UE20CS351- OOAD USING JAVA

PROJECT REPORT ON

HOTEL MANAGEMENT SYSTEM USING SPRING MVC

Submitted by:

VAIBHAV POKHRIYAL	**PES1UG21CS837**
DHANESH K MAHTO	**PES1UG21CS808**
VAISHNAVI K	**PES1UG21CS838**

Under the guidance of

Prof. Bhargavi Mokashi
Assistant Professor

January - May 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING
PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PROBLEM STATEMENT

A hotel management system is a software solution that streamlines the day-to-day operations of a hotel, allowing management to focus on the business side of things rather than spending time on administrative tasks. The purpose of this project is to develop a web-based hotel management system using the Spring MVC framework, which will automate most of the processes involved in managing a hotel.

The hotel management system will have different user roles with varying levels of access to the system. The administrator will have full access to the system and can manage all aspects of the hotel, including room management, guest management, and employee management. The receptionist will be able to perform tasks such as guest check-in and check-out, room allocation, and room status updates. The system will also provide a self-service portal for guests to make reservations, check room availability.

The system will store and manage data related to guests, rooms, reservations, and billing. The administrator will be able to manage room types, rates, and availability.

The main objectives of this project are:

1. To develop a web-based hotel management system using the Spring MVC framework.
2. To automate most of the processes involved in managing a hotel.
3. To provide a user-friendly interface for guests and staff.
4. To ensure data security and confidentiality.
5. To provide scalability for future growth of the hotel.

Overall, the project aims to provide a comprehensive and efficient solution for hotel management that will streamline the day-to-day operations of the hotel, enhance guest experience, and increase revenue.

REQUIREMENTS

Functional Requirements:

1. Room Management: The system should allow the hotel staff to manage different types of rooms, their availability, and rates.
2. Reservation Management: The system should allow customers to make reservations for rooms, either online or through the hotel's front desk.
3. Check-in/Check-out Management: The system should allow staff to check-in guests and manage their stay details, including room assignments, length of stay, and check-out.

Non-Functional Requirements:

1. Security: The system should have robust security measures to protect sensitive guest information and financial transactions.
2. Scalability: The system should be able to handle a large number of guests and transactions without any performance degradation.
3. Usability: The system should be user-friendly and easy to use for both guests and hotel staff.
4. Reliability: The system should be highly reliable and available 24/7, as any downtime can lead to lost business and a poor guest experience.
5. Performance: The system should have fast response times, and all the features should be efficient in terms of processing speed and memory usage.

ARCHITECTURE

In the context of the Hotel Management System, the MVC architecture is suitable as it allows us to separate the application's business logic, presentation, and user input processing into distinct components. This separation enhances the modularity and maintainability of the system.

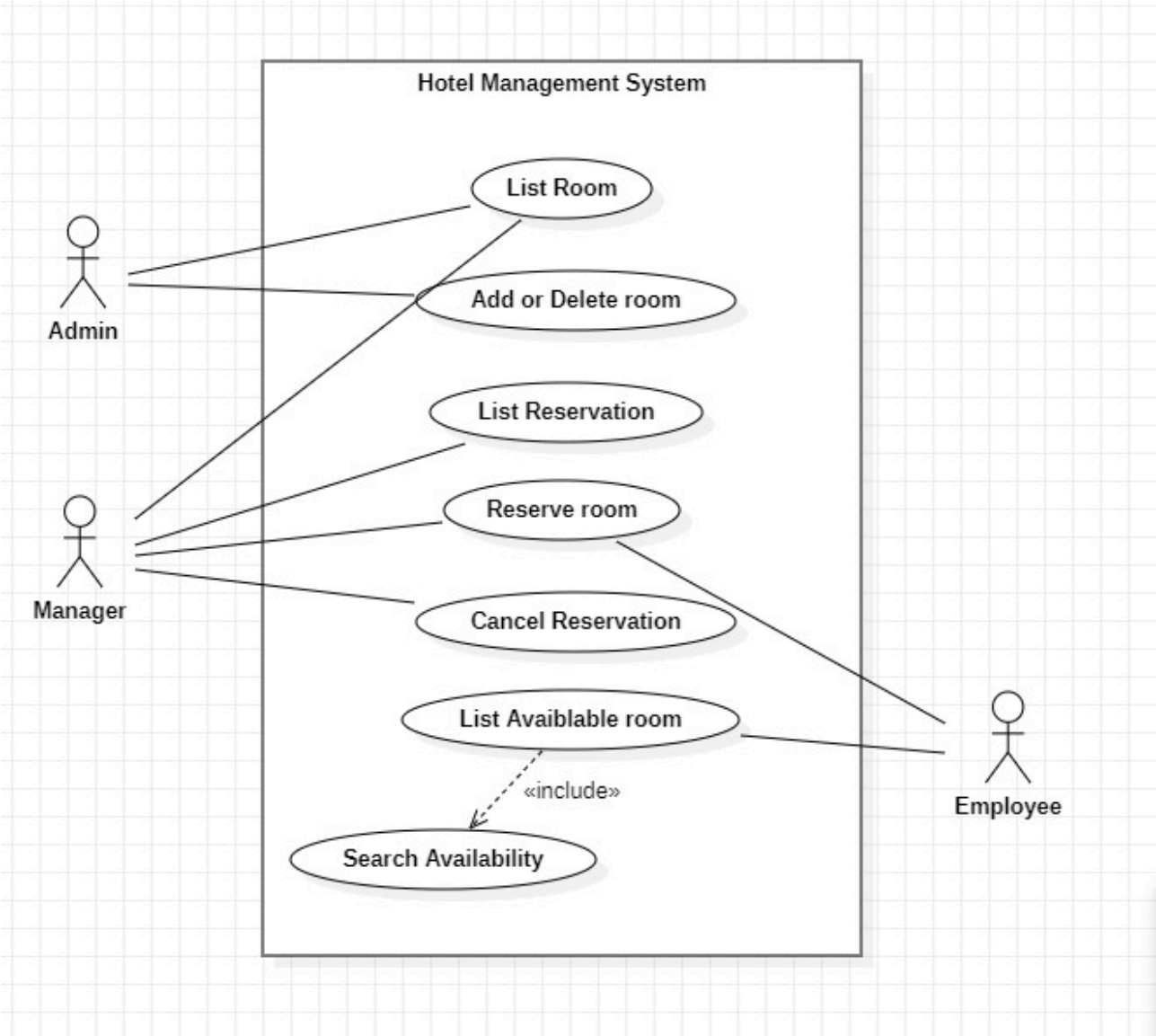
The three main components of the Hotel Management System in MVC architecture are:

1. Model: The model is responsible for managing the data and business logic of the system. In the Hotel Management System, the model component would include entities such as Room, Reservation, Guest and other necessary classes. The model component would handle data storage, retrieval, and manipulation, and interact with the persistence layer (database).
2. View: The view is responsible for presenting the data and user interface of the system to the user. In the Hotel Management System, the view component would handle the presentation of information to guests and staff, including information about available rooms, room rates, reservations. The view component would also include user interface elements such as forms.
3. Controller: In the Hotel Management System, the controller component would handle user input, validate user input, and update the model and view components accordingly.

