Project Final Report

CMPE 331

Report Date: December 13, 2023

Hotel Automation

A web application that reduces the workload at the front desk and improves guest experience in hotels.

Group No: 13

Tolga Çulha, Mustafa Özbalcı,

Ahmet Atum, Doğukan Şaki,

Harun Emre Yıldız

Istanbul Bilgi University

Hotel Automation

Aim of the Project

The aim of the project is to develop a web application that can be used by customers and employees of the hotel for faster, more consistent guest service. Using the application, guests will be able to request a spa appointment, order any product from the hotel stocks, apply for extra accommodation, and request an extension of their stay without speaking to the reception. Employees will be able to see requests from guests, check the product stocks and guide them accordingly. Application will enable new guests to access hotel rooms more easily and make reservations directly at the hotel price.

Importance of the Project

MVP features: By designing an interface for customers to keep their dialogue with employees to a minimum and submit their requests online, they will be able to submit their requests such as menu orders, extra person requests, stay extension requests and spa requests online and follow the approval process of the request. The application includes an online reservation system. An interface has been designed for employees where they can view customers' requests and have options such as approving or rejecting them. They also have an interface where they can track the amount of products in stock. There is also an interface where customers can view and control bill information.

Roles & Timeline

Tolga Çulha - Project Manager

Mustafa Özbalcı - Full Stack Developer, Manuel Tester

Doğukan Şaki - Developer

Ahmet Atum - UI Designer-System Designing

Harun Emre Yıldız - Tester-System Designing

Database Designing	1 Week
Backend Development	3 Weeks
Frontend Development-UI Designing	2 Weeks
Testing and Debugging	1 Week

REQUIREMENT LIST

Priority	Requirements
4	Creating and updating customer records.
5	Entering information for customer profiles (name, surname, contact information).
4	Making, changing and canceling reservations.
2	Room status tracking (empty, occupied, cleaned, not cleaned).
2	Room type options and pricing.
3	Making reservations for date ranges.
2	Reservation options according to room type and features.
3	Invoice reports (daily, weekly, monthly).
1	Reservation statistics.

3	Room occupancy rates.
5	Customer requests which are related to rooms, spa-fitness center and extra things.

Construction of the project and details

The project contains a large amount of backend codes and the necessary definitions for all methods connected to the database have been added as comment lines to the java controller classes. To briefly generalize the methods in the project, getmapping methods are used to display the information in the database; postmapping methods to add new information to the database; putmapping and deletemapping methods were used to update existing data.

Challenges and Solutions

While integrating HTML codes with the backend, the colors and photos in the background of the interface cannot be displayed. As a solution, making changes to the project file was followed.

