OOSD ASSIGNMENT DOCUMENT

Contents Page:

1. How to use and Notes, Pages 1.
2. Project Description, Pages 2 to 5
3. Agile Practices, Pages 6 to 8
4. Testing, Pages 9 to 12

5. Diagram Pictures, Pages 13

Instructions on how to use this project as a developer:

1. Copy project to desired folder or open project here.
2. Once opened, resolve the jars to the /jars directory located in the zip of this project.
3. Once resolved, Clean and build the application

Instructions on how to use as a non-developer:

1. After downloading run and execute the exe
2. Use the login details provided below or change/add as required.

Usernames and passwords can be changed if hardcoded:

Authorizations:

Admin = Access to halls and leases

Manager = Access to halls

These are in the app.class.

	Username	Authorization	Password
1.	p-alden	Admin	omg12345
2.	max-power	Warden	we123456

3. dragonborne Manager fusrodah
4. harambe SuperUser died2016

Notes and Comments:
The logins and their details are hardcoded along with the save to file part and this is not part of the specification.

Both the exe and code is provided.

The diagrams for the Project will have pictures below and the Astah file containing them provided.

OOSD Project Description

About the project

The UWE accommodation office has the responsibility to provide the necessary help for all the students that are using their accommodation on-campus. This office is made of a manager who supervises the hall operations and multiple wardens, each having a hall they are overseeing for regular cleaning and maintenance of all the rooms in that hall. This office is made up of multiple managers, wardens and office staff.

The hall manager supervises the hall operations such as generating and maintaining the room schedule and keeping a record of all the students in the halls and reviewing applications for future bookings.

Each warden oversees a hall for regular cleaning and maintenance of all the rooms in the hall. They have limited view of the room details and can only change the cleaning status of the room.

Each hall of residence has a name, number, address, telephone number. The halls provide only single rooms which have a room number and monthly rent rate. The total number of rooms provided by the accommodation office should also be available. The hall number uniquely identifies each room in all the halls controlled by the accommodation office and is used when renting a room to a student.

The UWE Bristol Accommodation Services is in charge allowing the students to rent rooms of the entire academic year. Each individual agreement has a uniquely lease number to keep track of the rent of every student. Each lease includes the lease number, duration of the lease (in months), address details of the hall, room number, student's name and ID number.

The UWE accommodation office requires the implementation of a system which helps the manager and wardens to schedule hall activities and keep track of the hall rooms in a simpler and easier way.

Everyone has login credentials to access the system to view and modify the status of the room. The manager can modify the details of a room in a hall, including the room number, status of whether being occupied or not, the monthly rent rate and the details describing the room. The manager can also decide if the room is suitable for the student to occupy. The warden can view the details of the rooms but only modify the room's cleaning status which can be 'clean', 'dirty' or 'off-line' if the room requires maintenance beyond normal cleaning.

The system will also have a "super user" that has full access to the whole system for administration purposes.

Requirements

- User Login
- Individual authorization(Login)

Optional Requirements

- Output in some form the information: (a CSV of the total entries)

Actors

- Manager
- Warden
- Accommodation Officer
- Super User

Use Cases

Manager

- getLogin
  Pre-condition: app working
  Post-condition: user can log in and access their authorized part
  If failed get an error message/try again

- getLease
  Pre-condition: user can access the leases of each room Post-condition:
  user can check modify the leases of each room.

- getRoom
  Pre-condition: user can access the rooms
  Post-condition: user can check modify the details of the rooms

Warden

- getLogin
  Pre-condition: app working
  Post-condition: user can log in and access their authorized part
  If failed get an error message/try again

- getRoom
  Pre-condition: user can access the rooms
  Post-condition: user can view the details of the room and change the cleaning status to “Clean”, “Dirty” or “Off-line”

Super User

- getLogin
  Pre-condition: app working
  Post-condition: user can log in and access the leases, halls and rooms
  If failed get an error message/try again

- getLeases
  Pre-condition: user can access the leases of each room/user login/login as super user
  Postcondition: user can check modify the leases of each room

- getHalls
  Pre-condition: user can access the halls/user login/ login as super user Post-Condition: user can check and modify the details of the halls

- getRoom
  Pre-condition: user can access the rooms/user login/ login as super user Post-condition: user can check modify the details of the rooms Accommodation officer

- getLogin
  Pre-condition: User can login to their authorized part
  Post-condition: user can log in and access their authorized part If failed get an error message/try again

- getHall
  Pre-condition: User can get the hall information
  Post-condition: User can update/create/delete hall information

- getLeases

Pre-condition: User can get the lease information
Post-condition: User can update/create/delete leases

Agile Practices

What is Agile?

