Object-Oriented Programming  
Assignment 2  
Design

Contents

[Screen Designs 2](#_Toc163911238)

[Individual Screen Designs 2](#_Toc163911239)

[Log In 2](#_Toc163911240)

[Create User 3](#_Toc163911241)

[Main Form 4](#_Toc163911242)

[Create Booking 5](#_Toc163911243)

[View Bookings 6](#_Toc163911244)

[View Cruises 8](#_Toc163911245)

[Wireflow Diagram 9](#_Toc163911246)

[Algorithm Design 10](#_Toc163911247)

[Luhn Algoirthm 10](#_Toc163911248)

[Pseudocode 10](#_Toc163911249)

[Flowchart 11](#_Toc163911250)

[Database Design 12](#_Toc163911251)

[Entity-Relationship Diagram 12](#_Toc163911252)

[UML 13](#_Toc163911253)

[Appendix 14](#_Toc163911254)

[Appendix A - Detailed Database Design 14](#_Toc163911255)

[UserRoles 14](#_Toc163911256)

[Users 14](#_Toc163911257)

# Screen Designs

## Individual Screen Designs

* The default form size will be 800 width, 500 height.
* The default font will be Segoe UI, 24pt in black (“Control Text”).
* The background colour will be “Control”.
* The individual screens are also in the “Individual Screens” subdirectory of the subdirectory “Images”.

### Log In

A screenshot of a computer

Description automatically generated

Image 1 - The Log-In Form

* The labels are as standard.
* The textboxes have the default font sizes. They have a width of 365 and height of 50.
* The txtPassword box, will be masked with asterixis.
* The buttons have the default font sizes. They have a width of 170 and a height of 105.

### Create User

A screenshot of a computer

Description automatically generated

Image 2 - The Create User Form, with Password Hidden

A screenshot of a computer

Description automatically generated

Image 3 - The Create User Form, with Password Visible

* All elements are designed to match the previous form, including item sizes and colours.
* The btnShowPassword is square with dimensions of 40. It is placed equally between the password inputs.
* On, toggle the button symbol will change to a lock.

### Main Form

A screenshot of a computer

Description automatically generated

Image 4 - The Main Form before Interpolation

A screenshot of a computer

Description automatically generated

Image 5 - The Main Form after Interpolation

* In general, the styling remains the same as above.
* The buttons are sized at 170 width and105 height.
* The lblWelcomeMessage on login, will change to show the user’s username.

### Create Booking

A screenshot of a computer

Description automatically generated

Image 6 - The Create Booking Form before Interpolation

A screenshot of a computer

Description automatically generated

Image 7 - The Create Booking Form after Interpolation

* This form has a width and height of 900.
* The “Full Name” label and text box is at y = 25. From there, each subsequent element’s y position:
* The text boxes and combo boxes have a width of 365.
* Validation will be run on the email and telephone numbers, via the use of regex.
* The combo boxes will be loaded via data from the database, depending on what has been booked.
* lblPriceValue will auto calculate the price based on the room and whether it is single occupancy.
* If cbBypassCardValidation is not selected, the card number will be validated using the Luhn Algorithm (Luhn, 1960).

### View Bookings

A screenshot of a computer

Description automatically generated

Image 8 - The View Bookings Form without Search

A screenshot of a computer

Description automatically generated

Image 9 - The View Bookings Form with Search

* This form is 800 by 500.
* The txtSearch has a width of 420, and has placeholder text set to “Search…”
* The two buttons btnSearch and btnClearSearch are each 55 by 55.
* btnClearSearch is only visible when the trimmed text content of txtSearch has a length greater than 0.
* The DataGridView dgvBookings is 660 by 300. It is read-only.
* It contains two text columns Name and Date.
* It contains two button columns Cancel and Amend.

A screen shot of a computer program

Description automatically generated

Image 10 - The Code for the View Bookings DataGridView

### View Cruises

A screenshot of a computer

Description automatically generated

Image 11 - The View Cruises Form

* This form has dimensions 950x500.
* The DataGridView dgvCruises has dimensions 920x355.
* The data grid will have a row for each cruise date.
* It will calculate the total income that cruise has calculated so far, as well as the number of rooms remaining.
* There will be a button column which will open the View Bookings form, filtered to only show bookings for that specific cruise.

## Wireflow Diagram

A screenshot of a computer

Description automatically generated

Image 12 - Wireflow Diagram Showing User Flow

The Wireflow diagram is also in the “images” subdirectory as “**Wireflow Diagram.png**”

# Algorithm Design

There is only one specific algorithm that will be used as part of this project. That is the Luhn algorithm, use to validate credit/debit card numbers.

## Luhn Algoirthm

### Pseudocode

A screenshot of a computer program

Description automatically generated

Image 13 - The Pseudocode for the Luhn Algorithm

### Flowchart

A diagram of a diagram

Description automatically generated with medium confidence

Image 14 - Flowchart Showing the Luhn Algorithm

See also in the images’ subdirectory as both a PDF and PNG. These are as **LuhnAlgorithmFlowChart.pdf** and **LuhnAlgorithmFlowChart.png** respectively.

