**Temasek Polytechnic**

**School of Informatics and IT**

**Diploma in Information Technology (IT)**

Software Design Specifications (DS)

**Project Particulars**

|  |  |
| --- | --- |
| **Tutor** | Qi Yutao |
| **Class** | P01 |
| **Project Title** | Delonix Regia Hotel Management System |

**Project Team’s Particulars**

|  |  |
| --- | --- |
| **Matric Number** | **Student Name** |
| 1601553I | Yip Xiu Han |
| 1603625C | Ch’ng Wai Kit Wesley |
| 1601705E | Qamarul Fattah Bin Hamdan |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mm/yy> | <x.x> | <details> | <name> |
| 13/11/2017 | 1.0 | Initial Drafting |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[**1.**](#_gjdgxs) **DISTRIBUTION OF WORKLOAD 4**

[**2.**](#_30j0zll) **ARCHITECTURE DESIGN 4**

[**3.**](#_1fob9te) **USER INTERFACE (UI) DESIGN 4**

[**4.**](#_3znysh7) **PROGRAM DESIGN 4**

[**5.**](#_2et92p0) **DATABASE DESIGN 4**

# **DISTRIBUTION OF WORKLOAD**

*[Determine which members of the team will be responsible for which areas of design. Individual’s responsibilities should be clearly spelt out.]*

*(Who wrote which one)*

|  |  |
| --- | --- |
| **Design** | **Members** |
| Architecture Design | Wesley |
| User Interface (UI) Design | Qamarul |
| Program Design | ALL |
| Database Design | ALL |

# 

# **ARCHITECTURE DESIGN**

*[Describe the proposed system architecture design e.g. 2-tier comprising of client-data? Or 3-tier comprising of client-business-data? Include diagram to illustrate the different tiers in the architecture.]*

*(Go and google, 2 tier systems, 3 tier system, what architecture is your system!)(Do not teach the reader about the tiers, only write the one you are going to do)*

**Multitier Architecture Background**

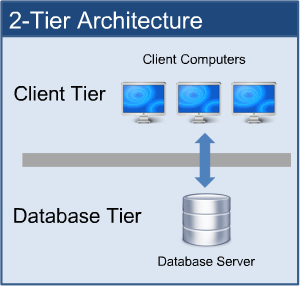
In software engineering, a multitier architecture (often referred to as n-tier architecture) or multilayered architecture is a client–server architecture in which presentation, application processing, and data management functions are physically separated.

The three most common layers in a multitier architecture are:

* Presentation Layer - The Presentation layer which is also known as the client layer which is the top most layer of an application. This layer is used to allow the users to view the application and to access the web pages. The main function of this layer is to communicate with the application layer. The presentation layer passes the information which is entered by the user to the application layer. For example, the login page in a hotel management system will require the user to enter values into the text fields of “Username” and “Password”, of which this information is passed onto the application layer.
* Application/Business Layer - The Application layer which is also known as the Business layer which contains the written business logic. For example, once the user keys in their login information into the provided text fields, the application layer interacts with the data layer which sends the login information to the presentation layer. The application layer provides users with the ability to carry out operations on the application. It also controls the application’s functionality by performing detailed processing. This layer acts as a medium between the presentation layer and the data layer.
* Data Access Layer - The Data Access layer consists of the application’s database. The application layer communicates with the data layer to request for the retrieval of data. It contains methods that connects the database and performs the required functions such as the create, retrieve, update and delete.

**Proposed System Architecture Design**

After careful analysis, our team has decided on implementing the 2-tier architecture design into this hotel management system.



The above is an example of a 2-tier architecture system. It is designed for a client-server communication. The application must be connected to a network, such a Azure SQL database, in order to function properly. The presentation layer and the application layer will be on the client side while the data layer will be on the server side. The “client” will be the program that is running on the hotel management device used by the hotel staffs to capture and upload guest information while the “server” will be the machine that the client gets content from. Many clients would be able to access the information on a single server thus, this architecture design is suitable for a hotel management system.

**Advantages of 2-tier architecture:**

Greater Performance - The database layer and application/business layer is physically close which offers a greater performance hence, allowing response time and communication to happen at a faster rate.