Agile is a discipline intended for project management and software development to help teams submit their deliverables faster to the clients without having any problems.

If used correctly, it will make the programmers and the code itself more flexible.

Why use Agile?

Using Agile will lead to code that has little to no issues and having an efficient and compact design that will work with all the intended features. This methodology is also aiming to reduce risks by minimizing the "leap-of-faith" that is needed before any evidence of value can be obtained.

What is Agile consisting of?

The core agile software programming practices are the following:

- Test-first programming (or perhaps Test-Driven Development)
- Rigorous, regular refactoring
- Continuous integration
- Simple design
- Pair programming
- Sharing the codebase between all or most programmers
- A single coding standard to which all programmers adhere
- A common “war-room” style work area

How will we implement Agile if it was a real-life project?

For starters, we will have a close collaboration with the client for each iteration like keeping in touch through email and frequent face-to-face or online meetings (i.e. Skype) so that we can modify the assumptions and requirements of the project, correct errors caused by misunderstandings and control the course of the process. Providing the latest version of the program on a regular basis is also a must because you will get feedback and new expectations form the client even in the late stages of development, also carrying out the changes at a minimum cost.

Developing our system using Agile

Task Distribution

One step is to distribute the required work evenly between members of the team. If the workload is uneven, then the delivery schedule may be affected if the imbalance is ignored. If one member has a high workload to finish, you should move some of the tasks to the member with the fewest tasks to complete if possible.

Setting up pair programming

Pair programming means when two team members work together on the same code at the same time. An alternative would be a show-and-tell hour or a developer scrum. In a show-and-tell hour, every team member demonstrates (using screen sharing) his or her work to the entire team and receives feedback. In a daily developer scrum, developers meet briefly to collaborate on technical issues or approaches.

Use Online Tools for Agile Artefacts

Some of the physical Agile Artefacts are objects like white boards or sticky notes. You can move these in a virtual environment using online tools such as “Microsoft Visual Studio Team System” or even “Google Docs”. These tools help the team organising user stories, product backlog and so on. They also can be used to share documents across members and modify them on the go if necessary.

Arranging face-to-face interactions

Especially for larger projects, face-to-face interactions are sometimes necessary to bring together key team members, both to attain buy-in and to facilitate team building. Make sure your budget allows for travel and let affected team members know about the travelling requirement.

Plan Frequent Demos and Retrospectives

Product demos with a retrospective at the end increase the visibility of your project, convey its status and provide instant feedback as well as opportunities for process adjustment. Involving your customers in these demos (at the appropriate times) fosters a shared understanding of the project’s business context and allows the business stakeholders to communicate with the development teams.

References

Joshi, S (2012). *Agile Development - Working with Agile in a Distributed Team Environment* [online]. Available from: https://msdn.microsoft.com/en-us/magazine/hh771057.aspx. [Accessed 5 February 2018].

InfoSec Institute (2012). *17 Best Practices of Agile Methodology* [online]. Available from: https://www.business2community.com/strategy/17-best-practices-of-agile-methodology-0260495. [Accessed 10 February 2018].

