

CIS016-1 Assignment 3 Hotel booking system

Group Report



# Introduction

For this group project the task required was to create a hotel booking application, containing of a client, a database and a server; the client consisting of the user (customer and hotel manager), database being where the information gathered is stored/altered/retrieved and the server being the link between the client and the database.

The task also required certain functions for both the user and hotel manager. The user to be able to create an account and use such account to log in and book a room in the hotel as well as can view, alter or delete previous bookings. The hotel manager should be able to view all customers’ bookings and assign rooms to each booking after signing in.

All the data shown on both the hotel manager and customer page must come from the database, pass through a server and sent to the customer. Our system consists of a registration, log in, booking, billing, hotel manager and customer to form one graphical user interfaces which connects with the server which communicates with the database.

# System Design

## UML Use Case Diagram

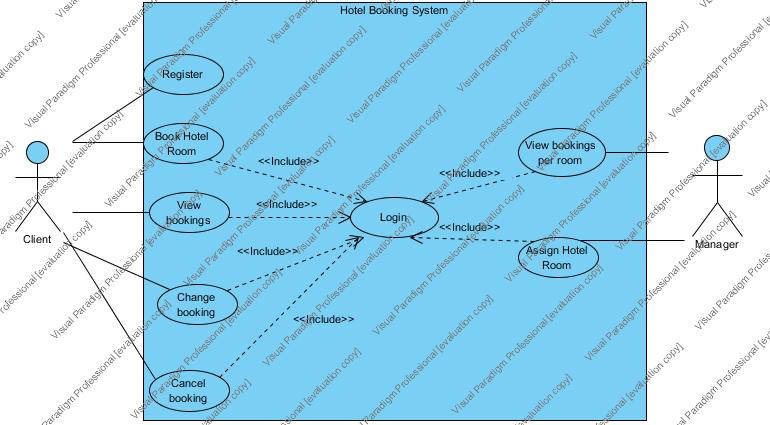
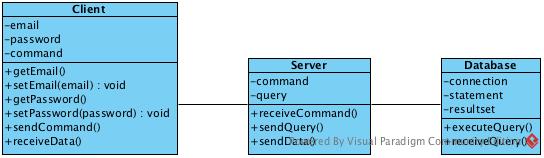


Figure 1

## UML Class Diagram

Figure 2



## Database Design

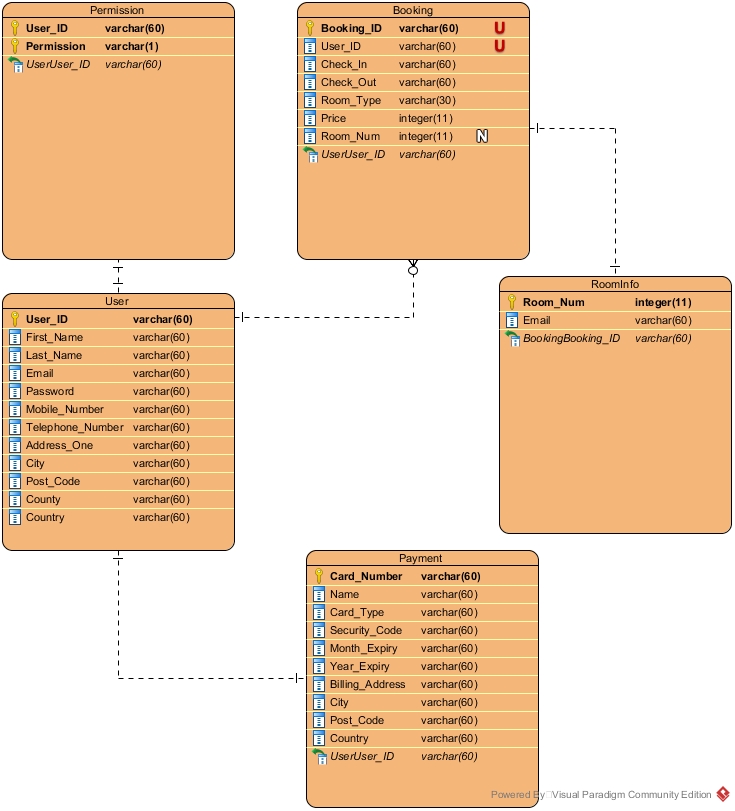


Figure 3

# Implementation

Our choice of implementation was a Client-Server system, the idea is when the server runs, it waits for a connection from a client, when a client does finally connect then the server waits for any commands or requests from the client.

