

University Of Zimbabwe

School Of Technology

Hotel Management System C++ Project

BY

LIONEL JOSEPH VAN HEERDEN

This is submitted in partial fulfillment of the requirements for the award of the diploma in software engineering (DAITSE)

**DECLARATION**: I hereby declare that the Hotel Management System is my own work, that has not been submitted for any degree or examination in any other university to my knowledge, and that all sources I have used or quoted have been indicated and acknowledged by complete references.

**ACKNOWLEDGEMENTS,** I would like to thank the following for their contributions to my work:

A big thank you to my family for the support they have given me, I could not have done this without them.

Mr. Solomon Kembo for whole heartedly teaching me programming, web design, how to conduct presentations and for the advice of what to expect from interviews and employers. Because of him, I have decided to learn other programming languages on my own.

And lastly, thank you to the School Of Technology for providing me with the opportunity to study software engineering.

**ABSTRACT**

Guests are the primary targets for hotel businesses, and in order for hotels’ to keep up the demand of constantly inviting guests, they have to have a software program that is fast and reliable, hence the reason for creating this management system. This system was created for Hotel Elizabeth.

The Hotel Management System (HMS) makes it easy for the employer or employees to keep track of all records, add new guests, search for names, addresses, and check room availability.

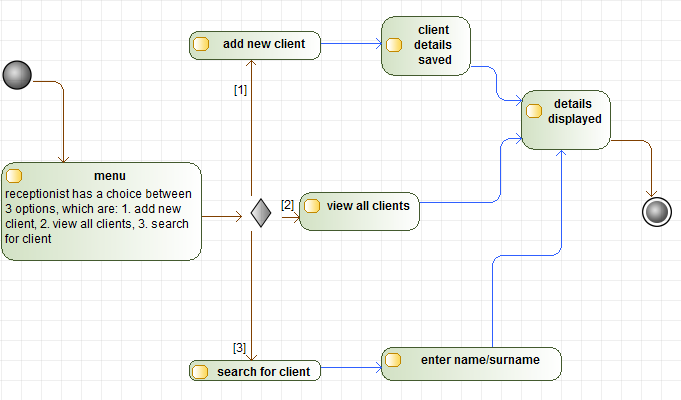
It is simple to understand and can be used by anyone who is not even familiar with a hotel management system. It is user friendly and just asks the user to follow step by step operations by giving easy to follow options. It is fast and can perform many operations for the company.

The code in the program is easy to understand as it has commented lines to explain how the code runs and its specific purpose. The program was initially designed and created to be compact to allow the machine it is running on to operate fast and smooth, and to focus on saving a vast amount of sensitive valuable client information, therefore, the entire program has only a total of 654 lines of code, which makes navigation around the code faster and easier.

Lastly, the program was created to help reduce paper work in a hotel, therefore making it easier to find information that can be readily available and accurate, no need to worry about paper work getting damaged, lost or stolen. The receipts printed by the system are given to the client and the client’s information is safely saved in the database.

OO Analysis

## Activity diagram

Activity diagrams are used to model different aspects of a system. The following diagram shows how the entire system operates, from what it displays to the options available to the tasks that it carries out. 

When the program starts, it displays a menu with a list of options for the user to choose from, which are:

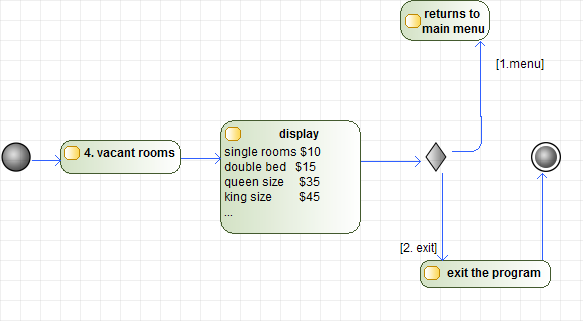
1. Add new guest
2. View all guest records
3. Search for guest
4. Check vacant rooms
5. Exit

If or when the user chooses to add a new guest, the system will prompt the user to enter valid information, such as the guest’s name, address, email address, age, the kind of room service their want and how long their want to rent a room for. When adding a new guest to the system, this information is saved to the “database” (text file), one of which is called ‘Bill’, here the information from all clients’ is saved.

The second option on the main menu is self-explanatory; this will display the entire list of all the guests to ever rent a room in the hotel, from the date they checked in, to their first name. If the user is looking for a certain name, then using this method to look for it is not advisable, as this will display a long list of client information.

