Project Report

Resort Management System.

Group Members:

- Aun Ali (20K-0286)
- Ammar Amin (20K-0285)
- Muhammad Anas (20K-0179)

Section: 6F

Course: Software Engineering

Course Instructor: Ms. Noureen Fatima

Contents

Overview

Purpose:

"Resort Management System" is a web application, that is designed to conduct various operations for both client (and potential client) and admin. The system provides the user an easy to use interface, which will allow an increase in productivity and efficiency.

Intended Audience:

This document is intended for course instructors and professionals that belongs to the field of computer science or tourism.

Tools and Language:

Resort Management System is a three-tier system including a front-end, back-end and a database.

- **Front End:**
    1. **HTML 5:** HTML stands for Hyper Text Markup Language, it is a standard markup language for creating web pages and defines a web page structure and produce web pages that include text, graphics and hyperlinks.
    2. **CSS:** CSS stands for Cascading Style Sheets, it describes how HTML elements are to be displayed on screen.
    3. **JavaScript:** JavaScript is one of the most popular programming language of the web. It is a lightweight, interpreted programming language, designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.
    4. **Bootstrap:** Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

- **Back End:**
    1. **Node.Js:** Node.js is a back-end JavaScript runtime environment. Node.js runs on the V8 JavaScript Engine and executes JavaScript code outside a web browser. It is designed to build scalable network applications.

- **Database:**
    1. **SupaBase:** Supabase is an open source Firebase alternative providing all the backend features that are required to build a product. It is a platform designed to help devs streamline the creation of modern apps. It covers, database, authentication and file storage.

Project Interfaces:

- **Guest Interface:**

Guest logs in on the website and carry various operations such as booking services, checking in/out, and ordering food.

- **Admin Interface:**

Managers have access to a separate interface when logging into their accounts, since some special tasks are only available to managers.

Operating Platforms and Applications Used

- **Hardware:**

1. 4GB RAM

2. Intel core "i5 10^{th} generation"

3. 5MB disk space.

- **Software:**
    1. Windows 10 operating system
    2. Visual Studio Code.
    3. Google Chrome (web browser).

Non-Functional Requirements

Performance Requirements

- System is designed to return results very quickly
- Higher workload will not have a negative impact on the performance

Portability and Compatibility Requirements

- System can work on any popular graphical or open source operating system like windows and Ubuntu
- System can work on any browser that support the use of tools and language mentioned above.

Usability Requirements

- It contains a relatively easy-to-use graphical user interface, which only requires a basic knowledge of English.

Security Requirements

- Supabase periodically takes data backup to secure it from accidental loss.
- Every log in session has a time out if there is not activity during a specific period.

Localization Requirements

- Date format used is DD/MM/YYYY

Application Features

Sign Up

User can create account by entering a valid email and setting a password.

Log In & Log Out

User can login using a registered e-mail and a valid password. System will redirect user to guest interface or admin interface according to their email domain.

After user click on log out button, they will be redirected to log in page and all cache will be cleared in the process.

Booking System

After logging in user can request to book following:

- Room
- Tour
- Vehicle
- Spa/ Massage appointment
- Pool slot
- Gym Slot
- Game Slot

To submit a request user have to enter start and end date of reservation, time and duration.

Room Service

User can order

- Food item that are available on menu.
- Laundry Service

Stock Management

Admin can **check count** of available stock and if the count is less than a set limit, more stock will be **ordered**, moreover chef can also send message to admin if a certain stock product is finished so it can be ordered.

Employee Management

After clicking “Manage Employees” button, admin can do the following:

- Add new employee, by inserting id, name, email, salary and role.
- Edit employee details by entering employee id and then change the attribute that needs an update.
- Delete an employee by entering their id.

Customer Complaint

- Customer can register complaint by entering it in designated text box, which can be found on guest dashboard.
- Admin can view the complaint by clicking “Read customer Complaints”

Wallet

Pre-paid service for customers that they can avail during their stay. Amount deposited will be deducted, if customer avail a service that requires payment.

Reservation

- After booking request admin will check if the date/slot is already booked, if no then request will be accepted otherwise request will be denied.
- At customer request admin can also cancel reservations

Check Status

Customer can check status of their upcoming reservations and wallet amount.

Billing System

System will display bill after customer booking request is accepted. Payment can be done by deducting amount from wallet or adding credit.

Assumptions:

- To deposit amount in wallet, guest have to be physically present or they can do it by transferring money online.
- Amount of products and services will be set by management themselves.
- Complaints and issue will be resolved by Resort staff.

User Guidelines:

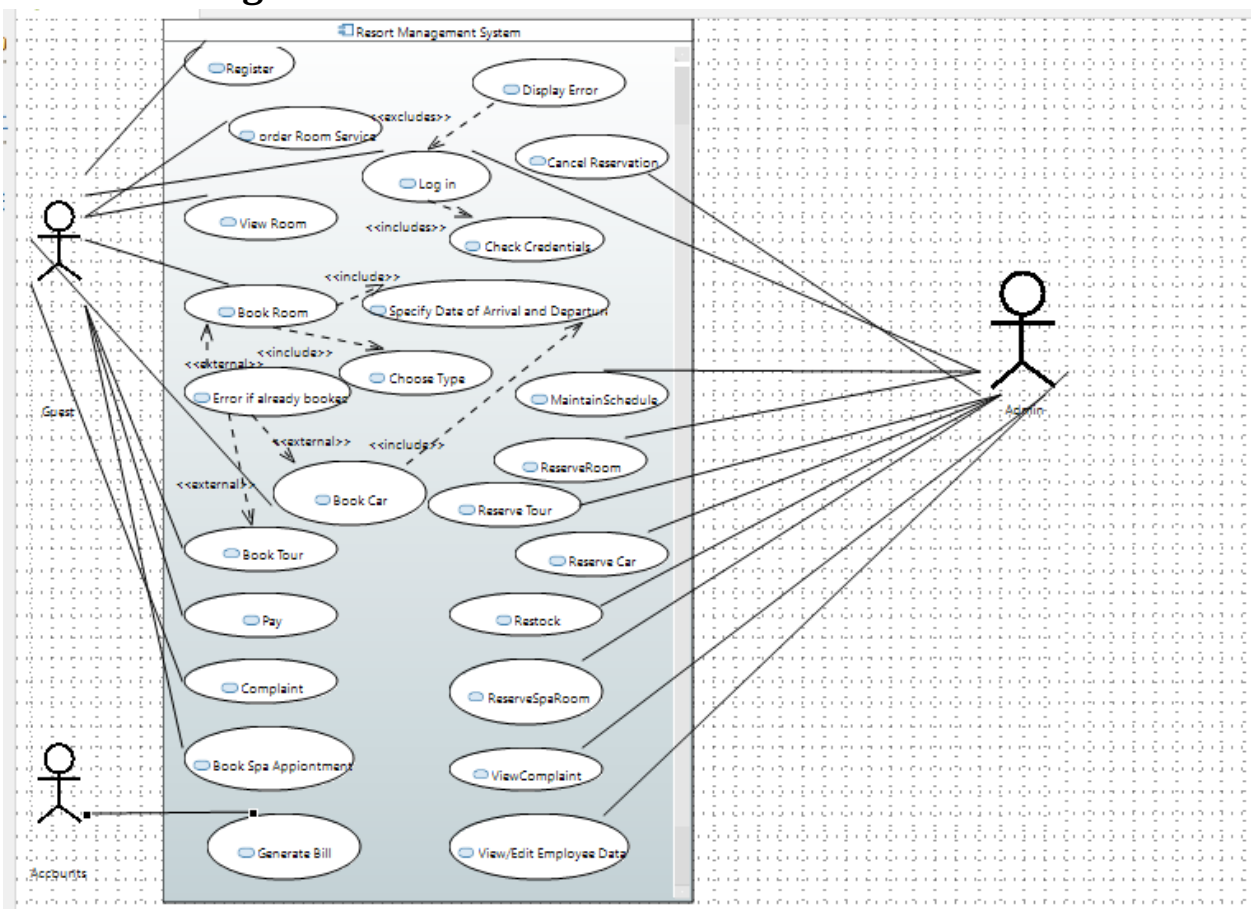
- User should know English Language.
- User should not share log in credentials, so any kind of malicious activity can be prevented.