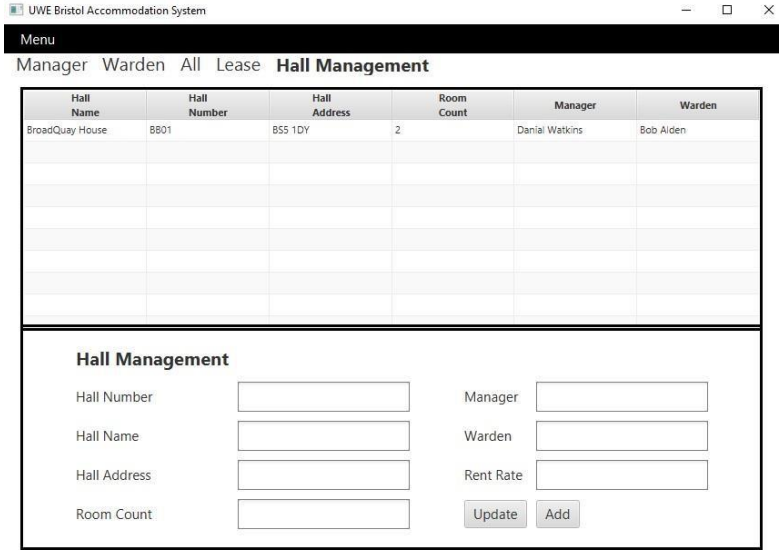
VersionOne (no date). *Agile Programming Best Practices* [online]. Available from: https://www.versionone.com/agile-101/agile-software-programming-best-practices/. [accessed 10 February 2018].

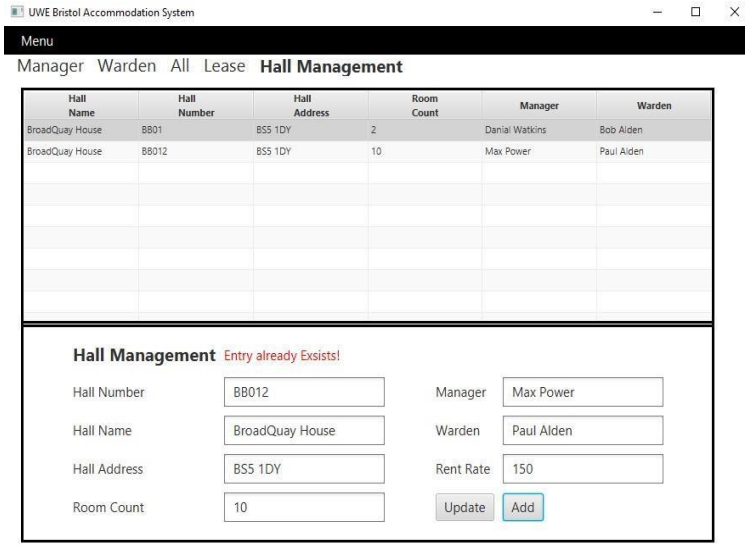
Testing Notes:

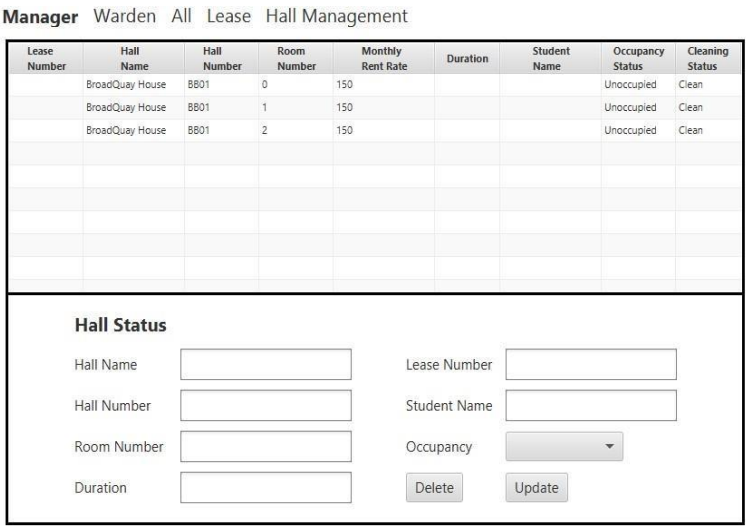
Black Box Testing:

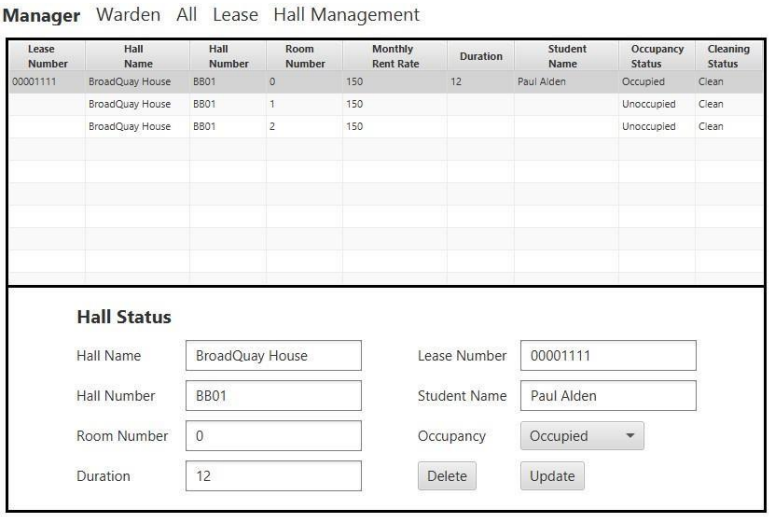
	Output Actions Preformed	Successful Result

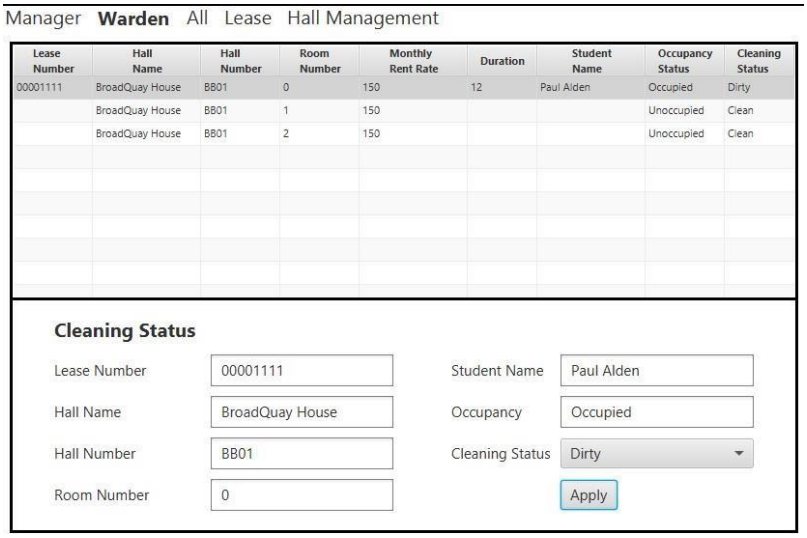
Manager	can it select values except cleaning status for rooms?	Yes
	Can we see changes to the Table that warden or all made?	Yes
Warden	can it set only cleaning status for rooms?	Yes
	Can we see changes to the Table that warden manager made?	Yes
All	Can we see changes to the Table that manager or warden made?	Yes
	can it set values for rooms?	Yes
Halls (Accommodation office)	Can it set values for the Halls?	Yes
	Do the halls add the rooms to the manager, warden and all tables?	Yes
Lease	Can it Create Leases?	Yes
Login	Can they login and only access what they are authorised to access	Yes
Overall	Do error messages show up?	Yes

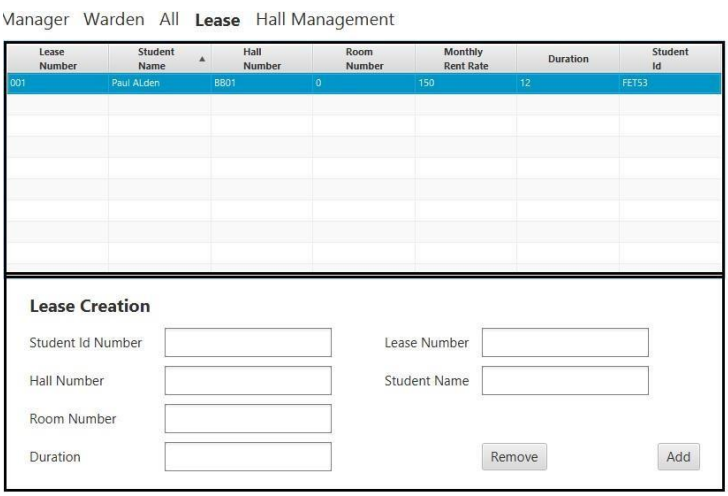

UWE Bristol Accommodation System
Menu
Manager Warden All Lease Hall Management
Hall Name
Hall Number
Hall Address
Room Count
Manager
Warden
Rent Rate
BroadQuay House
BB01
BS5 1DY
2
Danial Watkins
Bob Alden
Hall Management
Hall Number
Hall Name
Hall Address
Room Count
Manager
Warden
Rent Rate
Update
Add