Whenever a user interacts with the client, either pushing a button or anything interactive, then the server will receive a notification of sending information back to the client upon request. The server works by storing the information within a database, thus a Client to Server to Database implementation.

To begin we first designed the database, as we wanted to build the client according to the database and the server according to the client and database. Our initial requirements were having the Head Database Developer construct a database diagram with their Database Developers. Once we had the diagram we simply put the tables and data within the database, we knew we would have to change it later but we had our base.

The client was then a matter of implementing components according to the needs of the database, when we saw what we made we realized we needed to add more fields for the registration and more for that billing and so we did. We edited and changed the database according to the needs of the client, so any new JTextField or JComponents we would add the needs to the database accordingly.

The server was a matter of what the client needed to send to the database and vice versa, we knew the client needed to send all registration fields to the database and the client would need to retrieve the login details when logging in. The way we done it was creating object or class ‘Mules’ that would hold the data when every new object is created, and the objects are sent by the client, which then the server would process it and send it to the database, vice versa from database to server to client.

To code the program we used IntelliJ, Eclipse and BlueJ, however our primary IDE was IntelliJ, as the Head Developer would use that primarily. The usage of Eclipse and BlueJ was according to the other team members when they were tasked to create certain panels.

The way the client works is through objects of each individual class created and communicated within one Main Frame. The Main Frame utilizes 9 object classes, you can view them in the Implementation Images. The server communicates between both database and client, it uses SQL queries for the database communication and between itself and the client it uses ‘Command Phrases’, Implementation Images.

We chose a Client-Server system as it’s the more professional way to go and we knew nobody in our group knew how to make a server so we would learn a lot. The idea was, create the database and client first as we wanted the client to be almost web like; then if we had time we would build a server to help communicate between the two, if not the we would just simply use a Client-Database connection.

# Testing

## Testing Plans

The first stages of testing were to see if the client and server would run independently at separate times, afterwards would be to see if they run together simultaneously.

The next stages were to check if the requirements of the task were met, from the task description we decided the requirements were, access for two specific users the first, Customer, that would have to be able book a room, view their existing bookings and edit or make changes to the bookings only once logged in; before the customers’ login they must register an account, however the first time they register it should automatically log them in.

The second user, Hotel Manager, they would have the ability to view all bookings made, and assign any bookings without rooms a room according to their preference of the room type, single or double.

If the tests bring up errors, which most certainly they will then we would have to fix the errors and retest after fixing the errors.

## Test Log

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Client | Server | Register | Login | Book Room | View bookings | Edit bookings | Manager View | Database |
|  |  | P |  |  | P | P | P |  |  |
|  |  |  | F |  |  |  | F | F | P |
|  | F |  |  |  | F |  |  |  |  |
|  |  |  | P |  |  | F |  |  |  |
|  | P |  |  |  |  |  |  | F |  |
|  | P |  |  | P |  |  |  |  |  |

xxxv’s tasks were to check if the Server could run independently, if a room was able to be booked and if the bookings could be changed or deleted.

The Server could run without the client however, the server would just sit and wait for a connection, so it would not be doing anything other than just waiting. Rooms were bookable and when a booking was made the view bookings table would update accordingly which also passed, bookings were able to be deleted and changed when needed too.

xxx’s tasks were to check if the client could run independently and to check if registered users could login properly.

