UNIVERSITY OF SRI JAYAWARDENEPURA

Faculty of Applied sciences

Department of Computer Science

CSC 110 2.0 Object Oriented Programming

Final Assignment – 2023

Resource Allocation system – Project Report

P.K Thisara Sehan Kavinda

AS2021532

31.05.2023

Project Identification Data

Full Name :	P.K.Thisara Shehan Kavinda
Index Number :	AS2021532
Project Title :	Resource Allocation System
Program Language :	JAVA
Database :	SQL
Date of Commencement :	04.05.2023
Date of Completion :	31.05.2023

Declaration

I hereby declare that the project work entitled “Resource Allocation System” submitted to the Department of Computer Science in university of Sri Jayewardenepura, is a record of an original work done by me under the guidance of Miss Upeksha and Miss Malshani. The result embodied in this thesis have not been submitted to any other University or institute for the award of any degree or diploma.

Date : 31.05.2023

Signature of Candidate: _____

Introduction

A hotel hall booking management system is a standalone software application designed to facilitate the management of hall bookings within a hotel. The primary target users of the system are administrators who oversee the booking process, while certain functions are also provided for customers to make bookings and do there payments.

The system typically includes the following features:

1. Hall Management: The system allows administrators to manage the halls within the hotel, including defining hall details such as availability, and pricing. It provides a centralized database where administrators can add, edit, and remove hall information as needed.

2. Booking Management: The system enables administrators to handle hall bookings efficiently. It provides a user-friendly interface where administrators can view the availability of halls, accept or decline booking requests, manage conflicts, and generate booking confirmations. The system also allows administrators to handle cancellations and modifications to existing bookings.

3. Customer Management: The system maintains a customer database where customer profiles and contact information are stored. Administrators can manage customer information, view booking history, and provide personalized services. Customers may also have access to their profiles and booking details through a customer portal.

4. Billing and Invoicing: The system includes billing and invoicing capabilities to streamline the financial aspect of hall bookings. It generates invoices for customers, tracks payments, and integrates with accounting systems to maintain accurate financial records.

Overall, a hotel hall booking management system provides an efficient and organized approach to handle hall bookings within a hotel. It simplifies the reservation process, improves customer satisfaction, optimizes resource allocation, and streamlines administrative tasks. The system aims to enhance the overall efficiency of hall bookings and contribute to the success of various events and functions hosted by the hotel.

Abstract

The hotel hall booking management system is a standalone application designed to streamline the process of managing hall bookings within a hotel. The system caters to both administrators and customers, providing a comprehensive solution for efficient hall reservation and management.

The system offers various features such as hall management, booking management, customer management, billing and invoicing. Administrators can easily manage hall availability, handle booking requests, and generate booking confirmations. Customers can conveniently browse available halls, make bookings, and view their reservation details.

Throughout the development of the project, the guidance and support of the project supervisor have been invaluable.

In conclusion, the hotel hall booking management system provides a user-friendly interface for efficient hall reservation and management. Its features contribute to enhanced customer experiences, optimized resource allocation, and streamlined administrative tasks. The successful completion of this project is attributed to the collaborative efforts and contributions of all involved stakeholders.

Assumptions

1. The application has two privilege levels: admin and customer.
2. Customer can check the availability of halls for given time period.
3. Customer can pay the payment.
4. Customer can do both by without logging.
5. admin can add/remove/update/search/maintain the admins,customers,halls and bookings.

Functional Requirements

1. Hall Availability: Customers can view the availability of halls based on their desired date(s) or period.
2. Hall Booking: Customers can reserve halls for specific dates, continuous periods, or selected days within a particular time frame.
3. Capacity Management: The system ensures that bookings do not exceed the capacity of the chosen hall.
4. Reservation Modification and Cancellation: Customers can modify or cancel their reservations within a specified timeframe.
5. Admin Dashboard: Administrators have access to a dashboard to manage hall information, reservations, and customer details

Non Functional requirements

1. Performance: The system should handle concurrent user requests and provide quick response times for searching and booking halls.
2. Security: User authentication and authorization mechanisms are implemented to ensure data privacy and system integrity.
3. Usability: The system should provide an intuitive user interface with clear instructions, error handling, and feedback mechanisms.
4. Scalability: The system should be designed to handle an increasing number of halls, customers, and concurrent reservations
5. Reliability: The system should be robust, with proper error handling and backup mechanisms to ensure data integrity.

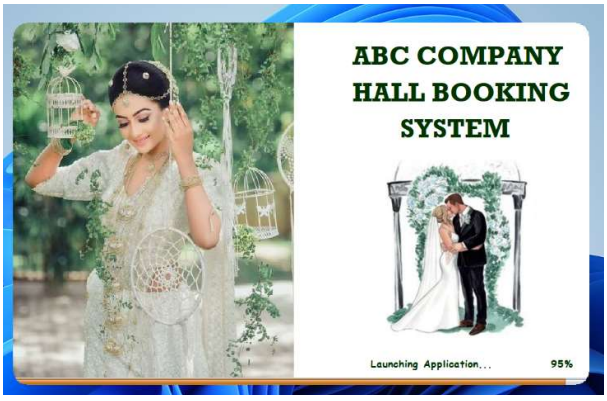
Contents

ABC Company Hall Booking System

1. Splash Form

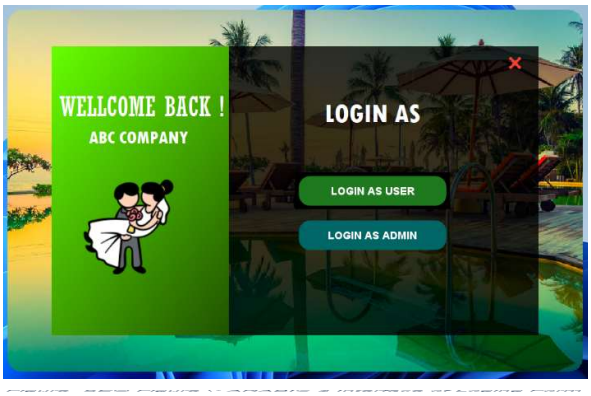
I. Splash Form's Interface

If you open the software, you get this loading form. You have to wait until complete the 100% progress to enter the login form. After complete the 100% progress you will enter to the login form.



2. Login Form

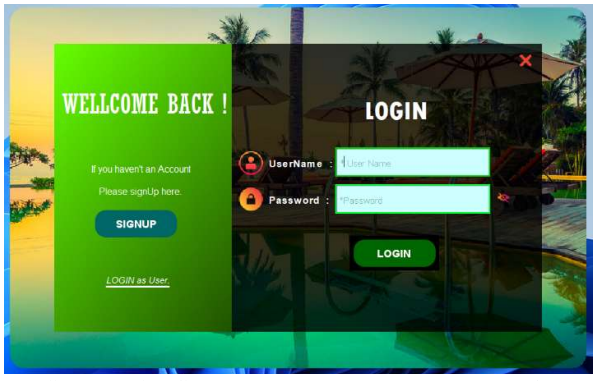
I. Interface of Loging form



In this form user can choose two option. If user is a admin the user can choose login as admin. If user is a customer user can choose login as customer.