Navigation - User maneuverability (for both staff and developers) with a 2-tier system is easily achieved as applications can be easily developed, modified and maintained.

**Disadvantages of 2-tier architecture:**

Scalability - This architecture lacks scalability as it only supports a very limited number of users. This architecture will not be suitable if Hotel Delonix Regia continues to expand in size as a very limited number of user connections can be supported before application performance is degraded as clients require separate connections and CPU memory to proceed.

Circumstantial usage - Businesses with rapidly changing rules and regulations are not suitable since the database server needs to handle the business logic which slows down database performance.

Alteration of Database Server - As most applications which are used for interaction is dependent on the database structure, this creates an issue if the owners of Hotel Delonix Regia decide to redesign the UI. As the database structure would have to be re-designed as well since it corresponds with the UI.

**Summary**

Hence, the 2-tier architecture design is recommended for the hotel management system as we plan on using a web form.

The presentation and business layer will be installed on the client’s machines that will access the Azure SQL Database so that there will be the least amount of downtime. The advantages of the 2-tier architecture have been discussed above and we have found out that this architecture design is useful for a hotel management system as multiple staff will be able to access and update the system at the same time.

The 2-tier architecture would be able to meet Mr Lim’s requirements and it is less complex than a 3-tier architecture and thus, it can be easily implemented maintained.

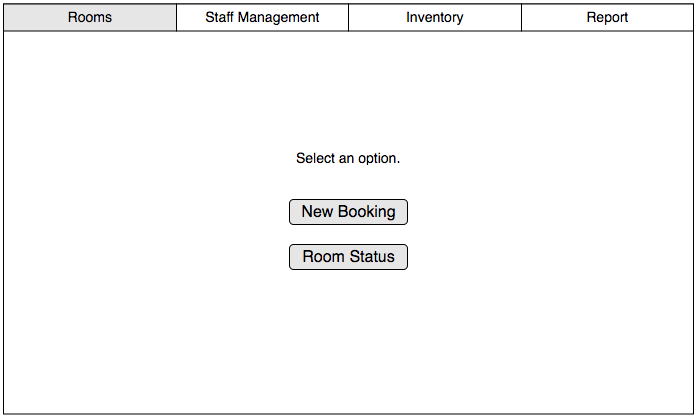
# 

# **USER INTERFACE (UI) DESIGN**

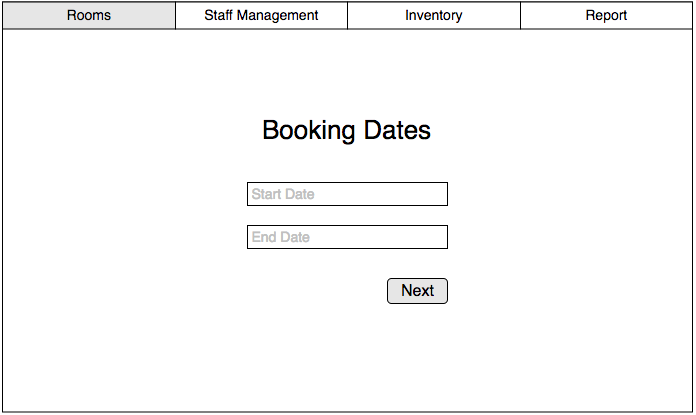
*[Document the user interface design decisions and considerations. This should not be a mere capture of all UI forms but a reflection of UI design considerations. Is web or windows or other forms of UI used? Why did you choose that particular UI? How does the UI design impact usability and why adopt certain UI controls than others?]*

