# UFCFB6-30-2 Group Coursework assignment OOSD “UWE Accommodation System”

# Adwait Chhetri, George Jones & Jamie Mills

## Project Description – Describes how the idea is turned into a project

The University of the West of England (UWE) has several on-site halls for accommodation of students that require it. In managing these halls, there are several different people involved in keeping them up to standard, and performing management manually would take too long. The project will involve creating a system to allow all users of the on-site accommodation, namely, the students, hall wardens and hall managers, to manage, lease, apply and otherwise view the details of the accommodation using a simple graphical user interface (GUI). Depending on the permission level of the user, the user will be able to perform tasks related to their role in the system, the details of which are as follows;

* Student – View halls of residence, apply for rooms
* Hall Warden – View details of halls and rooms, change cleanliness status
* Hall Manager – View details of halls, rooms, and applications. Modify hall and room details, approve or deny application requests.

## Use Case

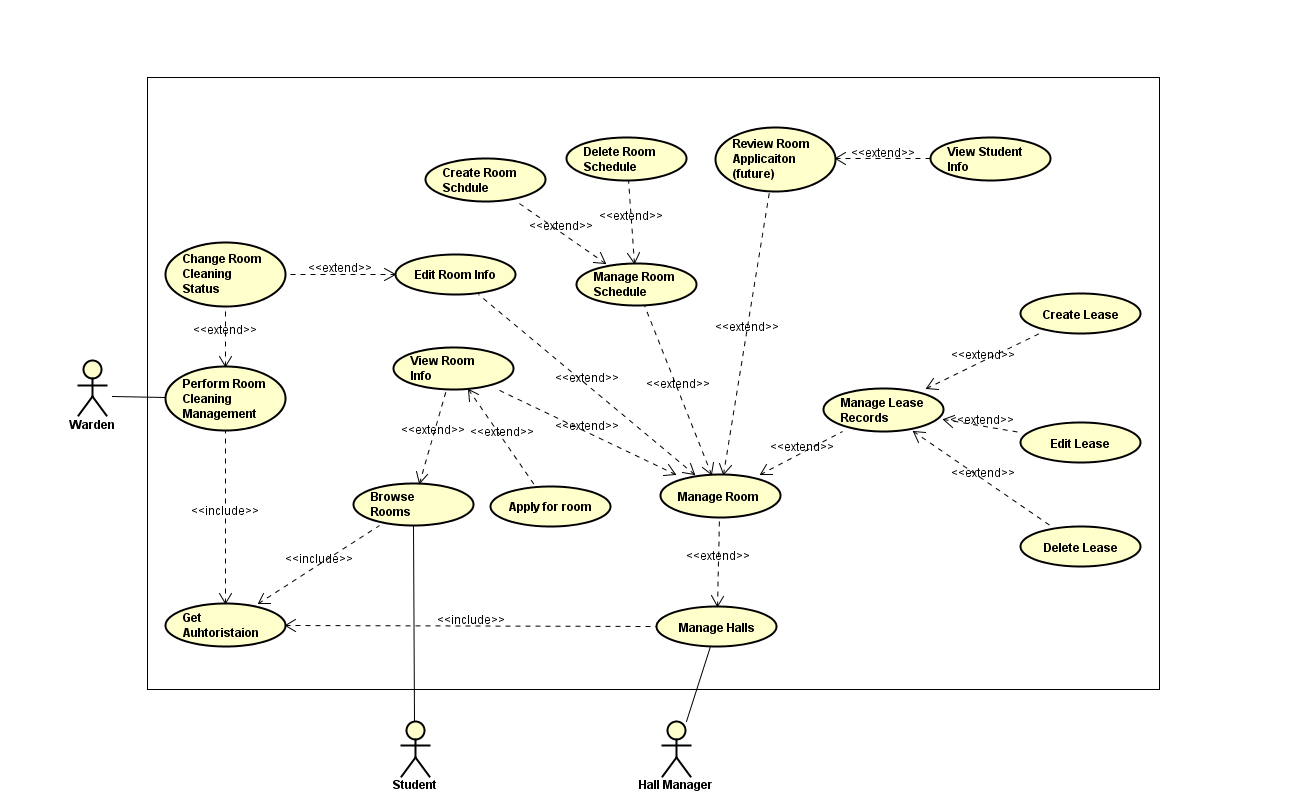


Figure 1. Use Case Diagram for UWE Accommodation System

The Use case contains three actors who will be using the system, namely, Students, Hall Managers, Wardens. The use case shows that all three actors must get authorization before they can access the system, depending on their user access the can use certain features. Hall managers have the authority manage lease records and schedule rooms, whereas wardens only have access to change the room cleaning status and student to view rooms and apply for rooms.

## Class Diagram

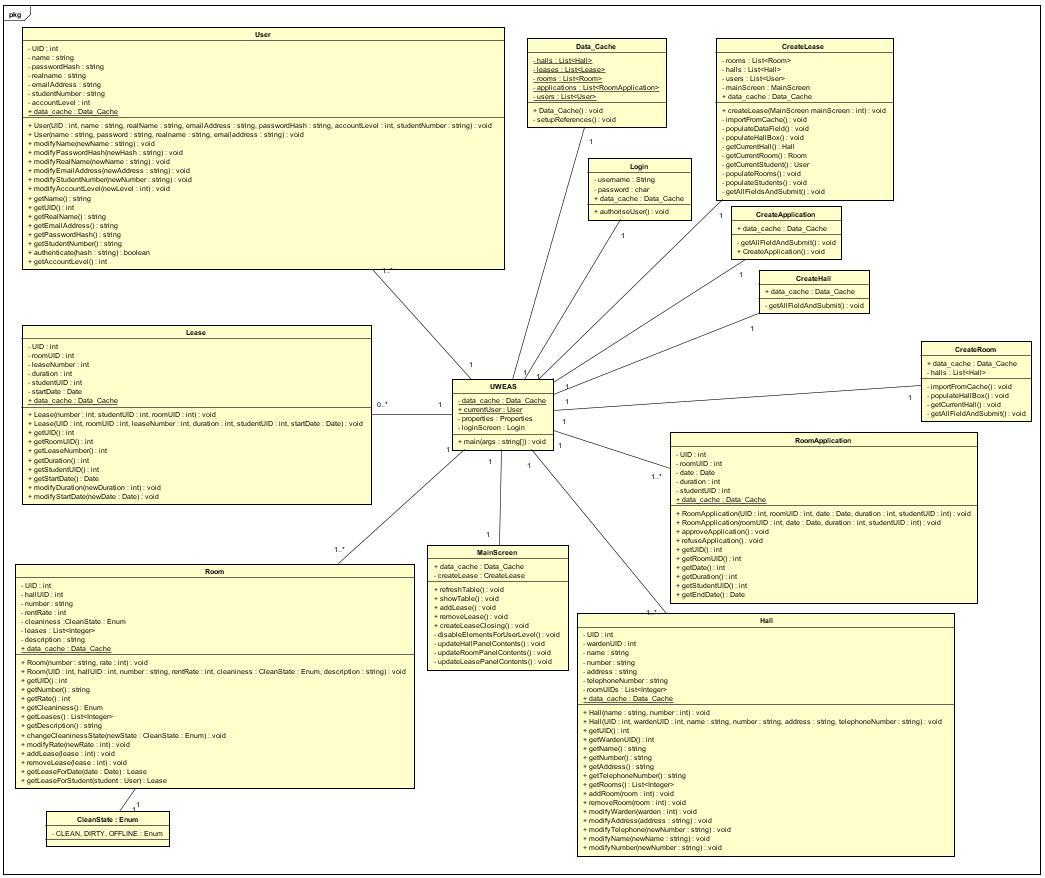


Figure 2 Class Diagram for UWE Accommodation System

Figure 2 contains all the classes in the UWE accommodation system. There is the main class called UWEAS which connects to the database and opens an instance of LoginGUI, after login is successful, the MainScreenGUi is displayed. Data\_Cache class is used to talk with the database through database\_controller. User, Lease, Room, RoomApplication and Hall classes are objects used within the system which are stored into database through the data cache class.