The client runs independently however throws an error with the Input/output streams on the client side socket. The registered accounts could login successfully and show the booking a room panel and view bookings panel.

xxx’s tasks were to, check if the client would automatically log in the user as soon as he/she registered, make sure that the changes made in the editing bookings page would update in the manager database tables, ensure the manager would have access to the necessary information when logged in to the system and guarantee that the database was efficient enough.

The client would not log the user in as soon as such registered with an account, instead it would take the user to the login page, making it not meet the requirements. When changed a booking would update in the customer “Show Bookings” table but it would not update in the manager table. The manager screen was missing a view all bookings table, which meant that it could only assign a room at the time making it not meet all the requirements of the manager view. The database was being used by the server on all aspects.

xxxx’s tasks were to check if there were any errors after fixing the errors according to Daniela’s feedback.

The client runs as intended, however upon logging into the system the program will display several popups informing the user that the password the entered was incorrect, even if the password entered was correct. This would occur approximately three times upon the initial login and five times upon logging out. Another issue with the system is that when a new booking is made, the corresponding table would be populated with incorrect data.

xxx’s tasks were to check if users were automatically logged in after registering a new account upon xxxx’s and xxxx’s feedback.

The client allows me to create an account, upon creation I’m automatically taken to the home page. However, an error occurs during the login process as it starts to freeze. I think the issue is with the view bookings table as it is trying to load data that is failing to send.

xxxx’s tasks were to check if the client had any errors once fixed according to xxxx’s, xxxx’s and xxxx’s feedback.

The customer load up programming is fine, when an account is registered, the record is consequently signed in, the screen does not encounter a freeze due to no bugs, and when a booking is made it refreshes the view appointments table for the client as needs be. The administrative side however, has not been refreshed or fixed since the past mistakes, the hotel manager still can't see existing appointments which comes up short the assignments necessities. Another table or board will be required for the administrative side, other than that the customer is by all accounts fitting the necessities of the client.

## Testing of the fully developed client/server/database

The first range of tests will check to see if the server and client can run independently without running the other before or after, and whether the errors are handled from previous tests.

The server interface runs and indicates the server is running (Figure 7 - Fully developed client/server/database testing) however, the database was not enabled at the time of the test, so the console gave off an error indicating a ‘Communications line failure’ (Figure 8 - Fully developed client/server/database testing).

The reason the error in Figure 8 appears is because within the ‘catch’ block we’ve specified the block to print the error stack trace (Figure 10 - Fully developed client/server/database testing), however that’s not very user friendly. Instead we’ve modified the catch block to present a JOptionPane dialog message (Figure 9 - Fully developed client/server/database testing), notifying the user upon runtime that the database is not connected and told the catch block to exit the system (Figure 11 - Fully developed client/server/database testing).

The server runs fine now that the database is running, the server can establish a connection and the dialog will no longer appear, unless the database is disconnected (Figure 12 - Fully developed client/server/database testing).

Next, the client runs fine, however as the server is not running, two dialogs appear, one notifying the user that the server is not running and that streams cannot establish a connection (Figure 13 - Fully developed client/server/database testing).

Next tests will see if the client meets the requirements of the task, we believe the requirements were: Customers can, register, login, book rooms, view rooms, edit rooms; Hotel managers can, login, view all bookings, assign rooms.

The register screen upon pressing the ‘Register’ button brings up two panels, the first being the register panel, where the users would enter their details accordingly (Figure 14 - Fully developed client/server/database testing). The second panel is the billing panel, where users must enter their billing details accordingly, (Figure 15 - Fully developed client/server/database testing). Upon creation, the users are taken straight to the main page, automatically logging them in (Figure 16 & 17 - Fully developed client/server/database testing).

Once the users get logged in they can view their bookings, as the user is new there would be no bookings upon login (Figure 17 - Fully developed client/server/database testing). Users are also able to book a room if they want to (Figure 18 - Fully developed client/server/database testing), booking prices are adjusted according to the number of days the user wish to stay for. After the booking a dialog screen appears notifying the user whether the booking was successful or not, then the view bookings table is refreshed and shows the new booking accordingly. (Figure 19 & 20 - Fully developed client/server/database testing).