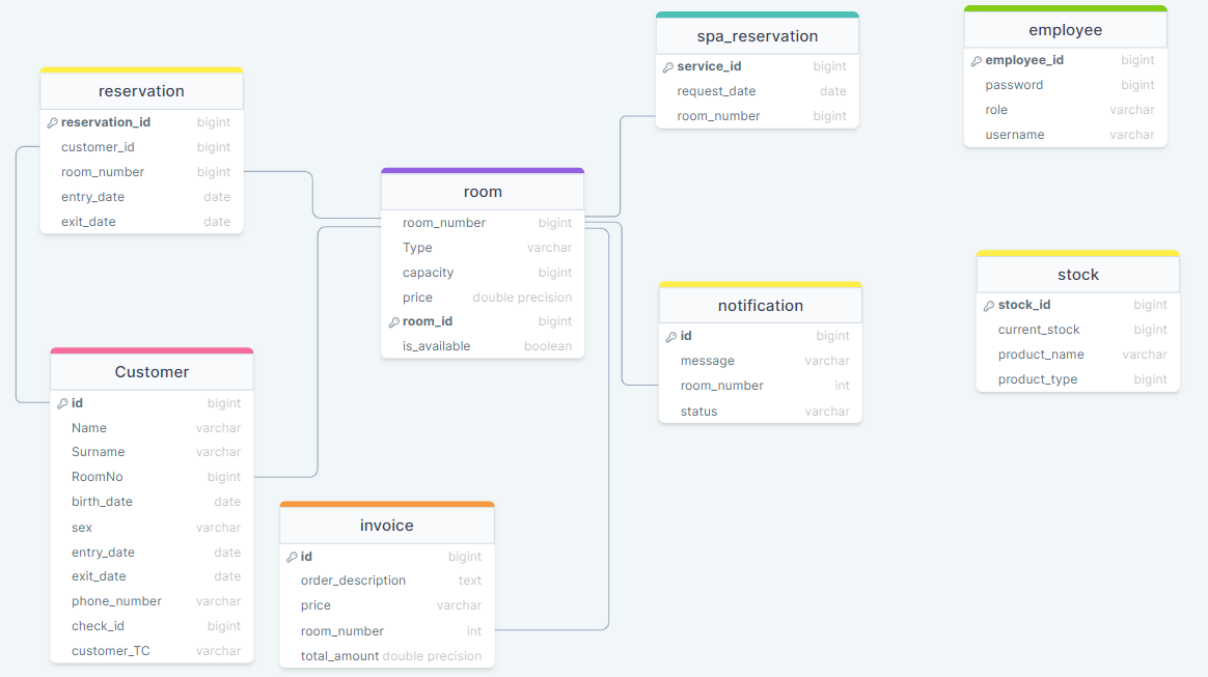
Since customer information is stored as session storage in the project, when a customer whose room number has not yet been assigned enters the system, we encountered a problem that records the room number as 0. As a solution, you need to use the incognito tab of the browser or clear cookies and log in again.

Future works

The project currently includes the existing stock system, and in the future, a system will be prepared to send orders directly to food and beverage companies via e-mail when the number of stocks falls below a certain amount.

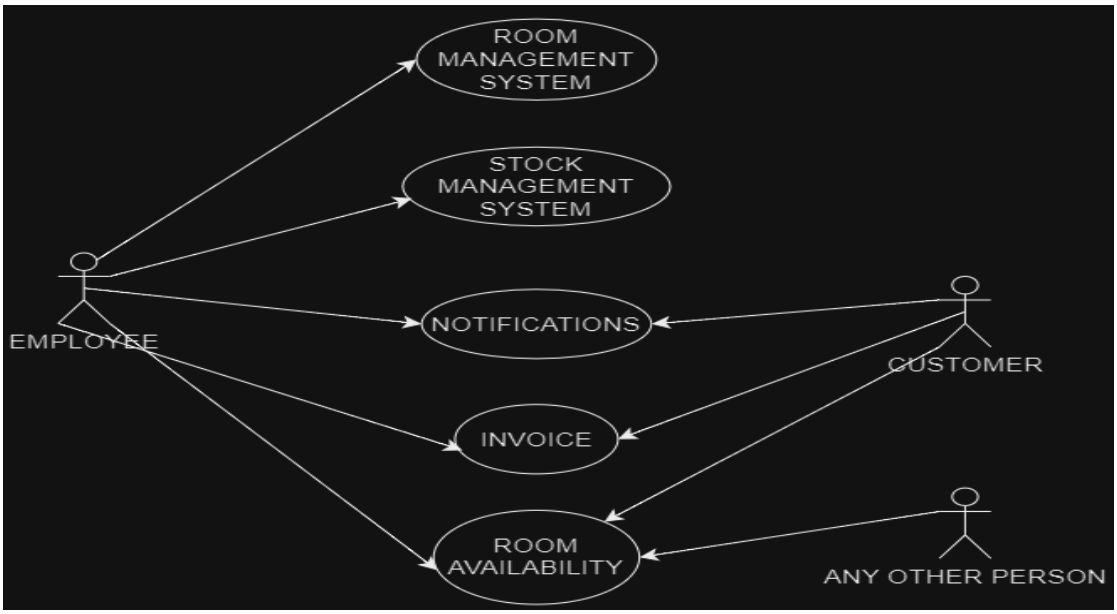
FULL DESIGN

Database Design:



Picture 1:Database Design image, shows the connections of database.

