Faculty of Engineering & Technology – Computer Science Department

Second Semester 2023-2024

COMP333, DATA BASE SYSTEMS

Phase -3-

Prepared by

Rasha Daoud – 1210382

Nadia Al-shaikh – 1210021

Donia Al-shaikh-1210517

Hala Sulaiman -1202997

Instructor: Bashar Tahayna

Section: 3

Date: 11th May 2024

Abstract

The goal of this stage is to connect our database (the Hotel Management system) designed by Workbench with the interface that we built using Java and Scene Builder. Through this interface, we will be able to perform all operations on the linked tables in the database like insert, delete, update etc.

2.Result of Interface

2.1 The Basic interface

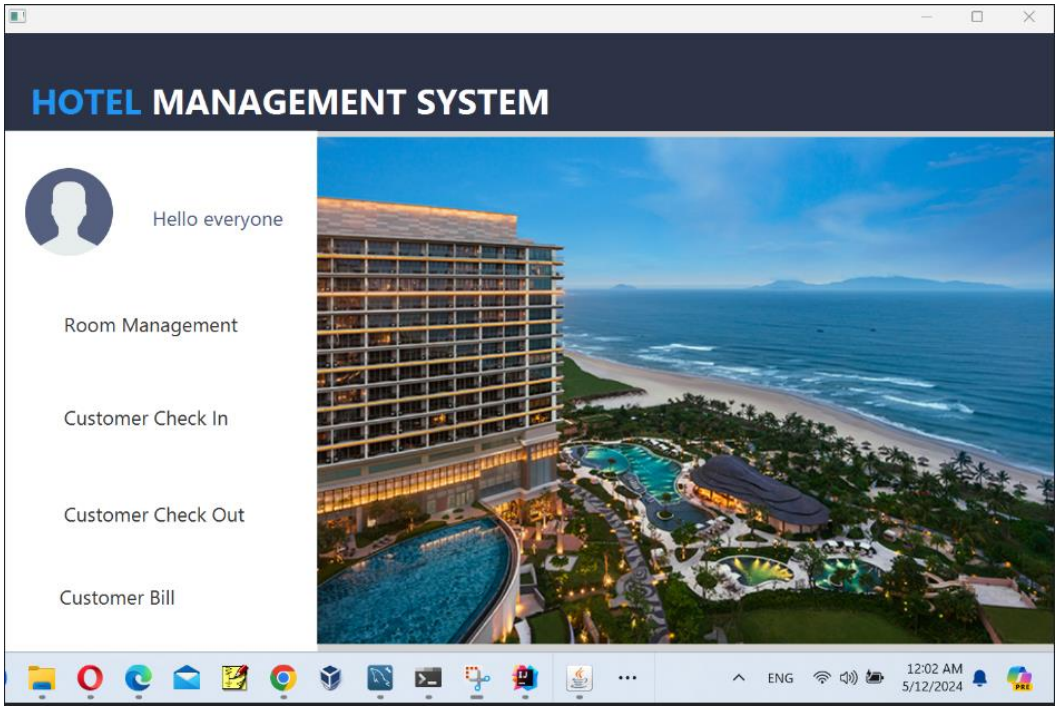


Figure 1:Interface of Hotel Mangement System.

From this basic interface we can manage our database and do all operation on the table, this operation can done when the user click on Room Management: this for add the Room on the hotel. Customer Check in this one used to insert The Customer on The Database of The Hotel. And the Customer Check Out is to log the customer out from The Database, this mean that the customer will be out from the hotel finally, the Customer Bill from this we can see the Bill of the Customer and Print It.

2.2 The Room Management

when we click on Add Room, the result will be :

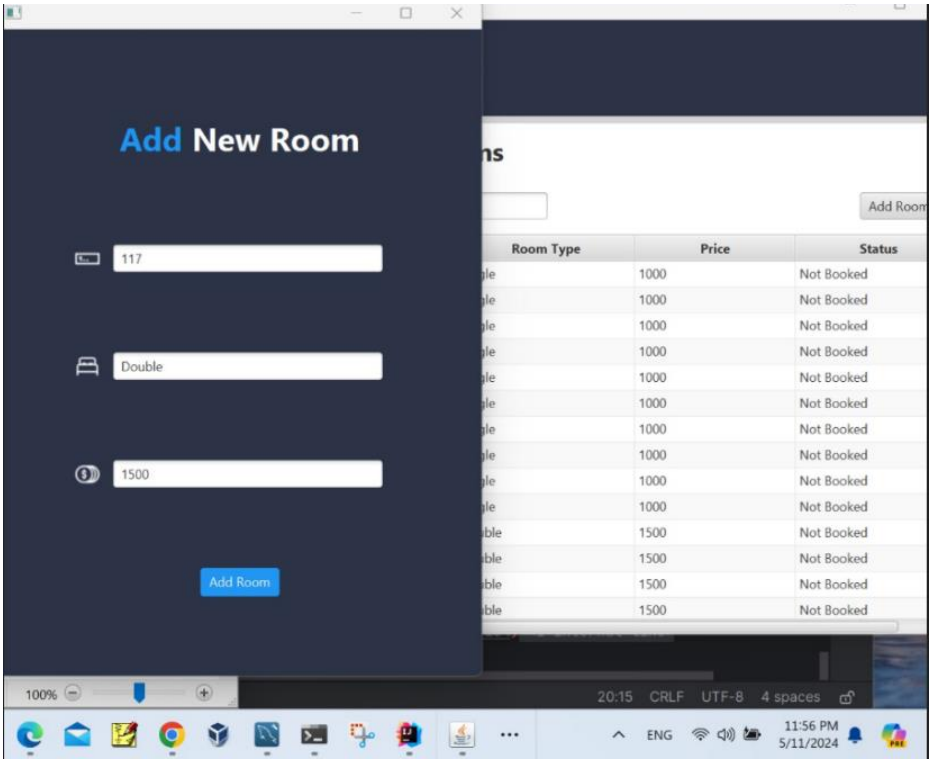


Figure 2:Add New Room InterFace .