3. Login as Admin Form

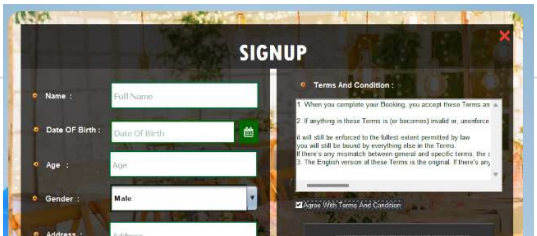
I. Interface of Loging



If user is a new admin then he or she can Signup using signup button. If admin already sign Up then admin can enter the username and password correctly and login to the admin form. There is a option of unhide and hide for the password text field. There is a link to go back to the login form also.

4. Signup Form

I. Interface of Signup

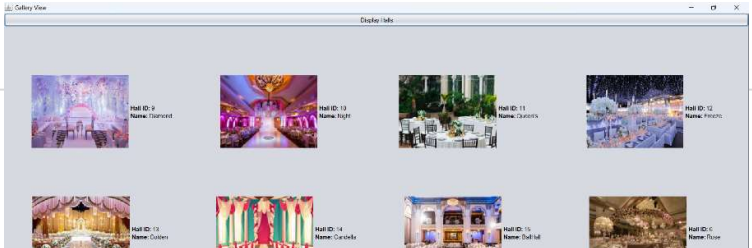


In here admin can Signup using filling all the details of admin. Admin should remember the password and username for login to the software. After the admin agree to terms and condition captcha code panel will be appear. Register button will enable if and only if the verification is correct. After the signup Admin can login now using password and username in the admin login form.

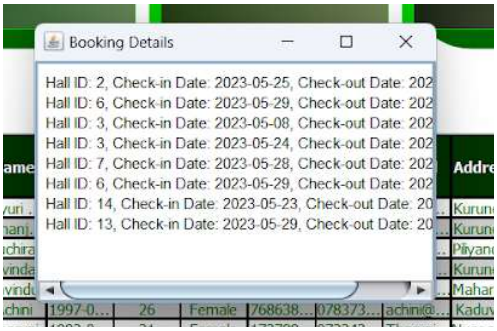
5. Admin's Home

I. Interface of Admin Home

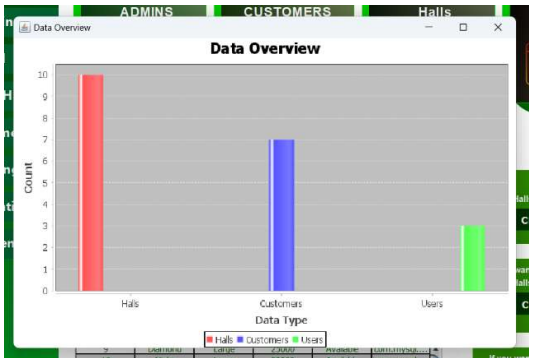
In this form admin can see all the things that admin can do. Admin management, hall management, customer management and all the booking management can do by admin. In home admin can view a quick view of the system. There are two tables that shows the details of customer and hall. There are 4 cards that represent the number of admins, number of customers, number of halls and number of bookings. There is a button that show all the hotel details that hall photo name and ID. This interface can see the customer also.



There is another button that display short description of unavailable hall with there check-in and check-out date. By that form admin can get a quick view of unavailable halls.

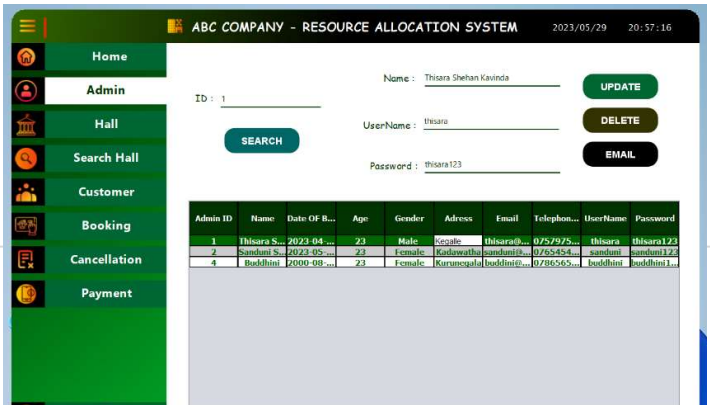


There is another button that show a bar graph of number of customers, admins and halls. From that admin can get a quick view of hall, admin and halls.

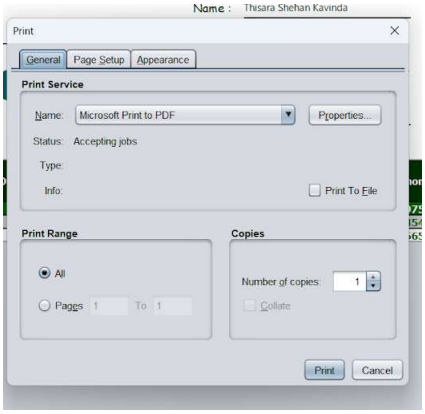


6. Admin Form

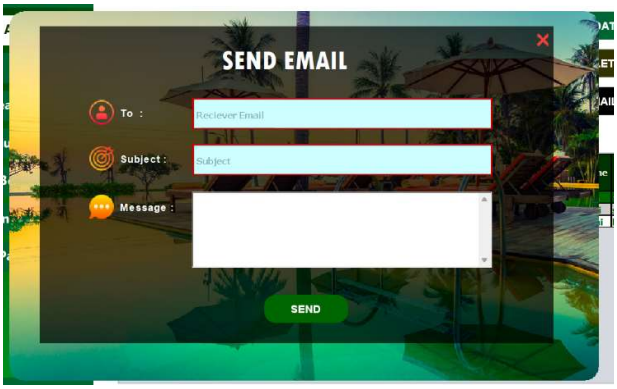
I. Interface of admin form



In here admin can Search , update and delete a admin using admin ID. There is a table that includes the data od admin. If we click a row that details will be display on relevant text field and combo boxes. There is a print option to print the table with all the details. From that admin can get a pdf file also.

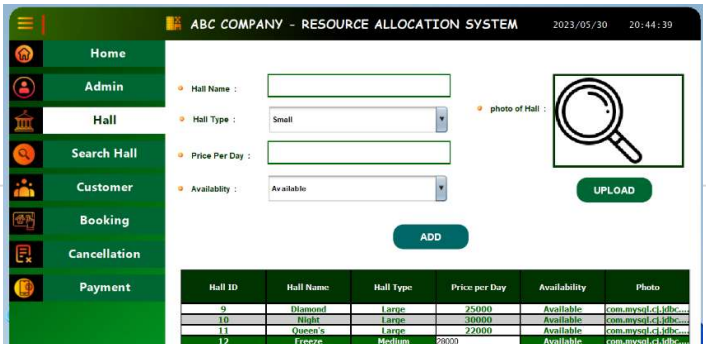


There is another option that can email the admins. It will useful when the admin forgot the password admin can get password or username via email.