The third option is used when a user is looking for something particular, such as a name, surname, email, date, amount or age. The program will then display all the information that is linked to the users input, but if the user wants pin point accurate information, then it’s better to input the whole name of the guest being searched for.

For the fourth option on the main menu list, the following diagram below displays what the program will display if the user chooses to view vacant rooms:

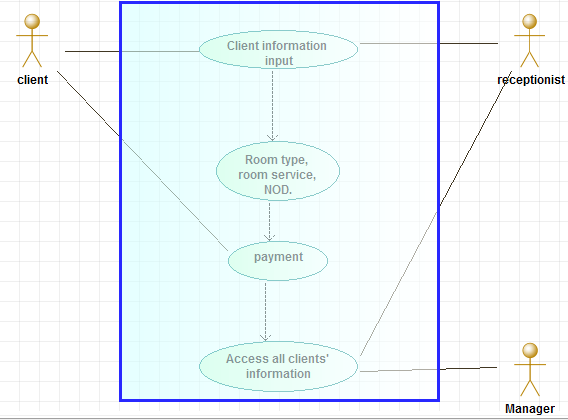


As the user chooses to view the vacant rooms, a list of the occupied and vacant rooms will be displayed, along with the total number of rooms that the hotel has. And just below this short list of rooms, are two options which the user can choose from, namely main menu (which allows the user to return to the main menu) and exit (which leaves the program completely).

How does the fourth option on the list work? Well, the code in the program basically reads through the lines in the text file of the designated folder. What this means is that, each room in the program has its own text file, and every time a user enters a guest into the system, their information is entered into the text file. And so, when this option is selected, the program runs through these text files and counts all the lines of information in the text file and displays the number as ‘number of occupied rooms’ and then this number is subtracted from the number of rooms that the hotel has. For example, there are 10 single bed rooms, if 5 are occupied, then 5 will be subtracted from 10, leaving 5 vacant. All of this kind of information is displayed.

Use case analysis

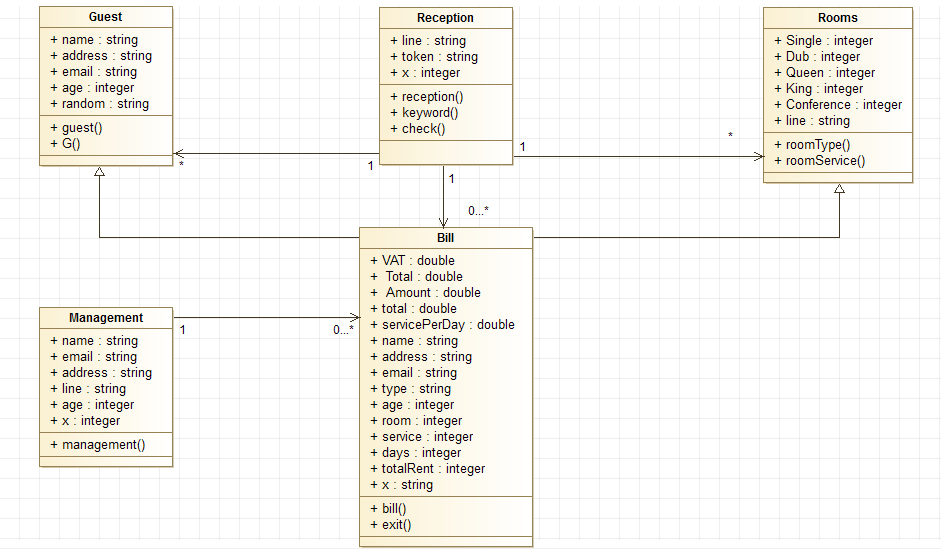
A use case defines a goal-oriented set of interactions between external users and the system under consideration. Thus, a use case scenario is a description that illustrates, step by step, how a user is intending to use the system, essentially capturing the system behavior from the user’s point of view.



The guest/client arrives at the hotel and gives the receptionist their information and chooses what kind of room service they require and what kind of room their want, along with payment. The receptionist enters this information into the system. The guests’ information is able to be view by both the receptionist and the manager. At the end of the client’s visit, they are given their receipt.