UML Diagrams

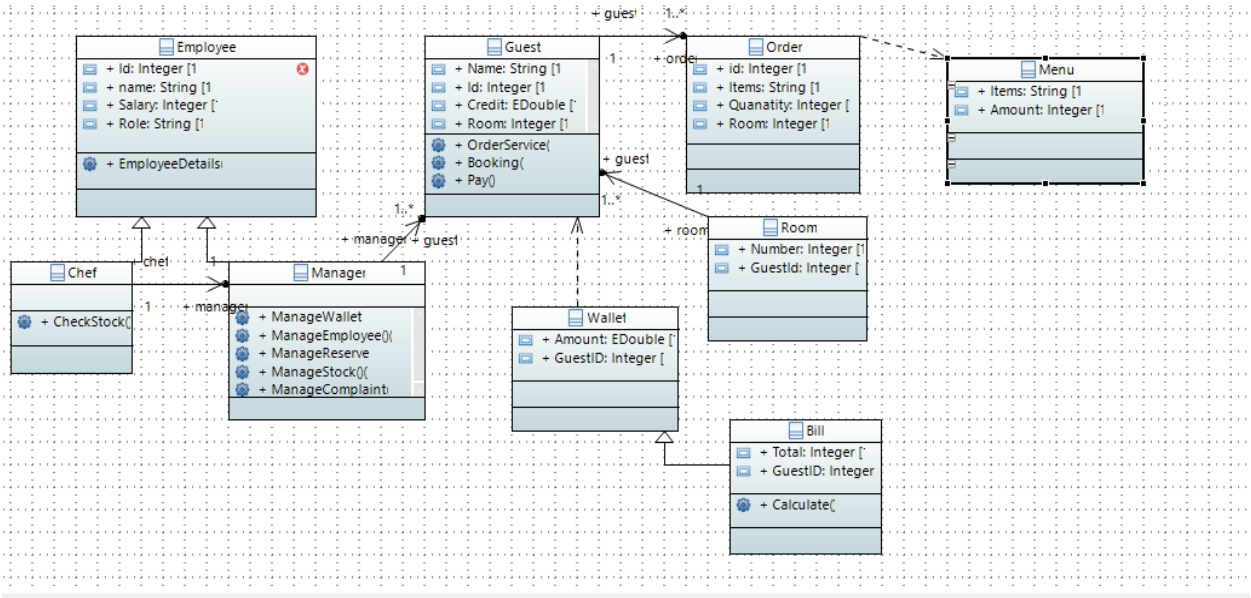
Following diagrams are presented below:

- Use Case Diagram
- Class Diagram
- Activity Diagrams
- Sequence Diagrams
- State Transition Diagram
- Component Diagram
- Timing Diagram
- Deployment Diagram