The interactions between the components of the Hotel Management System can be summarized as follows:

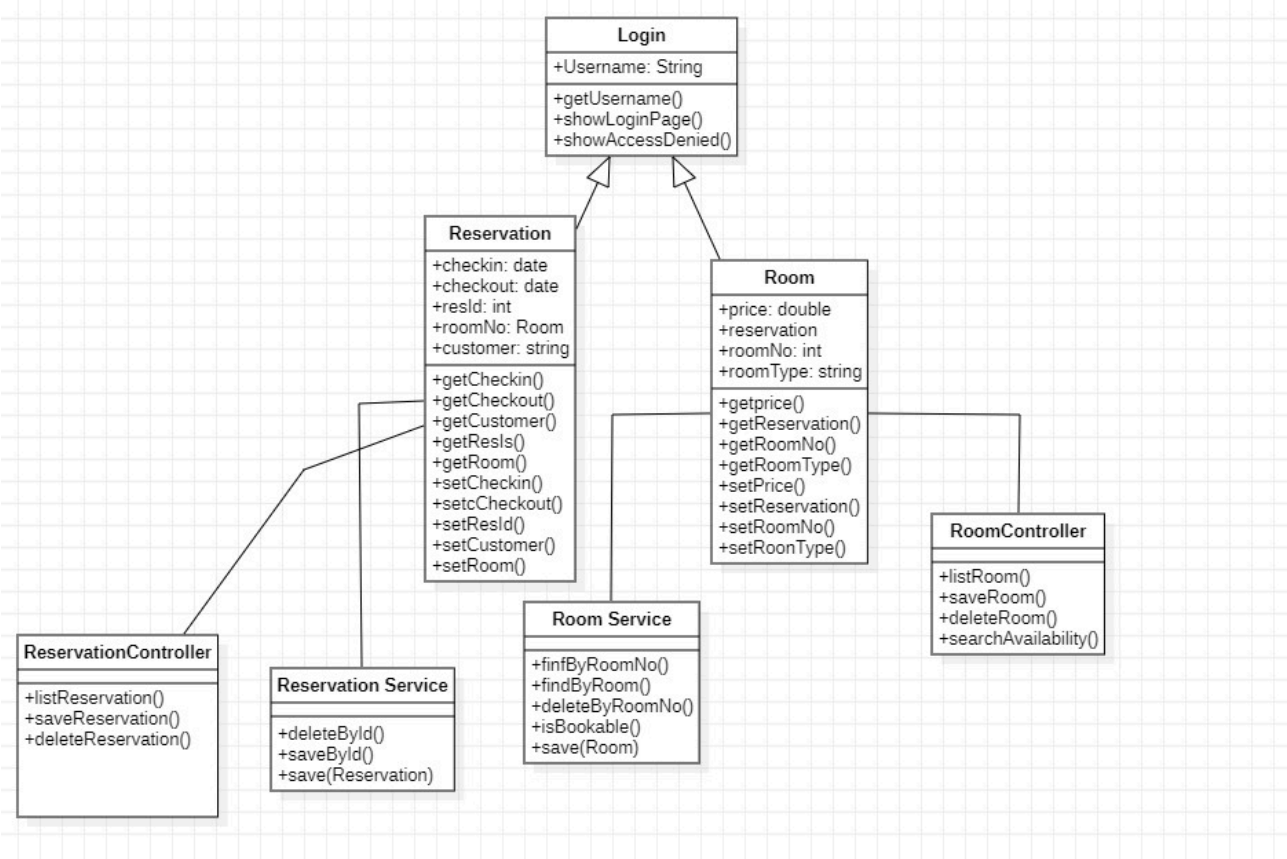
1. The user interacts with the view component by providing input through the user interface.
2. The view component sends the user input to the controller component for processing.
3. The controller component processes the user input and interacts with the model component to retrieve or update the necessary data.
4. The controller component updates the view component with the new data.
5. The view component presents the updated data to the user.