Notice: Once We click on The Room Management the table of room will appear and **button Add Room** once we click on it the interface of Add New Room will appear, and from it we can insert a new Room with specific feature and **notice the room will be in the first Not booked.**

So after we **insert all the rooms,** the result on the table of room will be:

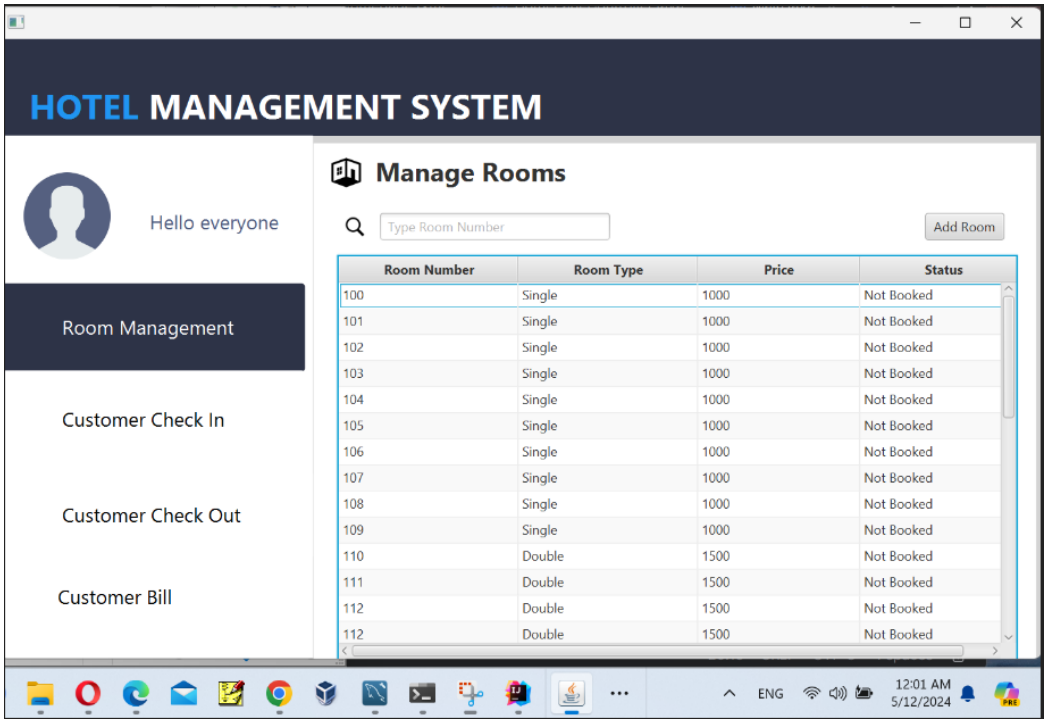


Figure 3:Room Table After Insert .

2.3 Customer Check in

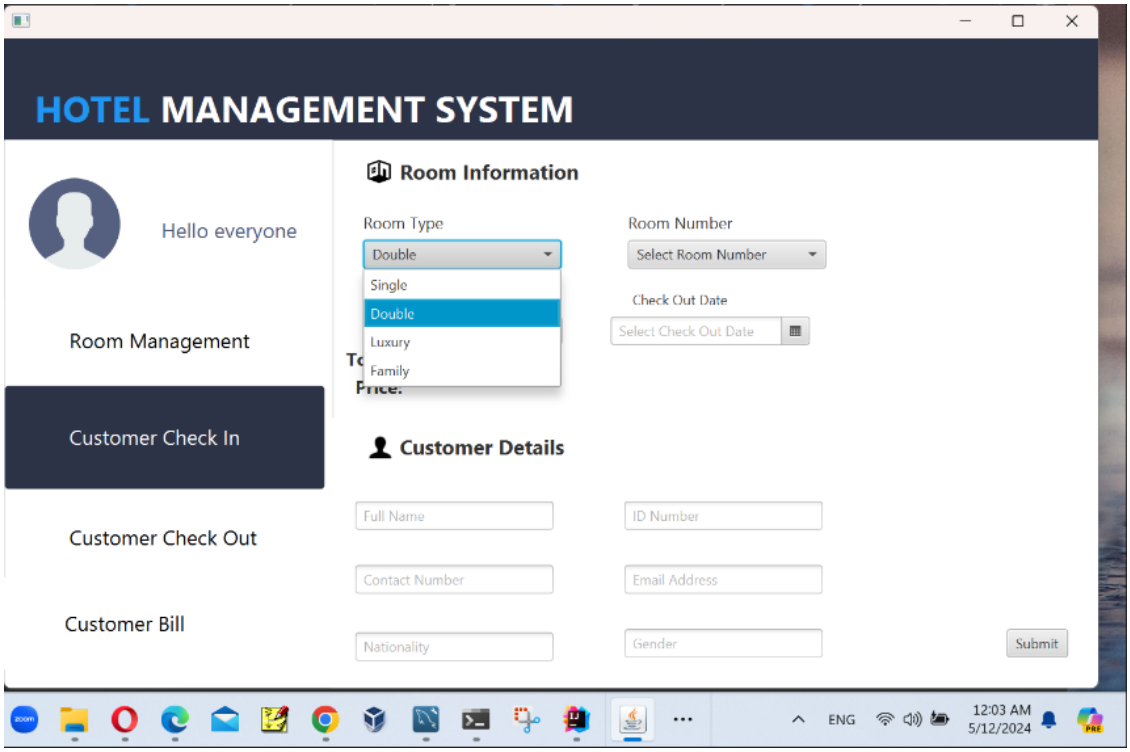


Figure 4:Customer Check In InterFace .

Here, when we click on the Customer Check In, the previous interface will appear. From it we can select the room type, the room number, write the Check Out Date and Check In Date, and the customer details, and this will be the insertion operation on The customer table.