The bookings made by the user can be edited to their liking, users are able to change the check-in or check-out date or even change the room type, users are also able to delete the booking completely, if necessary (Figure 21 & 22- Fully developed client/server/database testing). After the user is satisfied with their decisions, they can finally log out of the client, in case they wish to log into a different account, make a new one, or just simply end their session (Figure 23 - Fully developed client/server/database testing).

After the user chooses to logout they will be sent to the login page, where they can either log back in with the same or different account or create a new account by pressing register (Figure 24 - Fully developed client/server/database testing).

When a hotel manager logs in, the screen shown will be the same as a normal user, however they will get two new tabs, a ‘Manage Bookings’ tab and ‘Existing Bookings’ tab (Figure 25 & 26 & 27 - Fully developed client/server/database testing).

Hotel managers can assign rooms using the manage bookings tab according to the customers’ preference. The manage bookings tab shows the booking id, customer email and room type the customer is requesting, also a button to allow the ability to assign. Once a room is assigned then the existing bookings panel will update the room number accordingly (Figure 28 - Fully developed client/server/database testing).

We’ve also made the hotel manager a customer so they can book rooms and manage their own bookings if they wish to, the way hotel managers and customers are differentiated is based on their permissions. Upon the creation of a user’s account the user is given the ‘Permission’ ‘C’, ‘C’ standing for customer. Hotel managers must be assigned their permission of ‘M’ to give them access to the manager tabs (Figure 30 & 31 & 32 - Fully developed client/server/database testing).

Although all the success after fixing the errors initially, we still had issues within the code which we didn’t know what was causing them at first. So we called them ‘What’ statements, that catch leaked memory from the server side, as we didn’t have much time we had to just put these lines of codes around the code when the memory leak occurred (Figure 29 - Fully developed client/server/database testing).

# Reflections and Responsibilities

|  |  |
| --- | --- |
| **Name and ID** | **Roles** |
|  | Head Java Developer |
|  | Head Database Developer |
|  | Database and Java Developer |
|  | Database and Java Developer |
|  | Lead Presenter and Designer |
|  | Database Developer |

When we had our first group meeting we assigned everybody roles according to their skill sets and preference on what they wanted to learn, everybody in the group was eager to learn a bit of Java, MySQL, report writing and presentation speaking. So, we decided as a group to assign specific roles but allow everybody to take part in a little bit of everything as the members preferred.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Client | Server | Database |
|  | Lead the client creation.  Created the register screen, helped edit the login screen, manager screen and booking screens. | Created the connection between the database and server and client. | Helped edit database according to the client’s needs/updates. |
|  | Helped improve the login design and main screen design | Helped with the connection between the server and database and SQL queries. | Lead the database creation.  Created certain tables and data types. |
|  | Created the manager screen and booking screen and helped the design of the login and main screen. | Helped with SQL queries. | Helped create certain tables and data types. |
|  | Created the manager screen and booking screen and helped the design of the login and main screen. | Helped with SQL queries. | Helped create certain tables and data types. |
|  | Created the login screen and helped the design of the login and main screen. | Helped with the design of the server user interface. | Edited database information accordingly. |
|  | Helped with testing. | Helped with testing. | Helped with testing. |

We as a group got together to write up this report, everybody had a say and participated in writing the report, we each hold equal responsibility. The creation of the presentation was led by Michael, who assigned everybody their specific slides and what to discuss.

The things that we as a group decided went well was our organizational skills, as we could organize presentation and computer rooms, in the times we needed to group up and discuss the client; our communication skills considering we had to communicate to each other what times everyone was able to group up and acquire rooms accordingly and considering nobody left or was kicked from the group; our deadlines everyone was tasked with, every individual member would create or edit according to the task given and the creation of the client was finally finished with all requirements met and with nearly two weeks to spare, to be able to decide final touches and designs.

