

**GROUP ASSIGNMENT**

**TECHNOLOGY PARK MALAYSIA**

**CT044-3-1-IOOP  
INTRODUCTION TO OBJECT ORIENTED PROGRAMMING**

**APD-APU1F2011CS(CYB)-CGD-CS(IS)-MMT-MMT(VRAR)-SE**

**HAND OUT DATE: 17 JUNE 2021**

**HAND IN DATE: 20 September 2021**

**WEIGHTAGE: 35%**

page1image16720128

**INSTRUCTIONS TO CANDIDATES:**

**1  Submit your assignment at the administrative counter**

**2  Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing)**

**3  Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld**

**4  Cases of plagiarism will be penalized**

**5  The assignment should be bound in an appropriate style (comb bound or stapled).**

**6  Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**

**7  You must obtain 50% overall to pass this module.**

**CT044-3-1  
INTRODUCTION TO OBJECT ORIENTED PROGRAMMING**

**TITLE: UNIVERSITY ACCOMMODATION MANAGEMENT SYSTEM**

**INTAKE CODE:** APD1F2011CS(IS)/ APD1F2011(SE)

|  |  |  |
| --- | --- | --- |
| **Full Name** | **Student ID No.** |  |
| **ESLAM MAGDY REZK EBRAHIM HASSANIN** | **TP062816** | **1** |
| **Seitkerey Dinmukhamed** | **TP057390** | **2** |
| **ANAS YASSER MOHAMMED ABDELFATTAH HASSABOU** | **TP062438** | **3** |

**Date Assigned : 16 June, 2021**

**Date Due: 20 September, 2021**

Table of Contents

[Diagrams 3](#_Toc82198667)

[1. Use-case diagram (UML) 3](#_Toc82198668)

[2. class diagram 4](#_Toc82198669)

[Object-oriented programming concepts 5](#_Toc82198670)

[1. Classes 5](#_Toc82198671)

[2. Methods 6](#_Toc82198672)

[**2.1** **AddStudent** 6](#_Toc82198673)

[**2.2** **Check Availability for Assign rooms** 6](#_Toc82198674)

[**2.3** **Warden Login** 7](#_Toc82198675)

[3. Objects 8](#_Toc82198676)

[**3.1** **Get Student Information** 8](#_Toc82198677)

[**3.2** **Change Room SQL Lite** 8](#_Toc82198678)

[Test Plan and Test Cases 8](#_Toc82198679)

[Conclusion 9](#_Toc82198680)

[References 9](#_Toc82198681)

[Workload Matrix 9](#_Toc82198682)

**Table of Figures**

[Figure 1: UML 4](#_Toc82199130)

[Figure 2: UML class diagram 5](#_Toc82199131)

[Figure 3: Add student method 7](#_Toc82199132)

[Figure 4: Check Availability for Assign rooms method 8](#_Toc82199133)

[Figure 5: Warden login method 8](#_Toc82199134)

[Figure 6: Add student object 9](#_Toc82199135)

[Figure 7: SQLITE CHANGE ROOM 9](#_Toc82199136)