*(Copy Paste the UI Design Prototype from the Software Requirement)*

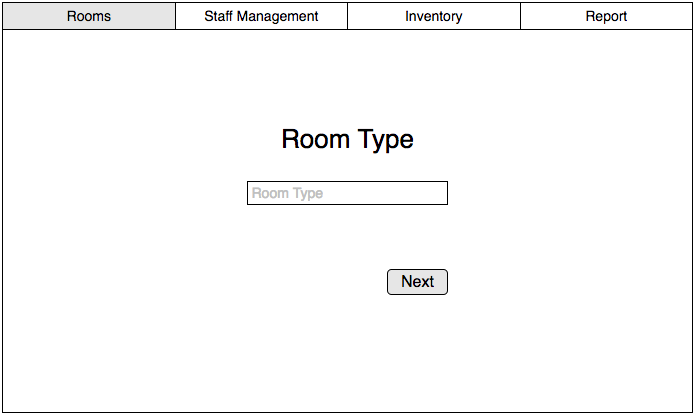
Room



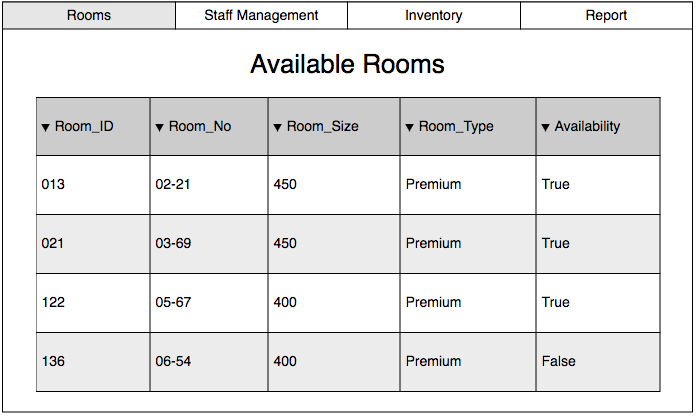
Room Booking



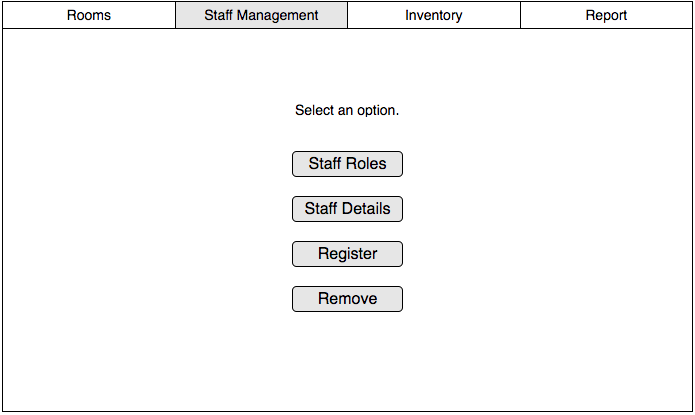
Room Type



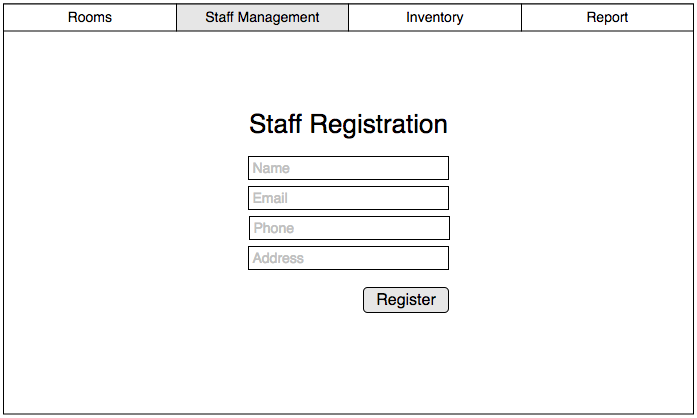
Room Availability



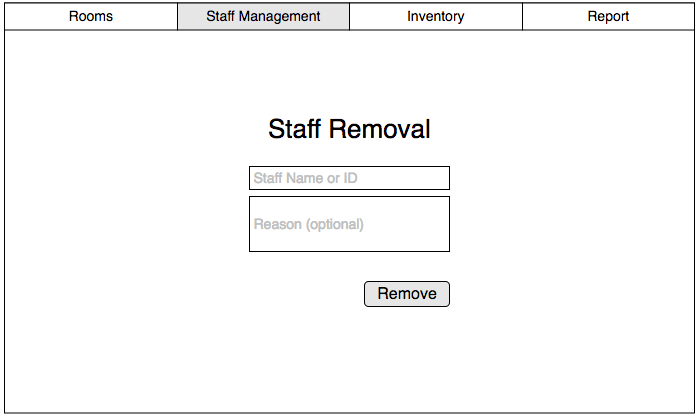
Staff Management



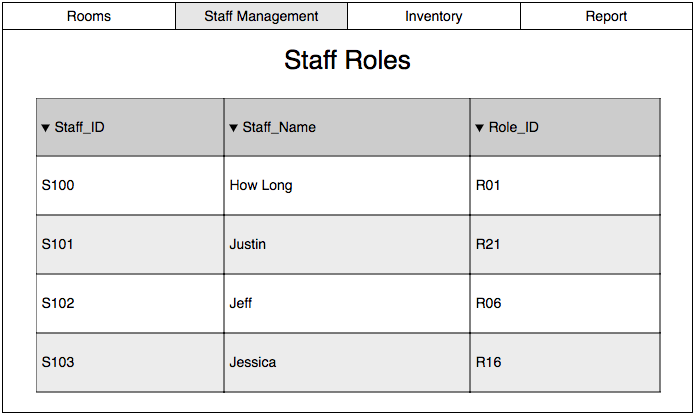
# Staff Registration



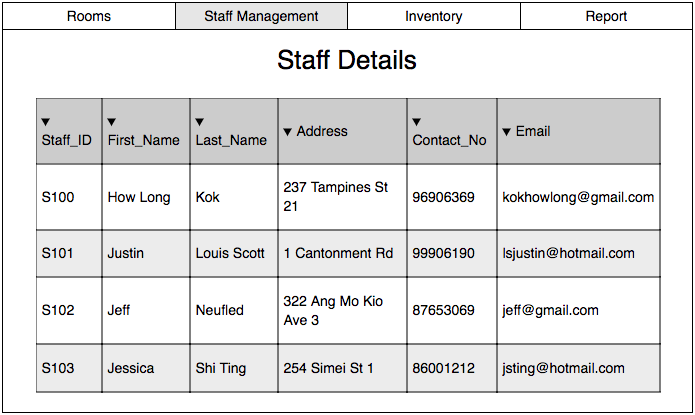
Staff Removal



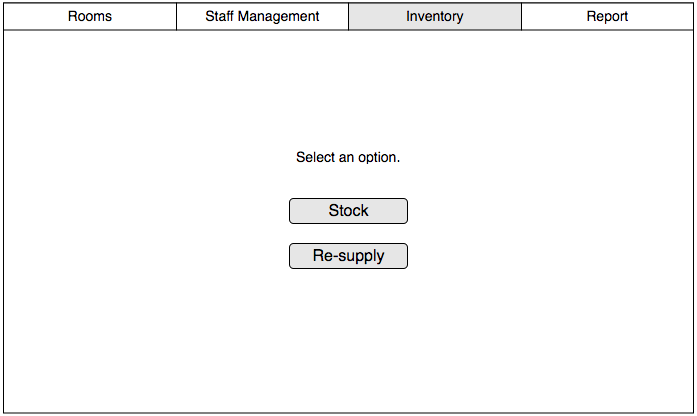
Staff Roles



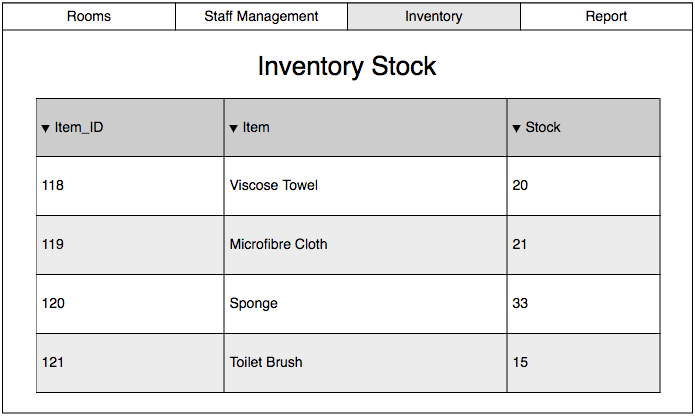
Staff Details



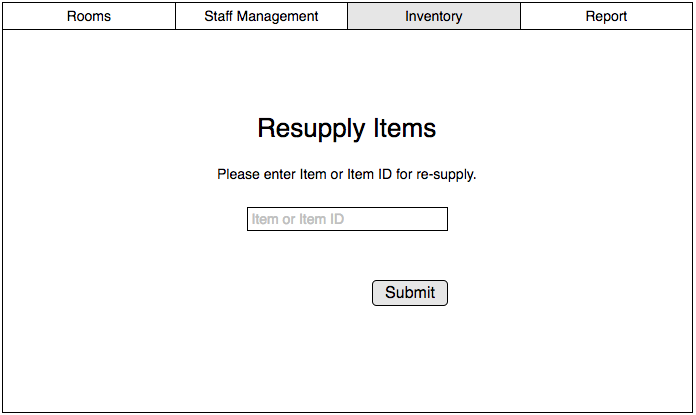
Inventory



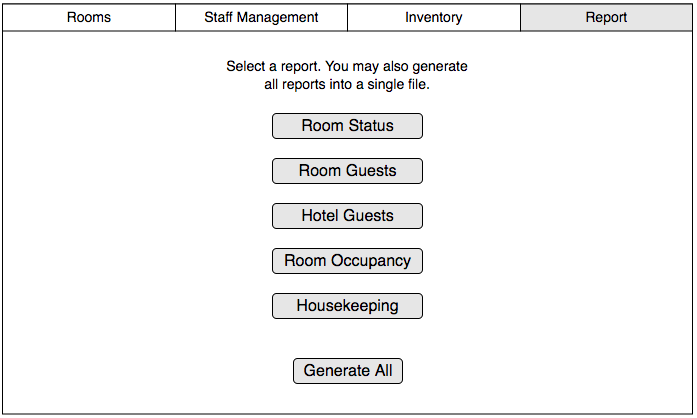
Inventory Stock



Resupply Items



Report



# **PROGRAM DESIGN**

*[Describe how program design is done. What are the programs making up each system/software module?]*

*(For every module : Write the* ***main flow + one alternate flow****)(No use case diagram, no class diagram, only flow)*