Hotel Project Use Case Diagram:

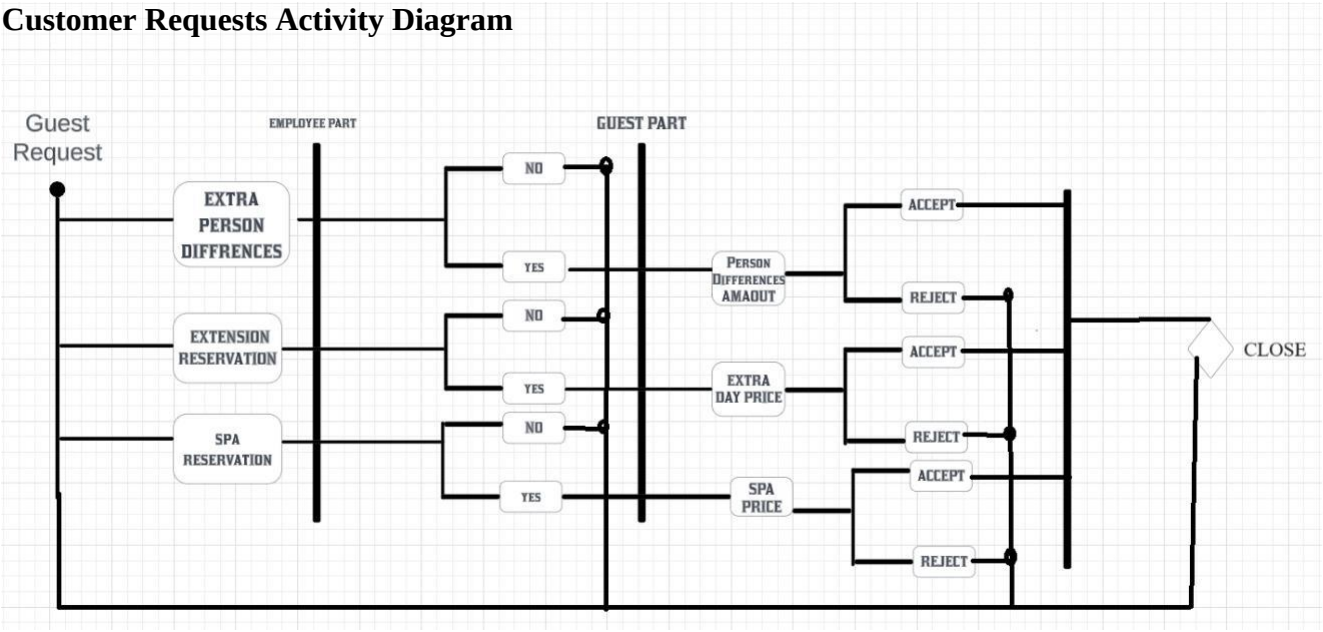


Picture 2:ANY OTHER PERSON means the person who tries to make a reservation through the hotel's online reservation system, so these people can only view room availability.

CUSTOMER's can see notification part because they will be able to add all kinds of requests and orders to the room from this system and they will be able to do many things, including accommodation extension requests, through this page. In addition, these people will be able to view the availability of the spa service at the hotel and make an appointment as they wish. Finally, by looking at the room status, these people will be able to request a room change from the room management request according to the information they view here.