# **Diagrams**

## **Use-case diagram (UML)**

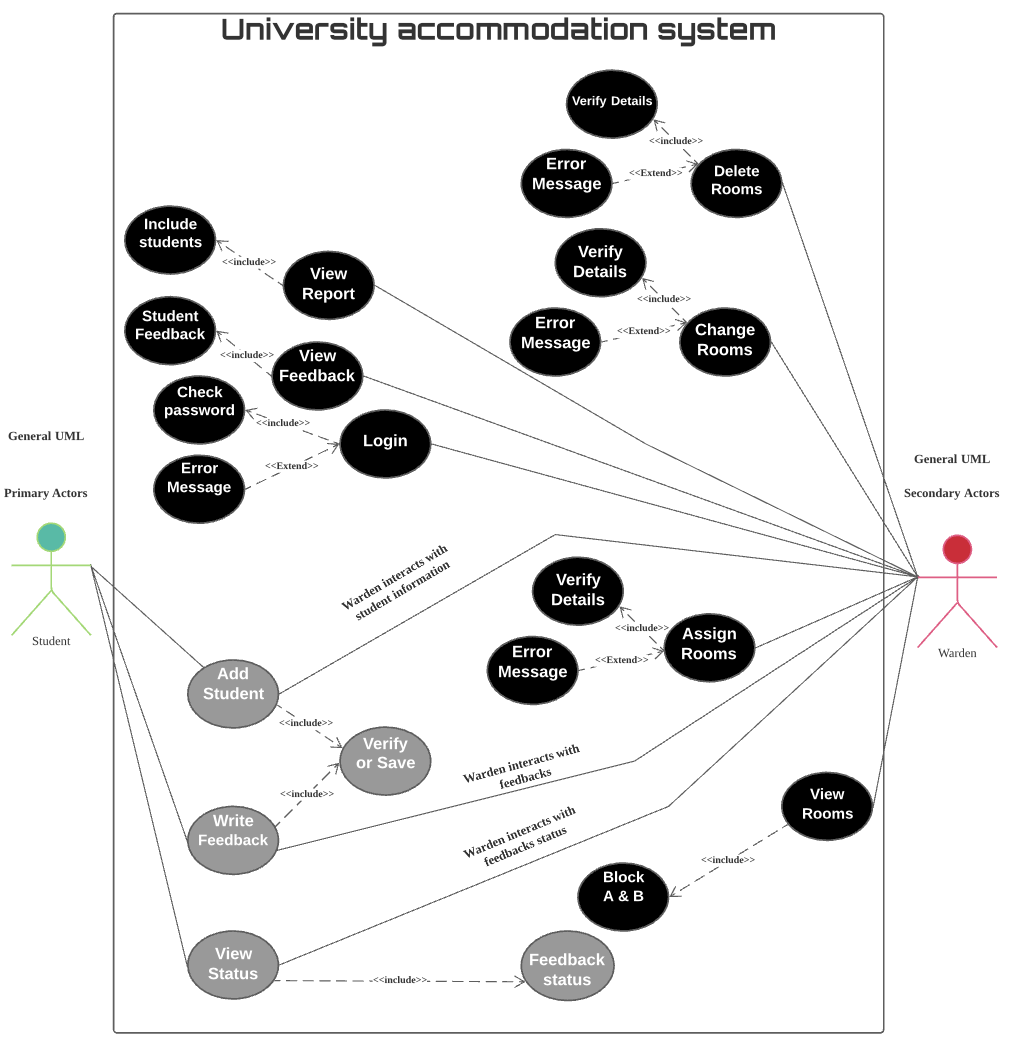


Figure 1: UML

## **class diagram**

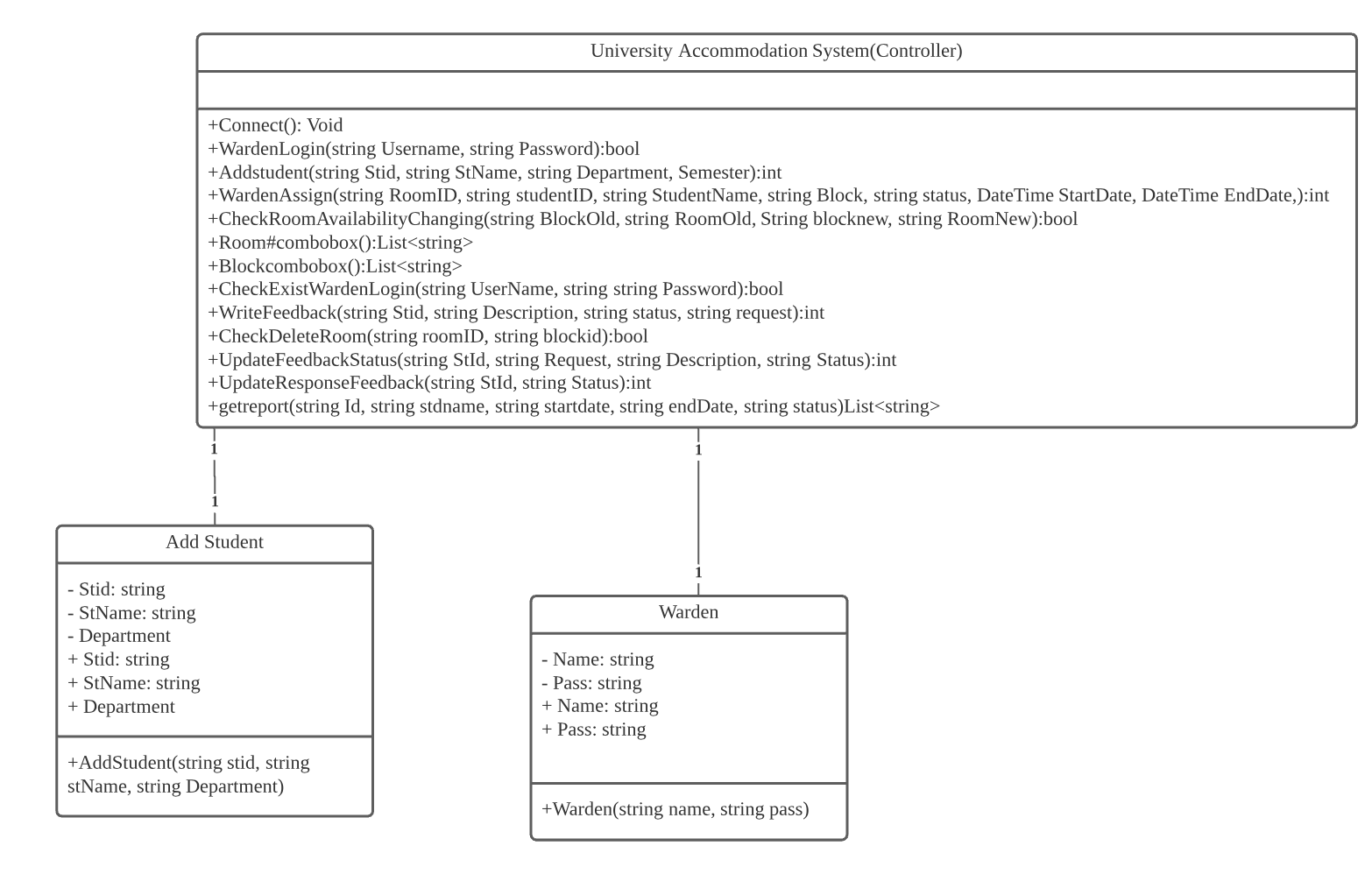
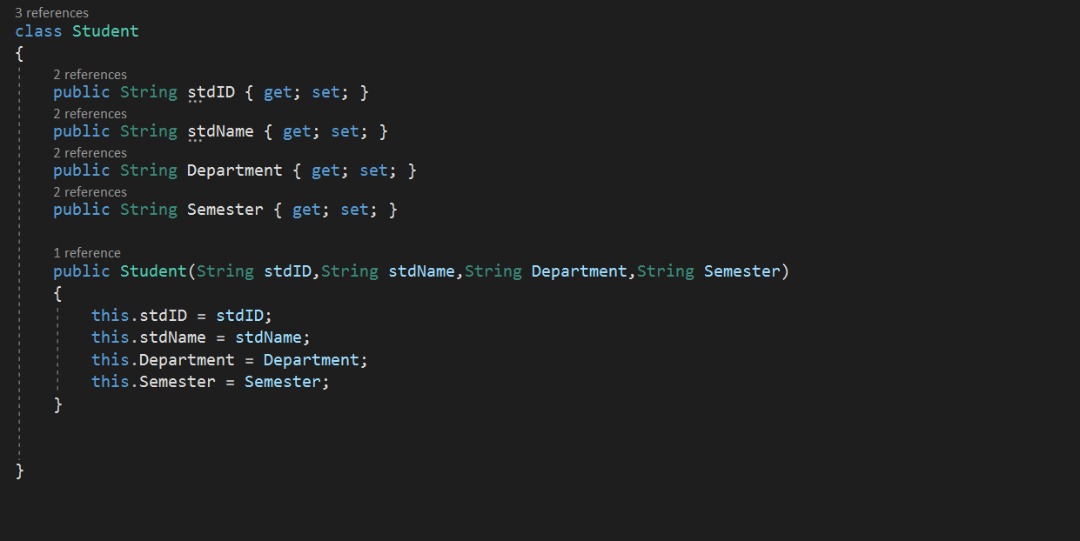


Figure 2: UML class diagram

# **Object-oriented programming concepts**

## **Classes**



Properties and encapsulation, which means hiding important data that is of paramount importance, or in a more precise sense, sensitive data. It must also be noted that the variables or rights are private or public. This must be specified at the beginning, and ways must also be provided for public users to access the field through the feature. And then it is updated (w3schools, n.d.).