Use Case Diagram

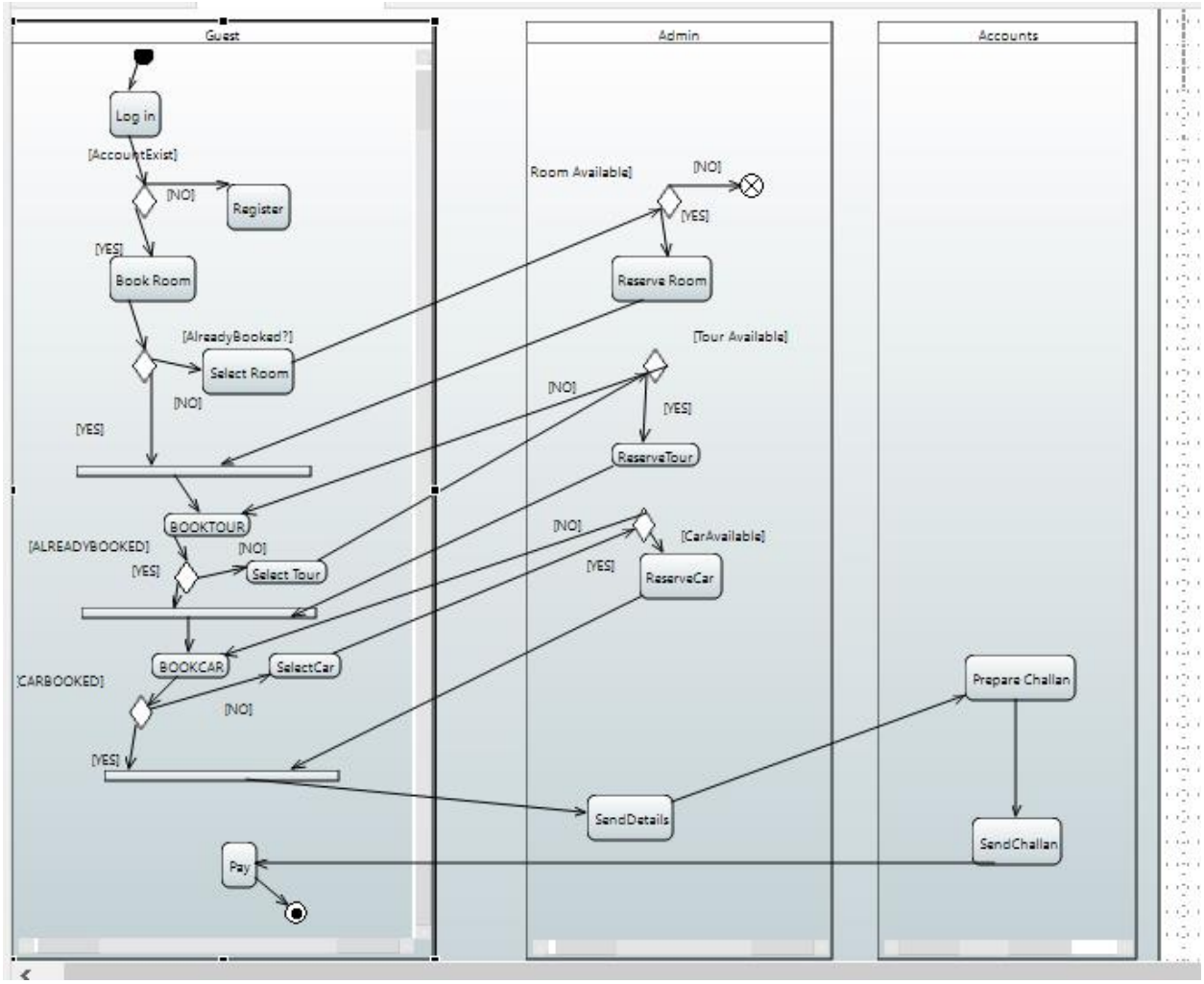


Class Diagram

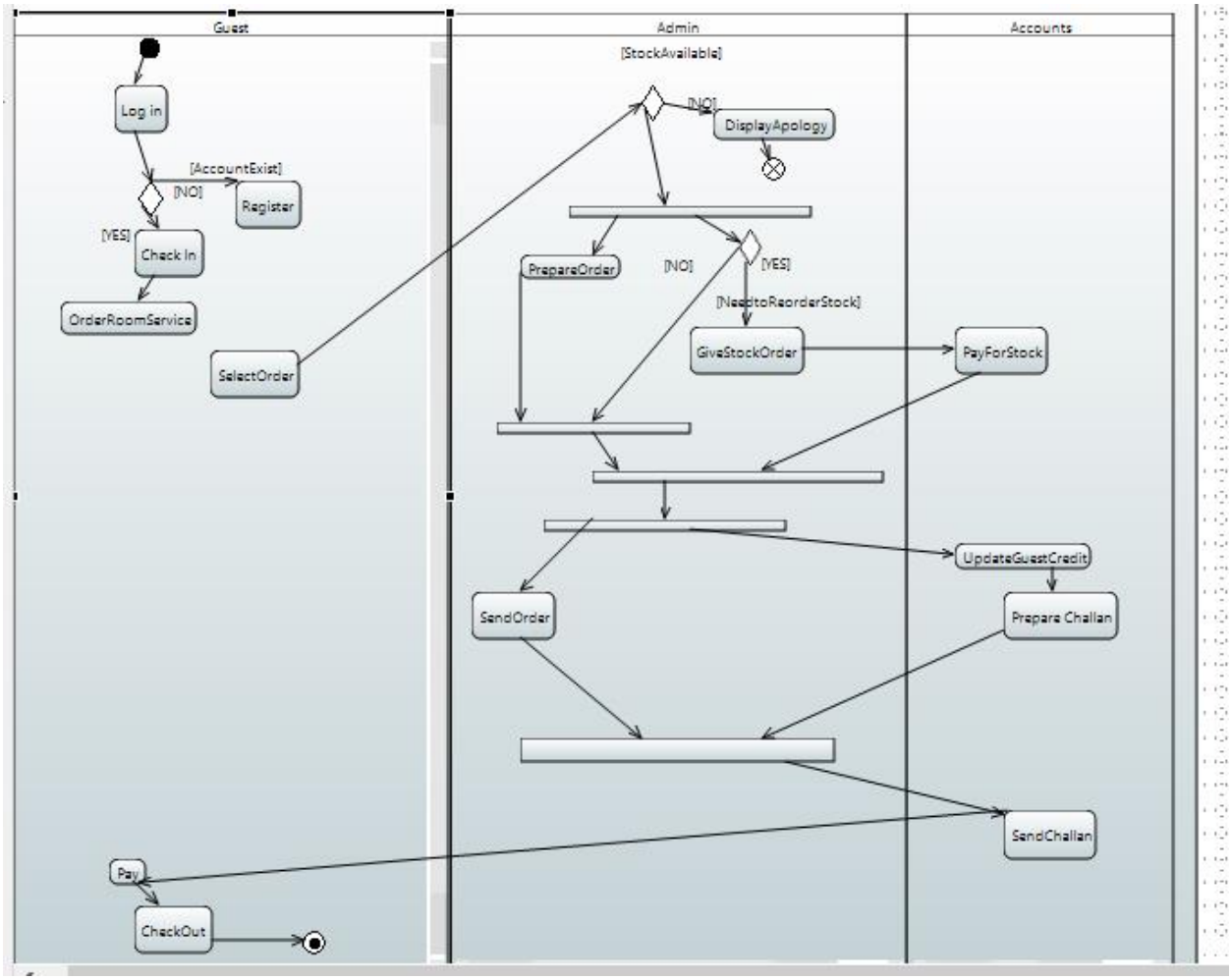


Activity Diagram.

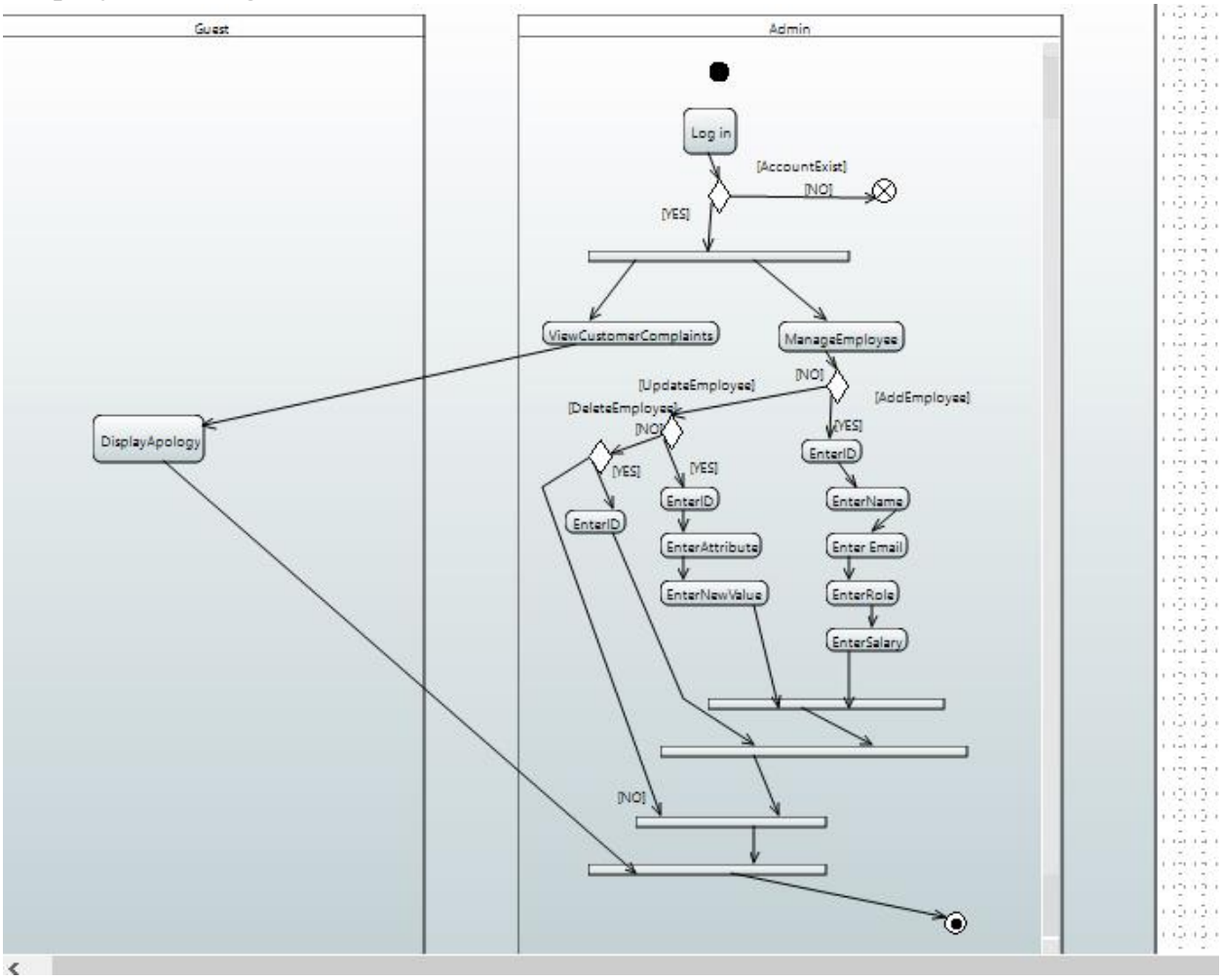
- Reservation Process:



- Room Service and Restocking

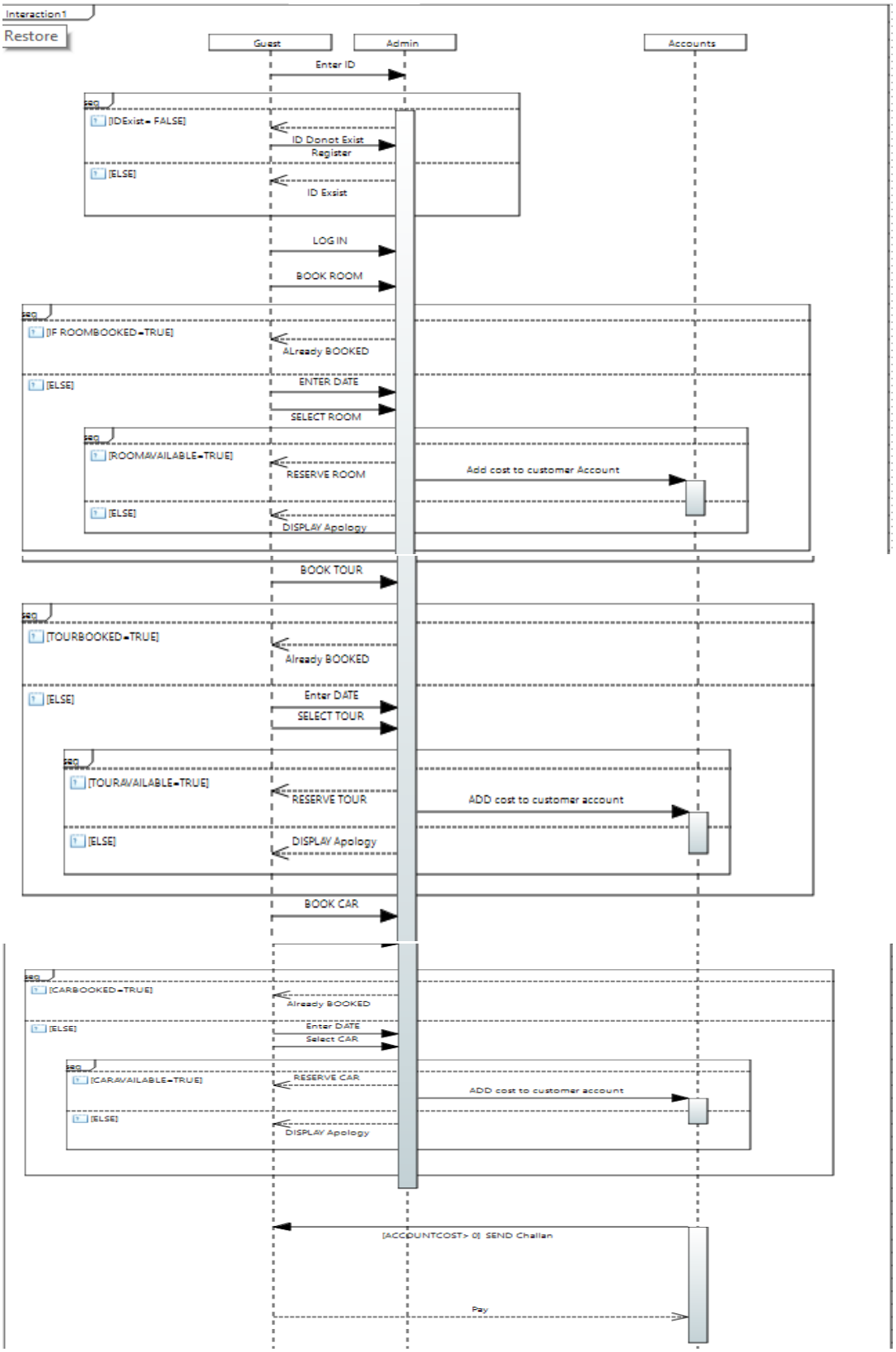


- Employee Management and Customer Service

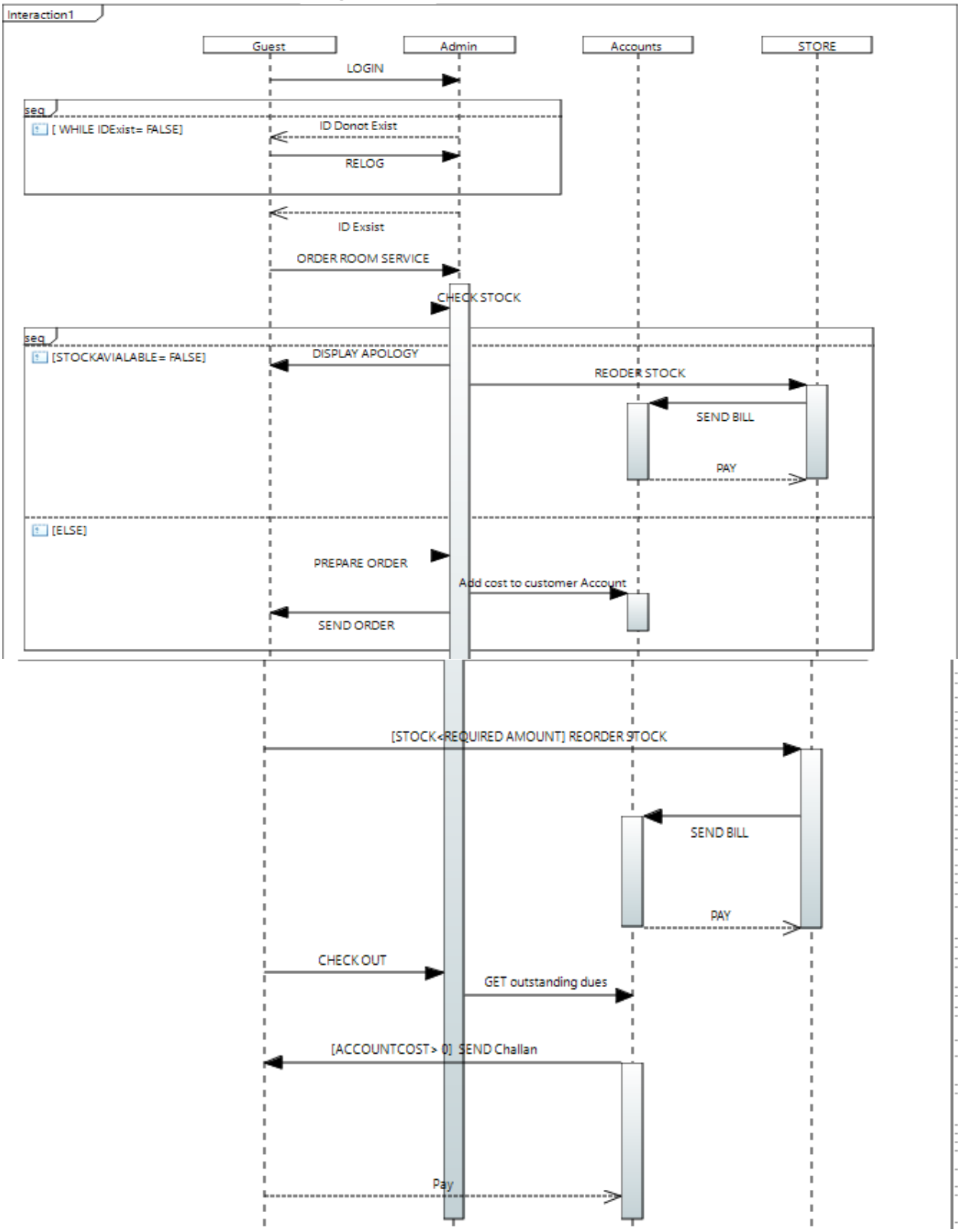


Sequence Diagram.

