRS.

Coding standards are important in any development project, but they are particularly important when many developers are working on the same project. Coding standards help ensure that the code is high quality, has fewer bugs, and can be easily maintained. Function names must always start with a lowercase letter. When a function name consists of more than one word, the first letter of each new word must be capitalized. Any file that contains PHP code should end with the extension ".*php*", with the notable exception of view scripts.

1.3. Definitions, Acronyms, and Abbreviations

No:	Terms/Acronyms	Definitions
1.	**ODD**	Object Design Document.
2.	**UI**	User Interface
3.	**DB**	Database is a collection of information that is organized so that it can easily be accessed.
4.	**PHP**	Hypertext Preprocessor
5.	**HTML**	Hyper Text Markup Language
6.	**SDD**	System Design Document

1.4. References

- http://searchsqlserver.techtarget.com/definition/database
- http://dioscuri.sourceforge.net/docs/ODD_Dioscuri_KBNA_v1_1_en.pdf

2. Packages

Packages are so important because, medium and large scale projects do not coding by one person, everyone who involves the coding side of project can see very easily. Before packaging we separate our system to subsystems in our SDD. Also subsystems must be relative with each other, when coding is finish, subsystems attach the code packages. Before the coding process first of all we create empty package, in this package we have necessary statements which is including in our use case. Inside our package hotel, and Client information's stored. After that our package connect to the DB then retrieval data from DB. Our second package is about creating hotel. Primarily entering information to GUI. Another .php page these information's taken and stored in DB. In this package we can edit anything room and hotel. The last package provide user needs. Finally last package includes Log-in, Sign-up, Search and homepage.