The development of the HMS will be split up into two groups, one will be programming the logic and the functions of the system, the other group will be designing the UI. Due to time constraint, these two development tasks will be carried out together, afterwards we will merge the interface with the codes at the end of the software development.  
  
We will be using object oriented way of programming to organise various variables/attributes, and use a database helper class to access the database.

**Module 1 - Room Availability and Booking Module (Wesley):**

**Use Case 1 : Check-In (List of Rooms)**

**Brief Description**

This use case occurs when the End Users / Management Users / Administrators wishes to check a guest into the hotel management system.

**Actors(s)**

End Users / Management Users / Administrators

**Main Flow**

1. This use case begins when a user selects the check in option on the hotel management system.
2. System displays the check in page.
3. System prompts the user to enter the guest’s room number.
4. User keys in the guest’s room number.
5. System displays the existing bookings for that specific room.
6. User asks for the guest’s name.
7. User checks for booking(s) under the guest’s name.
8. User selects the row that has the guest’s name.
9. User checks the other booking details with the guest (E.g. Name, No. of Guests, Room Details).
10. User presses the check in button.
11. System prompts the user for confirmation.
12. User selects confirm check-in button.
13. System saves the guest’s information and the room status is changed from “Booked” to “Occupied”. The use case ends here.

**Alternate Flow(s)**

Guest name recurs

7.1. User clarifies with guest on other booking details (E.g. Room Details) to get the correct information.

7.2. User continues from step 8.

Guest’s details are non-existent

7.1. User clarifies with guest on whether they have successfully booked a room (likelihood answer will be no).

7.2. User moves onto Use Case 2.

Guest’s details differ from database

7.1. User clarifies with guest on booking details.

7.2. User moves onto Use Case 5.

**Use Case 2: Check-In (Registration)**

**Brief Description**

This use case occurs when the End Users / Management Users / Administrators wishes to register a guest’s booking into the hotel management system.