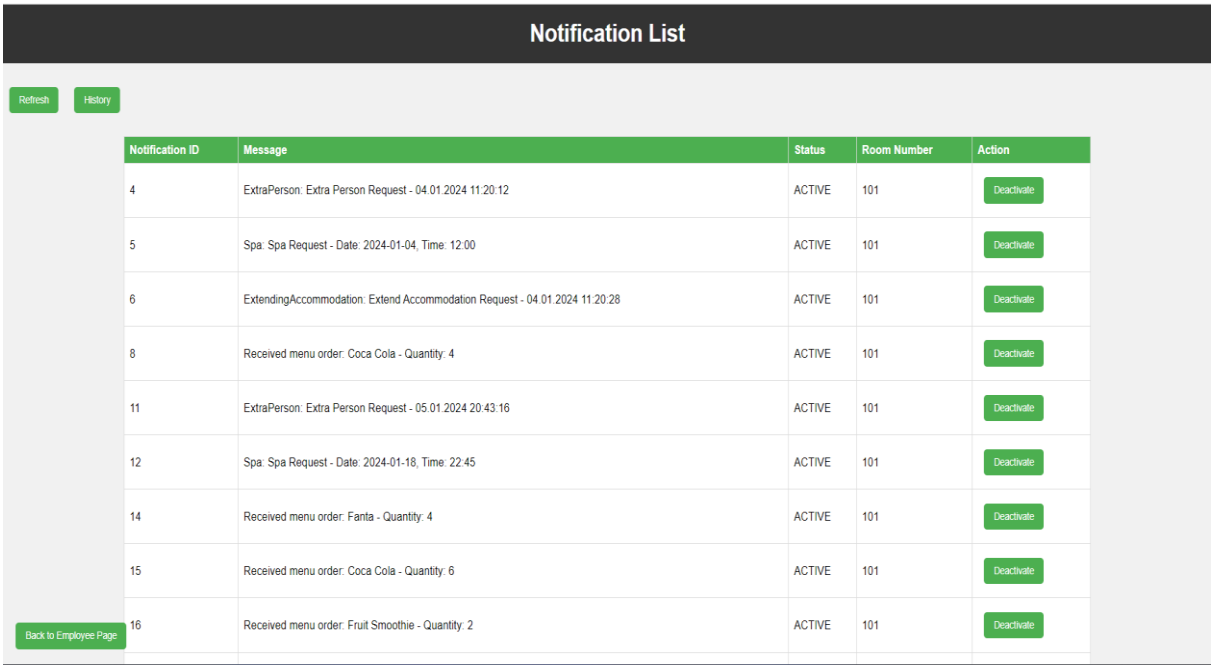
EMPLOYEE's will have access to all pages. They will be able to control the rooms and current guests from the room management system, follow the notifications in the Notification system, respond to incoming requests such as approving or rejecting them, and make the necessary checks according to spa availability.

Customer Requests Activity Diagram



Picture 3: In the **activity diagram**, 3 guest requests are sent to the employee side. These 3 requests will be answered by the employee by looking at the room management system and spa availability. If the answer is yes, an extra fee will be sent to the guest and the guest will be able to finalize his request according to this amount

Employee Notification :



Picture 4: Employee Notificaiton Picture, there is a system where employees can view the room number and the content of the request when customers enter orders or make any requests.

Notification Java

```
//TO FIND THE STATUS : "ACTIVE" NOTIFICATIONS
Mustafa Ozbalci
@GetMapping
public ResponseEntity<List<Notification>> getActiveNotifications() {
    List<Notification> activeNotifications = notificationRepository.findByStatus("ACTIVE");
    return new ResponseEntity<>(activeNotifications, HttpStatus.OK);
}
```

Picture 5:Notification Java image, There is a java code that returns the rows in the notification table as a list with the get mapping method.

Notification JS

```
2 usages   Mustafa Ozbalci
function refreshNotifications() {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/process-request", true);

    xhr.onreadystatechange = function () {
        if (xhr.readyState === 4) {
            if (xhr.status === 200) {
                var notifications = JSON.parse(xhr.responseText);
                updateNotificationTable(notifications);
            } else {
                alert("Error fetching notifications. Server status: " + xhr.status);
            }
        }
    };

    xhr.send();
}
```

Picture 6: Notification JS image, contains a method that allows obtaining the data in the notification table by sending a getmapping request to the /process-request extension.