HRS

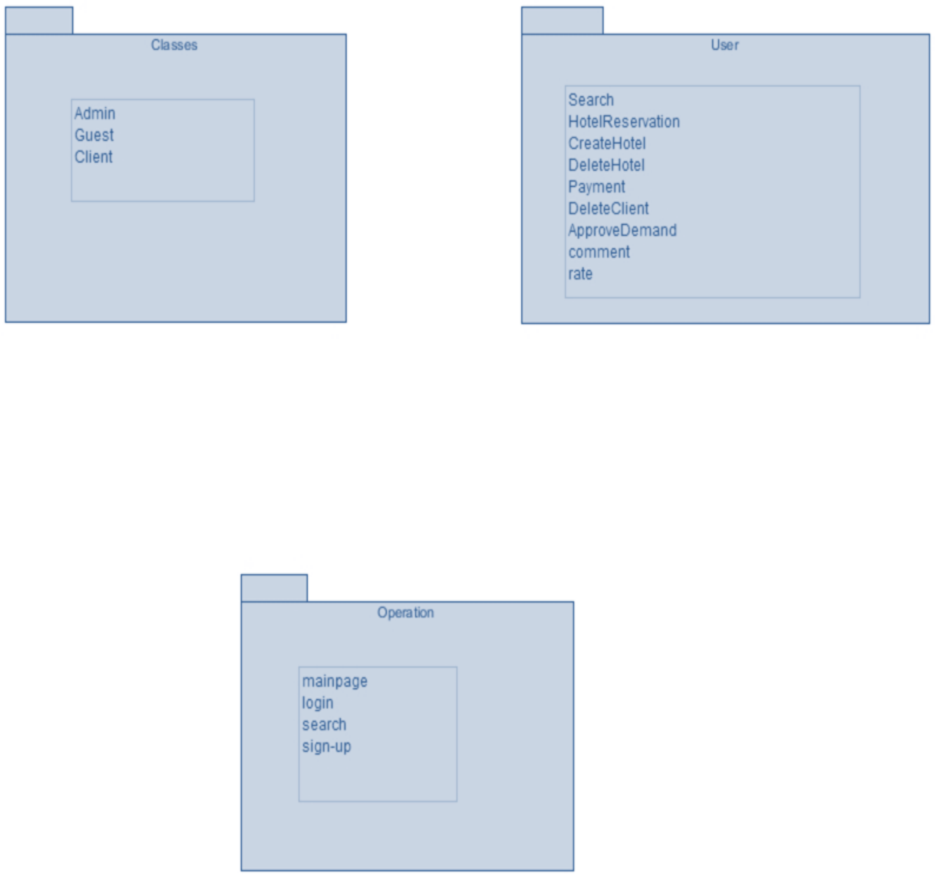


Figure 2.1- HRS packages

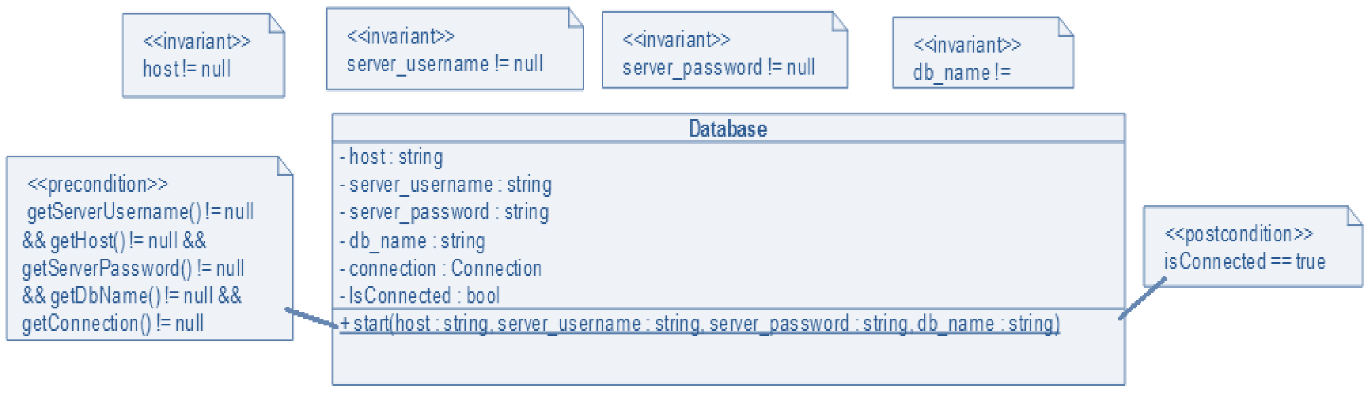
3. Class Interfaces

In class interfaces all classes' attributes are private. Some necessary part of methods have get and set methods. Guest and Client classes are extends from user class. Because every Client and Guest is a user. Both of their information are stored in DB and both of them must be log-in for make reservation, update their information, change hotel information or room type. Also user is abstract class.

Database

```
/**
 *Invariants:
 * These information cannot be null
 *@invariant db_name!=null
 *@invariant server_password!=null
 * @invariant server_username!=null
 * @invariant host!=null
 *
 * PostConditions:
 * start(host,server_username,server_password,db_name)
 * @post isConnected==true
```

HRS
 *
 *
 *
 *PreConditions:
 * start(host,server_username,server_password,db_name)
 * @pre getServerUsername()!=null && getHost!=null && getServerPassword!=null && getServerUsername!=null && getDbName!=null
*/



<<invariant>> host != null
<<invariant>> server_username != null
<<invariant>> server_password != null
<<invariant>> db_name !=
Database
- host : string
- server_username : string
- server_password : string
- db_name : string
- connection : Connection
- IsConnected : bool
+ start(host : string, server_username : string, server_password : string, db_name : string)
<<precondition>> getServerUsername() != null && getHost() != null && getServerPassword() != null && getDbName() != null && getConnection() != null
<<postcondition>> isConnected == true