Select a Room Number:

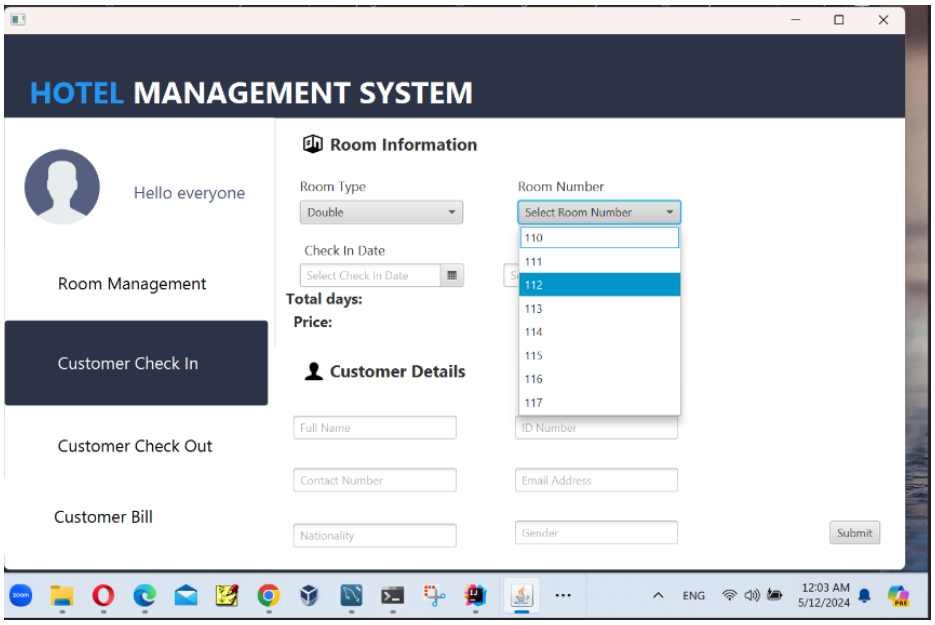


Figure 5:Select The RoomNumber.

Notice: here will display only the Number of room that are Not booked.

After we add all the details, the result will be:

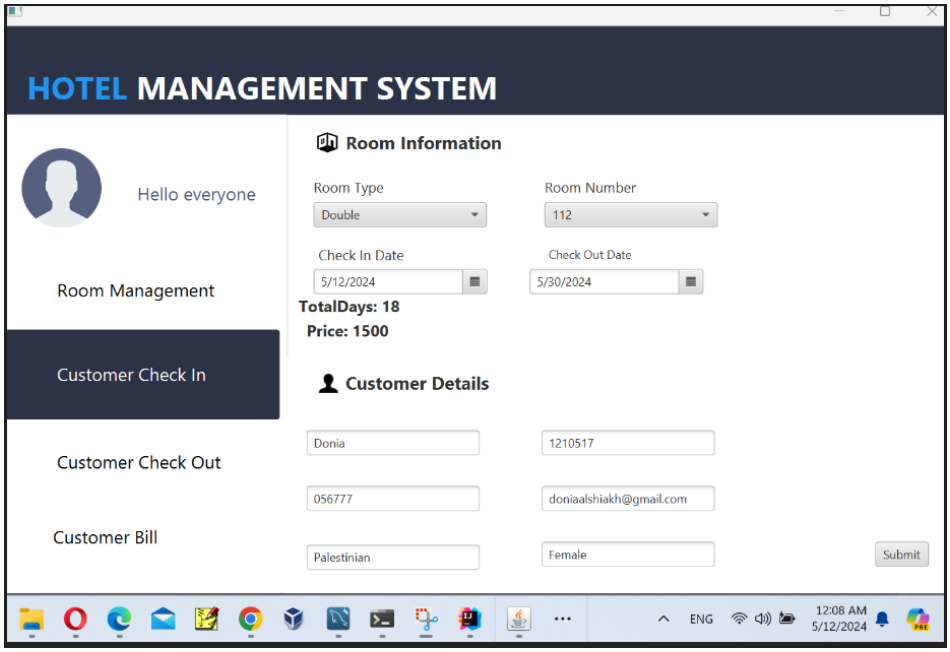


Figure 6:Customer Details.

When we click on **Submit Button**, the Room with number 102 will be booked this appear in:

Before Submit:

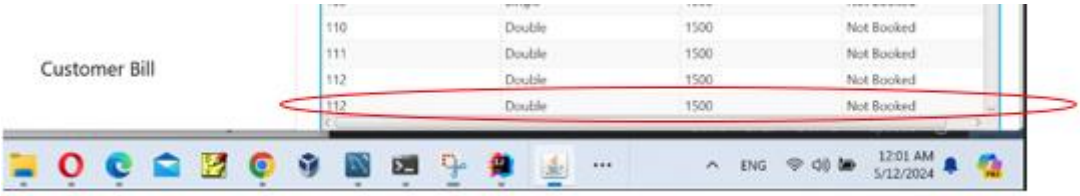
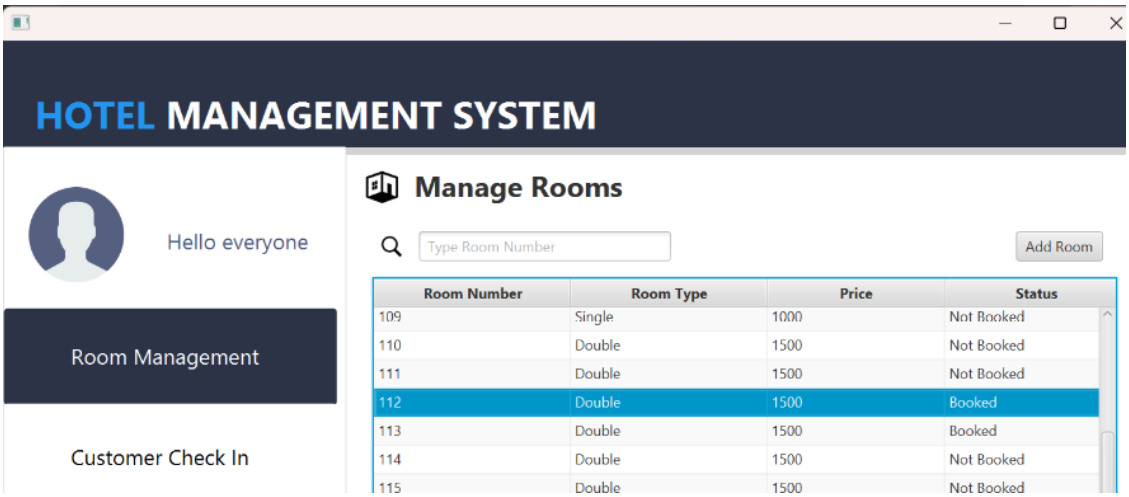