UpdateTableJS

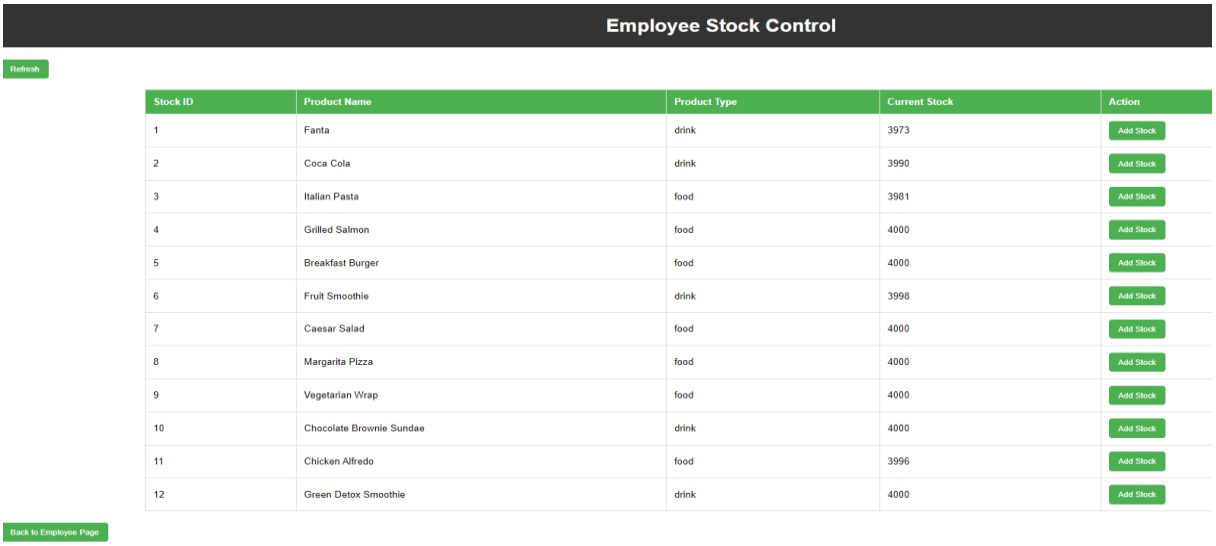
```
function updateNotificationTable(notifications) {
    var tableBody = document.getElementById("notificationTableBody");
    tableBody.innerHTML = "";

    notifications.forEach(function (notification) {
        var row = tableBody.insertRow();
        var cell1 = row.insertCell(0);
        var cell2 = row.insertCell(1);
        var cell3 = row.insertCell(2);
        var cell4 = row.insertCell(3);
        var cell5 = row.insertCell(4);

        cell1.innerHTML = notification.notificationId;
        cell2.innerHTML = notification.message;
        cell3.innerHTML = notification.status;
        cell4.innerHTML = notification.roomNumber;
        if (notification.status === "ACTIVE") {
            var button = document.createElement("button");
            button.innerHTML = "Deactivate";
            button.onclick = function () {
                deactivateNotification(notification.notificationId, row);
            };
            cell5.appendChild(button);
        } else {
            cell5.innerHTML = "N/A";
        }
    });
```

Picture 7: UpdateTableJS image, creates a table with the data previously received as response text and allows it to be entered line by line.

Employee Stock



Stock ID	Product Name	Product Type	Current Stock	Action
1	Fanta	drink	3973	Add Stock
2	Coca Cola	drink	3990	Add Stock
3	Italian Pasta	food	3981	Add Stock
4	Grilled Salmon	food	4000	Add Stock
5	Breakfast Burger	food	4000	Add Stock
6	Fruit Smoothie	drink	3998	Add Stock
7	Caesar Salad	food	4000	Add Stock
8	Margarita Pizza	food	4000	Add Stock
9	Vegetarian Wrap	food	4000	Add Stock
10	Chocolate Brownie Sundae	drink	4000	Add Stock
11	Chicken Alfredo	food	3996	Add Stock
12	Green Detox Smoothie	drink	4000	Add Stock

Picture 8: Employee Stock Image, there is a system where employees can view the stock status.

Stock Java

```java
//TO LIST CURRENT STOCKS
 Mustafa Ozbalci
@GetMapping("/allStock")
public ResponseEntity<List<Stock>> getAllStock() {
    List<Stock> allStock = stockRepository.findAll();
    return ResponseEntity.ok(allStock);
}
```

Picture 9: Stock Java Image, there is a java code that returns the rows in the Stock table as a list with the get mapping method.