- Reservation Process

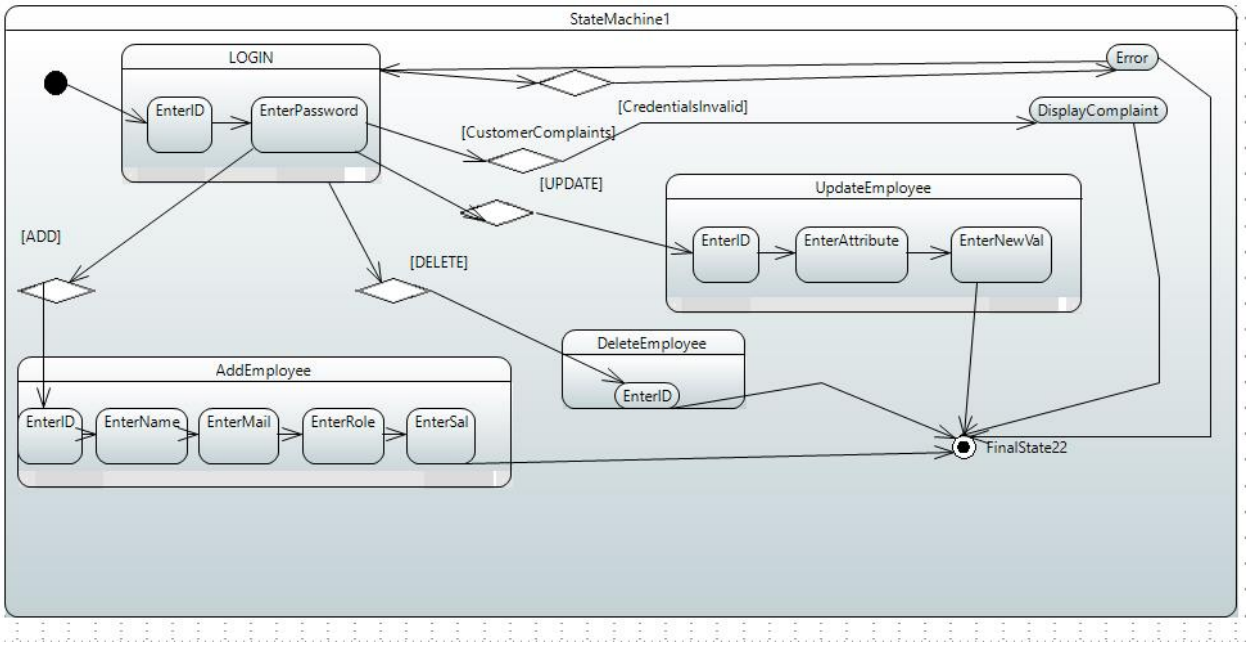


- Room Service and Restocking

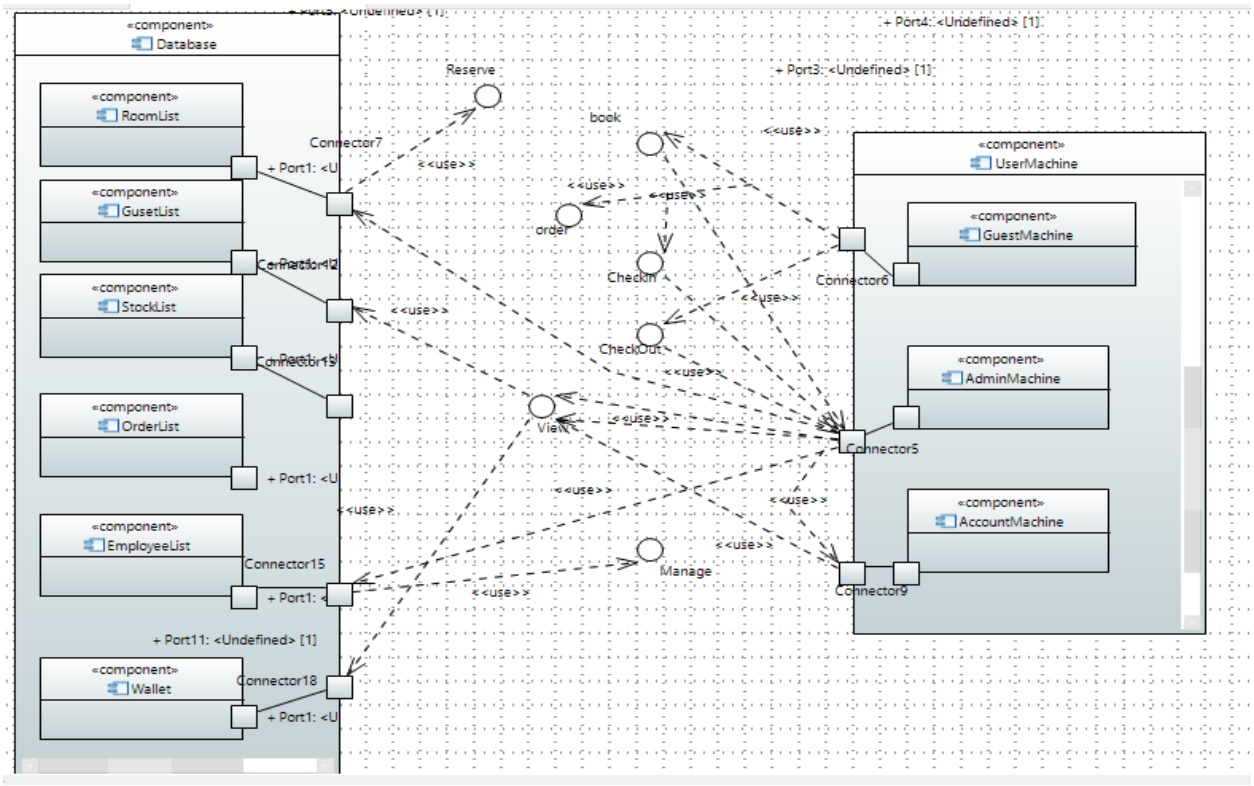


State Transition Diagram

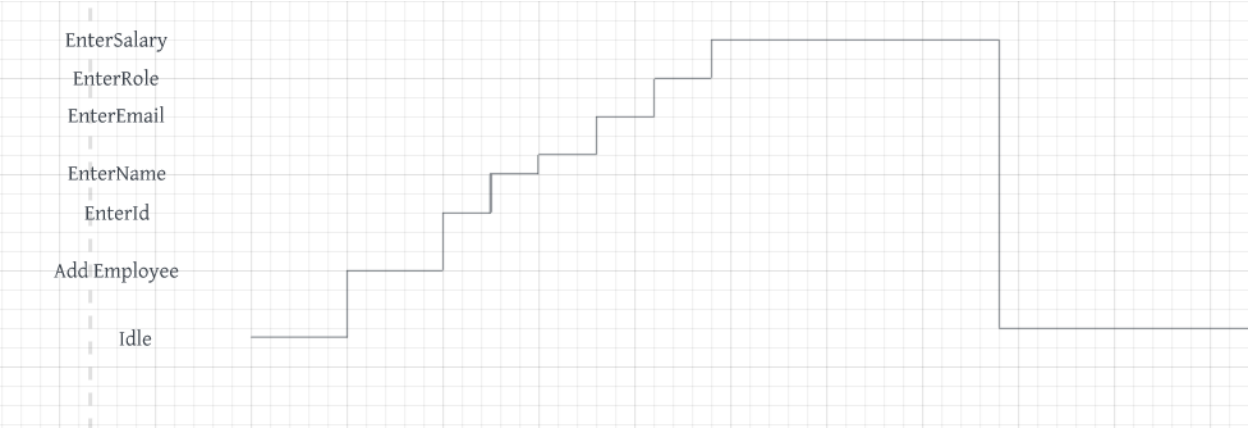
- Employee Management and Customer Service



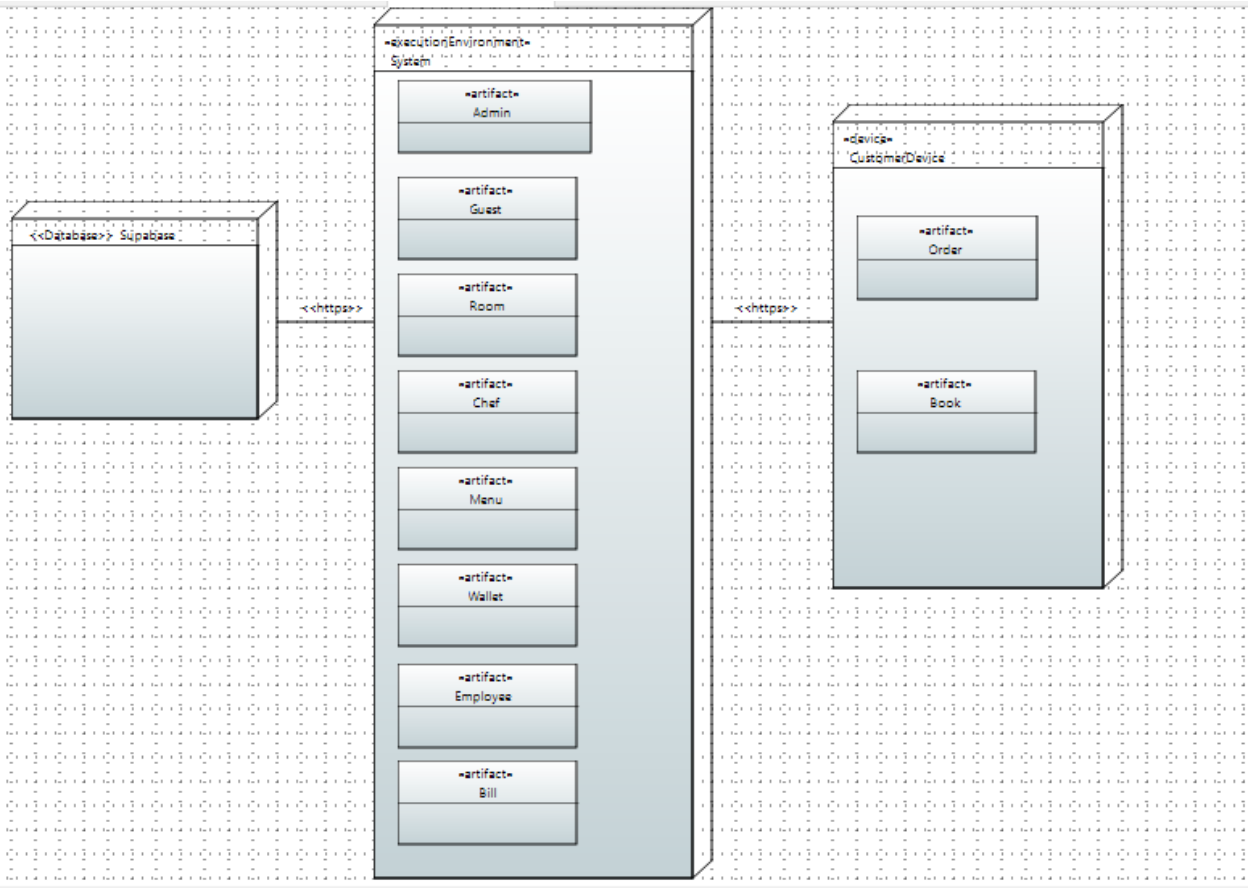
Component Diagram.



