CSC-20002: dATABASE sYSTEMS

Ease Ltd room hire system

Student Name: Alex Farrell

Student Number: 15005594

**Contents**

[Function Requirements 2](#_Toc482275963)

[Customers can book particular rooms in branches 2](#_Toc482275964)

[Invoicing 2](#_Toc482275965)

[Booking as many rooms as needed 2](#_Toc482275966)

[Store previous bookings 2](#_Toc482275967)

[Database Design and Implementation 2](#_Toc482275968)

[Design 2](#_Toc482275969)

[Implementation 2](#_Toc482275970)

[Appraisal 3](#_Toc482275971)

**Website Link:** [**http://www.teach.cs.keele.ac.uk/prin/w4s22/csc20002**](http://www.teach.cs.keele.ac.uk/prin/w4s22/csc20002)

# **Function Requirements**

## **Customers can book particular rooms in branches**

As shown on my logical ERD, the customer entity is split into two subclasses, standard and business customers. The standard customer has a link to standard booking only, whereas business customer has links to standard booking and executive booking. Due to this, there is no way of a standard customer booking an executive room.

## **Invoicing**

A business customer can pay by invoice as well as cash. In the booking entity, there is an attribute called ‘paymentMethod’. For a standard customer, this will always be cash. However, for a business customer, it will be either cash or invoice. In the business customer subclass, there is an attribute called ‘invoiceNumber’. If the business customer wishes to pay by invoice, the system will fetch the invoice number when the booking is made.

## **Booking as many rooms as needed**

In the logical model, it became clear that there was a many-to-many relationship between booking and room. As appose to creating a separate entity for this, the easier solution and therefore the one I employed, was that each room for a booking, had its own booking record. You would be able to find all rooms that the customer wants by using a select statement in SQL and selecting all bookings with the same customer id, date and time.

## **Store previous bookings**

This requirement is one that sorted itself out in the background. All bookings are stored in the booking entity and will not be deleted unless a customer contacts the company to say that they can’t make the booking, in which case, the booking will simply be removed from the database. There is an attribute in the booking entity that states whether a customer has turned up for a booking. In order to view previous bookings, you would simply search for all bookings where the ‘arrived’ attribute was true.

# **Database Design and Implementation**

## **Design**

My design began by listing all the necessary entities which would make up the conceptual model. This inevitably brought up a few many-to-many relationships which would need to be dealt with in the logical model. As well as this, I discovered opportunities to include subclasses. In the conceptual model, these were represented by arrows pointing towards the superclass. However, in the logical model, each subclass was represented by a separate entity, depicting a one-to-one relationship with its superclass.

## **Implementation**

The implementation began by mapping the logical model onto Oracle Server using SQL Plus. Each field was given a datatype and maximum size. On inserting dummy data into the database, I decided that realistically, 5 branches would be sufficient as sample data. I then decided that I would have a ratio of 1 employee to every 2 rooms. The amount of data inserted into the other entities reflected the same sample size. My web interface allows a user to search each table and display all results, as well as being able to search for specific stock of each branch, using an SQL join to display the product name and supplier name as appose to the ID’s, the attributes stored in the ‘BranchStock’ table.

# **Appraisal**

Overall, I feel that my system meets the requirements of the company to an adequate level and the entities are organised so that it is easy to expand on the system in the future. I feel that my SQL queries are neat and tidy, and the join allows multiple SQL queries to be condensed into one single query, which further benefits the output to the user in the way that you are able to show data form multiple tables in the same output table, for example I am able to display the names of products and the names of suppliers as appose to the ID’s which to a normal user, would not mean anything.

Improvements that could be made include adding constraints to the tables so that only certain values can be entered. Although this can be dealt with effectively using php, by implementing this in the creation of the database, you avoid having to check each time that the values the user is trying to enter, are valid. Also, I do not feel that I have sufficiently dealt with the issue that if rooms are not available at certain times, bookings cannot be made. This is something that could be implemented in php at a later date although, like the constraints, it would be much easier to implement these on the creation of the database.