Class diagram

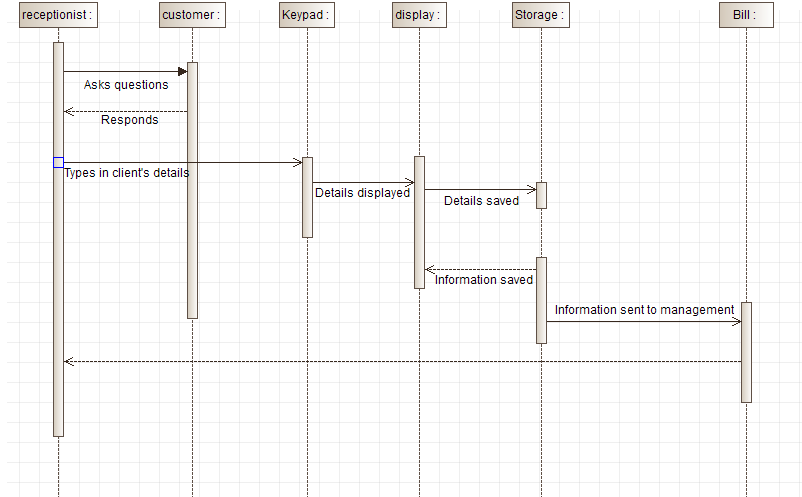
In the class diagram below, the Bill class inherits from the Guest class and the Rooms class. The management class is associated to the Bill class so that the employer has access to all the guest records (such as name, age, email, date checked-in, VAT, and so on). The Reception class is associated with the Rooms class so that, from the reception, the user can check if there are any rooms available before any guests can check in. The Reception class is also associated with the Guest class in case the user wants to search for a particular name.



## Sequence diagram

Sequence diagrams help in the identification of a detailed level of the operations required to implement the functionality depicted in the use case model.

Scenario 1: Receptionist adds a new guest

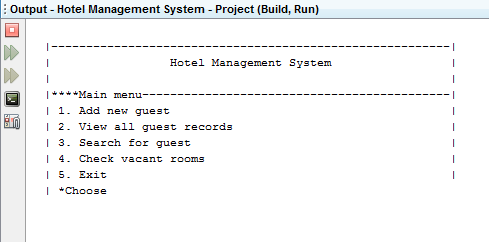
1. Guest(customer) arrives at the hotel and speaks with the receptionist and gives their information(such as name, age, room, etc)
2. The receptionist types the information in, checks room availablity and enters the guest.
3. When the information is saved in the system, a copy of the essential information(such as their name, nights paid for, chosen room, room service and Value Added Tax) is given to the guest on a bill or receipt. 

### **Systems testing**

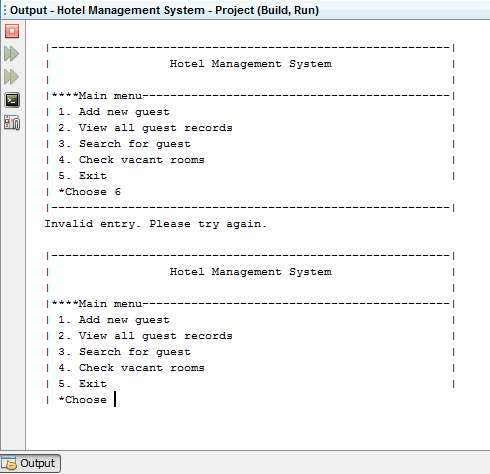
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Test Purpose** | **Test Condition** | **Expected Outcome** | **Actual Result** |
| Menu | Gives the user options such as add new user, view all guests, search for a guest, check vacant rooms or exit. | If the input is incorrect, the system will display an error message and return to the menu screen. | Access is given to the selected option. | The system will display the information as the user has selected. |
| Add a new guest | To ensure that the new guest is added to the system. | The program has to make sure that the user enters information into the system. For example, if a “field” is left empty, the program will make sure that the field is filled. | The new guest is added to the system. | Once information has been entered, the program saves the data and allows the user to continue. |
| Search for guest | The user should receive the information there are searching for. | When the user wants to search for a particular name, age, address (or area), amount paid or date. If what the user is searching for is not in the system, then the program will display an empty screen. | The system must search for the request and display all information containing the given characters. | System displays the requested information if the information is found. |
| Room availability | To search for which rooms are vacant. | If a certain type of room is not available, the user will not be given the option to choose from that type. | The system must display the available rooms and the rooms occupied. | The system will display the rooms vacant, the rooms occupied and the total number of rooms. When there is no space available in a certain type of room, the user is not given the option to select it. |
| View all client guest records | Test to view all client records in the system. | If there are any clients recorded in the system, all information should be displayed. | Information about all clients that have visited the hotel must be displayed. | All information about every client that has been in the hotel is displayed, from their name, age, number of days, to the date their checked-in is all displayed. |