StockJS

```javascript
function fetchStockData() {
    fetch('http://localhost:8080/api/stock/allStock')
        .then(response => response.json())
        .then(data => {
            const stockTableBody = document.getElementById('stockTableBody');
            stockTableBody.innerHTML = '';

            // Loop through the stock data and create table rows
            data.forEach(stock => {
                const row = document.createElement('tr');
                row.innerHTML = `
                    <td>${stock.stockId}</td>
                    <td>${stock.productName}</td>
                    <td>${stock.productType}</td>
                    <td>${stock.currentStock}</td>
                    <td><button onclick="openAddStockModal('${stock.productName}')">Add Stock</button></td>
                `;
                stockTableBody.appendChild(row);
            });
        })
        .catch(error => {
            console.error('Error fetching stock data:', error);
        });
}
```

Picture 10: StockJS Image, displays the information in json format obtained by sending a request to http://localhost:8080/api/stock/allStock address in a table.

Customer Page

Picture 11: Customer Page, customers can view their information on the left and invoice information on the right.

Invoice Java

```java
//TO SHOW INVOICE HISTORY BY ROOM NUMBER FILTER
 Mustafa Ozbalci
@GetMapping("/getByRoomNumber/{roomNumber}")
public ResponseEntity<Invoice> getInvoiceByRoomNumber(@PathVariable int roomNumber) {
    Optional<Invoice> optionalInvoice = invoiceRepository.findByRoomNumber(roomNumber);
    if (optionalInvoice.isPresent()) {
        Invoice invoice = optionalInvoice.get();
        return new ResponseEntity<>(invoice, HttpStatus.OK);
    } else {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}
```

Picture 12: Invoice Java, It calls a getmapping method by filtering the specified address by room number and gives other information about the matching row.

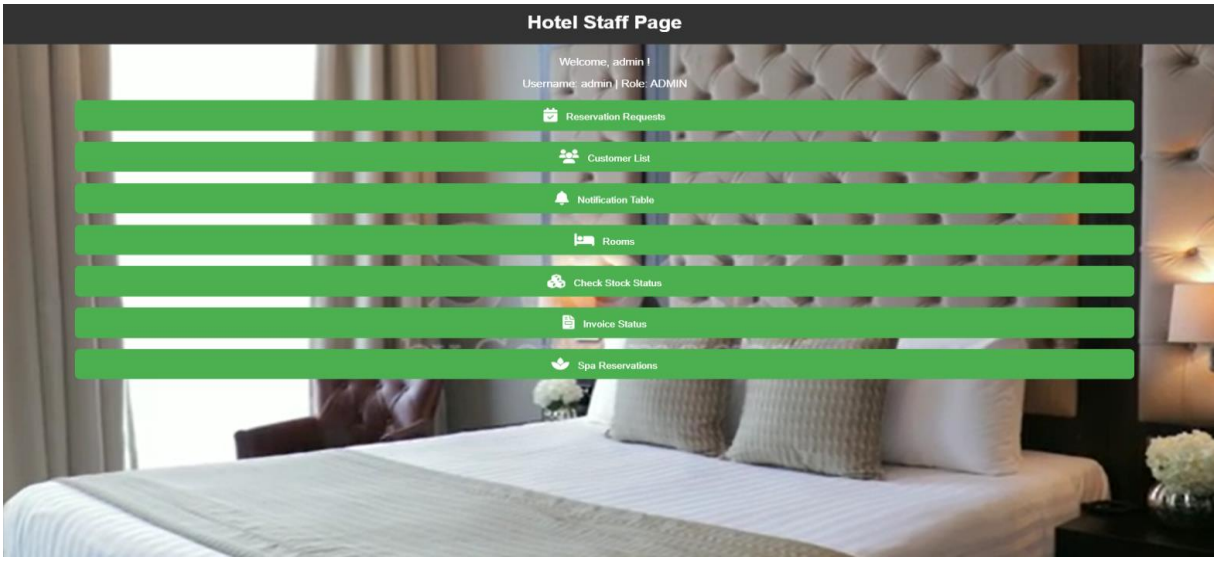
InvoceJS

```javascript
function updateInvoiceContent() {
    var roomNumber = sessionStorage.getItem('roomNumber');
    var apiUrl = "/invoice-status/getByRoomNumber/" + roomNumber;

    fetch(apiUrl)
        .then(response => {
            if (!response.ok) {
                throw new Error("Invoice not found for room number: " + roomNumber);
            }
            return response.json();
        })
        .then(invoice => {
            var totalAmount = invoice.totalAmount;
            var totalAmountElement = document.getElementById('totalAmount');
            totalAmountElement.innerText = totalAmount + " $";
        })
        .catch(error => {
            console.error("Error fetching invoice data:", error);
        });
}
// Refresh Invoice Content
document.addEventListener("DOMContentLoaded", function () {
    updateInvoiceContent();
});
```