UML MODELS



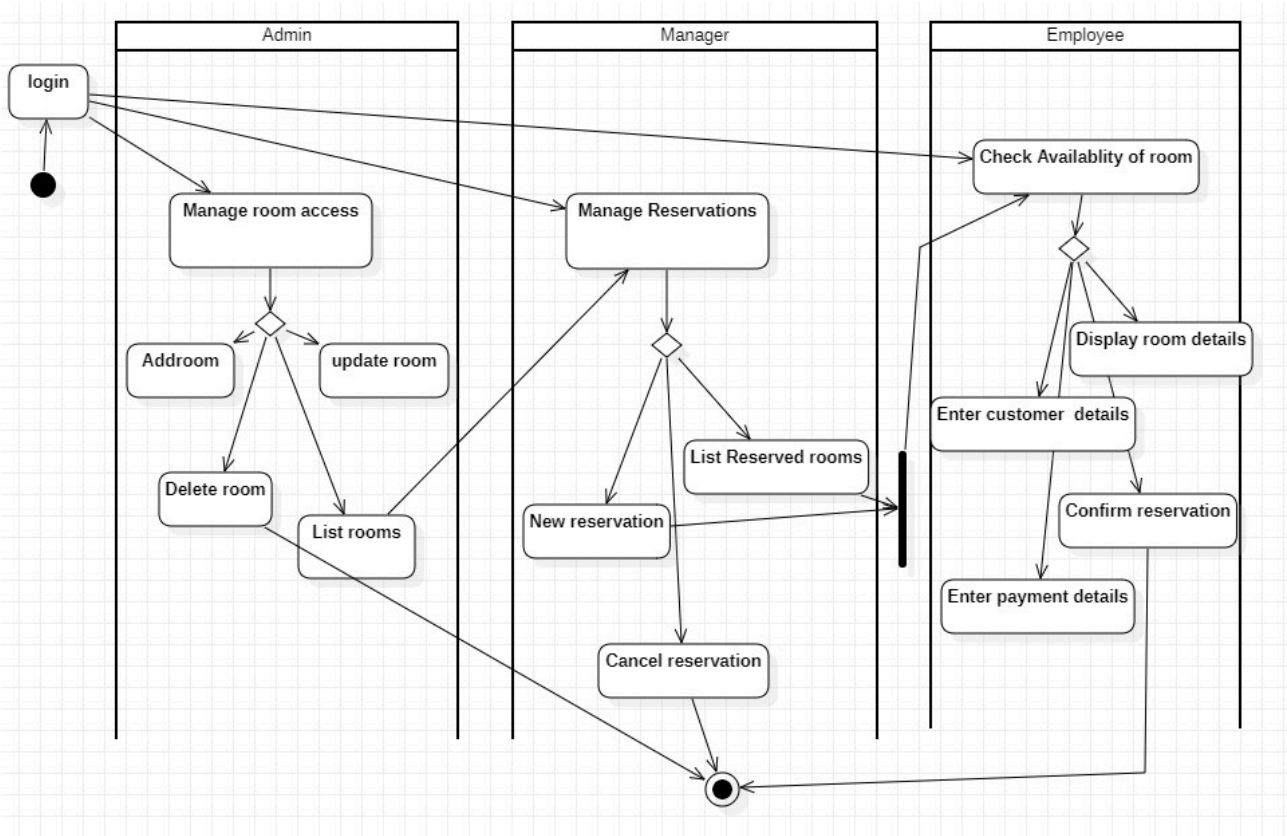
USE CASE DIAGRAM

UML MODELS



CLASS DIAGRAM

UML MODELS



ACTIVITY DIAGRAM

DESIGN PRINCIPLES USED

1. Dependency Injection (DI): Dependency Injection is a design pattern used to remove hard-coded dependencies between objects by injecting dependencies into the object at runtime. It enables loose coupling between objects and makes the code more modular and testable. In Spring Boot, DI is achieved using the Inversion of Control (IoC) principle, where the framework manages the creation and injection of dependencies.
2. Factory Pattern: The Factory pattern is a creational design pattern used to create objects without exposing the instantiation logic to the client. It provides an interface to create objects of a class but delegates the actual creation of the object to a separate factory class. This pattern allows for flexibility in object creation and decouples the client code from the object creation process.
3. Singleton Pattern: The Singleton pattern is a creational design pattern that restricts the instantiation of a class to a single instance and provides a global point of access to that instance. This pattern is commonly used in scenarios where a single instance of a class needs to be shared across the entire application.
4. Template Method Pattern: The Template Method pattern is a behavioral design pattern used to define the skeleton of an algorithm in a base class while allowing subclasses to override specific steps of the algorithm. This pattern promotes code reuse and enables customization of the algorithm's behavior by subclasses.
5. Separation of concerns: Separation of concerns is an important principle in software design, and it can be applied to many programming languages including Java. In Java, separation of concerns refers to the practice of organizing code into distinct modules or components that handle specific tasks or responsibilities, without interfering with each other.

IMPLEMENTATION

GITHUB REPOSITORY LINK :-

https://github.com/VaibhavPokhriyal2510/UE20CS352_OOADJ_Mini_Project-MVC.git

DATABASE USED FOR IMPLEMENTATION

MySQL is an open-source relational database management system (RDBMS) that is widely used in web development and other applications that require the storage, retrieval, and management of large amounts of data. MySQL was first released in 1995 and is now owned and maintained by Oracle Corporation.

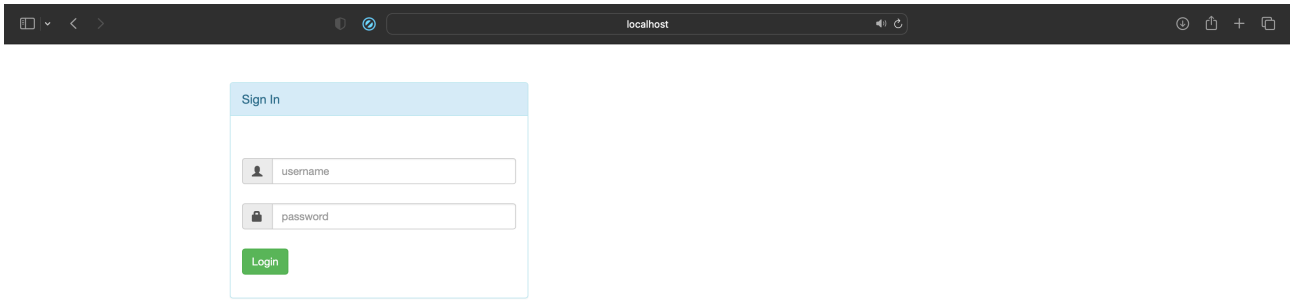
MySQL supports a variety of programming languages, including Java, Python, PHP, and C++, and can be used with many different operating systems, including Linux, Windows, and macOS. It is known for its high performance, scalability, and reliability, and is used by many large organizations, including Facebook, Twitter, and Google.

MySQL uses a client-server architecture, where the MySQL server manages the database and responds to client requests for data. Clients can be applications running on the same machine as the server, or they can be remote applications connecting to the server over a network. MySQL supports a variety of storage engines, including InnoDB, MyISAM, and MEMORY, which provide different features and trade-offs between performance and data integrity.