**Actors(s)**

End Users / Management Users / Administrators

**Main Flow**

1. This use case begins when a user selects the new booking option in the check-in module of hotel management system.
2. System prompts the user to enter the start date and end date the guest wishes to book for.
3. User asks the guest for the booking dates.
4. User keys in the booking dates.
5. System prompts user to select the room type.
6. User asks the guest for their room preference.
7. User selects the guest’s room preference.
8. System displays rooms that are available on the booking dates that coincides with the guest’s room preference.
9. User selects a room number for the guest.
10. System prompts the user to enter the room, customer and booking details.
11. User asks the guest for their details.
12. User keys in the room, customer and booking details.
13. System prompts the user on whether they would require any additional requirements (E.g. Extra Pillow).
14. User asks the guest if they would require any additional requirements.
15. User keys in the special requirements.
16. System prompts the user for payment.
17. User asks the guest if they would like to pay by cash or card.
18. User collects the cash from the guest.
19. System prompts the user to confirm the booking.
20. User selects the confirm booking button.
21. System saves the information the database and changes the room status of the room from “Available” to “Booked”. Use case ends.

**Alternate Flow(s)**

Guest pays by card.

18.1. User prompts the guest to insert card into the payment terminal and enter pin.

18.2. User hands the card back to the guest and clarifies transactional information.

18.3. User continues from step 19.

Invalid information entered (can occur from any step from step 4 onwards)

4.1. User clarifies with the guest again on whatever information is required.

4.2. User re-enters the information and continues with the flow of the use case.

Booking already exists.

20.1. System displays booking information already exists.

20.2. User rectifies the problem by clarifying with guest on whether they have already made a booking and other booking details.

**Use Case 3: Check-Out**

**Brief Description**

This use case occurs when the End Users / Management Users / Administrators wishes to check out a guest from the hotel management system.

**Actors(s)**

End Users / Management Users / Administrators

**Main Flow**

1. This use case begins when a user selects check-out module of hotel management system.
2. System displays form interface and prompts the user to select the room number.
3. User selects the room number.
4. System takes the values of the room number and displays the details on the form interface.
5. User adds new row in items consumed.
6. System takes in the values and shows the row in the form.
7. System will calculate the total amount through the quantity of items consumed E.g. (Minibar) and price.
8. User clicks generate invoice button.
9. System prompts for confirmation.
10. User confirms the action.
11. User keys in the room, customer and booking details.
12. System generates and updates the invoice in the database).
13. System changes the room status from “Occupied” to “Available but not cleaned”. Use case ends here.

**Alternate Flow(s)**

Requested room check-out not in use.

3.1. User clarifies with customer if their desired checkout number of room is correct.

3.2. User repeats step 3.

**Use Case 4: Room (Current Room Status)**

**Brief Description**

This use case occurs when the End Users / Management Users / Administrators wishes view the current room status.

**Actors(s)**

End Users / Management Users / Administrators

**Main Flow**

1. This use case begins when a user selects room module of hotel management system.
2. System displays form interface and prompts user to enter room number to search.
3. User keys in the room number.
4. System takes in the value of the room number and displays the room details with status on the form interface. Use case ends here.

**Alternate Flow(s)**

Entered room number not found.

4.1. System displays “Room number not found” error message.

4.2. System redirects user to Step 2.

**Use Case 5: Room (Edit Details of Room)**

**Brief Description**

This use case occurs when the End Users / Management Users / Administrators wishes to edit the details of a room using the hotel management system.

**Actors(s)**

Management Users / Administrators

**Main Flow**

1. This use case begins when a management user/administrator selects selects the edit room details option the room module of hotel management system.
2. System prompts the user to enter the room number and room type he/she wishes to search for.
3. Manager/Administrator keys in room number and room type and clicks search.
4. System takes in the values of the inputs and displays the result in the form of a table.
5. Manager/Administrator clicks on the specific row of the table.
6. System displays the details of the room on the “details” tab.
7. Manager/Administrator updates the text fields of the room details and clicks the update button.
8. System prompts for confirmation.
9. Manager/Administrator selects confirm.
10. System saves the changes and updates the database. Use case ends.

**Alternate Flow(s)**

Entered room number not found.