7. Hall Form

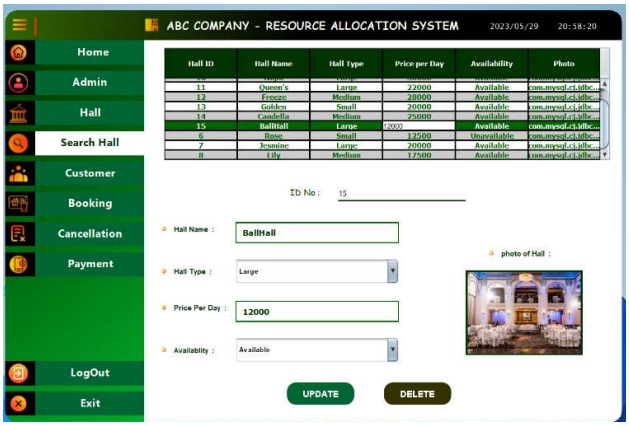
I. Interface of Hall



In here admin can add hall with hall's photo. There is table for show all the details of hotel. After hall is added then table is refresh and show newly added hall also.

8. Search Hall Form

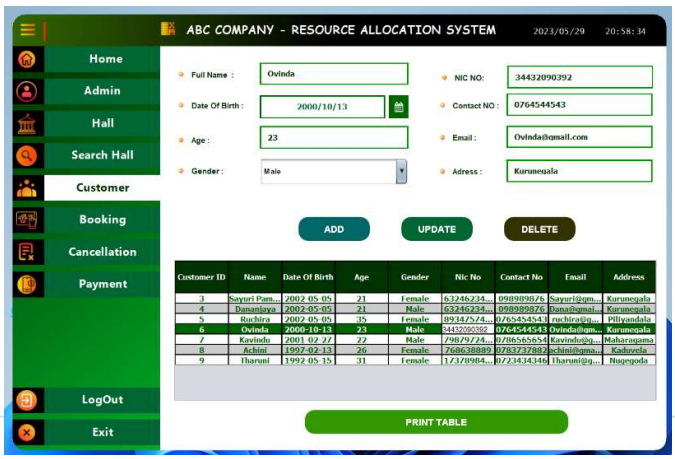
I. Interface of Search Hall



In here Admin can search every available and unavailable halls. In here display all the details of individual hall in text Fields. Update and delete options are also here.

9. Customer Form

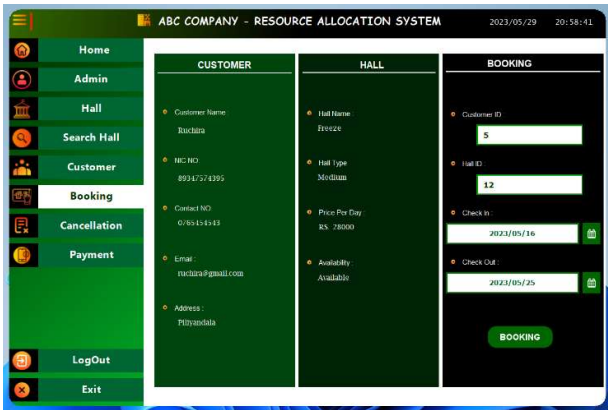
I. Interview of customer

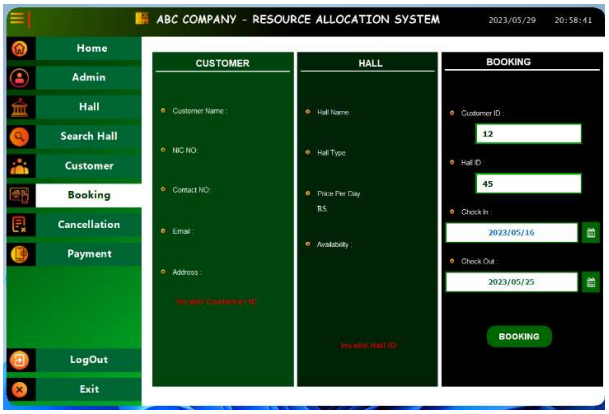


In here admin can add a customer update and delete. Customers details are in the table. To print the table there is a print option also. When click the row of table it shows the detail of that customer in relevant text field. Filters and validations are done in every fields.

10. Booking Form

I. Interface of Booking form

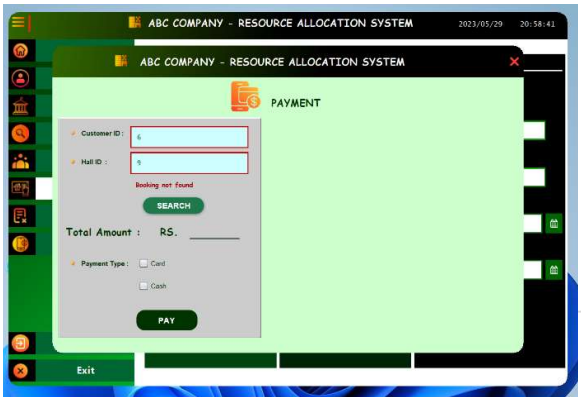


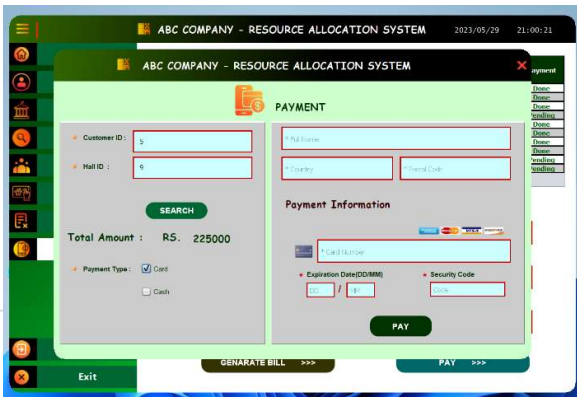


In here admin can booking. If admin want to book admin can enter the customer id in relevant field then in customer panel if there is a customer relevant to the ID number shows the detail. If not it shows that invalid ID. After that hall id should enter in the hall ID field. Then if there is hall in that ID then it shows all the details of hall. After fetching the check-in date and checkout date then admin can book the hall. Then admin enter to the payment Form.

11. Payment Form

I. Interface of Payment

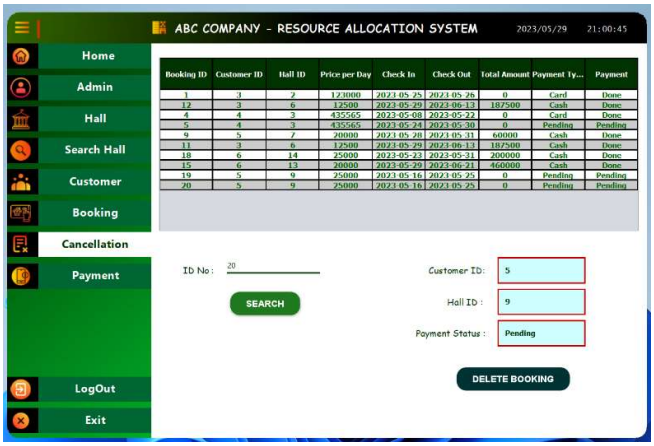




After booking is done then admin enter to the payment Frame. Here admin should enter the customer ID and hall ID after search if there is booking matching the customer ID and hall ID it display the total amount in total amount label. If there is no booking it shows in a label that booking is not found. Aftre display the total amount then admin can choose the cash or card method to pay. If he choose the payment method as card then card payment panel is appear and after filling the details then admin can pay. After payment hall is unavailable and booking table is updated the column payment type , total amount and payment status. If admin choose the payment method as cash then he can directly pay the system.