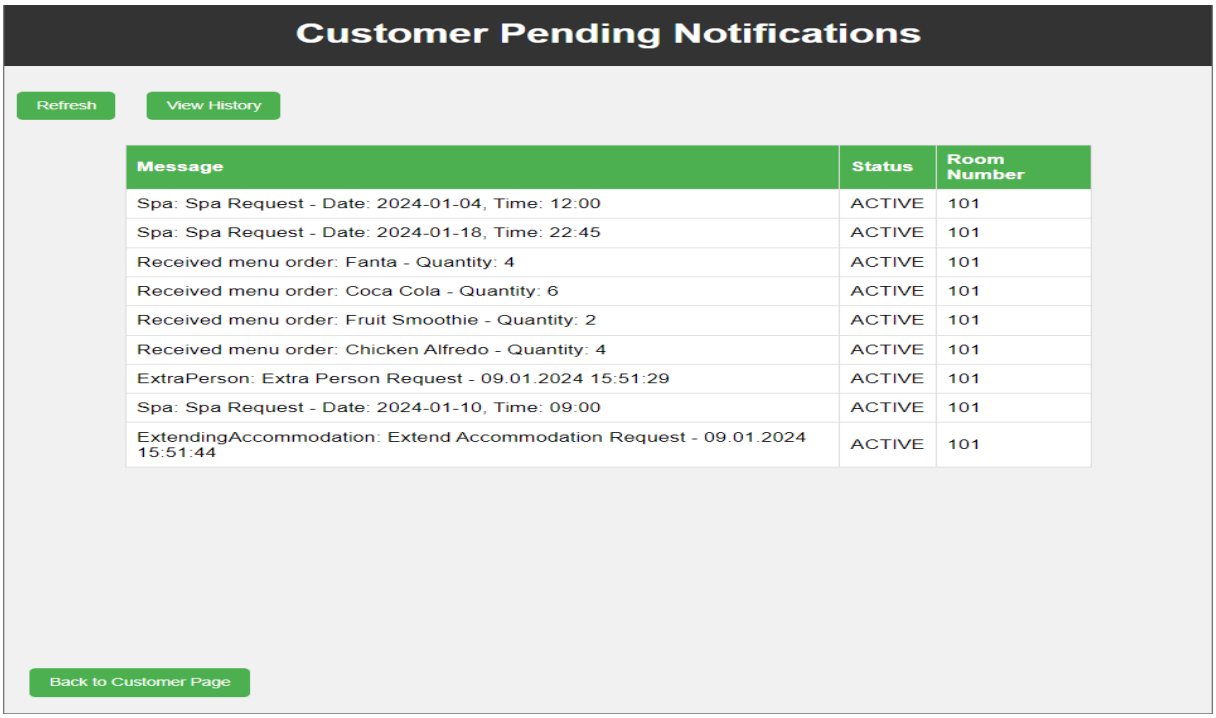
Picture 13: InvoiceJS, contains a js method that sends a request to /invoce-status/getByRoomNumber/{roomNumber} with the room number information obtained through sessionStorage and updates and displays the total amount value according to the result.