Figure 3.1- Database UML

HotelRoom
/**
 *Invariants:
 * roomid cannot be null because of each room belong any room type
 *@invariant roomid!=null
 *@invariant roomname!=null
*@invariant roomprice!=null
*@invariant roomsize!=null
*@invariant roomtype!=null
*@invariant roomdescription!=null
*@invariant roomhotel!=null
*@invariant isreserved!=null
*@invariant outdate!=null
*@invariant indate!=null
 *
 * Preconditions:
 * user_login(ssn, password)
 * @pre getpassword()!=null
 * @pre getusertype_id()!=null
 *
 * PostConditions:
 * isLogin(ssn, password)

```
 * @post isLogin == true
 *
 *
```

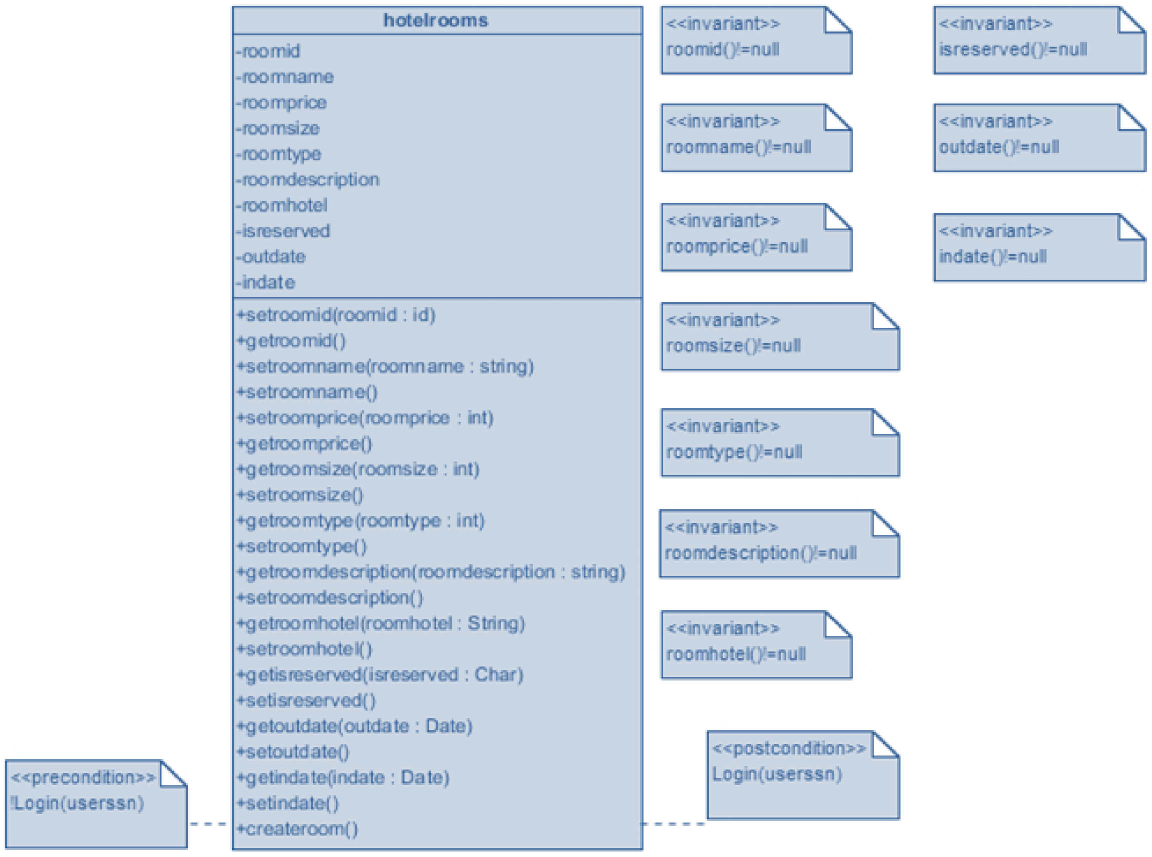

hotelrooms
-roomid
-roomname
-roomprice
-roomsize
-roomtype
-roomdescription
-roomhotel
-isreserved
-outdate
-indate
+setroomid(roomid : id)
+getroomid()
+setroomname(roomname : string)
+setroomname()
+setroomprice(roomprice : int)
+getroomprice()
+getroomsize(roomsize : int)
+setroomsize()
+getroomtype(roomtype : int)
+setroomtype()
+getroomdescription(roomdescription : string)
+setroomdescription()