12. Cancellation form

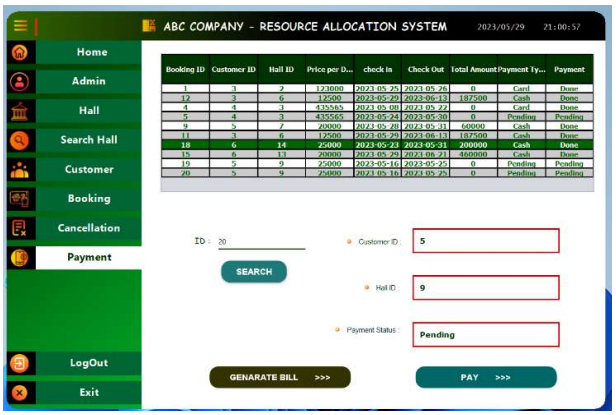
I. Interface of Cancellation form

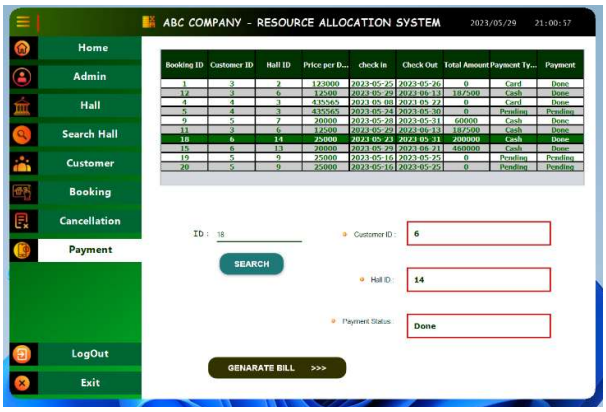


In here admin can view all the booking details. Admin can search for booking by booking ID. After that if want admin can delete the booking. After deleting booking record then that hall is available in hall Table.

13. Payment admin form

I. Interface of payment Admin form

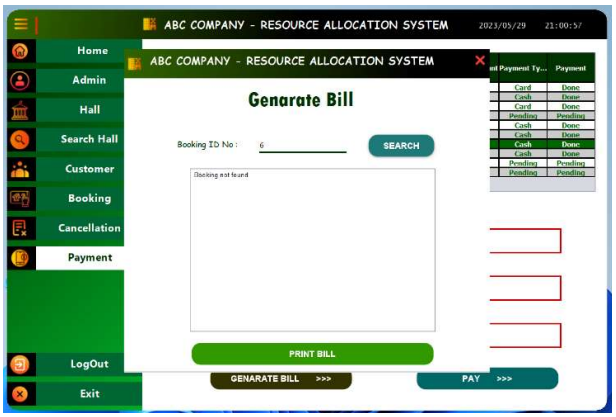


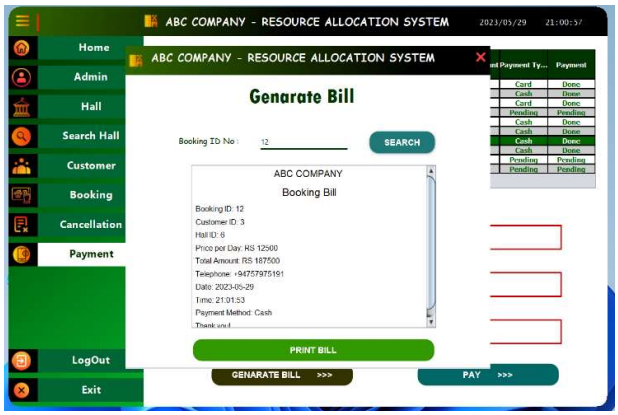


In here also admin can do the payment if payment is not done. After the searching booking ID if booking is done the payment pay button will not appear. If payment is not done then pay button will appear and admin can do the payment by click the button. If Payment is done then can generate a bill for that by clicking the generate bill.

14. Generate Bill form

I. Interface of generating Bill





In here admin can generate the bill by searching the booking ID. If there is a no booking record relevant to the booking id then it shows that no booking. If there is booking relevant to the booking ID then it print the bill by all the details including total price and date time that the bill issued.

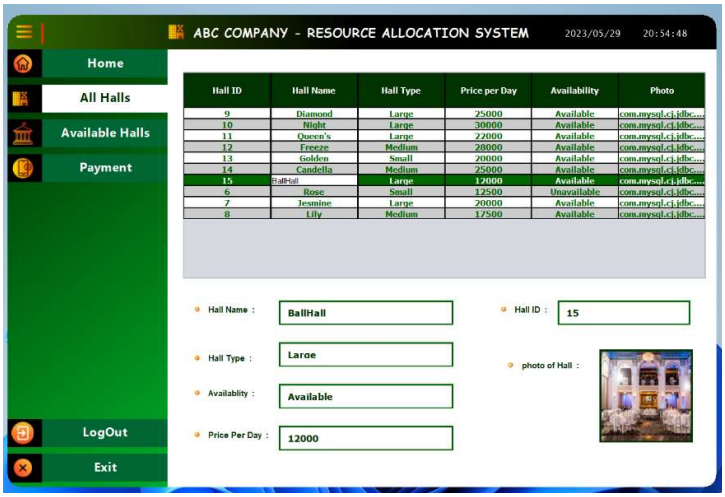
15. Customer Home Form

I. Interface of customer Home

After the customer login then he enter to the customer Home. In this form customer can get quick view of the company. There are three options to select that are all halls, Available halls and do the payement.

16. All halls form

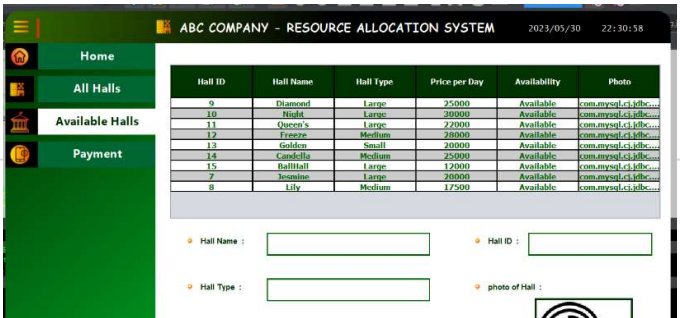
I. Interface of all halls form

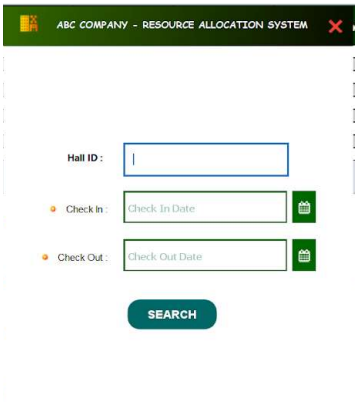


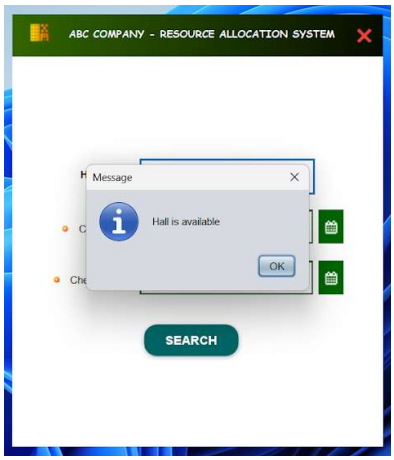
In here customer can see all the halls and all the halls details with its photo. By clicking rows in the table customer can all the individual details of the hall.