Figure 7:Submit in Customer Check in .

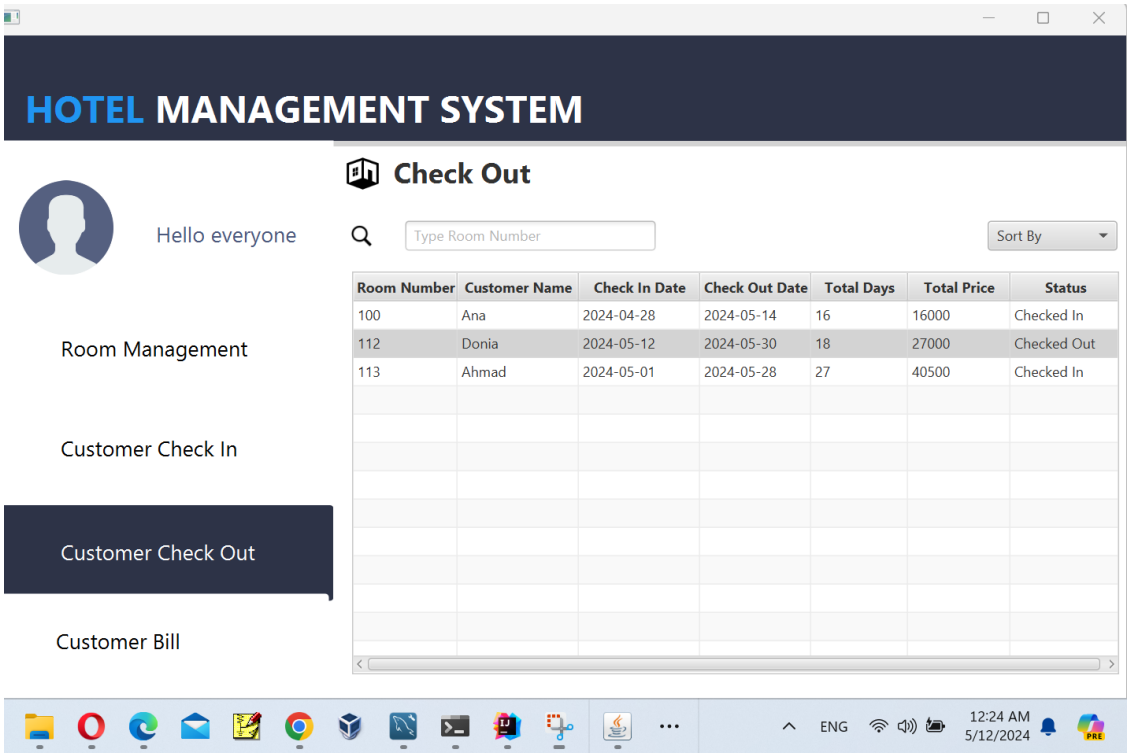
After Submit:



If we click on the tuple of the customer who check in the result will appear the customer information as follow:

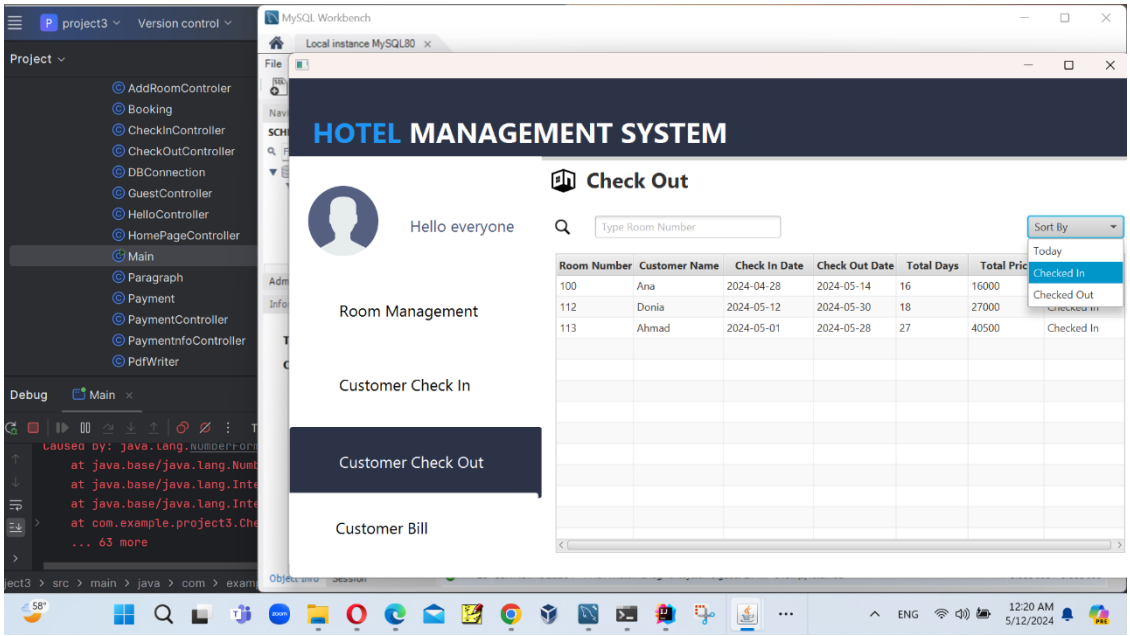
Figure 8:Customer Information .