Employee Page



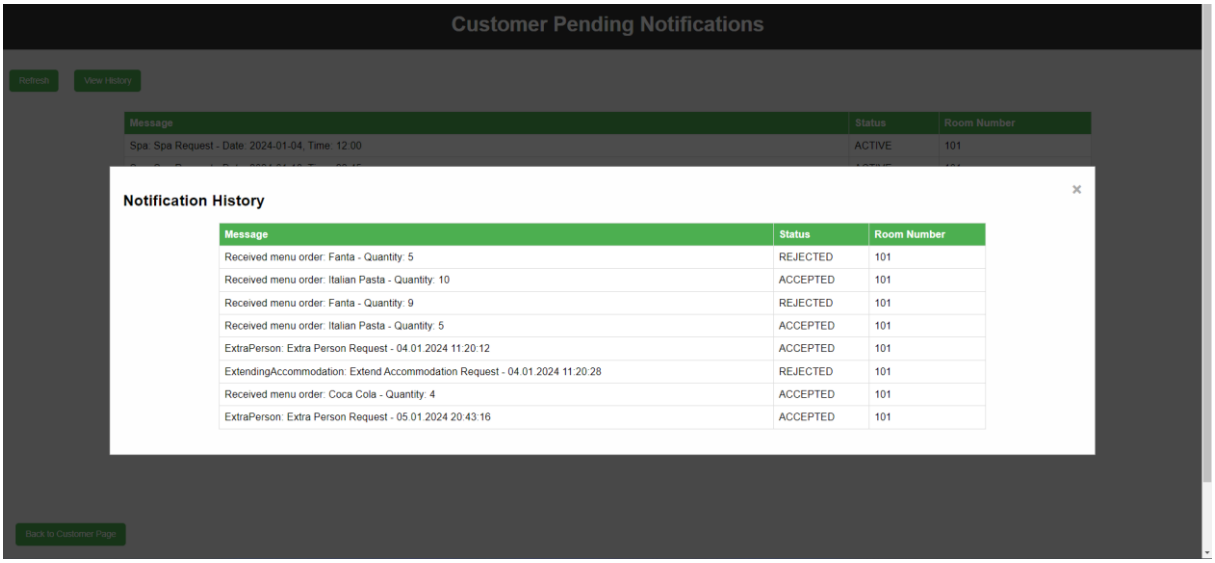
Picture 14: Employee Page, It is the home page where employees can view reservation requests, room status, current customer list, notifications, stock information, billing information and spa reservations.

Customer Notification Image



Picture 15: Customer notification image, there is a system that displays the system where customers track their active orders.

Customer Notification History Image



Picture 16: Customer Notification History Image, there is a system that displays the system where customers track their accepted or rejected orders.

Test Cases and Expected System Responses Table

Test Case Objective	Test Case Description	Test Case Expected Result
Validate the transition from log-in to account information	Enter login information and press login button	To be redirected to guests to customer page and the employee to employee page
Validate the new customer request arrive to employee	Customers create a new person request for their room, the employee receives the necessary information	Employee answers yes or no based on room availability
To confirm that the order system is working correctly when customers enter orders.	Checking stock status when customers place orders.	Depending on whether the entered order is in stock, forwarding the order otherwise sending a warning message to the employee.

Guest Check-In	Attempt to check in a guest with valid information	Successful check-in message displayed
Incomplete Check-In Information	Attempt to check in a guest with missing details	Error message prompts user for missing information
Room Availability Check - Fully Occupied Room	Verify room availability for fully occupied room	Alert indicates the room is fully occupied
Room Service Request	Place a room service request	Notification to staff for the requested service
Generate Daily Occupancy Report	Request a report on the daily occupancy	Report displays the number of occupied rooms per day
Room Availability Check - Available Room	Verify room availability for specified dates	Confirmation that the room is available
Guest Check-Out	Attempt to check out a guest	Successful check-out message displayed

References

1. Hotel Automation: Tools & Tips for Better Operations Performance (2023). Retrieved from https://operto.com/hotel-automation
2. Spring Boot : Tools&Applications for Creating Web Development Projects. Received from https://spring.io/projects/spring-boot
3. Java: Programming Language Retrieved from https://www.java.com/tr/
4. PostgreSQL : Database Application Retrieved from https://www.postgresql.org/
5. HTML : Language to Create User Interface Retrieved from https://www.w3schools.com/html/
6. Tons of Infromation Gathered from https://www.w3schools.com/