Timing Diagram



Deployment Diagram



Database Systems

- Structured Query Language (SQL) is a High Level Data manipulation language (DML) that specifies which data to retrieve not how to retrieve it. Also called Declarative Languages.
- Every Supabase project comes with a full **Postgres** database.
- Object-relational database system PostgreSQL is powerful and open source and uses and extends the SQL language with a number of features that allow it to safely store, process, and scale the most complex data sets.

Tables Created

- Customer

Id	Name	Email	Bill	Complaint	Complaint Status	Wallet

  - **Id:** Unique id assigned to guest (Primary key)
  - **Name:** Guest Name
  - **Email:** Guest email address
  - **Bill:** Guest bill amount
  - **Complaint:** Complaint submitted by guest
  - **Complaint_Status:** Check if there is any complaint registered by the guest.
  - **Wallet:** Amount in Guest Wallet

- Employee

Id	First Name	Email	Role	Salary

  - **Id:** Unique Id assign to employee. (Primary key)
  - **First_Name:** Employee first name
  - **Email:** Employee email address
  - **Role:** Employee Job
  - **Salary:** Employee Salary

- Manager

Id	First Name	Email	Role desc

  - **Id:** Unique Id assign to Manager. (Primary key)
  - **Fisrt_Name:** Manager first name
  - **Email:** Manager email address
  - **Role_Desc**

- Room

Room id	Cust_id

  - **Room_Id:** Room Number. (Primary key)
  - **Cust_id :** Id of guest who have reserved the room. (Foreign Key from Customer table)

- Massage

Massage id	Cust_id

  - **Massage_id:** Massage appointment unique id. (Primary Key).
  - **Cust_id:** Id of guest who have an appointment. (Foreign Key from Customer table).

- Service

Token

- Food_Stock

Item_Name	Qty

  - **Item_Name:** Stock item name. (Primary Key).
  - **Qty:** Stock item quantity.

- Gym

Gym_Id	Cust_id

  - **Gym_Id:** Gym room unique id. (Primary Key).
  - **Cust_id:** Id of guest who have reserved the gym. (Foreign Key from Customer table)

- Car

Car Id	Cust_id

  - **Car_Id:** Car registration number. (Primary Key).
  - **Cust_id:** Id of guest who have reserved the Car. (Foreign Key from Customer table)

- Spa

Spa Id	Cust_id

  - **Spa_Id:** Spa appointment unique id. (Primary Key).
  - **Cust_id:** Id of guest who have an appointment. (Foreign Key from Customer table)

- Game

Game_Id	Cust_id

  - **Game_Id:** Game room unique id. (Primary Key).
  - **Cust_id:** Id of guest who have reserved the game room. (Foreign Key from Customer table)

- Pool

Pool_Id	Cust_id

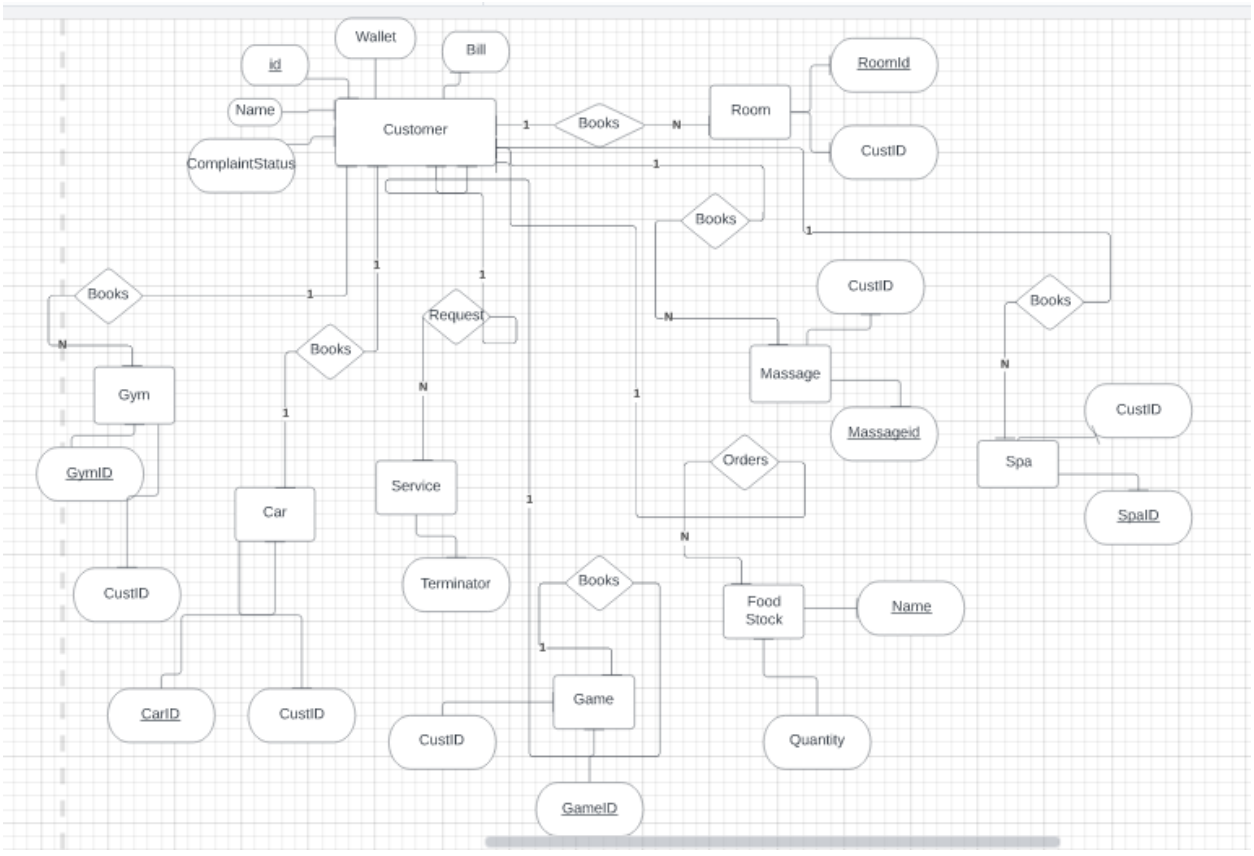
  - **Pool_Id:** Pool unique id. (Primary Key).
  - **Cust_id:** Id of guest who have reserved the pool. (Foreign Key from Customer table)