+getroomhotel(roomhotel : String)
+setroomhotel()
+getisreserved(isreserved : Char)
+setisreserved()
+getoutdate(outdate : Date)
+setoutdate()
+getindate(indate : Date)
+setindate()
+createroom()
<<invariant>> roomid()!=null
<<invariant>> roomname()!=null
<<invariant>> roomprice()!=null
<<invariant>> roomsize()!=null
<<invariant>> roomtype()!=null
<<invariant>> roomdescription()!=null
<<invariant>> roomhotel()!=null
<<invariant>> isreserved()!=null
<<invariant>> outdate()!=null
<<invariant>> indate()!=null
<<precondition>> !Login(userssn)
<<postcondition>> Login(userssn)

Figure 3.2- HotelRoom UML

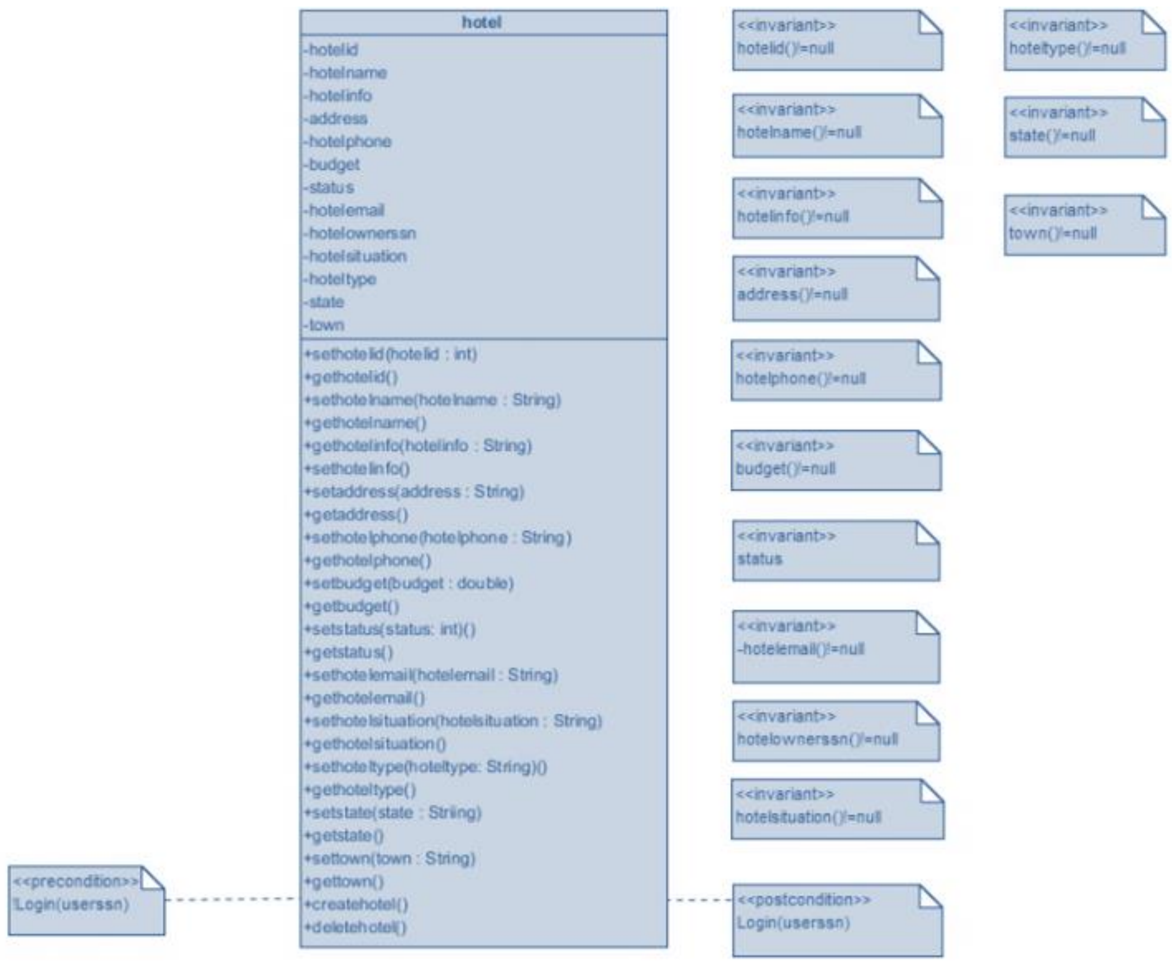
Hotel

```
/**
 *Invariants:
 * hotelid cannot be null because of each hotel belong any hotel type
 *@invariant hotelid!=null
 *@invariant hotelname!=null
*@invariant hotelinfo!=null
*@invariant address!=null
*@invariant hotelphone!=null
*@invariant budget!=null
*@invariant status!=null
*@invariant hotelemail!=null
*@invariant hotelownerssn!=null
*@invariant hotelsituation!=null
```