A class is used to create objects, and is provided with initial values, such as functions, or methods for users. The name of the class is used, the same as the name used by the template, and the name of the constructor is also used. These concepts are often confused (wikipedia, n.d.).

So, there we defines our field in student class, like, student id, student name, Department of student and semester, to can use it by methods then storing these fields at database by parameters.

## **Methods**

### **AddStudent**

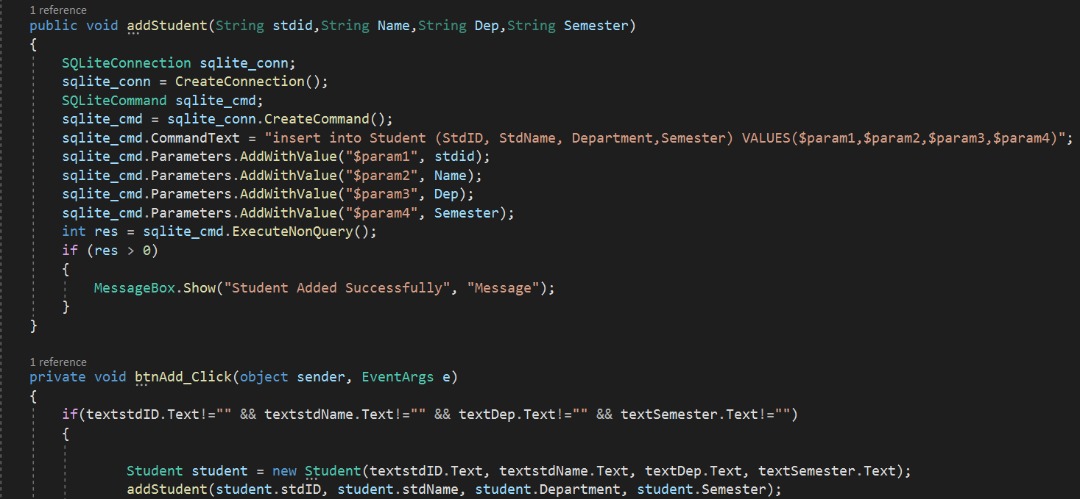


Figure 3: Add student method

When the developer creates objects, he must allocate ways to run and operate those objects, and the method is also bound to the class, when the method and class are linked together is called binding. For example, here the class is adding a student and it consists of several objects, such as the student's number and his semester, also the university department (D., n.d.).

Here in the method, a Database is created, and then the class and the fields in which it is used are used and linked with the database, with specifying the cells that will be linked, whether in the Database or the class, and then the data is stored by the user by the parameter, and the data is stored in the position assigned to it.