17. Available Halls

I. Interface of available halls







18. Model Classes

I. Admin Model

```
package abcweddinghallbooking_ms.Model;

import java.util.Date;
public class Admin {
    private int adminID;
   private String name;
   private Date dateOfBirth;
   private int age;
   private String gender;
   private String address;
   private String email;
   private String telephoneNo;
   private String userName;
   private String password;

public int getAdminID() {
   return adminID;
}

public void setAdminID(int adminID) {
   this.adminID = adminID;
}

public String getName() {
   return name;
}
```

```
public void setName(String name) {
    this.name = name;
}

public Date getDateOfBirth() {
    return dateOfBirth;
}

public void setDateOfBirth(Date dateOfBirth) {
    this.dateOfBirth = dateOfBirth;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String getGender() {
    return gender;
}

public void setGender(String gender) {
    this.gender = gender;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getTelephoneNo() {
    return telephoneNo;
}

public void setTelephoneNo(String telephoneNo) {
    this.telephoneNo = telephoneNo;
}

public String getUserName() {
    return userName;
}

public void setUserName(String userName) {
    this.userName = userName;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}
}
```

II. Customer Model

```
package abcweddinghallbooking_ms.Model;

import java.util.Date;
public class Customer {
    private int customer_id;
   private String name;
   private Date date_of_birth;
   private int age;
   private String gender;
   private String nic_no;
   private String contact_no;
   private String email;
   private String address;

   public int getCustomer_id() {
      return customer_id;
   }

   public void setCustomer_id(int customer_id) {
      this.customer_id = customer_id;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public Date getDate_of_birth() {
      return date_of_birth;
   }

   public void setDate_of_birth(Date date_of_birth) {
      this.date_of_birth = date_of_birth;
   }

   public int getAge() {
      return age;
   }

   public void setAge(int age) {
      this.age = age;
   }

   public String getGender() {
      return gender;
   }

   public void setGender(String gender) {
      this.gender = gender;
   }

   public String getNic_no() {
      return nic_no;
   }

   public void setNic_no(String nic_no) {
      this.nic_no = nic_no;
   }

   public String getContact_no() {
      return contact_no;
   }

   public void setContact_no(String contact_no) {
      this.contact_no = contact_no;
   }

   public String getEmail() {
      return email;
```

```
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

}
```

III. Hall Model

```
package abcweddinghallbooking_ms.Model;

import java.sql.Blob;
public class Hall {
    private int hallID;
    private String hallName;
    private String hallType;
    private int pricePerDay;
    private String availablity;
    private Blob photo;

    public int getHallID() {
        return hallID;
    }

    public void setHallID(int hallID) {
        this.hallID = hallID;
    }

    public String getHallName() {
        return hallName;
    }

    public void setHallName(String hallName) {
        this.hallName = hallName;
    }

    public String getHallType() {
        return hallType;
    }

    public void setHallType(String hallType) {
        this.hallType = hallType;
    }

    public int getPricePerDay() {
        return pricePerDay;
    }

    public void setPricePerDay(int pricePerDay) {
        this.pricePerDay = pricePerDay;
    }

    public String getAvailablity() {
        return availablity;
    }

    public void setAvailablity(String availablity) {
        this.availablity = availablity;
    }
```

```
    public Blob getPhoto() {
        return photo;
    }

    public void setPhoto(Blob photo) {
        this.photo = photo;
    }
```

IV. Booking Model

```
import java.util.Date;

/**
 *
 * @author sanduni punchihewa
 */
public class Booking {

    private int booking_id;
    private int customer_id;
    private int hall_id;
    private int price_per_day;
    private Date check_in;
    private Date check_out;
    private int total_amount;
    private String payment_type;
    private String payment;

    public int getBooking_id() {
        return booking_id;
    }

    public void setBooking_id(int booking_id) {
        this.booking_id = booking_id;
    }

    public int getCustomer_id() {
        return customer_id;
    }

    public void setCustomer_id(int customer_id) {
        this.customer_id = customer_id;
    }

    public int getHall_id() {
        return hall_id;
    }

    public void setHall_id(int hall_id) {
        this.hall_id = hall_id;
    }

    public int getPrice_per_day() {
        return price_per_day;
    }

    public void setPrice_per_day(int price_per_day) {
        this.price_per_day = price_per_day;
    }

    public Date getCheck_in() {
        return check_in;
    }

    public void setCheck_in(Date check_in) {
        this.check_in = check_in;
    }

    public Date getCheck_out() {
        return check_out;
```

```
    }

    public void setCheck_out(Date check_out) {
        this.check_out = check_out;
    }

    public int getTotal_amount() {
        return total_amount;
    }

    public void setTotal_amount(int total_amount) {
        this.total_amount = total_amount;
    }

    public String getPayment_type() {
        return payment_type;
    }

    public void setPayment_type(String payment_type) {
        this.payment_type = payment_type;
    }

    public String getPayment() {
        return payment;
    }

    public void setPayment(String payment) {
        this.payment = payment;
    }
```