2.4 Customer Check out

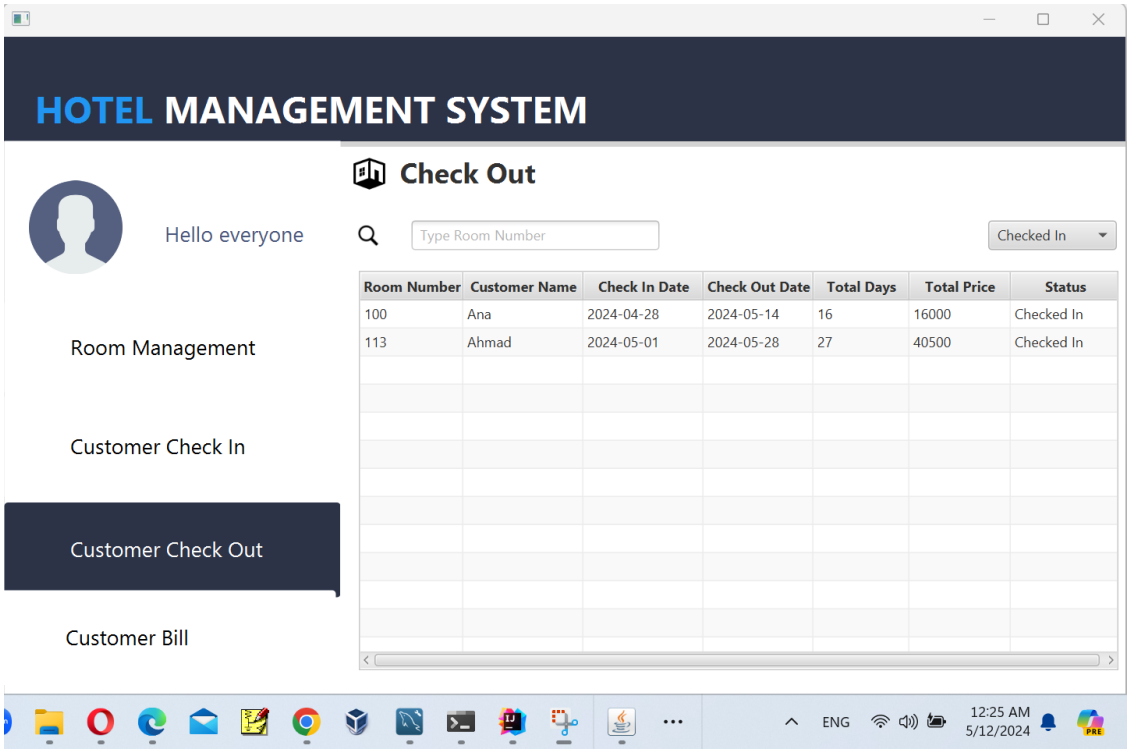


When we click on “Customer check Out”, the information of the operation of booked the room appears. Including date of check in and check out, and the price that must be paid after a number of days.

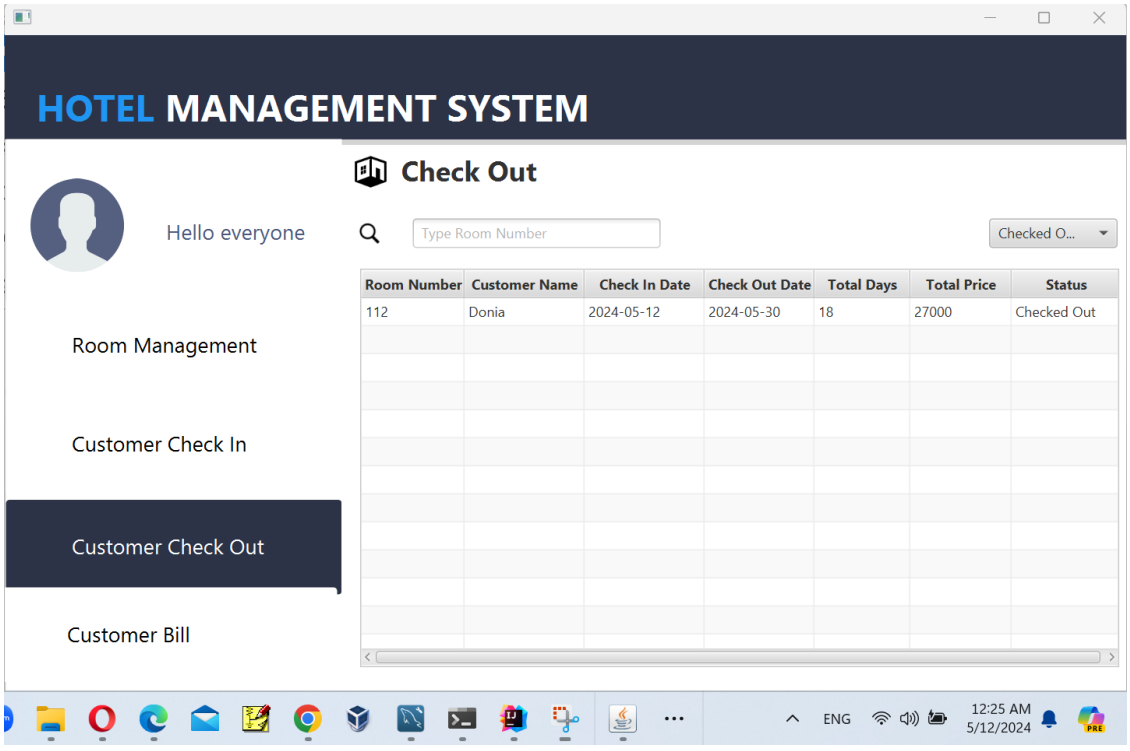
Then we can choose to show the room we were checked in to, the room we were checked out of, or the room we were checked out of today.