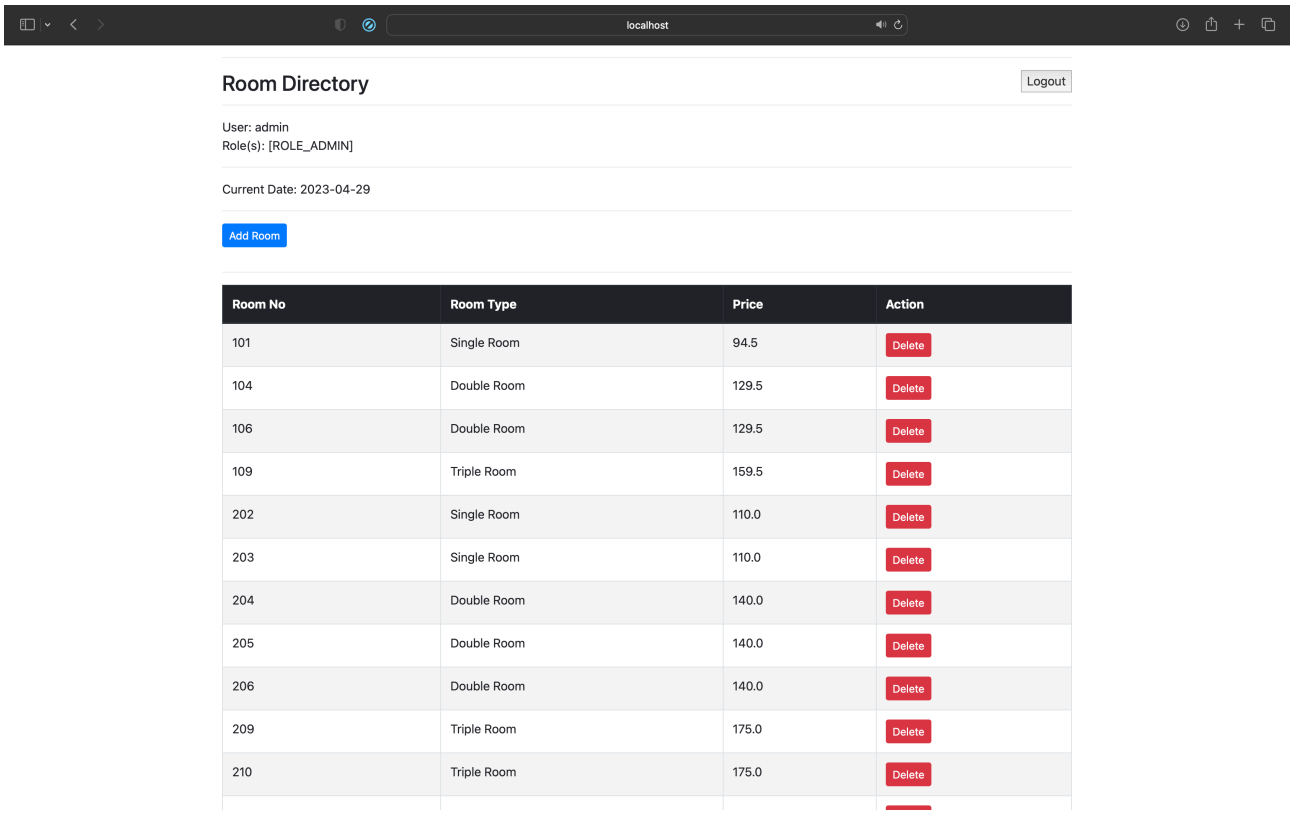
MySQL uses a SQL (Structured Query Language) interface to interact with the database, allowing users to create, read, update, and delete data using a set of standard commands. MySQL also supports advanced features like triggers, stored procedures, and transactions, which allow developers to build complex database-driven applications.

Overall, MySQL is a powerful and widely used database management system that offers a variety of features and benefits for developers and organizations looking to manage large amounts of data efficiently and reliably.