19. Controller Classes

I. Admin

```
package abcweddinghallbooking_ms.Controller;

import abcweddinghallbooking_ms.Model.Admin;
import java.util.List;

/**
 *
 * @author sanduni punchihewa
 */
public interface AdminDAO {
    public void save(Admin admin);
      public void update (Admin admin);
      public void delete (Admin admin);
      public Admin get(int id);
      public List<Admin> list();
}
```

```
package abcweddinghallbooking_ms.Controller;
import abcweddinghallbooking_ms.DBConnection;
import abcweddinghallbooking_ms.Model.Admin;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import java.sql.*;
public class AdminDAOImp implements AdminDAO {

  public void save(Admin admin) {
       java.util.Date dateOfBirth = admin.getDateOfBirth();
```

```
    try{

        Connection con =  DBConnection.getConnection();
        String sql = "INSERT into
admin(name,date_of_birth,age,gender,address,email,telephon
e_no,username,Upassword) values (?,?,?,?,?,?,?,?,?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, admin.getName());
        ps.setDate(2, new java.sql.Date(dateOfBirth.getTime()));
        ps.setInt(3, admin.getAge());
        ps.setString(4, admin.getGender());
        ps.setString(5,admin.getAddress());
        ps.setString(6,admin.getEmail());
        ps.setString(7,admin.getTelephoneNo());
        ps.setString(8,admin.getUserName());
        ps.setString(9,admin.getPassword());

        ps.executeUpdate();
        JOptionPane.showMessageDialog(null, "Registered!");

    }catch(Exception e){
        e.printStackTrace();
    }

  }

  public void update(Admin admin) {

    try{

        Connection con = DBConnection.getConnection();
        String sql = "UPDATE admin set
name=?,username=?,Upassword=? where admin_id=?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, admin.getName());
        ps.setString(2, admin.getUserName());
        ps.setString(3, admin.getPassword());
        ps.setInt(4, admin.getAdminID());
        ps.executeUpdate();
        JOptionPane.showMessageDialog(null, "Updated!");

    }catch(Exception e){
        e.printStackTrace();
    }
  }

  public void delete(Admin admin) {
   try{

      Connection con = DBConnection.getConnection();
  String sql = "DELETE FROM admin WHERE admin_id = ?";
  PreparedStatement ps = con.prepareStatement(sql);
  ps.setString(1, admin.getName());

  int affectedRows = ps.executeUpdate();
  if (affectedRows > 0) {
    JOptionPane.showMessageDialog(null, "Delete
successful!");
  } else {
    JOptionPane.showMessageDialog(null, "No rows
deleted!");
  }

    }catch(Exception e){
        e.printStackTrace();
    }
  }

  public Admin get(int id) {
```

```
Admin adm = new Admin();

try{
   Connection con = DBConnection.getConnection();
   String sql = "select * from admin where admin_id=?";
   PreparedStatement ps = con.prepareStatement(sql);
   ps.setInt(1, id);
   ResultSet rs = ps.executeQuery();
   if(rs.next()){
    adm.setAdminID(rs.getInt("admin_id"));
        adm.setName(rs.getString("name"));
         adm.setUserName(rs.getString("username"));
        adm.setPassword(rs.getString("Upassword"));

   }
}catch(Exception e){
   e.printStackTrace();
}

return adm;

}

public List<Admin> list() {
    List<Admin> list = new ArrayList<Admin>();
   try{

      Connection con = DBConnection.getConnection();
      String sql = "select * from admin";
      PreparedStatement ps = con.prepareStatement(sql);
      ResultSet rs = ps.executeQuery();

      while(rs.next()){
         Admin adm = new Admin();
         adm.setAdminID(rs.getInt("admin_id"));
         adm.setName(rs.getString("name"));
         adm.setDateOfBirth(rs.getDate("date_of_birth"));
         adm.setAge(rs.getInt("age"));
         adm.setGender(rs.getString("gender"));
         adm.setAddress(rs.getString("address"));
         adm.setEmail(rs.getString("email"));
         adm.setTelephoneNo(rs.getString("telephone_no"));
         adm.setUserName(rs.getString("username"));
         adm.setPassword(rs.getString("Upassword"));

         list.add(adm);
      }
   }catch(Exception e){

      e.printStackTrace();

   }

   return list;

}

 public boolean login(String username, String password) {
   try {
     Connection con = DBConnection.getConnection();

      // Create SQL query
      String query = "SELECT * FROM admin WHERE username
= ? AND Upassword = ?";
      PreparedStatement statement =
con.prepareStatement(query);
      statement.setString(1, username);
      statement.setString(2, password);

      // Execute the query
      ResultSet resultSet = statement.executeQuery();

      // Check if a matching record exists
      boolean loggedIn = resultSet.next();
```

```
        // Close resources
        resultSet.close();
        statement.close();
        //con.close();

        return loggedIn;
      } catch (SQLException e) {
          e.printStackTrace();
          return false;
      }
  }

}
```

II. Customer

```
package abcweddinghallbooking_ms.Controller;

import abcweddinghallbooking_ms.Model.Customer;
import java.util.List;
public interface CustomerDAO {
    public void save(Customer customers);
     public void update (Customer customers);
     public void delete (Customer customers);
     public Customer get(int id);
     public List<Customer> list();
}
```

```
package abcweddinghallbooking_ms.Controller;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import java.sql.*;
import abcweddinghallbooking_ms.DBConnection;
import abcweddinghallbooking_ms.Model.Customer;
/**
 *
 * @author sanduni punchihewa
 */
public class CustomerDAOImp implements CustomerDAO{
   @Override
   public void save(Customer customers) {

       java.util.Date dateOfBirth = customers.getDate_of_birth();


       try{


           Connection con =  DBConnection.getConnection();
           String sql = "INSERT into
customer(name,date_of_birth,age,gender,nic_no,contact_no,e
mail,address) values (?,?,?,?,?,?,?,?)";
           PreparedStatement ps = con.prepareStatement(sql);
           ps.setString(1, customers.getName());
           ps.setDate(2, new java.sql.Date(dateOfBirth.getTime()));
           ps.setInt(3, customers.getAge());
           ps.setString(4, customers.getGender());
           ps.setString(5,customers.getNic_no());
           ps.setString(6,customers.getContact_no());
           ps.setString(7,customers.getEmail());
           ps.setString(8,customers.getAddress());

           ps.executeUpdate();
           JOptionPane.showMessageDialog(null, "Saved!");

       }catch(Exception e){
           e.printStackTrace();
       }

   }
```

```
    @Override
    public void update(Customer customers) {

        java.util.Date dateOfBirth = customers.getDate_of_birth();

        try{

            Connection con = DBConnection.getConnection();
            String sql = "UPDATE customer set
name=?,date_of_birth=?,age=?,gender=?,nic_no=?,contact_no
=?,email=?,address=? where customer_id=?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, customers.getName());
            ps.setDate(2, new java.sql.Date(dateOfBirth.getTime()));
            ps.setInt(3, customers.getAge());
            ps.setString(4, customers.getGender());
            ps.setString(5,customers.getNic_no());
            ps.setString(6,customers.getContact_no());
            ps.setString(7,customers.getEmail());
            ps.setString(8,customers.getAddress());
            ps.setInt(9, customers.getCustomer_id());
            ps.executeUpdate();
            JOptionPane.showMessageDialog(null, "Updated!");

        }catch(Exception e){

            e.printStackTrace();
        }
    }

    @Override
    public void delete(Customer customers) {
        try{

            Connection con = DBConnection.getConnection();
            String sql = "DELETE FROM customer WHERE customer_id =
?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setInt(1, customers.getCustomer_id()); // Replace
'customerId' with the actual value you want to delete

            int affectedRows = ps.executeUpdate();
            if (affectedRows > 0) {
                JOptionPane.showMessageDialog(null, "Delete
successful!");
            } else {
                JOptionPane.showMessageDialog(null, "No rows
deleted!");
            }

        }catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public Customer get(int id) {
        throw new UnsupportedOperationException("Not
supported yet."); // Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/Gen
eratedMethodBody
    }

    @Override
    public List<Customer> list() {

        List<Customer> list = new ArrayList<Customer>();
        try{

            Connection con = DBConnection.getConnection();
            String sql = "select * from customer";
            PreparedStatement ps = con.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
```

```
        while(rs.next()){
            Customer cs = new Customer();
            cs.setCustomer_id(rs.getInt("customer_id"));
            cs.setName(rs.getString("name"));
            cs.setDate_of_birth(rs.getDate("date_of_birth"));
            cs.setAge(rs.getInt("age"));
            cs.setGender(rs.getString("gender"));
            cs.setNic_no(rs.getString("nic_no"));
            cs.setContact_no(rs.getString("contact_no"));
            cs.setEmail(rs.getString("email"));
            cs.setAddress(rs.getString("address"));

            list.add(cs);
            }
        }catch(Exception e){

            e.printStackTrace();

        }

        return list;

    }
}
```