Though, no group is perfect and we did have problems, not necessarily with each other but problems with how to tackle the task itself. As we all prefer to be practical rather than theoretical, we struggled with the creation of the UML diagrams, as that part was not one of our strong points, we did give the diagrams a go and created what we think would help us and it did help as a starting base.

We found it difficult to identify our skills properly as everyone was eager to learn new things, and considering we needed to learn new codes to program the task in the direction the task wanted us to, we had to do things in a more structured and methodical way so we wouldn’t run out of time. We lacked the knowledge to build a server so we had to just focus on the skills we already knew and had then if we had time we would develop the skills even more to build a server.

Overall, we believe we meshed very well as a group and there was no complain as we all had a more laid back attitude, we knew what was required and we all led the group together as a team rather than one specific leader.

# Conclusion

In this report, we discussed the design and implementation of a hotel booking system; this task was tackled by the creation of a GUI, a database and a server. The specified requirements were met for both the users of this hotel booking system (customer and hotel manager). The system that we as a group created consists of a Server-Client relationship, allowing back-end system upgrades for any changes in the future.

# References

To build the client in the design the lead designer wanted the client to look like, we needed to implement a button within a JTable, as we did not have enough time to implement the button ourselves we used a class borrowed from the internet that would create buttons within the JTable, the class is called ButtonColumn.

Camick, R. (2009) ‘Table Button Column’ *Rob Camick’s Java Tips Weblog,* 12th July. Available at: <https://tips4java.wordpress.com/2009/07/12/table-button-column/> (Accessed: 20th April 2017).

# Appendix

## ../../Screen%20Shot%202017-04-26%20at%2019.05.50.pngImplementation Images

Figure 4 – Object Classes

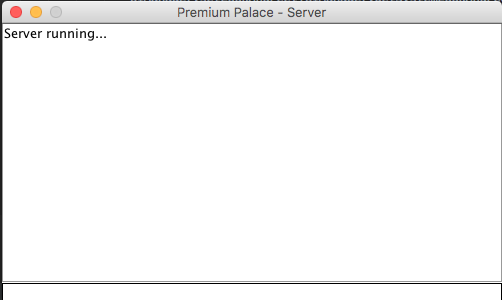
../../Screen%20Shot%202017-04-26%20at%2019.10.41.png

Figure 5 - SQL Queries

../../Screen%20Shot%202017-04-26%20at%2019.11.08.png

Figure 6 - Command Phrases

## Fully developed client/server/database testing

Figure 7 – Server interface

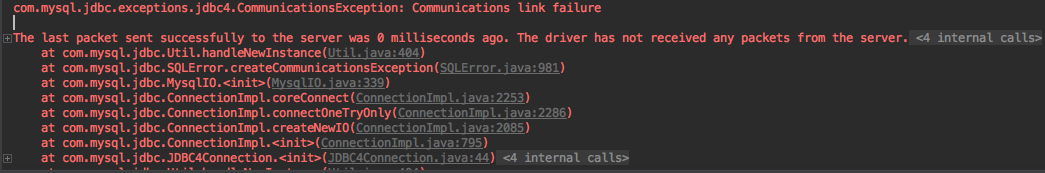
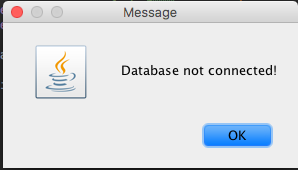