### **Check Availability for Assign rooms**

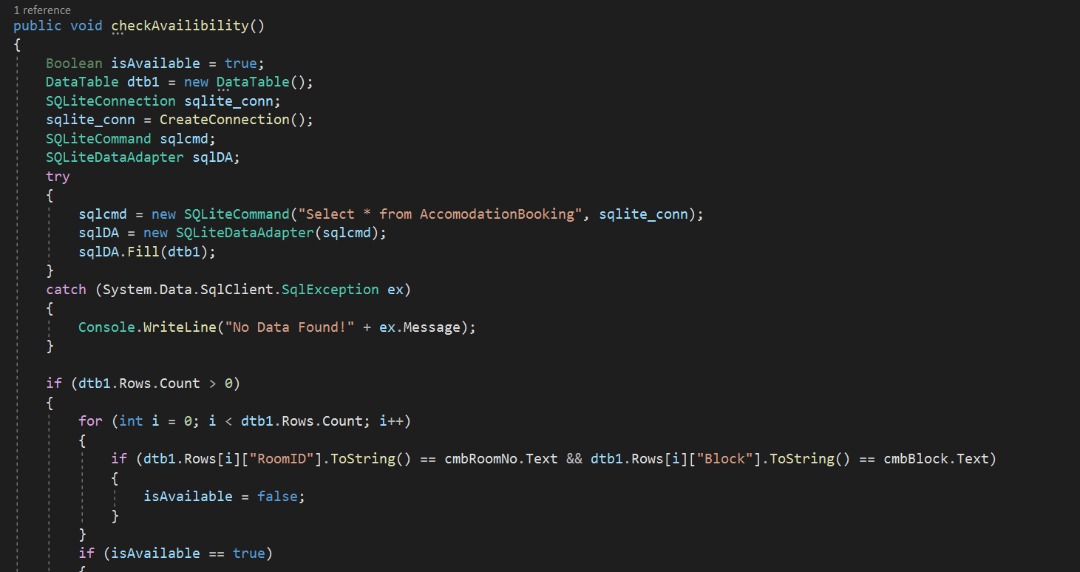




Figure 4: Check Availability for Assign rooms method

Here, the method of Boolean and verifying the integrity of the data entered in the student data registration list is used, and the database is linked to the entered data and the fields in which it is saved, as we said before, is used by the parameter.

Here, storage is done on a logical basis, which is that the start date must be less than the end, and then the rooms stored in the system are checked. If it is empty, a message will appear that the registration has been successfully completed, and then the fields such as student name, department, semester, room number, block number and entry date will be filled in. Exit date and status if it is occupied or empty, But if the data is wrong, a message will be shown that the data is wrong. On the other hand, if the data was previously registered, a message will be shown that the data was previously registered.

### **Warden Login**

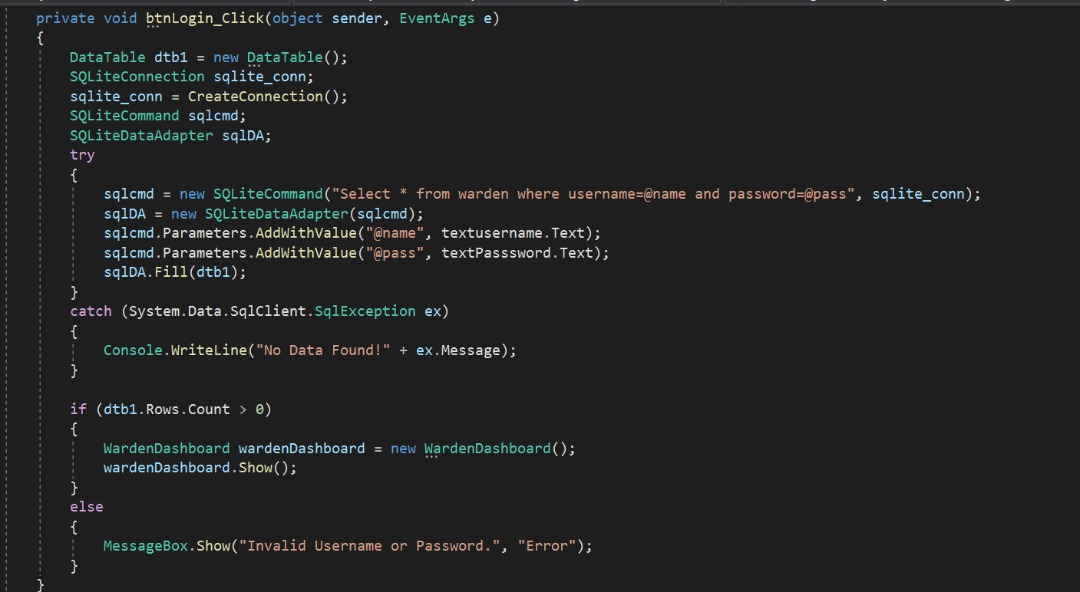


Figure 5: Warden login method

Here, the correctness of the data for the Campus Warden is also checked. If the password and the username match a rule, the database will be entered, and if it is wrong, a message will appear with an error entered data.