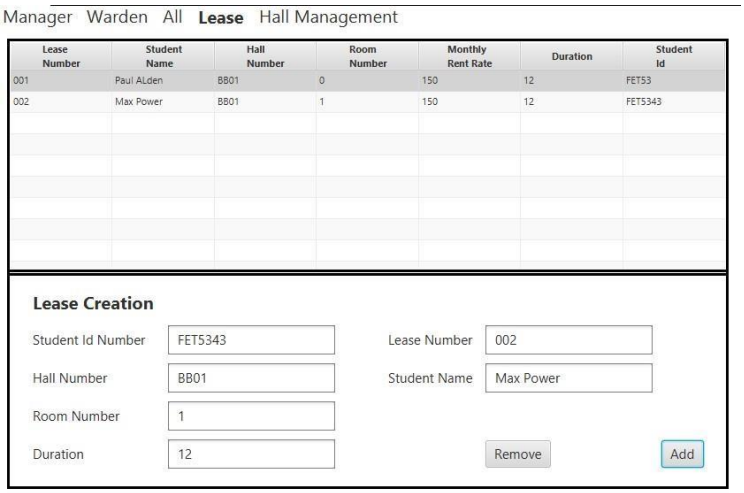

UWE Bristol Accommodation System
Menu
Manager Warden All Lease Hall Management
Hall Name
Hall Number
Hall Address
Room Count
Manager
Warden
Rent Rate
BroadQuay House
BB01
BS5 1DY
2
Danial Watkins
Bob Alden
BroadQuay House
BB012
BS5 1DY
10
Max Power
Paul Alden
Hall Management
Entry already Exsists!
Hall Number
Hall Name
Hall Address
Room Count
Manager
Warden
Rent Rate
BB012
BroadQuay House
BS5 1DY
10
Max Power
Paul Alden
150
Update
Add


Manager Warden All Lease Hall Management
Lease Number
Hall Name
Hall Number
Room Number
Monthly Rent Rate
Duration
Student Name
Occupancy Status
Cleaning Status
BroadQuay House
BB01
0
1
2
150
Unoccupied
Clean
Hall Status
Hall Name
Hall Number
Room Number
Duration
Lease Number
Student Name
Occupancy
Delete
Update


Manager Warden All Lease Hall Management
Lease Number
Hall Name
Hall Number
Room Number
Monthly Rent Rate
Duration
Student Name
Occupancy Status
Cleaning Status
00001111
BroadQuay House
BB01
0
1
2
150
12
Paul Alden
Occupied
Unoccupied
Clean
Hall Status
Hall Name
Hall Number
Room Number
Duration
Lease Number
Student Name
Occupancy
Delete
Update


Manager Warden All Lease Hall Management
Lease Number
Hall Name
Hall Number
Room Number
Monthly Rent Rate
Duration
Student Name
Occupancy Status
Cleaning Status
00001111
BroadQuay House
BB01
0
1
2
150
12
Paul Alden
Occupied
Unoccupied
Dirty
Clean
Cleaning Status
Lease Number
Hall Name
Hall Number
Room Number
Student Name
Occupancy
Cleaning Status
Apply