HRS

*@invariant hoteltype!=null
*@invariant state!=null
*@invariant town!=null

*
* Preconditions:
* login(userssn, password)
* @pre getpassword()!=null
*
* PostConditions:
* Login(userssn, password)
* @post Login == true
*
*



hotel
-hotelid
-hotelname
-hotelinfo
-address
-hotelphone
-budget
-status
-hotelemail
-hotelownerssn
-hotelsituation
-hoteltype
-state
-town
+sethotelid(hotelid : int)
+gethotelid()
+sethotelname(hotelname : String)
+gethotelname()
+gethotelinfo(hotelinfo : String)
+sethotelinfo()
+setaddress(address : String)
+getaddress()
+sethotelphone(hotelphone : String)
+gethotelphone()
+setbudget(budget : double)
+getbudget()
+setstatus(status: int)()
+getstatus()
+sethotelemail(hotelemail : String)
+gethotelemail()
+sethotelsituation(hotelsituation : String)
+gethotelsituation()
+sethoteltype(hoteltype: String)()
+gethoteltype()
+setstate(state : String)
+getstate()
+settown(town : String)
+gettown()
+createhotel()
+deletehotel()
<<invariant>> hotelid()!=null
<<invariant>> hotelname()!=null
<<invariant>> hotelinfo()!=null
<<invariant>> address()!=null
<<invariant>> hotelphone()!=null
<<invariant>> budget()!=null
<<invariant>> status
<<invariant>> -hotelemail()!=null
<<invariant>> hotelownerssn()!=null
<<invariant>> hotelsituation()!=null
<<invariant>> hoteltype()!=null
<<invariant>> state()!=null
<<invariant>> town()!=null
<<precondition>> Login(userssn)
<<postcondition>> Login(userssn)

Figure 3.3- Hotel UML

User
/**
 *Invariants:
 * id cannot be null because of each user belong any user id

```
 *@invariant id!=null
 *@invariant username!=null
*@invariant password!=null
*@invariant firstname!=null
*@invariant lastname!=null
*@invariant birthdate!=null
*@invariant email!=null
*@invariant gender!=null
*@invariant telephone!=null
*@invariant ssn!=null
*@invariant usertype_id!=null
*@invariant address!=null
*@invariant situation!=null

 *
 * Preconditions:
 * login(userssn, password)
 * @pre getpassword()!=null
 * @pre getusertype_id()!=null
 *
 * PostConditions:
 * Login(userssn, password)
 * @post Login == true
 *
 *
```