## **Objects**

Objects have lots of classes and structures, classes are grouped together and functions also can be modified by field state (cs.uah, n.d.).

The object is characterized by some characteristics, including:

* **State,** which is by storing its state in the generated fields.
* **Defined behaviors**, which is that the object expresses the behavior that will be used and depends entirely on the internal state of the object and then the connection and linking from one object to another object is done by the principle The basic of programming is by encapsulating data.
* **Multiple ways to modify state** must use methods to enumerate its own state and it must be difficult to modify by another object.

### **Get Student Information**

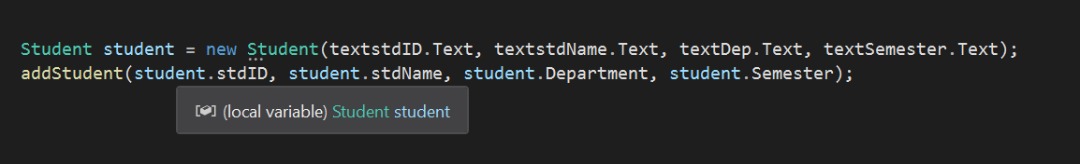
****

Figure 6: Add student object

Student, It is an abbreviation for the method of “add student”, and this object consists of many variables related to adding a student, which is stdID, stdName, Dep, Semester.

### **Change Room SQL Lite**

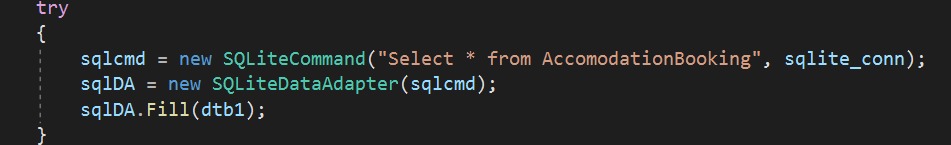


Figure 7: SQLITE CHANGE ROOM

Change Room, It is an abbreviation for the method of “create connection”, and this object consists of one variable “Accommodation Booking”.

# **Test Plan and Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Accommodation university system | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnWarden | To allow warden access to login page | Go directly to warden login page | Successfully directed to dashboard | Only for Warden usage |
| 2 | btnStudent | To Allow the student to enter this menu and enroll at accommodation | Go directly to student dashboard without login | Successfully directed to dashboard | Only for student usage |
| 3 | btnExit | To Exit the application | Exit the application directly | Successfully for shutting down App | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Warden Login | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnLogin | To allow warden access to login page | Test availability of username and password for warden the go directly to dashboard | * Successfully directed to dashboard * If the username or password error Display ”Error at username or password” | Warden usage only |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Warden dashboard | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnViewRooms | Allow warden to access to View room page | Go directly to view room page, and get information about vacant or occupied rooms | Successfully directed to view rooms page and get the details for rooms | Blue color room means that the room is occupied and white color for vacant rooms |
| 2 | btnAssignRooms | Allow warden to enroll the student at the accommodation | Go directly to Assign room page | Successfully directed to Assign room | None |
| 3 | btnExit | To Exit the application | Exit the warden dashboard directly | Successfully exit warden dashboard | None |
| 4 | btnChangeRoom | Allow warden to change room or block to another room or block | Go directly to change room page | Successfully run new form of change room | None |
| 5 | btnDeleteRoom | Allow warden to access delete room form | Go directly to delete room form | Successfully run new form to delete room | None |
| 6 | btnViewFeedback | Allow warden to access view feedback form | Go directly to Feedback form | Successfully run view feedback form | None |
| 7 | btnViewReport | Allow warden to access view report form | Go directly to view report form | Successfully run view report form | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Assign Rooms (warden dashboard) | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnAdd | Allow warden to save all information for student at accommodation Like check in and out | Store all enrolled information in the system of accommodation | * If the information of student correct, so the information will be succussed and saved * If there is any incorrect information Display “There is error in information” | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Change Rooms (warden dashboard) | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnChange | Allow warden to change the room to another rooms also for blocks and save the information | Send the information directly to data base to change the rooms | * If the information is corrected display “the room is changed successfully” * If the information is incorrected display “the changed room is occupied” | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Delete Rooms (warden dashboard) | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnDelete | Allow warden to delete rooms for students | Send the information directly to data base to delete the rooms | * If the information is corrected display “the room is deleted successfully” * If the information is incorrected display “the changed room is already vacant” | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Response feedback (warden dashboard) | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnApprove | Allow warden to approve the requests from students | Send the information for student that his request approved | * If the warden approved the request display “Request updated successfully” | None |
| 2 | btnReject | Allow warden to reject the requests from students | Send the information for student that his request rejected | * If the warden rejected the request display “Request updated successfully” | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| View Report (warden dashboard) | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | Textbox1(autosearch) | Allow the warden to search for student by his name without press any button (its an event and mix between Show data base and data grid view all of things changed when the warden search for the student and the search done from the main database also the data grid will change into database table | The main details inside database will appear | * The name of student will appear | * Write only the first word of student to get all names of students who started with the first letter, if you need specially person write the full name. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Student dashboard | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnAddStudent | Allow student to access to Registration form for accommodation | Go directly to enroll the information for students | Successfully directed to view Registration form | None |
| 2 | btnViewStatus | Allow student to see his status and if his request approved or rejected | Go directly to Status form | Successfully directed to status form | None |
| 3 | btnExit | To Exit the student dashboard | Exit the student dashboard directly | Successfully exit student dashboard | None |
| 4 | btnWriteFeedback | Allow student to access to feedback form | Go directly to feedback form | Successfully run feedback from | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Add student (student dashboard) | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnAddStudent | Allow student to store all his information into database | Send the information for warden | * If the information correct the request display “Add student successfully” * If the fields are vacant display “fill up the form” | None |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Write feedback (student dashboard) | | | | | |
| Test case | Function name | Test objective | Expected Result | Actual Result | Remarks |
| 1 | btnPost | Allow warded to store all of his information for his own request and send it to the warden | Send the information of student to warden | * Send the request successfully | None |