Figure 8 – Communications failure

Figure 9 – Dialog



../../Screen%20Shot%202017-04-30%20at%2010.48.40.png

Figure 10 – Stack trace

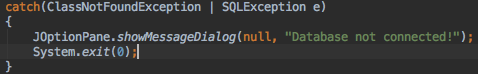


Figure 11 - JOptionPane code

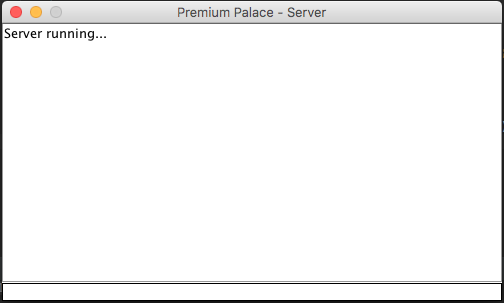


Figure 12 – Server with database running

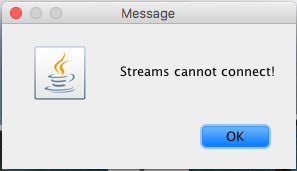
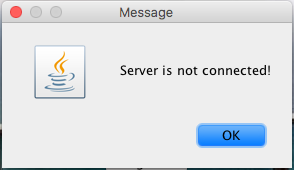


Figure 15 – Billing panel

Figure 14 - Register panel

Figure 13 – Server is not connected, Streams cannot connect

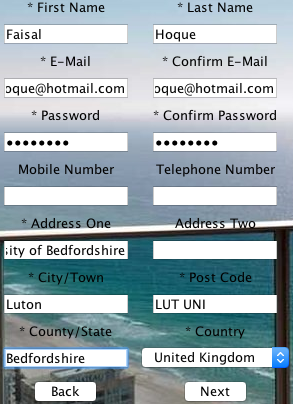
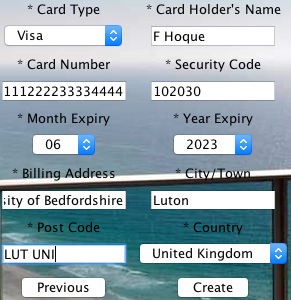
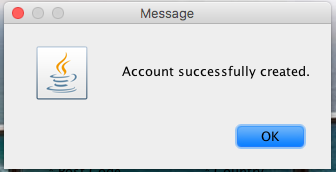


Figure 16 - Account created

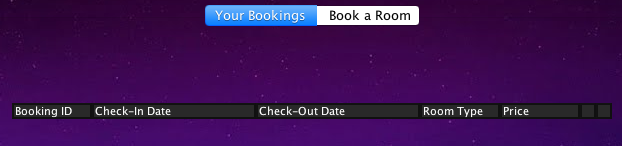


Figure 17 - After registration

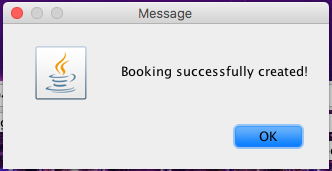
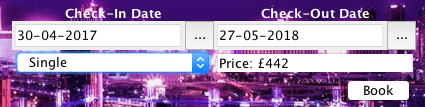
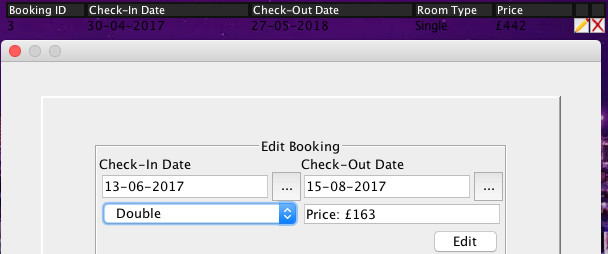