III. Hall

```
package abcweddinghallbooking_ms.Controller;

import abcweddinghallbooking_ms.Model.Hall;
import java.util.List;

/**
 *
 * @author sanduni punchihewa
 */
public interface HallDAO {
    public void save(Hall hall);
    public void update (Hall hall);
    public void delete (Hall hall);
    public Hall get(int id);
    public List<Hall> list();
}
package abcweddinghallbooking_ms.Controller;
import abcweddinghallbooking_ms.DBConnection;
import abcweddinghallbooking_ms.Model.Hall;
import java.sql.Connection;
import java.util.List;
import javax.swing.JOptionPane;
import java.sql.PreparedStatement;
import java.util.ArrayList;
import java.sql.*;
/**
 *
 * @author sanduni punchihewa
 */
public class HallDAOImp {

public void save(Hall hall) {
   try {
      Connection con = DBConnection.getConnection();
      String sql = "INSERT INTO hall(hall_id, hall_name, hall_type,
price_per_day, availability, photo) VALUES (?, ?, ?, ?, ?, ?)";

      PreparedStatement ps = con.prepareStatement(sql);

      ps.setInt(1, hall.getHallID());
      ps.setString(2, hall.getHallName());
      ps.setString(3, hall.getHallType());
      ps.setInt(4, hall.getPricePerDay());
      ps.setString(5, hall.getAvailablity());
      ps.setBlob(6, hall.getPhoto());
```

```
      ps.executeUpdate();
      JOptionPane.showMessageDialog(null, "Saved!");
    } catch (Exception e) {
      e.printStackTrace();
    }
}



    public void update(Hall hall) {



      try{


          Connection con = DBConnection.getConnection();
          String sql = "UPDATE hall set
hall_name=?,hall_type=?,price_per_day=?,availablity=?,photo=
? where customer_id=?";
          PreparedStatement ps = con.prepareStatement(sql);
          ps.setString(1,hall.getHallName());



          ps.setString(2, hall.getHallType());
           ps.setInt(3,hall.getPricePerDay());
          ps.setString(4,hall.getAvailablity());


          ps.setBlob(5,hall.getPhoto());


          ps.setInt(6, hall.getHallID());
          ps.executeUpdate();
          JOptionPane.showMessageDialog(null, "Updated!");



      }catch(Exception e){
          e.printStackTrace();
      }
    }


    public void delete(Hall hall) {
         try{


          Connection con = DBConnection.getConnection();
    String sql = "DELETE FROM hall WHERE hall_id = ?";
    PreparedStatement ps = con.prepareStatement(sql);
    ps.setInt(1, hall.getHallID());


    int affectedRows = ps.executeUpdate();
    if (affectedRows > 0) {
      JOptionPane.showMessageDialog(null, "Delete
successful!");
    } else {
      JOptionPane.showMessageDialog(null, "No rows
deleted!");
    }



      }catch(Exception e){
          e.printStackTrace();
      }
    }


    public Hall get(int id) {
      throw new UnsupportedOperationException("Not
supported yet."); // Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/Gen
eratedMethodBody
    }



    public List<Hall> list() {
      List<Hall> list = new ArrayList<Hall>();
      try{


          Connection con = DBConnection.getConnection();
          String sql = "select * from hall";
```

```
        PreparedStatement ps = con.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();

        while(rs.next()){
            Hall hall = new Hall();
            hall.setHallID(rs.getInt("hall_id"));
            hall.setHallName(rs.getString("hall_name"));

            hall. setHallType(rs.getString("hall_type"));
            hall.setPricePerDay(rs.getInt("price_per_day"));
            hall.setAvailablity(rs.getString("availablity"));
            hall.setPhoto(rs.getBlob("photo"));


            list.add(hall);
        }
    }catch(Exception e){

        e.printStackTrace();

    }

    return list;
}

public List<Hall> list1(String availability) {
    List<Hall> list = new ArrayList<Hall>();
    try{

        Connection con = DBConnection.getConnection();
        String sql = "select * from hall where availablity = ?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, availability);
        ResultSet rs = ps.executeQuery();

        while(rs.next()){
            Hall hall = new Hall();
            hall.setHallID(rs.getInt("hall_id"));
            hall.setHallName(rs.getString("hall_name"));

            hall. setHallType(rs.getString("hall_type"));
            hall.setPricePerDay(rs.getInt("price_per_day"));
            hall.setAvailablity(rs.getString("availablity"));
            hall.setPhoto(rs.getBlob("photo"));


            list.add(hall);
        }
    }catch(Exception e){

        e.printStackTrace();

    }

    return list;
}

public void updateAvailability(int hallID, String available) {

    PreparedStatement statement = null;

    try {
        // Obtain a database connection
        Connection con = DBConnection.getConnection();

        // Prepare the SQL statement
        String sql = "UPDATE hall SET availablity = ? WHERE
hall_id = ?";
        statement = con.prepareStatement(sql);

        // Set the parameters
        statement.setString(1, available);
        statement.setInt(2, hallID);

        // Execute the update statement
        statement.executeUpdate();
```

```
        } catch (SQLException e) {
            e.printStackTrace();
    }
    }
```

IV. Booking

```
package abcweddinghallbooking_ms.Controller;

import abcweddinghallbooking_ms.Model.Booking;
import java.util.List;

/**
 *
 * @author sanduni punchihewa
 */
public interface BookingDAO {
   public void save(Booking booking);
      public void update (Booking booking);
      public void delete (Booking booking);
      public Booking get(int id);
      public List<Booking> list();
} package abcweddinghallbooking_ms.Controller;

import abcweddinghallbooking_ms.DBConnection;
import abcweddinghallbooking_ms.Model.Booking;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import java.sql.*;


/**
/**
 *
 * @author sanduni punchihewa
 */
public class BookingDAOImp implements BookingDAO {

   /**
    *
    * @param booking
    */
   @Override

public void save(Booking booking) {
   java.util.Date checkin = booking.getCheck_in();
   java.util.Date checkout = booking.getCheck_out();

   try {
      Connection con = DBConnection.getConnection();
      String sql = "INSERT into
hall_book(customer_id,hall_id,price_per_day,check_in,check_o
ut,total_amount,payment_type,payment) values
(?,?,?,?,?,?,?,?)";
      PreparedStatement ps = con.prepareStatement(sql);
      ps.setInt(1, booking.getCustomer_id());
      ps.setInt(2, booking.getHall_id());
      ps.setInt(3, booking.getPrice_per_day());
      ps.setDate(4, new java.sql.Date(checkin.getTime()));
      ps.setDate(5, new java.sql.Date(checkout.getTime()));
      ps.setInt(6, booking.getTotal_amount());
      ps.setString(7, booking.getPayment_type());
      ps.setString(8, booking.getPayment());

      ps.executeUpdate();
      JOptionPane.showMessageDialog(null, "Saved!");

   } catch (Exception e) {
      e.printStackTrace();
   }
}
```

```
}

    @Override
    public void update(Booking booking) {
        throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
    }

    @Override
    public void delete(Booking booking) {
      try{

         Connection con = DBConnection.getConnection();
    String sql = "DELETE FROM hall_book WHERE booking_id = ?";
    PreparedStatement ps = con.prepareStatement(sql);
    ps.setInt(1, booking.getBooking_id());

    int affectedRows = ps.executeUpdate();
    if (affectedRows > 0) {
        JOptionPane.showMessageDialog(null, "Delete successful!");
    } else {
        JOptionPane.showMessageDialog(null, "No rows deleted!");
    }



        }catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public Booking get(int id) {
        Booking adm = new Booking();

    try{
    Connection con = DBConnection.getConnection();
    String sql = "select * from hall_book where booking_id=?";
    PreparedStatement ps = con.prepareStatement(sql);
    ps.setInt(1, id);
    ResultSet rs = ps.executeQuery();
    if(rs.next()){
     adm.setCustomer_id(rs.getInt("customer_id"));
            adm.setHall_id(rs.getInt("hall_id"));
            adm.setPayment(rs.getString("payment"));



    }
    }catch(Exception e){
    e.printStackTrace();
    }

    return adm;
    }



    @Override
    public List<Booking> list() {
    List<Booking> list = new ArrayList<Booking>();
    try{

        Connection con = DBConnection.getConnection();
        String sql = "select * from hall_book";
        PreparedStatement ps = con.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();

        while(rs.next()){
            Booking book = new Booking();
            book.setBooking_id(rs.getInt("booking_id"));
```

```
            book.setCustomer_id(rs.getInt("customer_id"));
            book.setHall_id(rs.getInt("hall_id"));
            book.setPrice_per_day(rs.getInt("price_per_day"));
            book.setCheck_in(rs.getDate("check_in"));
            book.setCheck_out(rs.getDate("check_out"));
            book.setTotal_amount(rs.getInt("total_amount"));
            book.setPayment_type(rs.getString("payment_type"));
            book.setPayment(rs.getString("payment"));



            list.add(book);
            }
        }catch(Exception e){

            e.printStackTrace();

        }

        return list;
    }

}
```