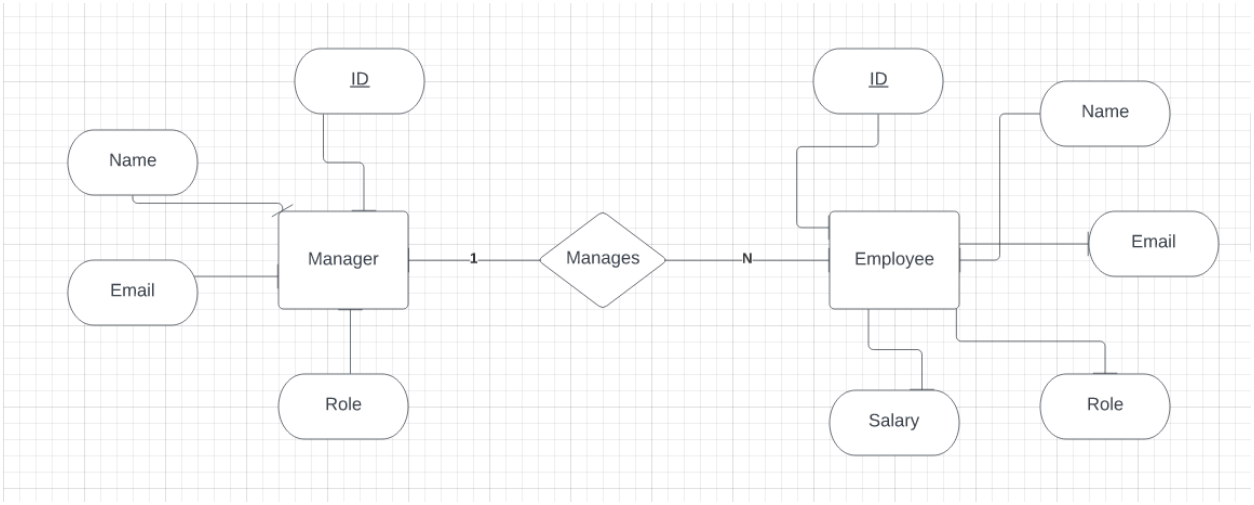
Schema Diagram

- The schema of a database defines how a relational database is logically configured.
- It is both a visual representation and a set of formulae known as integrity constraints that govern databases.
- These formulas are expressed in a data definition language, such as SQL.
- A database schema identifies how tables, views, stored procedures, and other entities in the database relate to one another.

ER Diagram

- Entity Relationship Diagrams (ERDs) illustrate how individuals, objects, or concepts relate to one another within a given system.
- Software engineers, business information systems professionals, educators, and researchers utilize ER diagrams to design or debug relational databases.


Wallet
Bill
id
Name
ComplaintStatus
Customer
Books
Room
RoomId
CustID
Books
Books
Request
Books
Gym
GymID
Car
CustID
CarID
CustID
Service
Terminator
Books
Game
CustID
GameID
CustID
Massage
MassageId
Orders
Food Stock
Name
Quantity
Books
Spa
CustID
SpaID


ID
Name
Email
Manager
Role
1
Manages
N
ID
Name
Employee
Email
Salary
Role

Implementation

- **Js code for Sign Up:**

```
const { data, error } = await _supabase.auth.signUp({
    email: email_1.value,
    password: password.value,
  });
```

- **Js code for Log in:**

```
const { data, error } = await _supabase.auth.signInWithPassword({
   email: loginEmail.value,
   password: loginPassword.value,
 });
```

- **Js code for updating Customer database table :**

```
 const givedata = async()=>{
   const { data, error } = await _supabase
   .from('customer')
   .insert([
     { id: id_1.value, Name: name_1.value, email: email_1.value },
   ]);
   if (error) {
     alert("Constraint Violation Detected, Record Not Created!");
     window.location.href = "/signup.html";
 console.log(error);
 } else {
 window.location.href = "/login.html";
 }}
```

- **Js code for Updating Bill:**

```
const upd_bill_deduct = async()=>{
```

```
curr_bill=curr_bill-500;
const { data, error } = await _supabase
.from('customer')
.update({ bill: curr_bill})
.eq('id', customer_id);
alert("Bill Updated!");
if (error) {
  alert("Not Updated, Try Again...");
  window.location.href = "/curr_status.html";
  console.log(error);
}}
```

- **Js code for Delete Data:**

```
const delete_data = async()=>{
    const { data, error } = await _supabase
    .from(entity)
    .update({ Cust_id: 0})
    .eq(entity.concat("_id"), selected);
    if (error) {
        alert("Unable to delete, Try Again");
        console.log(error);
    } else {
        upd_bill_deduct();
    }
}
```

- **Js code for Fetching Data:**

```
const upd_table_fetch = async()=>{
  const { data, error } = await _supabase
  .from(entity)
  .select(entity_id)
  .eq("Cust_id", 0)
  .limit(1);
  if (error) {
    console.log(error);
    alert(error);
  }
}
```

System Testing

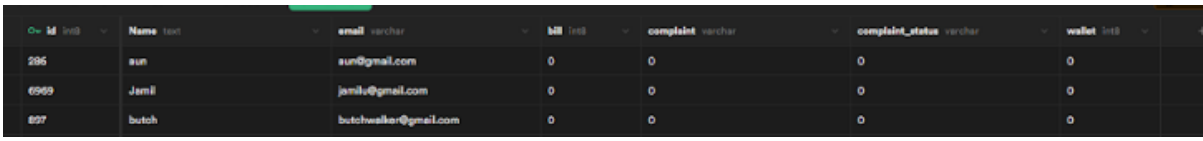
- Software testing helps identify whether developed software is correct, complete, secure, and of high quality.
- The purpose of this is to find errors in a program by running it.
- It is an evaluation of a system or its components to determine whether it meets specified requirements.
- Software testing aims to find faults in existing software as well as to improve the software's efficiency, accuracy, and usability.

Testing Example

Signing Up:

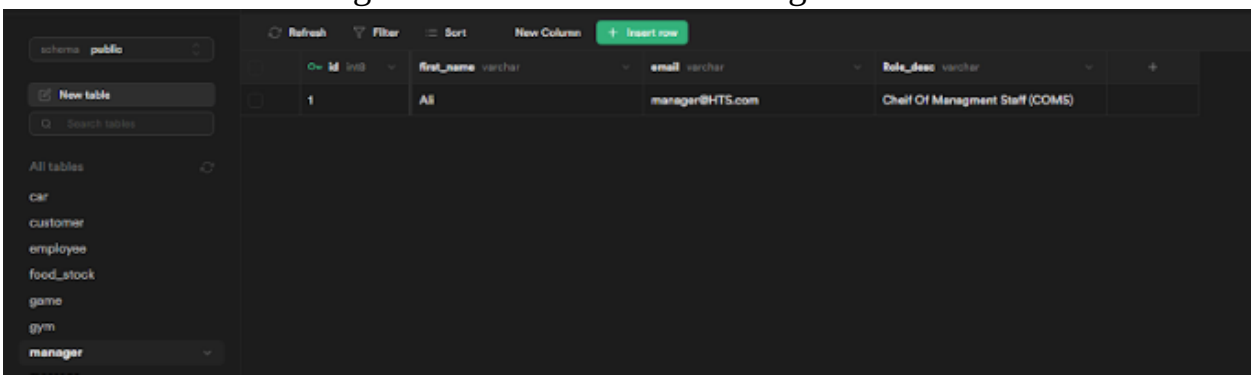
1. Create an account:

2. Check in Database if the account entered in the Customer Table:



Create Manager Account

Check if Manager Account entered in Manager Table



Testing Table

Case	Input	Output	Description
1	Username: johndoe, Password: 12345	Login Successful	Tests if a valid username and password combination allows the user to log in successfully
2	Username: janedoe, Password: abcde	Login Failed	Tests if an invalid username and password combination prevents the user from logging in
3	Press Book Button	Required Field Booked	Updates the database and reserves the slot for the customer
4	Press Book Button	Required Field Not Booked	All the Slots are at full capacity
5	Press Pay Button	Payment Paid	Amount is deducted from the wallet
6	Press Pay Button	Payment Not Paid	Insufficient Funds in the wallet
7	Submit Complaint	Complaint Submitted	Updates the database along with the manager
8	Press Status Button	Shows Current Status	Fetch data from database and displays
9	Press Logout Button	Take user to Login Page	Current active user is logged out
10	Press Update Food Stock Button and input wrong code	Shows Stock for updating	Displays List of Stock and updates the database accordingly
11	Press Update Food Stock Button and input wrong code	Shows Stock for updating but gives error	Displays List of Stock and but does not update the database
12	Press Manage Employees Button and enter correct option	Shows list of options to choose from	Updates the database if option is right
13	Press Manage Employees Button but enter incorrect option	Shows list of options to choose from but later gives error	Error handling prevents accidental changes in database
14	Press Read Complaints button	Shows customer complaints	Fetch data from database