# **Conclusion**

The summary is to learn how to create a system and use the database, also use object oriented, classes, methods, objects and how to connect them, also the project is concerned with developing the ideas and plans needed to implement the project in a very high quality and efficiency, in addition to that learning how to create a class diagram. Increase experience in creating large and great projects using object oriented, Also, this project is completely dependent on the programming language C Sharp. It also takes care of the basic concepts of object oriented such as inheritance.

# **References**

w3schools, n.d. *C# Properties (Get and Set) ❮ PreviousNext ❯.* [Online]   
Available at: https://www.w3schools.com/cs/cs\_properties.php  
[Accessed 10 09 2021].

wikipedia, n.d. *wikipedia.* [Online]   
Available at: https://en.wikipedia.org/wiki/Class\_(computer\_programming)  
[Accessed 10 09 2021].

D., A., n.d. *Object-Oriented Programming: Objects, Classes & Methods.* [Online]   
Available at: https://study.com/academy/lesson/oop-object-oriented-programming-objects-classes-interfaces.html  
[Accessed 10 09 2021].

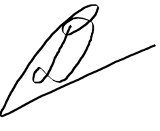
cs.uah, n.d. *The Objects of Object Oriented Programming.* [Online]   
Available at: https://www.cs.uah.edu/~rcoleman/CS307/OO%20Basics/Objects.html  
[Accessed 10 09 2021].

# **Workload Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Assigned Task & Brief Description** | **Assigned Member Name** | **Completion Status / Comment** |
| **1.** | **Warden dashboard + Warden login + SQLite for warden dashboard by contributing with my team to connect all database with each other** | **ESLAM MAGDY REZK EBRAHIM HASSANIN** | **Completion = 100%** |
| **2.** | **Student dashboard + SQLite for student dashboard by contributing with my team to connect all database with each other** | **SEITKEREY DINMUKHAMED** | **Completion = 100%** |
| **3.** | **Absent(didn’t help his team)** | **ANAS YASSER MOHAMMED ABDELFATTAH HASSABOU** | **Absent(didn’t help his team)** |



Workload matrix Signature: 1)



2)