4.1. System displays “Room number not found” error message.

4.2. System redirects user to Step 3.

**Module 2 - Housekeeping and Staff Management Module (Qamarul):**

**Main Flow:**

1. The use case starts when the user (hotel owner) wants to register or remove a staff
2. System prompts user ‘Register’ or ‘Remove’
3. If user selects ‘Register’, go to Register Alternate Flow
4. If user selects ‘Remove’, go to Remove Alternate Flow

**Register Alternate Flow:**

1. System displays a staff registration form
2. User enters staff details, and submits form
3. System updates Staff table

**Remove Alternate Flow:**

1. System displays a form, prompts user to enter Staff ID
2. User enters Staff ID
3. System prompts user to confirm staff removal
4. User confirms and submits form
5. System removes entered Staff ID from Staff table.

**Module 3 - Reporting Module (Xiu Han):**

**Main Flow:**

1. The use case starts when the user (staff) wishes to generate a type of report for the hotel.
2. If user selects the report generation for the status of the rooms in the hotel, go to Room Status Report alternate flow.
3. If user selects the report generation for listing the guests in a room, go to Room Guests alternate flow.
4. If user selects the report generation for listing all the guests in the hotel, go to Hotel Guests alternate flow.
5. If user selects the report generation for checking the statistics of room occupancy in the hotel, Room Occupancy alternate flow.
6. If user selects the report generation for Housekeeping report generation, go to Housekeeping alternate flow.

**Room Status Report Alternate Flow:**

1. User searches for an existing room in the hotel by its room number to modify its status.
2. User keys into the system whether the selected room is vacant, occupied, or scheduled for cleaning.
3. User repeats this process for as many times as needed.
4. User generates the report.

**Room Guests Report Alternate Flow:**

1. User keys into the system the room number of the hotel room.
2. User logs the name of the guests in the room along with other details.
3. User generates the report.

**Hotel Guests Report Alternate Flow:**

1. User selects the rooms by their room numbers to be compiled for the report (to account for the guests in those rooms).
2. User generates the report.

**Room Occupancy Report Alternate Flow:**

1. User selects an existing room in the hotel by its room number.
2. System returns the number of times the room was occupied during that current month.
3. User generates the report.