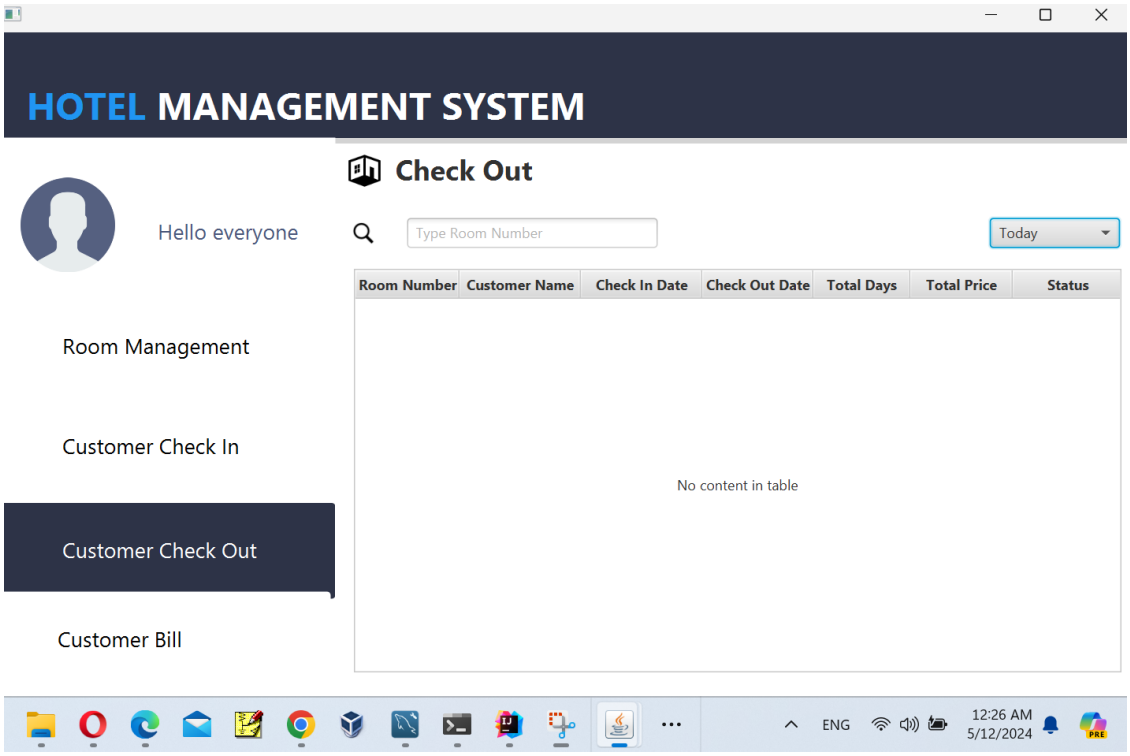
If we choose “Checked In”, then the result will be:



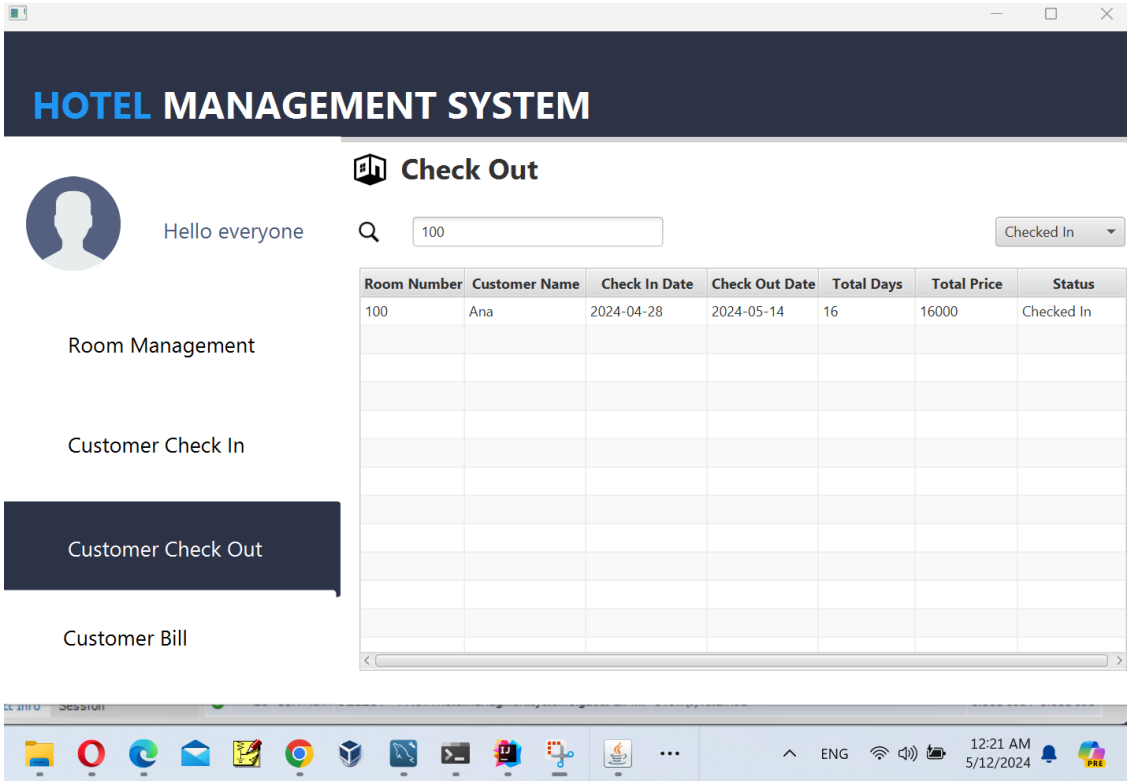
And if we choose “Checked Out”, then the result will be:



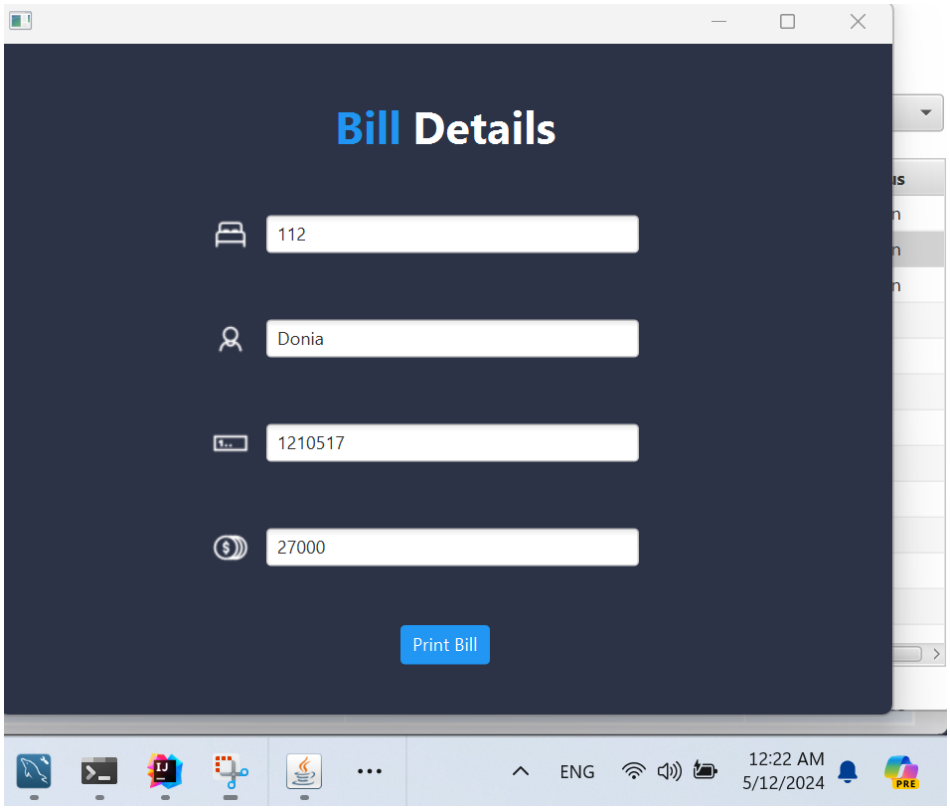
And if we choose “Today”, the result will be:



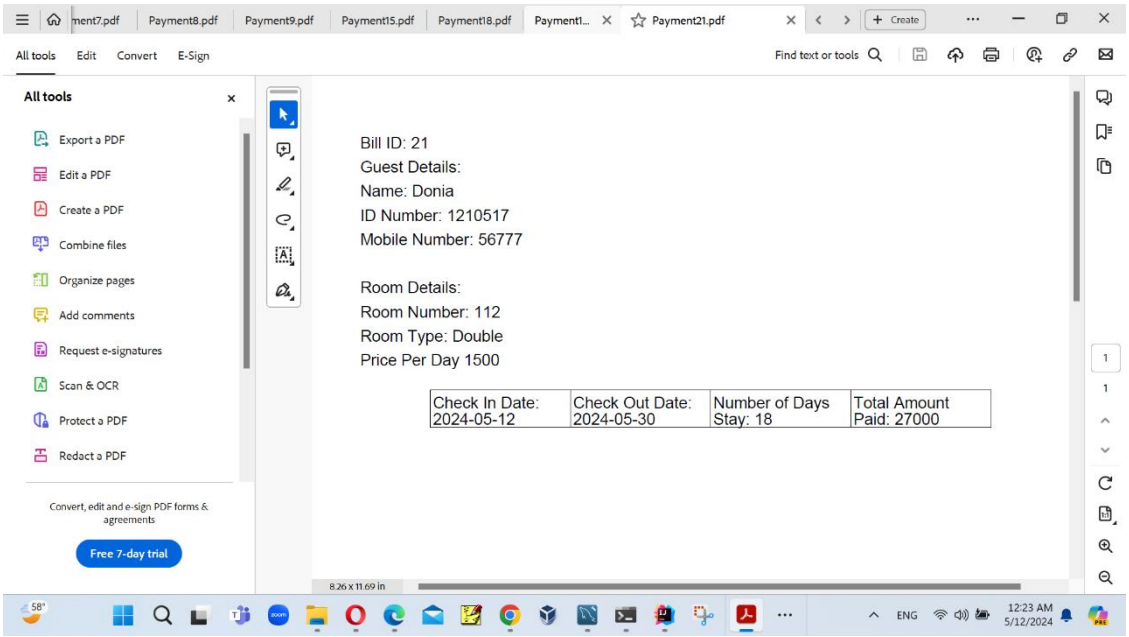
Also, we can check if the room booked or not, by enter the number of a specific room:



Also, by clicking on a tuple in the previous table, Bill details will appear in a separate window as the following figure:

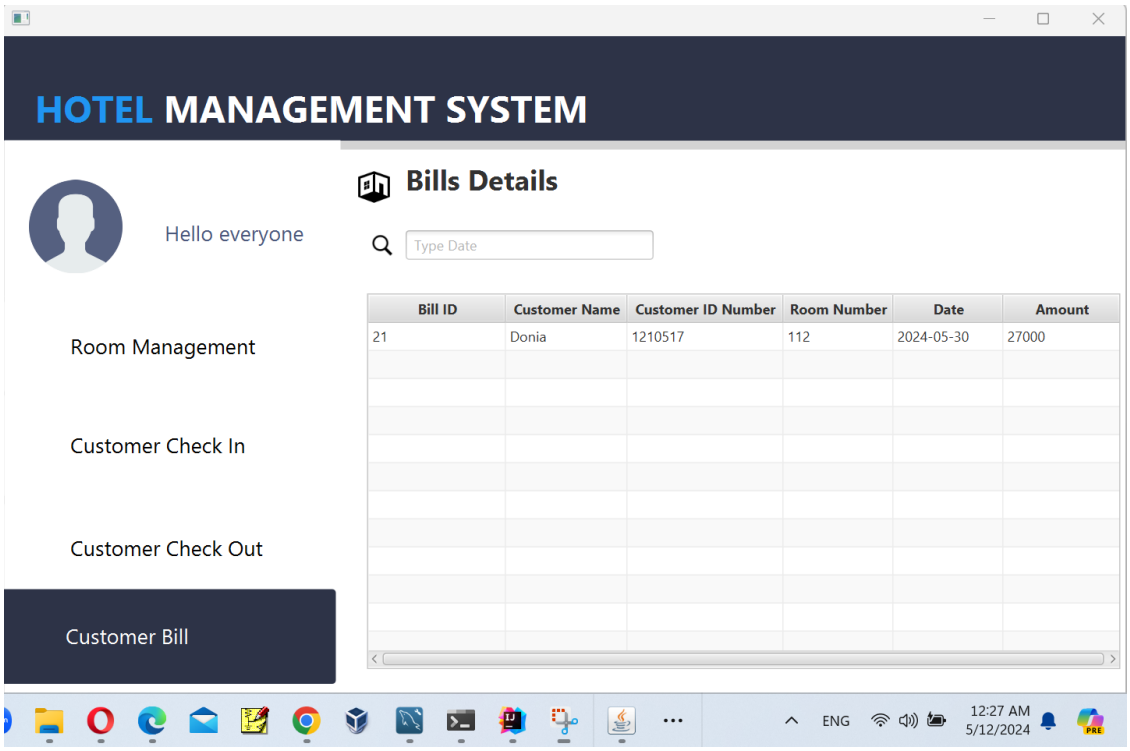


Then, by clicking in the “Print Bill” button, the Bill will be printed in your own device:



2.5 Customer Bill

When we click on “Customer Bill” in the interface, bill details will appear as follow:



These details are for the Bill that has already been paid.

Database Connection and Command Integration

Connection Setup

The connection between our Java application and the MySQL database is established using the DBConnection class. This class utilizes the JDBC API to create a connection object, which is essential for executing SQL queries and updating the database. The connection setup is initiated as soon as the initialize method is called in our controller classes, ensuring that the database is ready to interact with our Java application from the start.

```
dbConnection = new DBConnection();
connection = dbConnection.getConnection();
```

Upon calling the getConnection method, the class first attempts to load the MySQL JDBC driver. This is critical as it registers the driver with the JDBC Driver Manager, enabling it to manage the database-specific drivers.

Establishing the Connection

Once the driver is loaded, the class attempts to establish a connection to the database using the credentials and URL provided. The DriverManager.getConnection method is used for this purpose, which requires the database URL, username, and password as parameters.

So, the DBConnection class is a foundational component of our Hotel Management System, facilitating robust and secure connections to the MySQL database. It ensures that the database operations performed by other parts of the application are executed over a stable and secure connection, thereby enhancing the reliability and security of the system.

Command Execution

The integration of SQL commands within our Java application was handled through Prepared Statement objects. These objects allow us to execute SQL queries efficiently and securely, preventing SQL injection attacks.

here are the key operations performed in our application:

```
String id = "";
String insertBills = "INSERT INTO payment (BookingID, date, Amount) VALUES (?, ?, ?)";
String updateRoom = "UPDATE room SET status=\"Not Booked\" WHERE RoomNumber=?";
String updateReservation = "UPDATE booking SET status=\"Checked Out\" WHERE BookingID=?";
String selectBill = "SELECT PaymentID FROM payment WHERE BookingID=?";
if (!selectedReservation.getStatus().equals("Checked Out")) {
```

Insert Operations

When new bookings are made, or new rooms are added, our application executes insert commands. For example, in the PaymentInfoController, the booking details are inserted into the database once the user completes the payment process.

```
pst = connection.prepareStatement(insertBills);
pst.setString(1, String.valueOf(selectedReservation.getBookingID()));
pst.setString(2, selectedReservation.getCheckoutDate());
pst.setString(3, String.valueOf(selectedReservation.getTotalPrice()));
pst.executeUpdate();
```

Update Operations

The application updates the room status and reservation details upon customer check-out. The room's availability is reset, and the booking status is updated to reflect the check-out.

```
try {
    pst = connection.prepareStatement(updateReservation);
    pst.setString( parameterIndex: 1, String.valueOf(selectedReservation.getBookingID()));
    pst.executeUpdate();
} catch (SQLException e) {
    e.printStackTrace();
}
```

Selection and Retrieval

To display the bill details or get the current booking details, the application performs select operations. This is crucial for functions like printing the bill, where specific details need to be retrieved and formatted.

```
133         try {
134             pst = connection.prepareStatement(selectBill);
135             pst.setString( parameterIndex: 1, String.valueOf(selectedReservation.getBookingID()));
136             ResultSet rs = pst.executeQuery();
137             while (rs.next()) {
138                 id = rs.getString( columnLabel: "PaymentID");
139             }
140         } catch (SQLException e) {
141             e.printStackTrace();
142         }
```

Error Handling and Exceptions:

The code also includes try-catch blocks to handle SQL exceptions that may occur during database operations. This ensures the application remains efficient, usable and can recover gracefully from unexpected database errors.

Security Considerations:

While the DBConnection class includes hardcoded credentials for simplicity and ease of demonstration, in a production environment, it is important to handle these credentials more securely. Options include using environment variables, encrypted configuration files, or secure vaults to store sensitive information, thus protecting the database from unauthorized access.

Summary

Afterall, the seamless integration of the database with our Java application via JDBC and the careful handling of SQL commands have been pivotal in ensuring that the Hotel Management System functions as intended. This integration allows for real-time data processing and management, essential for maintaining an up-to-date and reliable system.