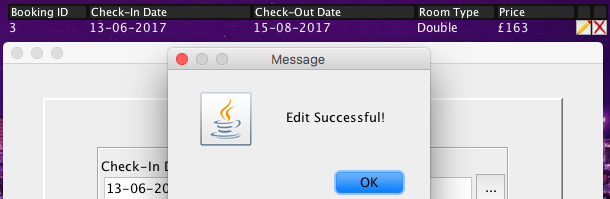
Figure 21 – Edit bookings

Figure 20 – Table updated accordingly

Figure 19 - After booking

Figure 18 - Booking panel

../../Screen%20Shot%202017-04-30%20at%2011.41.30.png



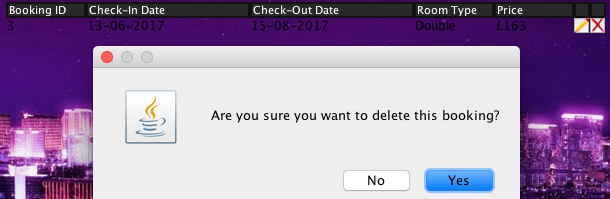


Figure 22 – Delete booking

../../Screen%20Shot%202017-04-30%20at%2012.03.22.png

Figure 22 – After booking deleted

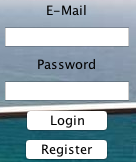
../../Screen%20Shot%202017-04-30%20at%2012.12.00.png

Figure 24 – After logout

Figure 23 – Logout

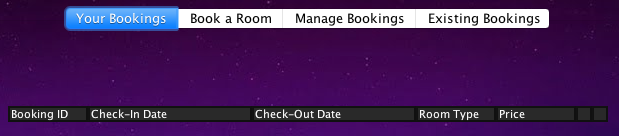


Figure 25 – Upon hotel manager logging in

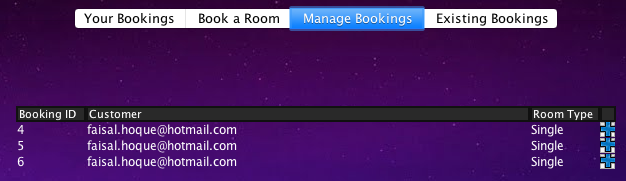
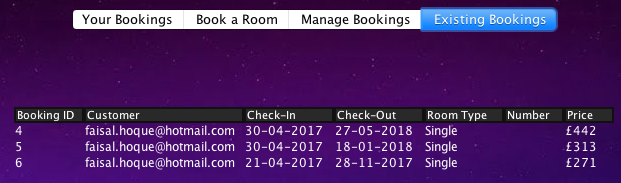


Figure 27 – View all existing bookings

Figure 26 – Manage bookings



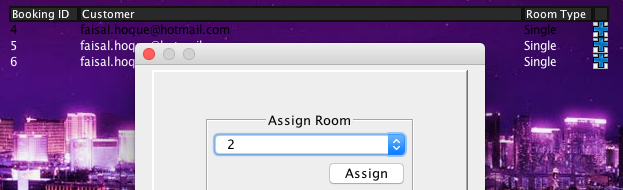
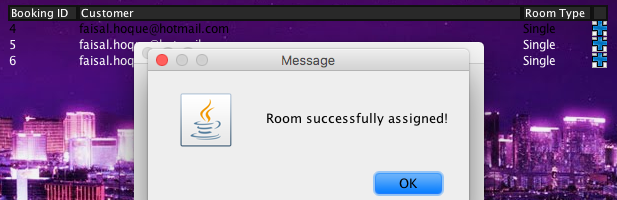
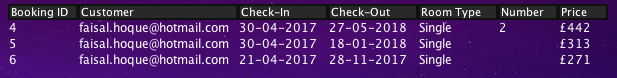


Figure 28 – Assigning a room

Figure 29 – What, memory leaks

../../Screen%20Shot%202017-05-01%20at%2013.46.35.png

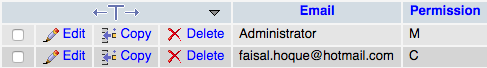


Figure 30 – Database permissions

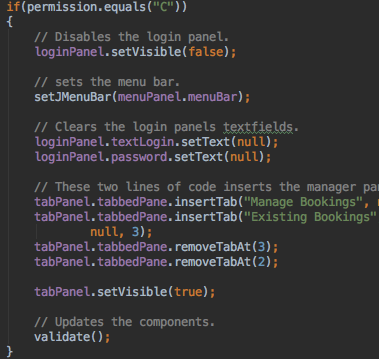


Figure 31 – Permission ‘C’

Figure 32 – Permission ‘M’