# Database Design

For detailed information on the database structure, please see [Appendix A](#_Appendix_A_-).

## Entity-Relationship Diagram

A diagram of a computer

Description automatically generated

See also in the images’ directory, the file **Database-ERD.jpg**.

The Visual Paradigm Project file, is also available in the diagrams’ subdirectory.

# UML

**Note:** The below diagram is not comprehensive. Additional classes, methods and instance variables may be needed. For example, not all getters and setters are presented. As well as this, I have not included the classes for the forms themselves, as they will contain minimal application logic. Instead, the diagram focuses on utility and model classes. As well as this, the UML diagram is also for the main project of the solution. The test project is not featured as part of this diagram.

A blue rectangles with black text

Description automatically generated

Image 15 - The UML Diagram

See also in the images’ directory, the file **UML.jpg**.

The Visual Paradigm Project file, is also available in the diagrams’ subdirectory.

# Appendix

## Appendix A - Detailed Database Design

As part of this project, I will need to include a database. I believe the below tables to be necessary.

### UserRoles

#### Overview

This table will contain information around the different types of user role.

At this point, this will include the ‘**Admin**’ role, which will be able to log in and access the system, as well as the ‘**Customer**’ role which will be used as part of the booking.

The ‘**UserRoles**’ table has a one-to-many relationship with the `**Users**` table. That is, one role can belong to multiple users.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| UserRolesId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| RoleName | Text, Unique, Not Null |

Table 1 - The Structure of the UserRoles Table

#### Example

|  |  |
| --- | --- |
| UserRolesId | RoleName |
| 1 | Admin |
| 2 | Customer |

Table 2 - An Example of the Data in UserRoles

### Users

#### Overview

This table will contain the minimal amount of information on a specific user.

The ‘**UsernameOrEmail**’ will be used either for the email address of a ‘**Customer**’ or for the username of an ‘**Admin**’.

The ‘**Users’** table has a one-to-many relationship with the ‘**UserRoles**’ table. That is, one role can belong to multiple users.

The ‘**Users’** table has a one-to-one relationship with the ‘**UserCredentials**’ table. This is because one user will have one hashed password.

The ‘**Users’** table has a one-to-one relationship with the ‘**UserContactDetails**’ table. This is because one user will have one set of contact details.

The ‘**Users’** table has a many-to-one relationship with the ‘**Bookings’** table. This is because one user can have multiple bookings.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| UsersId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| UsernameOrEmail | Text, Unique, Not Null |
| UserRoles.UserRolesId | Integer, Foreign Key, Not Null |

Table 3- The Structure of the Users Table

#### Example

|  |  |  |
| --- | --- | --- |
| UsersId | UsernameOrEmail | UserRoles.UserRolesId |
| 1 | DefaultAdmin | 1 |
| 2 | john.smith@mail.me | 2 |
| 3 | jane.doe@mail.com | 2 |
| 4 | AnotherAdmin | 1 |

Table 4 - An Example of the Data in Users table

### UserCredentials

#### Overview

This table will contain the hashed password for a user.

Only ‘**Admin**’ users at this stage will have credentials.

The ‘**UserCredentials**’ table has a one-to-one relationship with the ‘**Users’** table. This is because one user will have one hashed password.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| UserCredentialsId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| Users.UsersId | Integer, Foreign Key, Not Null |
| HashedPassword | Text, Not Null |

Table 5 - The Structure of the UserCredentials Table

#### Example

|  |  |  |
| --- | --- | --- |
| UserCredentialsId | Users.UsersId | HashedPassword |
| 1 | 1 | 5f4dcc3b5aa765d61d8327deb882cf99 |
| 2 | 4 | 126fcc5a5dd775d64e9327deb882cf12 |

Table 6 - An Example of the Data in the UserCredentials table

UserContactDetails

#### Overview

This table will contain the contact details for a user.

Only ‘**Customer**’ users at this stage will have contact details.

The ‘**UserContactDetails**’ table has a one-to-one relationship with the ‘**Users’** table. This is because one user will have one set of contact details.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| UserContactDetailsId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| FullName | Text |
| Users.UsersId | Integer, Foreign Key, Not Null |
| Telephone | Text |

Table 7 - The Structure of the UserContactDetails table

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| UserContactDetailsId | FullName | Users.UsersId | Telephone |
| 1 | NULL | 2 | 0800001066 |
| 2 | Jane Doe | 3 | NULL |

Table 8 - Example of the data in the UserContactDetails table

Tours

#### Overview

This table will contain data around the tours that are available.

The ‘**Tours**’ table will have a many-to-one relationship with the ‘**RoomTypesTours**’ table. This is because, one tour type will have many entries in the ‘**RoomTypesTours**’ table, due to the different room types.

The ‘**Tours**’ table will have a many-to-one relationship with the ‘**Bookings**’ table. This is because, a tour can have multiple bookings.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| ToursId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| TourStartDate | Date, Unique, Not Null |

Table 9 - The Structure of the Tours table

#### Example

|  |  |
| --- | --- |
| ToursId | TourStartDate |
| 1 | 2022-06-06 |
| 2 | 2022-06-13 |

Table 10 - Example of the data in the Tours table

RoomTypes

#### Overview

This table will store information about the different types of room.