V. Hall Availability

```
package abcweddinghallbooking_ms.Controller;


import abcweddinghallbooking_ms.DBConnection;
import java.time.LocalDate;
import java.sql.*;
import javax.swing.JOptionPane;
public class HallAvailabiltyCheck {
   public boolean isbooked(int hallID, LocalDate startDate,
LocalDate endDate){
      try{
         Connection con = DBConnection.getConnection();
         String sql = "select * from hall_book where hall_id=? and
((check_in <=?and check_out>=?) or (check_in<=? and
check_out>=?)or (check_in>=? and check_out <=?))";
         PreparedStatement ps = con.prepareStatement(sql);
         ps.setInt(1, hallID);
         ps.setString(2, startDate.toString());
         ps.setString(3, startDate.toString());
         ps.setString(4, endDate.toString());
         ps.setString(5, endDate.toString());
         ps.setString(6, startDate.toString());
         ps.setString(7, endDate.toString());


         try(ResultSet rs = ps.executeQuery()){
            if(rs.next()){
               JOptionPane.showMessageDialog(null, "Hall is
Unavailable");
               return true;
            }
         }


         JOptionPane.showMessageDialog(null, "Hall is
available");
         return false;



      }catch(Exception ex){
         ex.printStackTrace();
         JOptionPane.showMessageDialog(null, "Error occured
while checking availablity");
         return false;
      }
```

VI. Slide Show

```
package abcweddinghallbooking_ms.Controller;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JLabel;
```

```
import javax.swing.Timer;


public class SlideshowController {


    private JLabel[] labels = new JLabel[5];


    public SlideshowController(JLabel lbl1, JLabel lbl2, JLabel lbl3,
JLabel lbl4, JLabel lbl5) {

    labels[0] = lbl1;

    labels[1] = lbl2;

    labels[2] = lbl3;

    labels[3] = lbl4;

    labels[4] = lbl5;

}
```

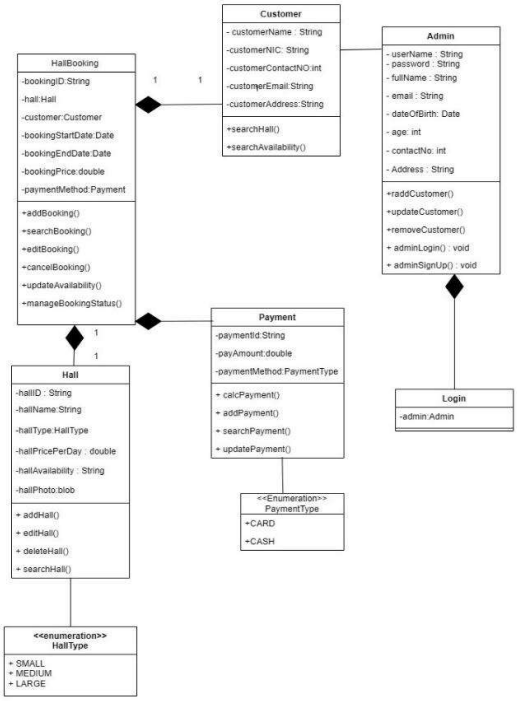
20. DBConnection class

```
package abcweddinghallbooking_ms;


import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.*;


/**

 *

 * @author PC

 */

public class DBConnection {

 private static final String DRIVER_CLASS_NAME =
"com.mysql.cj.jdbc.Driver";

   private static final String DATABASE_URL =
"jdbc:mysql://localhost:3307/resource_allocation_ms";

   private static final String USERNAME = "root";

   private static final String PASSWORD = "";


   private static Connection conn = null;


   // get database connection

   public static Connection getConnection() {


      try {

         if (conn == null) {

            Class.forName(DRIVER_CLASS_NAME);

            conn = DriverManager.getConnection(DATABASE_URL,
USERNAME, PASSWORD);

         }


         return conn;

      } catch (ClassNotFoundException ex) {

         ex.printStackTrace();

         return null;

      } catch (SQLException ex) {

         ex.printStackTrace();

         return null;

      }

   }


   // close database connection

   public static void closeConnection() {

      try {

         if (conn != null) {

            conn.close();

            conn = null;

         }

      } catch (SQLException ex) {

         ex.printStackTrace();

      }

   }


}
```

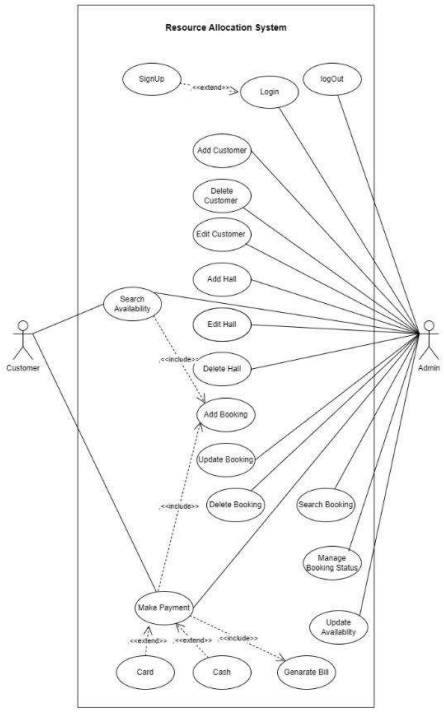
```
    public void startSlideshow() {
    int delay = 2000; // milliseconds
    Timer timer = new Timer(delay, new ActionListener() {
        int currentIndex = 0;

        @Override
        public void actionPerformed(ActionEvent e) {
            labels[currentIndex].setVisible(false);
            currentIndex = (currentIndex + 1) % labels.length;
            labels[currentIndex].setVisible(true);
        }
    });
    timer.start();
}
}
```

21. Diagrams

I. Class Diagram



II. Use case Diagram



VII. ER Diagram