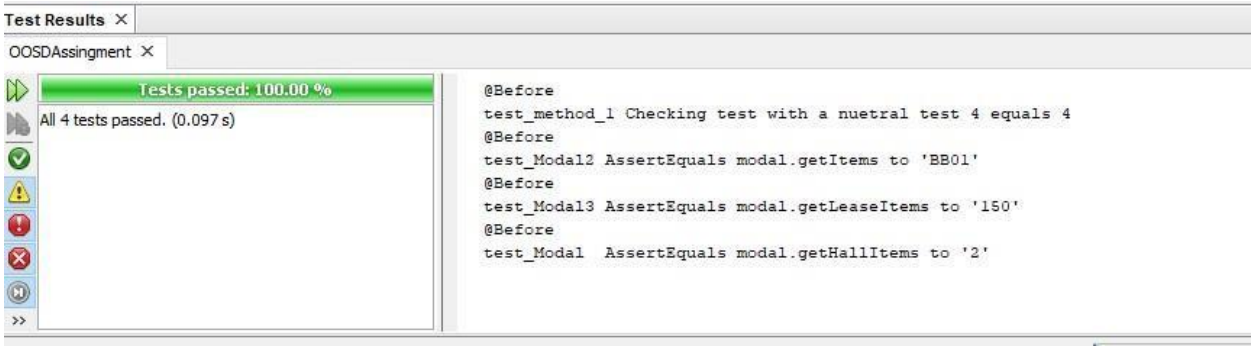
Junit Testing:

Focused on the table view modals.

Asserting that table view related functions work as specified and correct classes are called.

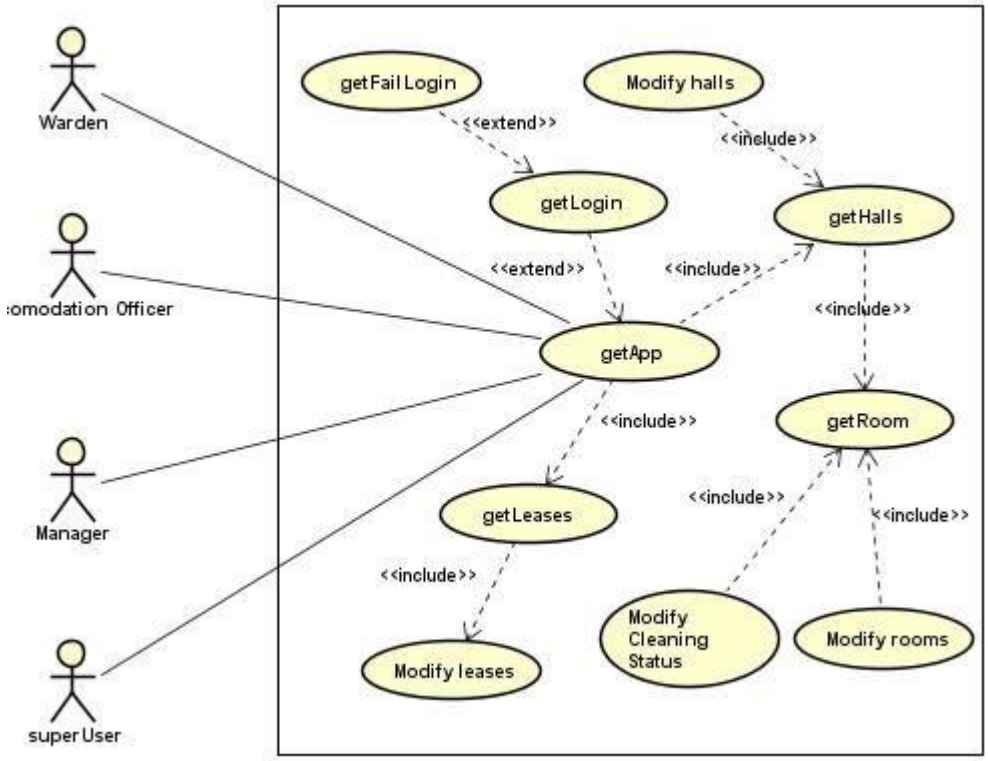
```
@Test      public void
test_method_1() {
        System.out.println("test_method_1 Checking test with a nuetral test 4 equals 4");          assertEquals(4,
4);
    }
     // AppTest of test_Modal method, Testing the hall modal list
     @Test      public void test_Modal(){
        System.out.println("test_Modal  AssertEquals modal.getHallItems to '2' ");
assertEquals(modal.getHallItems().get(0).getRoomCount(), "2");
    }



// AppTest of test_Modal2 method, Testing the Table data modal (getItems)
    @Test      public void test_Modal2(){
```