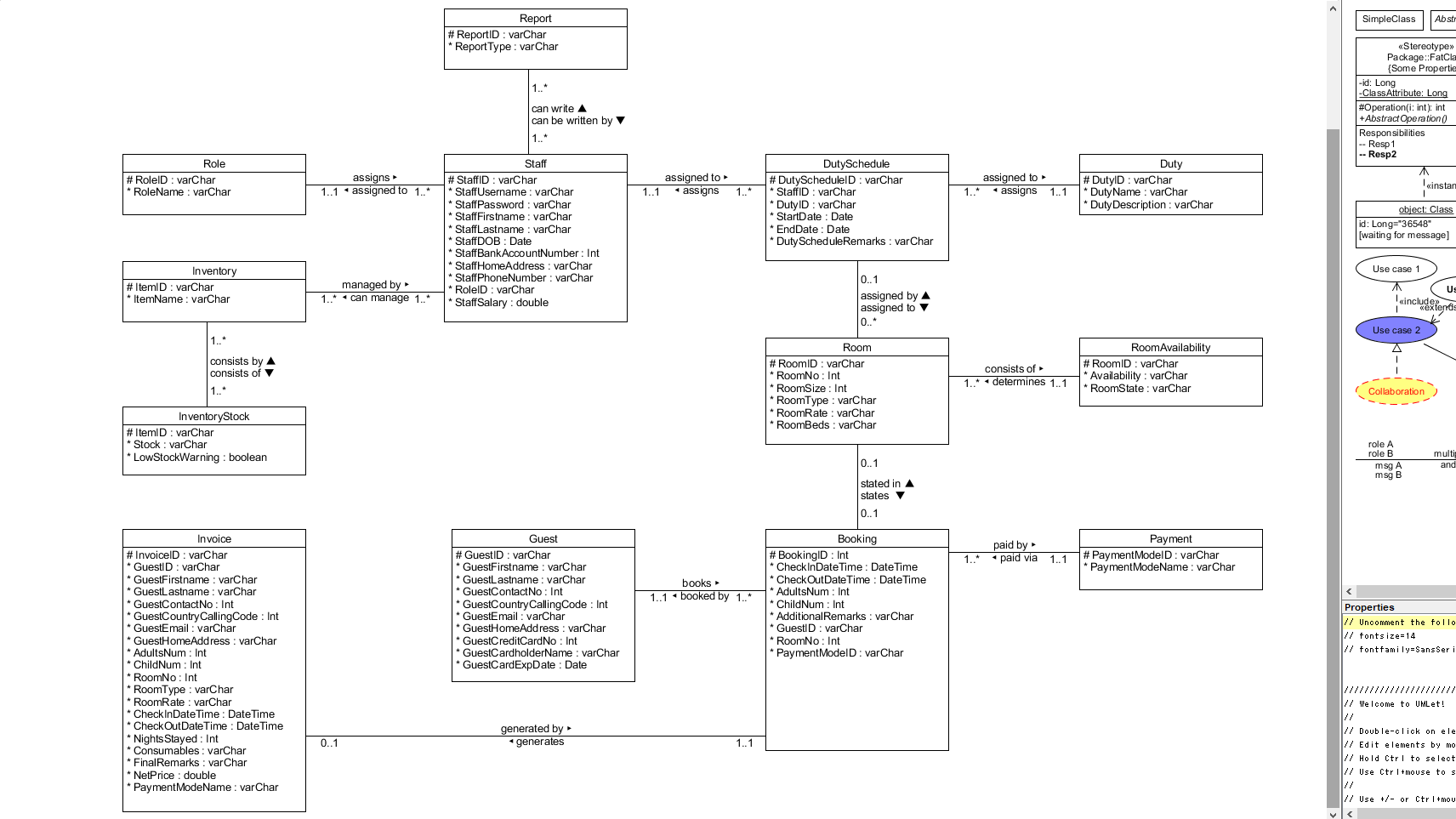
**Housekeeping Report Alternate Flow:**

1. User selects a staff in the hotel by their ID.
2. User keys in what duties the staff is assigned to.
3. User generates the report.

# **DATABASE DESIGN (If possibl**e place a Database Table) (Database Schema is REQUIRED)

*[Document the database design. What considerations and decisions went into the design of the database schema? How does your database design solve your data storage requirements? Are there any limitations? Highlight portions of the database schema that needs explanation. Highlight also interesting/innovative portions of your database design.]*

**Database Table**

**

*(Database Schema with arrows)*

*(For every table, put the schema)(Use SQL)*

*PaymentDetails (ownerID, name, expiration\_date, creditcard\_no, guestID)*

**Database Schema**

**Booking** (**BookingID**, CheckInDateTime, CheckOutDateTime, AdultsNum, ChildNum, AdditionalRemarks, GuestID, RoomNo, PaymentModeID)

**Payment** (**PaymentModeID**, PaymentModeName)  
 **Guest** (**GuestID**, GuestFirstname, GuestLastname, GuestContactNo, GuestCountryCallingCode, GuestEmail, GuestHomeAddress, GuestCreditCardNo, GuestCardholderName, GuestCardExpDate)

**Role** (**RoleID**, RoleName)

**Duty** (**DutyID**, DutyName, DutyDescription)

**DutySchedule** (**DutyScheduleID**, StaffID, DutyID, StartDate, EndDate, DutyScheduleRemarks)

**Invoice** (**InvoiceID**, GuestID, GuestFirstname, GuestLastname, GuestContactNo, GuestCountryCallingCode, GuestEmail, GuestHomeAddress, AdultsNum, ChildNum, RoomNo, RoomType, RoomRate, CheckInDateTime, CheckOutDateTime, NightsStayed, Consumables, FinalRemarks, NetPrice, PaymentModeName)

**Staff** (**StaffID**, StaffFirstname, StaffLastname, StaffDOB, StaffBankAccountNumber, StaffHomeAddress, StaffPhoneNumber, RoleID, StaffSalary)

**Report** (**ReportID**, ReportType)

**Room** (**RoomID**, RoomNo, RoomSize, RoomType, RoomRate, RoomBeds)

**RoomAvailability** (**RoomID**, Availability, RoomState)

**Inventory** (**ItemID**, ItemName)

**InventoryStock** (**ItemID**, Stock, LowStockWarn)

# References

<https://en.wikipedia.org/wiki/Multitier_architecture>

<http://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/>

<https://www.tuturself.com/posts/view?menuId=90&postId=465>