## Sequence Diagram

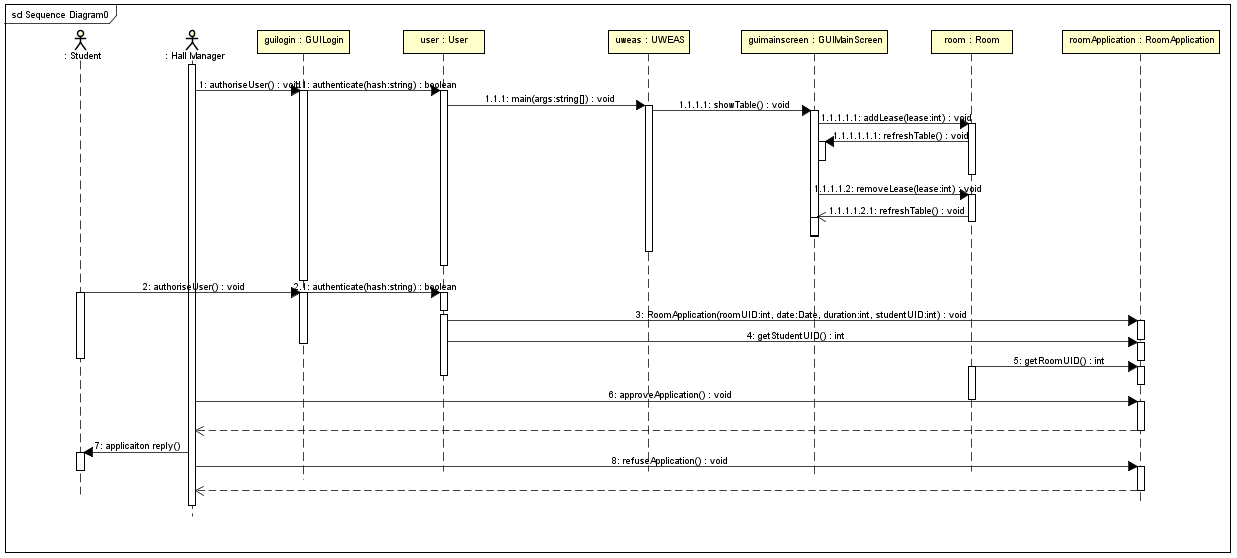


Figure 3 Sequence Diagram Application Process

The above sequence diagram represents an instance of Student applying for a room and room manager responding to the application request. The system requires all users to login, depending on the login details stored on the database students, wardens and hall manager have different access. Once hall manager is logged in, he can create or remove leases in the chosen room. Student applies for an application which gets details from the user class on the chosen room, then the hall manager decides whether to accept the application or decline it.

## Agile Practices

Agile development is an approach to software development. Agile development is the opposite to a plan driven approach where the software development process is well documented and planned, whereas agile is an iterative process where design, implementation and testing is a cyclic process. In agile development, a developer can go back to design phase whilst another developer is working on the code updating both design and code simultaneously. Agile has a few practices that makes it a unique, compared to traditional software development approaches. Such as, agile modeling, pair programming, test driven development, iterative and incremental development. Agile modeling is good for fast changing environments, it practices are documented continuously throughout project life cycle, the documentation is made late, requirements are executable specifications and information stored in one place to avoid confusion. Pair programming is where two people work in one workstation, driver and observer, one person writes code(driver) and the other checks the code (observer), they switch roles frequently. A disadvantage of pair programming is an increase in man hours compared to workers working separately. Furthermore, the design quality is increased due to different coding backgrounds, and reduces defects in program. Test driven development are short development life cycles, the requirements are specific test cases, once the software passes these test new tests are created. Iterative and incremental has their phases which are repeated until the final artefact is complete. Agile practices also include a single coding standard used by the all of the programmers.