## Test plan screen dumps

**Menu**

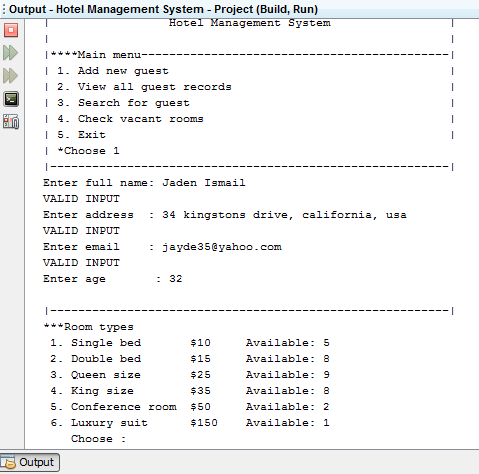


This is the program’s main menu with its few options to choose from. If a user enters a number that is not provided on the list, let’s say they enter 6; the program will respond with an error message and return to the main menu, as shown below.

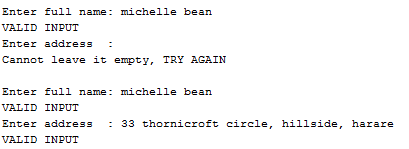


**Adding a new guest**

When adding a new client to the system, the program will display whether or not the information provided or entered in valid or invalid. The example below shows information that was entered is valid and therefore the program proceeds to the room menu.

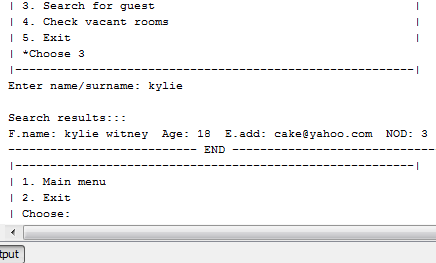


When the user leaves an empty space when entering a client’s details, the program will display an error message and return to a prompt for the user to re-enter the information, just like below:



**Search for a guest**

When the user chooses the option to search for a guest(s), they can choose to enter a name, surname, room type, age, rent, date and the program will display the information that it has concerning the input from the user, an example is provided below:

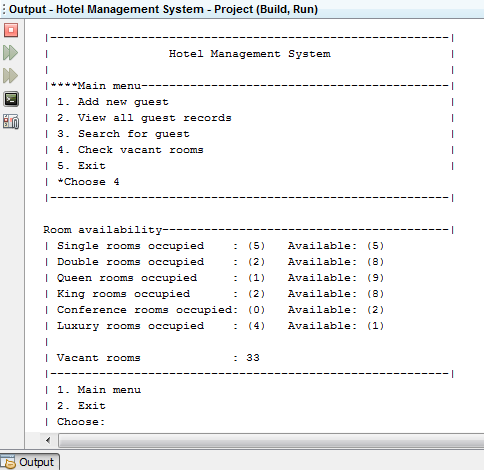


The following diagram shows the rest of the previous client’s information.



**Room availability**

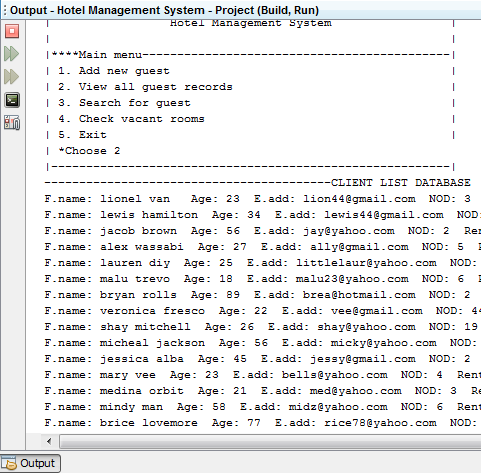
Checking room availability is important for the user (receptionist) to know which rooms are available before they try to check-in a new guest. When the user chooses to check vacant rooms, the program goes to another screen which only provides the user with the information of which rooms are occupied and which are vacant so that the user can inform the guest on what is available. An example is shown below:



This information is kept track of in text files; each room type has its own text to record its activity, such as each time a guest is added to a room, all of the guest’s information is saved to the room type’s text file.

**View all clients’ information**

When the user selects the option to view the entire client’s records, a list of all the guests to ever visit the hotel are displayed. This information is saved on a text file separate from the room text files, but they work the same. Each time a user is entered into a room, their information is automatically saved to the client database. Here is an example below: (which is incomplete because of the large amount of information in the database)



----------------------------------------------------------------END--------------------------------------------------------------------