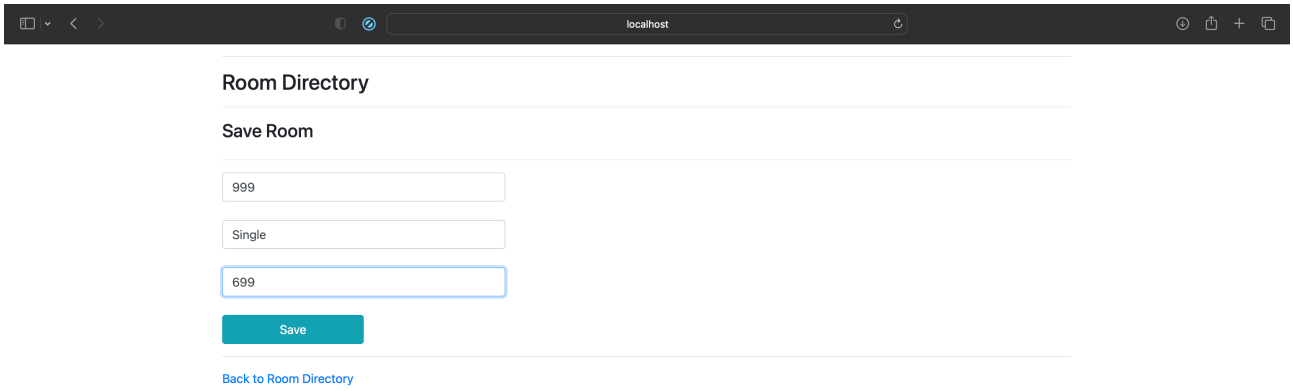
SCREENSHOTS



LOGIN PAGE



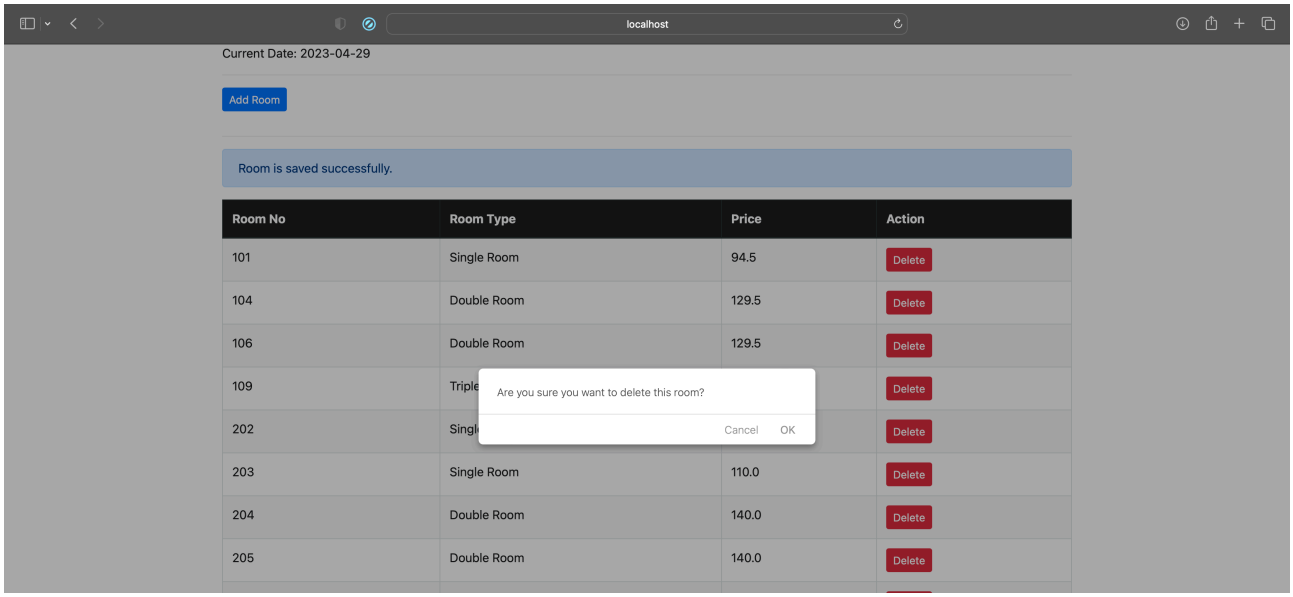
ADMIN PAGE



ADDING ROOM

```
MariaDB [hotel_management_app]> select * from room;
+-----+-------------+-------+
| no  | type        | price |
+-----+-------------+-------+
| 101 | Single Room |  94.5 |
| 104 | Double Room | 129.5 |
| 106 | Double Room | 129.5 |
| 109 | Triple Room | 159.5 |
| 202 | Single Room |   110 |
| 203 | Single Room |   110 |
| 204 | Double Room |   140 |
| 205 | Double Room |   140 |
| 206 | Double Room |   140 |
| 209 | Triple Room |   175 |
| 210 | Triple Room |   175 |
| 600 | hjjh        |   678 |
+-----+-------------+-------+
12 rows in set (0.001 sec)

MariaDB [hotel_management_app]> select * from room;
+-----+-------------+-------+
| no  | type        | price |
+-----+-------------+-------+
| 101 | Single Room |  94.5 |
| 104 | Double Room | 129.5 |
| 106 | Double Room | 129.5 |
| 109 | Triple Room | 159.5 |
| 202 | Single Room |   110 |
| 203 | Single Room |   110 |
| 204 | Double Room |   140 |
| 205 | Double Room |   140 |
| 206 | Double Room |   140 |
| 209 | Triple Room |   175 |
| 600 | hjjh        |   678 |
| 999 | Single      |   699 |
+-----+-------------+-------+
13 rows in set (0.002 sec)

MariaDB [hotel_management_app]>
```