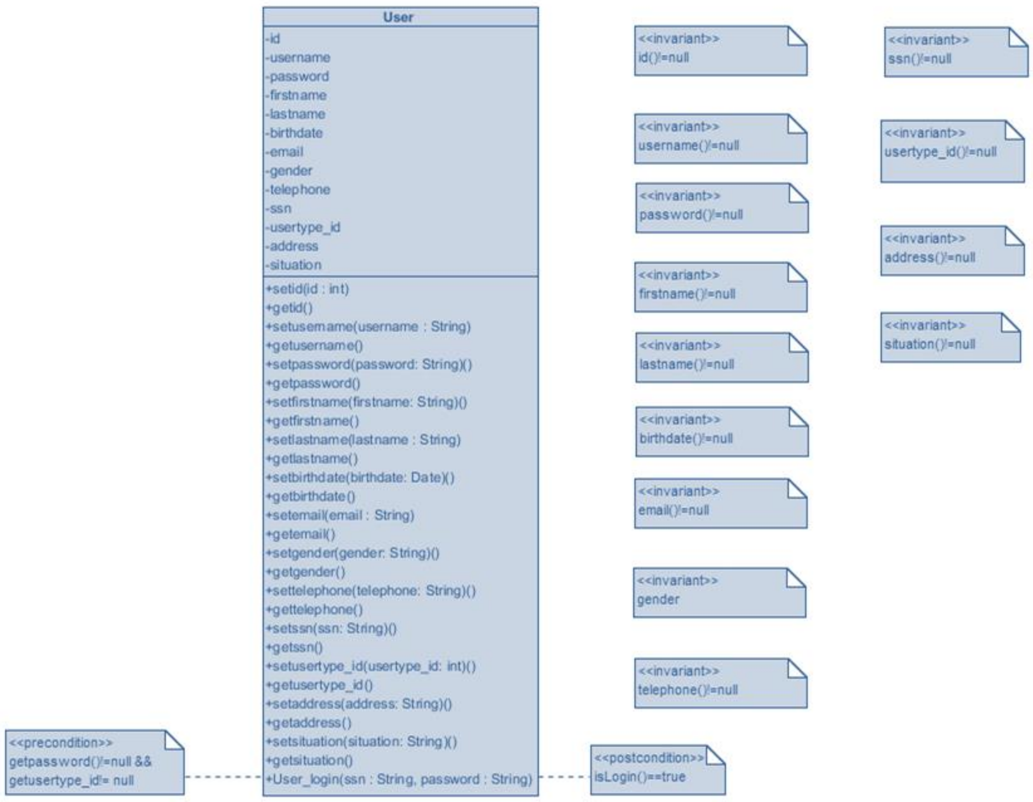

User
-id
-username
-password
-firstname
-lastname
-birthdate
-email
-gender
-telephone
-ssn
-usertype_id
-address
-situation
+setid(id : int)
+getid()
+setusername(username : String)
+getusername()
+setpassword(password: String)()
+getpassword()
+setfirstname(firstname: String)()
+getfirstname()
+setlastname(lastname : String)
+getlastname()
+setbirthdate(birthdate: Date)()
+getbirthdate()
+setemail(email : String)
+getemail()
+setgender(gender: String)()
+getgender()
+settelephone(telephone: String)()
+gettelephone()
+setssn(ssn: String)()
+getssn()
+setusertype_id(usertype_id: int)()
+getusertype_id()
+setaddress(address: String)()
+getaddress()
+setsituation(situation: String)()
+getsituation()
+User_login(ssn : String, password : String)
<<precondition>> getpassword()!=null && getusertype_id!= null
<<postcondition>> isLogin()==true
<<invariant>> id()!=null
<<invariant>> username()!=null
<<invariant>> password()!=null
<<invariant>> firstname()!=null
<<invariant>> lastname()!=null
<<invariant>> birthdate()!=null
<<invariant>> email()!=null
<<invariant>> gender
<<invariant>> telephone()!=null
<<invariant>> ssn()!=null
<<invariant>> usertype_id()!=null
<<invariant>> address()!=null
<<invariant>> situation()!=null

Figure 3.4- User UML