The ‘**RoomTypes**’ table will have a one-to-one relationship with the ‘**RoomCostInfo**’ table. This is because one room type will have one set of pricing.

The ‘**RoomTypes**’ table will have a many-to-one relationship with the ‘**RoomTypesTours**’ table. This is because, one room type will have many entries in the ‘**RoomTypesTours**’ table, due to the different room types.

The ‘**RoomTypes**’ table will have a many-to-one relationship with the ‘**Bookings’** table. This is because, a room type can belong to multiple bookings.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| RoomTypesId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| RoomName | Text, Unique, Not Null |

Table 11 - The Structure of the RoomTypes table

#### Example

|  |  |
| --- | --- |
| RoomTypesId | RoomName |
| 1 | Penthouse |
| 2 | Luxury |

Table 12 - Example of the data in the RoomTypes table

RoomCostInfo

#### Overview

This table will contain costing information about a room.

The ‘**CurrentPricePence**’ field will store the price of the room in pence. This has been chosen to reduce the risk of floating-point errors (Sierra, 1962).

The ‘**DiscountRate**’ filed will be the discount for single occupancy. It will be stored as an integer and converted in the code. For example. 20% will be store in the database as 20.

The ‘**RoomCostInfo**’ table will have a one-to-one relationship with the ‘**RoomTypes**’ table. This is because one room type will have one set of pricing.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| RoomCostInfoId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| RoomTypes.RoomTypesId | Integer, Foreign Key, Not Null, Unique |
| DefaultQuantity | Integer, Not Null |
| CurrentPricePence | Integer, Not Null |
| DiscountRate | Integer, Not Null |

Table 13 - The Structure of the RoomCostInfo table

#### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RoomCostInfoId | RoomTypes.RoomTypesId | DefaultQuantity | CurrentPricePence | DiscountRate |
| 1 | 1 | 1 | 78500 | 20 |
| 2 | 2 | 2 | 56500 | 18 |

Table 14 - Example of the Data in the RoomCostInfo table

RoomTypesTours

#### Overview

This table will link a room and a tour, to track the current quantity.

The ‘**RoomTypesTours**’ table will have a many-to-one relationship with the ‘**Tours’** table. This is because, one tour type will have many entries in the ‘**RoomTypesTours**’ table, due to the different room types.

The ‘**RoomTypesTours**’ table will have a many-to-one relationship with the ‘**RoomTypes’** table. This is because, one room type will have many entries in the ‘**RoomTypesTours**’ table, due to the different room types.

The ‘**CurrentQuantity**’ field will need controlling in the code, to stop it going above the default quantity of ‘**RoomCostInfo**’.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| RoomTypesToursId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| Tours.ToursId | Integer, Foreign Key, Not Null |
| RoomTypes.RoomTypesId | Integer, Foreign Key, Not Null |
| CurrentQuantity | Integer, Not Null |

Table 15 - The Structure of the RoomTypesTours table

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| RoomTypesToursId | Tours.ToursId | RoomTypes.RoomTypesId | CurrentQuantity |
| 1 | 1 | 1 | 0 |
| 2 | 1 | 2 | 1 |

Table 16 - Example of the data in the RoomTypesTours table

BookingStates

#### Overview

This table will contain the different states of a booking.

At this stage, it will include ‘**Booked**’, ‘**Refunded**’, and ‘**Cancelled**’.

The ‘**BookingStates**’ table will have a one-to-many relationship with the ‘**Bookings**’ table. This is because, a booking state can belong to multiple bookings.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| BookingStatesId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| BookingState | Text, Unique, Not Null |

Table 17 - The Structure of the BookingStates table

#### Example

|  |  |
| --- | --- |
| BookingStatesId | BookingState |
| 1 | Booked |
| 2 | Refunded |
| 3 | Cancelled |

Table 18 - Example of the data in the BookingStates table

Bookings

#### Overview

This table will contain information about bookings.

The ‘**Bookings’** table has a one-to-many relationship with the ‘**Users’** table. This is because one user can have multiple bookings.

The ‘**Bookings**’ table will have a one-to-many relationship with the ‘**Tours’** table. This is because, a tour can have multiple bookings.

The ‘**Bookings**’ table will have a one-to-many relationship with the ‘**RoomTypes’** table. This is because, a tour can have multiple room types.

The ‘**Bookings**’ table will have a one-to-many relationship with the ‘**BookingStates’** table. This is because, a booking state can belong to multiple bookings.

#### Structure

|  |  |
| --- | --- |
| Column Name | Column Information |
| BookingsId | Integer, Primary Key, Not Null, Unique, Auto Increment |
| Users.UsersId | Integer, Foreign Key, Not Null |
| Tours.ToursId | Integer, Foreign Key, Not Null |
| RoomTypes.RoomTypesId | Integer, Foreign Key, Not Null |
| BookingStates.BookingStatesId | Integer, Foreign Key, Not Null |
| IsSingleOccupancy | Boolean, Not Null, Default False |