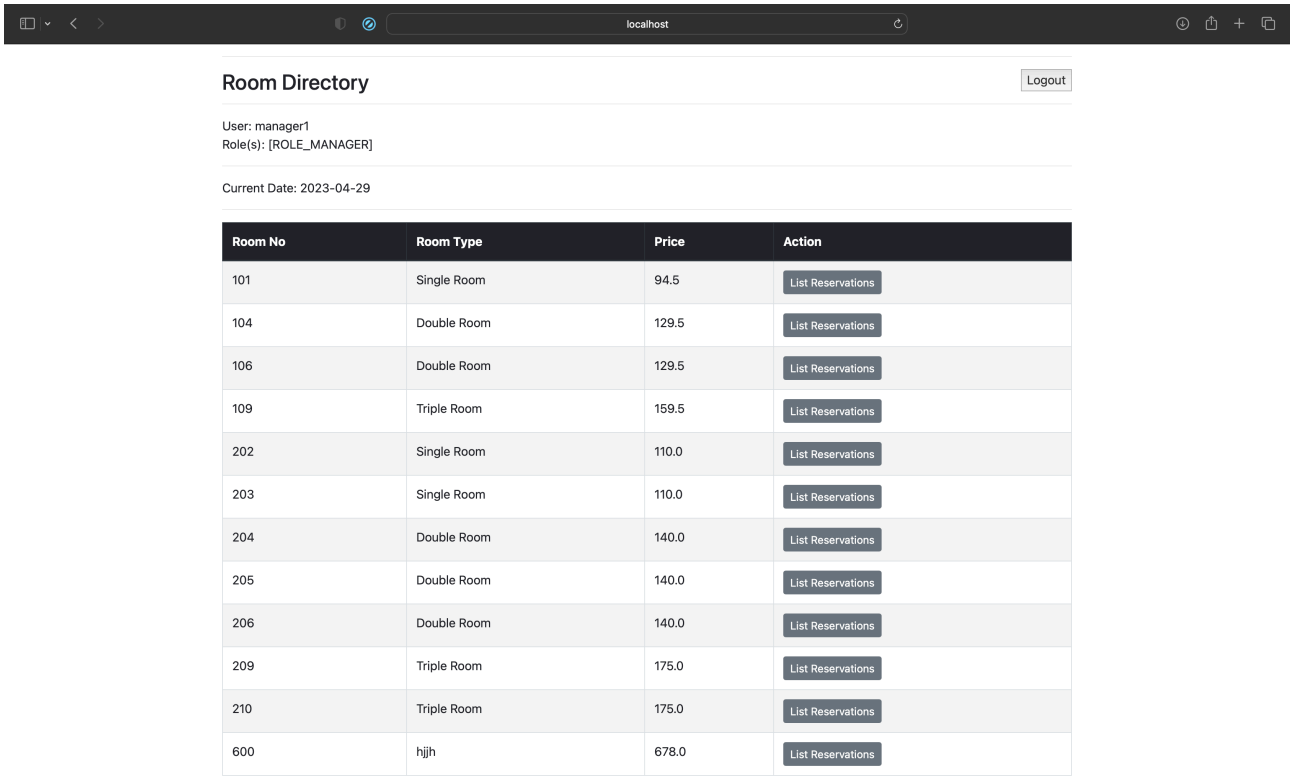
DATABASE BEFORE AND AFTER UPDATION



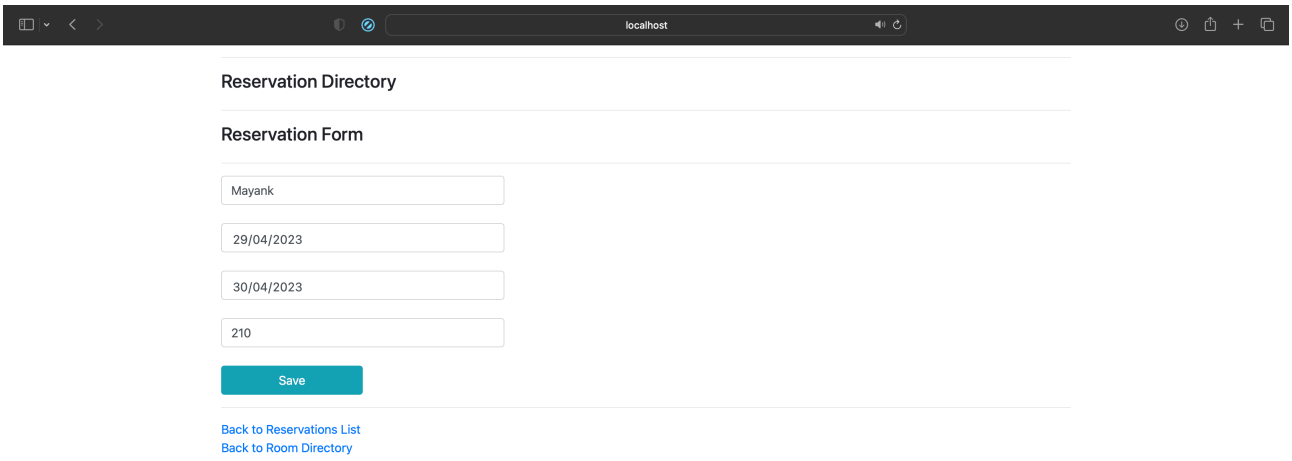
DELETING ROOM NO-999

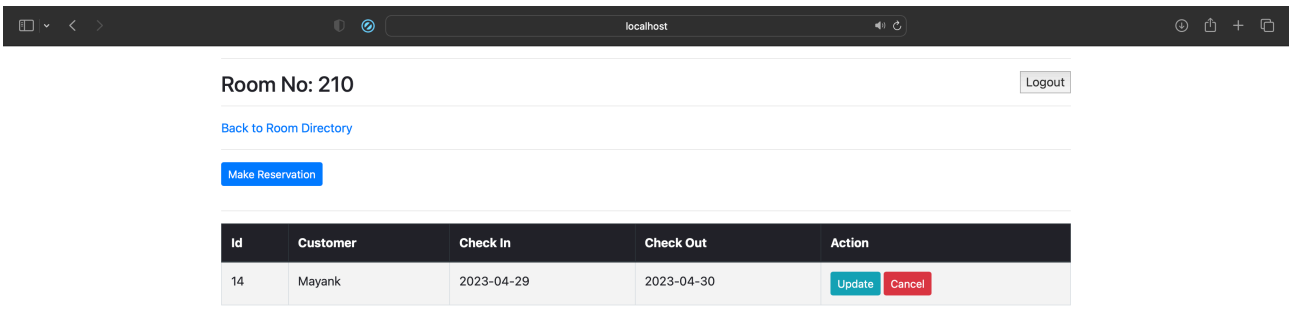
```
MariaDB [hotel_management_app]> select * from room;
+-----+-------------+-------+
| no  | type        | price |
+-----+-------------+-------+
| 101 | Single Room |  94.5 |
| 104 | Double Room | 129.5 |
| 106 | Double Room | 129.5 |
| 109 | Triple Room | 159.5 |
| 202 | Single Room |   110 |
| 203 | Single Room |   110 |
| 204 | Double Room |   140 |
| 205 | Double Room |   140 |
| 206 | Double Room |   140 |
| 209 | Triple Room |   175 |
| 210 | Triple Room |   175 |
| 600 | hjjh        |   678 |
| 999 | Single      |   699 |
+-----+-------------+-------+
13 rows in set (0.002 sec)

MariaDB [hotel_management_app]> select * from room;
+-----+-------------+-------+
| no  | type        | price |
+-----+-------------+-------+
| 101 | Single Room |  94.5 |
| 104 | Double Room | 129.5 |
| 106 | Double Room | 129.5 |
| 109 | Triple Room | 159.5 |
| 202 | Single Room |   110 |
| 203 | Single Room |   110 |
| 204 | Double Room |   140 |
| 205 | Double Room |   140 |
| 206 | Double Room |   140 |
| 209 | Triple Room |   175 |
| 210 | Triple Room |   175 |
| 600 | hjjh        |   678 |
+-----+-------------+-------+
12 rows in set (0.001 sec)
```