## Coding

## Testing

Test Strategy:

The Important features that need to be tested in this system are as follows;

* update, delete GUI buttons work as intended
* occupancy status
* cleaning status
* creating a new lease
* deleting previous lease
* the different states shown in the table,
* selecting a lease, warden restricted views (cannot edit)

The type of testing we will follow is first test the functional aspects of the system making sure that the systems works as intended follow up with UI testing making sure that the GUI works as intended then load/stress testing checking that any errors that come up with wrong or large values are solved. Most of these testing will be done manually as we are following agile practices, we will focus more on the test plan and as to bugs in the program we plan on fixing them manually oppose to automatic bug fixing. The test entry criteria will revolve around the important features that need to be tested for example with the occupancy testing the test entry condition will be, occupancy status is occupied and exit condition will be to solve the error/ bug. We will be tracking the testing through with the test plan below.

Test Plan:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Purpose | Expected Result | Actual Result |
| Occupancy status is occupied | To create a lease, occupancy status must be unoccupied. | Appropriate error message, “Lease already exists” |  |
| Occupancy status is occupied | To delete a lease, occupancy status must be occupied | Deletes selected lease |  |
| Occupancy status is unoccupied | A lease can only be created when occupancy state is unoccupied | Create a lease |  |
| Occupancy status is unoccupied | A lease can only be deleted when occupancy status occupied | Error Message (Lease does not exist) |  |
| Occupancy status is unoccupied and Room status is offline | A lease can only be created when room is unoccupied and not offline. | Error message (Room is offline, cannot create lease) |  |
| Room status is offline | To create a lease room status must be clean or dirty and unoccupied. | Error message pop up. |  |
|  |  |  |  |
| Warden restricted views | Warden can only change the cleaning status. | Appropriate error message when clicked on delete, add lease, no response on text fields. |  |
| Creating a new lease(GUI) | After creating a lease, it should display new lease in the table. | Displays new lease on table. |  |
| Deleting old lease | After deleting a lease, it should remove lease number and student name in the table. | Displays empty lease number and student names in table. |  |
| Table updates as values are changed | After changing a value in the table; room status, new lease, delete lease, it should display the result in table. | Displays appropriate value according to values changed. |  |
| Occupancy status changed | To change the occupancy status from occupied to unoccupied when old lease deleted | Occupancy status change depending on the lease. |  |
| Check User login with valid data | To check the login GUI pulls methods from user class to verify users. | Proceed into the main screen GUI. | Proceeds into the Main Screen GUI |
| Check User login with invalid data | To check the login GUI pulls methods from user class to verify users. | Display invalid login details | Displays invalid login details |
| Hall manager edit and view details | To check Hall manager permissions in the system. | Allowed access to change and view details of leases and students. | Access is allowed. |
| Row selection, gets values from row | To check if the values are same as text field | Values on selected row displayed in text field |  |
| GUI side pane | Check whether GUI panel change works | Error message if access not allowed |  |
| Add lease button | To create a lease, lease panel must have values | Error message “please enter all fields” |  |
| Student Name valid values | Checks if method are working | No change expected |  |
| Student Name invalid values | To check if the student name takes other data types | Error message “please enter a name” |  |
| Lease number valid values | Checks for valid lease number | Message” Lease number updated” |  |
| Lease number invalid values | Checks for null or non-integer values | Error message “please enter a number” |  |
| Add new lease gui(student drop down list) | New student is added to drop down list | New student in student name drop down list |  |
| Lease number stress test | Largest value for lease number | Error message “lease number too high” |  |
|  |  |  |  |