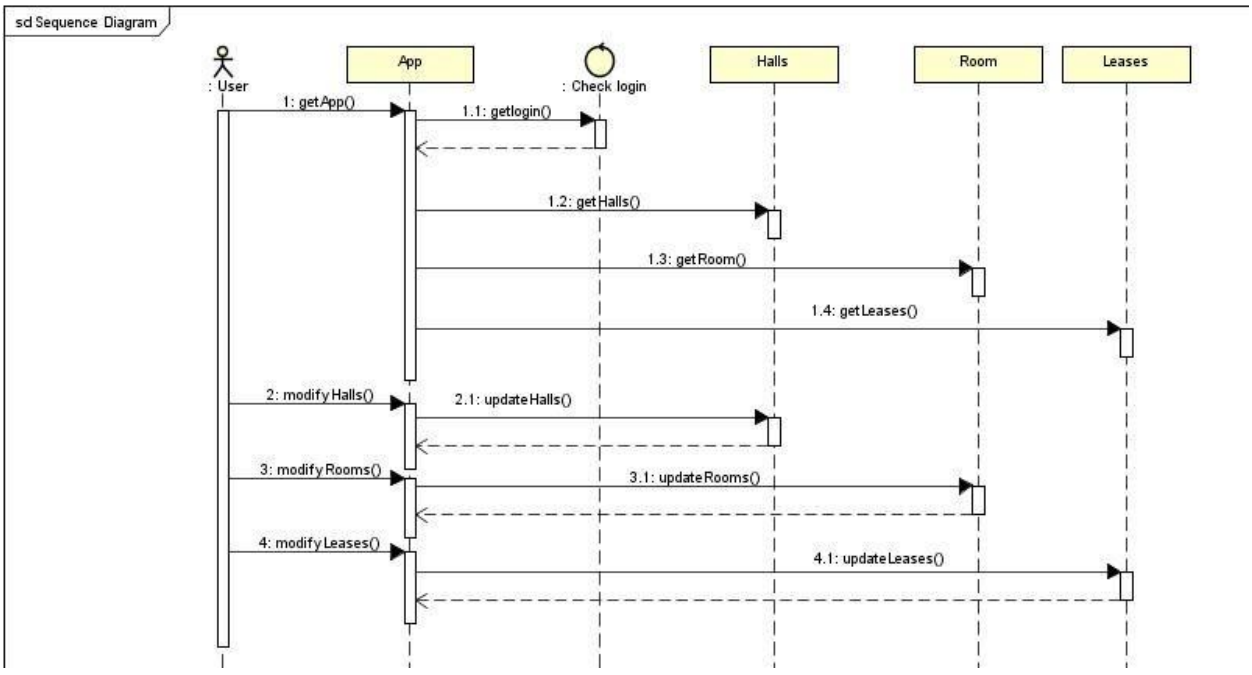
```
    System.out.println("test_Modal2 AssertEquals modal.getItems to 'BB01'");
assertEquals(modal.getItems().get(0).getHallNumber(), "BB01");
  }

   // AppTest of test_Modal3 method, Testing the lease modal list
  @Test    public void test_Modal3(){
    System.out.println("test_Modal3 AssertEquals modal.getLeaseItems to '150'");
assertEquals(modal.getLeaseItems().get(0).getRentRate(), "150");
  }
```



Diagrams

Class diagram is too big to be put on a picture, However the Astah for the diagram is provided.


Warden
:omodation Officer
Manager
super User
getFail Login
Modify halls
<<extend>>
<<inelude>>
getLogin
getHalls
<<extend>>
<<include>>
<<include>>
getApp
<<include>>
getRoom
getLeases
<<include>>
<<include>>
<<include>>
Modify Cleaning Status
Modify rooms
Modify leases


sd Sequence Diagram
: User
App
: Check login
Halls
Room
Leases
1: getApp()
1.1: getlogin()
1.2: getHalls()
1.3: getRoom()
1.4: getLeases()
2: modifyHalls()
2.1: updateHalls()
3: modifyRooms()
3.1: updateRooms()
4: modifyLeases()
4.1: updateLeases()