DATABASE BEFORE AND AFTER DELETION



LOGGED IN AS MANAGER

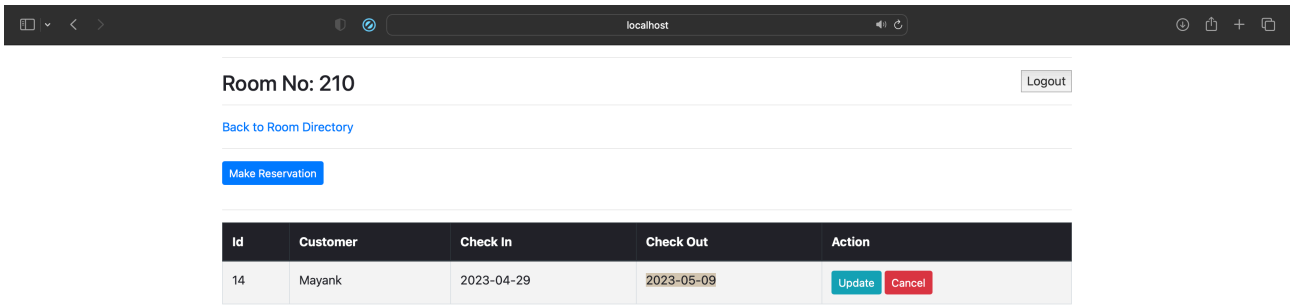




ADDING RESERVATION FOR ROOM 210

```
MariaDB [hotel_management_app]> select * from reservation;
+----+-------------------+------------+------------+---------+
| id | customer          | check_in   | check_out  | room_no |
+----+-------------------+------------+------------+---------+
|  2 | Rueben Tucker     | 2023-01-14 | 2023-01-16 |     101 |
|  3 | Owain Mccall      | 2023-01-16 | 2023-01-21 |     101 |
|  4 | Zaid Stevens      | 2023-01-21 | 2023-01-23 |     101 |
|  5 | Ava-Rose Byrd     | 2023-01-24 | 2023-01-27 |     101 |
|  6 | Thomas House      | 2023-01-28 | 2023-02-01 |     101 |
|  7 | Joseph Dawson     | 2023-02-04 | 2023-02-10 |     101 |
|  8 | Jaya Rose         | 2023-02-15 | 2023-02-18 |     101 |
|  9 | Keiran Valencia   | 2023-02-19 | 2023-02-26 |     101 |
| 10 | Lillie Kramer     | 2023-02-26 | 2023-03-03 |     101 |
| 11 | Vaibhav           | 2023-04-27 | 2023-04-28 |     600 |
| 13 | Vaibhav Pokhriyal | 2023-04-28 | 2023-04-29 |     104 |
+----+-------------------+------------+------------+---------+
11 rows in set (0.003 sec)

MariaDB [hotel_management_app]> select * from reservation;
+----+-------------------+------------+------------+---------+
| id | customer          | check_in   | check_out  | room_no |
+----+-------------------+------------+------------+---------+
|  2 | Rueben Tucker     | 2023-01-14 | 2023-01-16 |     101 |
|  3 | Owain Mccall      | 2023-01-16 | 2023-01-21 |     101 |
|  4 | Zaid Stevens      | 2023-01-21 | 2023-01-23 |     101 |
|  5 | Ava-Rose Byrd     | 2023-01-24 | 2023-01-27 |     101 |
|  6 | Thomas House      | 2023-01-28 | 2023-02-01 |     101 |
|  7 | Joseph Dawson     | 2023-02-04 | 2023-02-10 |     101 |
|  8 | Jaya Rose         | 2023-02-15 | 2023-02-18 |     101 |
|  9 | Keiran Valencia   | 2023-02-19 | 2023-02-26 |     101 |
| 10 | Lillie Kramer     | 2023-02-26 | 2023-03-03 |     101 |
| 11 | Vaibhav           | 2023-04-27 | 2023-04-28 |     600 |
| 13 | Vaibhav Pokhriyal | 2023-04-28 | 2023-04-29 |     104 |
| 14 | Mayank            | 2023-04-29 | 2023-04-30 |     210 |
+----+-------------------+------------+------------+---------+
12 rows in set (0.002 sec)

MariaDB [hotel_management_app]>
```