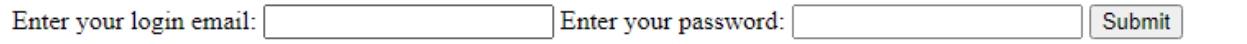
15	Press Update Complaints button and input customer id and details accordingly	Displays “Successfully updated!” message	Updates database for the complaints section
16	Press Update Complaints button and input incorrect customer id and details	Displays “Not Updated!” message	Does not Update database for the complaints section
17	Press update Customer Wallet and select right options	Displays “Successfully updated!” message	Updates database for the wallet section
18	Press update Customer Wallet and select wrong options	Displays “Not Updated!” message	Does not Update database for the wallet section
19	Press Delete Customer Account Button and select right options	Displays “Successfully updated!” message	Updates database for the customer section
20	Press Delete Customer Account Button and select incorrect options	Displays “Not Updated!” message	Does not Update database for the customer section
21	Press Order Food from Inventory button and input valid amount	Displays “Successfully updated!” message	Updates database for the quantity of food section
22	Press Order Food from Inventory button and input invalid amount	Displays “Not Updated!” message	Does not update database for the quantity of food section
23	Enter Incorrect manager credentials	Logins fails	Does not allow user to access manager account
24	Enter Incorrect chef credentials	Logins fails	Does not allow user to access chef account

Project Snapshots

- Sign In page

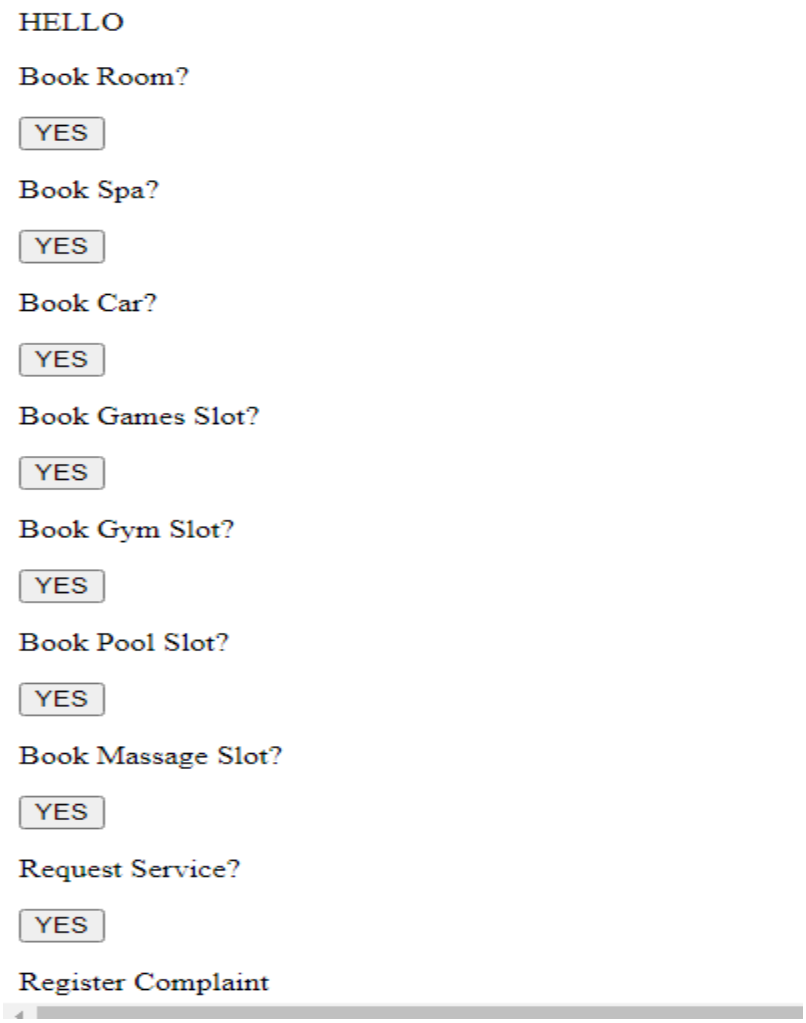
login account

- Log in page

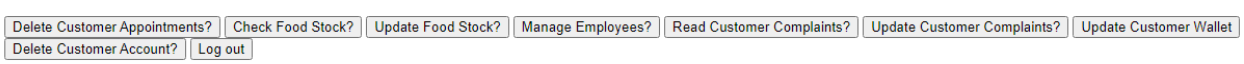


signup account

- Customer Dashboard



- Manager Dashboard



Gantt Chart

Task Names	Start Date	End Date	Duration (Border Days inclusive)	Dependencies	Milestones
Project Planning	Nov 1	Nov 1	0 days	None	Project Planning Complete
Website Design	Nov 1	Nov 3	3 days	Project Planning	Website Design Complete
Customer Registration & Login Function	Nov 4	Nov 5	2 days	Website Design	Customer Registration & Login Function Complete
Room Booking Function	Nov 6	Nov 7	2 days	Customer Registration & Login Function	Room Booking Function Complete
Spa Booking Function	Nov 6	Nov 7	2 days	Customer Registration & Login Function	Spa Booking Function Complete
Games Booking Function	Nov 6	Nov 7	2 days	Customer Registration & Login Function	Games Booking Function Complete
Service Booking Function	Nov 6	Nov 7	2 days	Customer Registration & Login Function	Service Booking Function Complete
Gym Booking	Nov 6	Nov 7	2 days	Customer Registration & Login Function	Gym Booking Function Complete
Massage Booking	Nov 6	Nov 7	2 days	Customer Registration & Login Function	Massage Booking Function Complete
Pool Booking	Nov 6	Nov 7	2 days	Customer Registration & Login Function	pool Booking Function Complete
Complaint Registration Function	Nov 8	Nov 8	1 days	Customer Registration & Login Function	Complaint Registration Function Complete

View Booking Status Function	Nov 9	Nov 9	1 days	Room Booking, Spa Booking, Games Booking, Service Booking, Pool Booking, Gym Booking, Massage Booking	View Booking Status Function Complete
Payment Gateway Integration	Nov 10	Nov 10	1 days	Room Booking, Spa Booking, Games Booking, Service Booking, Pool Booking, Gym Booking, Massage Booking	Payment Gateway Integration Complete
Manager Login Function	Nov 11	Nov 11	1 days	Website Design	Manager Login Function Complete
View and Update Complaints Function	Nov 12	Nov 12	1 days	Manager Login Function, Complaint Registration Function	View and Update Complaints Function Complete
Food Stock Management Function	Nov 13	Nov 13	1 days	Manager Login Function	Food Stock Management Function Complete
Delete Customer Appointments Function	Nov 14	Nov 14	1 days	Manager Login Function, Complaint Registration Function, Room Booking, Spa Booking, Games Booking, Service Booking, Pool Booking, Gym Booking, Massage Booking	Delete Customer Appointments Function Complete

Employee Management Function	Nov 15	Nov 15	1 days	Manager Login Function	Employee Management Function Complete
Virtual Wallet Management Function	Nov 16	Nov 17	2 days	Manager Login Function, Payment Gateway Integration	Virtual Wallet Management Function Complete
Chef Login Function	Nov 18	Nov 19	2 days	Website Design	Chef Login Function Complete
Food Ordering Function	Nov 20	Nov 20	1 days	Chef Login Function, Food Stock Management Function	Food Ordering Function Complete
Testing & Debugging	Nov 20	Nov 26	6 days	All Functions Complete	Testing

Results

- User can create an account successfully.
- Guest and admin can successfully log in if they enter correct credentials.
- Guest can successfully submit booking requests and complaints.
- Guest can successfully order resort services.
- Guest can keep track of their wallet amount.
- Admin can grant reservations if certain conditions are met (mentioned in "application features" section).
- Admin can easily manage employee data and stock.

Future Improvements

- Create a better and more beautiful user interface.
- Enhance the payment system, by allowing payment via credit cards. .
- Create an online store where guests can purchase products from resort stores online

References.

https://www.w3schools.com/html/

https://www.w3schools.com/css/css_intro.asp

https://www.w3schools.com/js/

https://supabase.com/docs/reference/javascript

https://www.tutorialspoint.com/javascript/index.htm

https://nodejs.org/en/about/

https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

https://www.altexsoft.com/blog/non-functional-requirements/#:~:text=Non%2Dfunctional%20requirements%20or%20NFRs,attempt%20to%20improve%20its%20functionality.

https://en.wikipedia.org/wiki/SQL

https://supabase.com/docs/guides/database

https://www.lucidchart.com/pages/er-diagrams

https://www.lucidchart.com/pages/database-diagram/database-schema