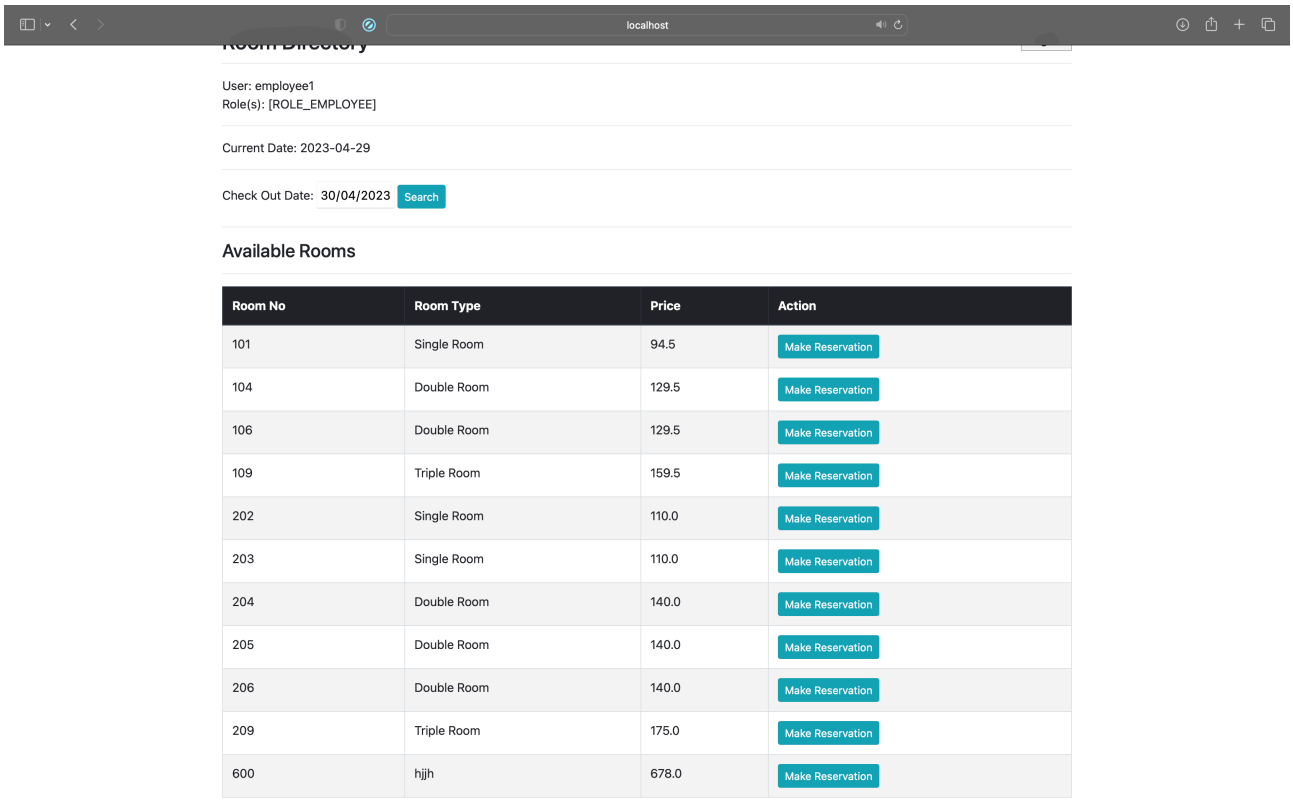
DATABASE BEFORE AND AFTER RESERVATION



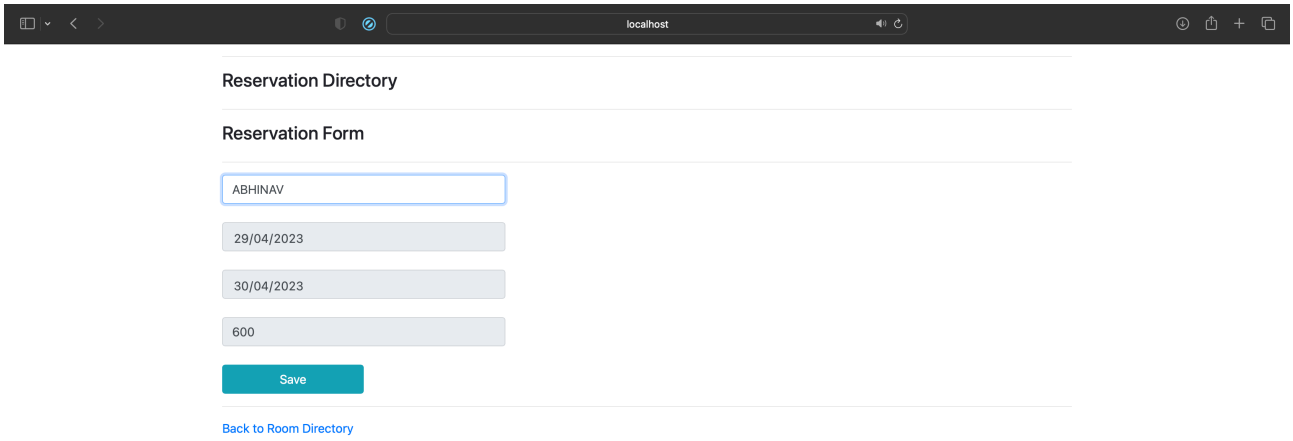
UPDATING DATE OF ROOM RESERVATION

```
MariaDB [hotel_management_app]> select * from reservation;
+----+------------------+------------+------------+---------+
| id | customer         | check_in   | check_out  | room_no |
+----+------------------+------------+------------+---------+
|  2 | Rueben Tucker    | 2023-01-14 | 2023-01-16 |     101 |
|  3 | Owain Mccall     | 2023-01-16 | 2023-01-21 |     101 |
|  4 | Zaid Stevens     | 2023-01-21 | 2023-01-23 |     101 |
|  5 | Ava-Rose Byrd    | 2023-01-24 | 2023-01-27 |     101 |
|  6 | Thomas House     | 2023-01-28 | 2023-02-01 |     101 |
|  7 | Joseph Dawson    | 2023-02-04 | 2023-02-10 |     101 |
|  8 | Jaya Rose        | 2023-02-15 | 2023-02-18 |     101 |
|  9 | Keiran Valencia  | 2023-02-19 | 2023-02-26 |     101 |
| 10 | Lillie Kramer    | 2023-02-26 | 2023-03-03 |     101 |
| 11 | Vaibhav          | 2023-04-27 | 2023-04-28 |     600 |
| 13 | Vaibhav Pokhriyal | 2023-04-28 | 2023-04-29 |     104 |
| 14 | Mayank           | 2023-04-29 | 2023-04-30 |     210 |
+----+------------------+------------+------------+---------+
12 rows in set (0.002 sec)

MariaDB [hotel_management_app]> select * from reservation;
+----+------------------+------------+------------+---------+
| id | customer         | check_in   | check_out  | room_no |
+----+------------------+------------+------------+---------+
|  2 | Rueben Tucker    | 2023-01-14 | 2023-01-16 |     101 |
|  3 | Owain Mccall     | 2023-01-16 | 2023-01-21 |     101 |
|  4 | Zaid Stevens     | 2023-01-21 | 2023-01-23 |     101 |
|  5 | Ava-Rose Byrd    | 2023-01-24 | 2023-01-27 |     101 |
|  6 | Thomas House     | 2023-01-28 | 2023-02-01 |     101 |
|  7 | Joseph Dawson    | 2023-02-04 | 2023-02-10 |     101 |
|  8 | Jaya Rose        | 2023-02-15 | 2023-02-18 |     101 |
|  9 | Keiran Valencia  | 2023-02-19 | 2023-02-26 |     101 |
| 10 | Lillie Kramer    | 2023-02-26 | 2023-03-03 |     101 |
| 11 | Vaibhav          | 2023-04-27 | 2023-04-28 |     600 |
| 13 | Vaibhav Pokhriyal | 2023-04-28 | 2023-04-29 |     104 |
| 14 | Mayank           | 2023-04-29 | 2023-05-09 |     210 |
+----+------------------+------------+------------+---------+
12 rows in set (0.002 sec)

MariaDB [hotel_management_app]>
```

DATABASE BEFORE AND AFTER UPDATION



ROOM 210 IS NOT DISPLAYED TO USER AS IT IS RESERVED



MAKING RESERVATION FOR ROOM 600

```
MariaDB [hotel_management_app]> select * from reservation;
+----+------------------+------------+------------+---------+
| id | customer         | check_in   | check_out  | room_no |
+----+------------------+------------+------------+---------+
|  2 | Rueben Tucker    | 2023-01-14 | 2023-01-16 |     101 |
|  3 | Owain Mccall     | 2023-01-16 | 2023-01-21 |     101 |
|  4 | Zaid Stevens     | 2023-01-21 | 2023-01-23 |     101 |
|  5 | Ava-Rose Byrd    | 2023-01-24 | 2023-01-27 |     101 |
|  6 | Thomas House     | 2023-01-28 | 2023-02-01 |     101 |
|  7 | Joseph Dawson    | 2023-02-04 | 2023-02-10 |     101 |
|  8 | Jaya Rose        | 2023-02-15 | 2023-02-18 |     101 |
|  9 | Keiran Valencia  | 2023-02-19 | 2023-02-26 |     101 |
| 10 | Lillie Kramer    | 2023-02-26 | 2023-03-03 |     101 |
| 11 | Vaibhav          | 2023-04-27 | 2023-04-28 |     600 |
| 13 | Vaibhav Pokhriyal | 2023-04-28 | 2023-04-29 |     104 |
| 14 | Mayank           | 2023-04-29 | 2023-05-09 |     210 |
+----+------------------+------------+------------+---------+
12 rows in set (0.004 sec)

MariaDB [hotel_management_app]> select * from reservation;
+----+------------------+------------+------------+---------+
| id | customer         | check_in   | check_out  | room_no |
+----+------------------+------------+------------+---------+
|  2 | Rueben Tucker    | 2023-01-14 | 2023-01-16 |     101 |
|  3 | Owain Mccall     | 2023-01-16 | 2023-01-21 |     101 |
|  4 | Zaid Stevens     | 2023-01-21 | 2023-01-23 |     101 |
|  5 | Ava-Rose Byrd    | 2023-01-24 | 2023-01-27 |     101 |
|  6 | Thomas House     | 2023-01-28 | 2023-02-01 |     101 |
|  7 | Joseph Dawson    | 2023-02-04 | 2023-02-10 |     101 |
|  8 | Jaya Rose        | 2023-02-15 | 2023-02-18 |     101 |
|  9 | Keiran Valencia  | 2023-02-19 | 2023-02-26 |     101 |
| 10 | Lillie Kramer    | 2023-02-26 | 2023-03-03 |     101 |
| 11 | Vaibhav          | 2023-04-27 | 2023-04-28 |     600 |
| 13 | Vaibhav Pokhriyal | 2023-04-28 | 2023-04-29 |     104 |
| 14 | Mayank           | 2023-04-29 | 2023-05-09 |     210 |
| 15 | ABHINAV          | 2023-04-29 | 2023-04-30 |     600 |
+----+------------------+------------+------------+---------+
13 rows in set (0.002 sec)

MariaDB [hotel_management_app]>
```

DATABASE BEFORE AND AFTER RESERVATION