THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

DATABASE PRINCIPLES

INFS7901

Online Property Booking System (OPBS)

Submitted By:
Jawaher ALGHAMDI ID. s4474982
Daniel HERR ID. s44556994
Siddharth KRISHNAKUMAR ID. s4651743
Megha SHAKTAWAT ID. s4619872

April 15, 2021

Contents

1 Project Outline

The project proposal outlines an Online Property Booking System (OPBS), which concentrates on a property rental domain and provides different property types to registered clients.

The system will allow registered customers to reserve short-stay self-catering accommodation for a chosen duration of time. For a customer to make a booking, they need to enrol into the system by populating a form that registers their name, address, phone, password, email address, and credit card information (card number and expiry date) and unique customer ID. A registered customer can then add in details of their guests (i.e., those guests are described by their names and DoB and are unable to be uniquely identified without a customer ID), select the desired property for a defined date range and register their intent to rent the property by making a reservation (the reservation is tentative until the property manager has confirmed it). Each reservation a customer makes — of which there could be more than one, will be identifiable with a unique reservation number (R-id). Additionally, the OPBS allows the user to make special requests and calculate booking costs when the date range is encoded by the customer. There is only one property manager involved with the OPBS, and they can be described by ManagerID (unique), email, name and phone number. In addition to the aforementioned task of confirming reservation status, the property manager will create the bookings that will be available for customers to reserve and host guests. Once a property manager has approved the booking request after reviewing the customer, guest and reservation details, the customer is then able to make a payment and confirm the status of their reservation. Should the property manager decide against the proposed booking or should there be a conflict with the booking schedule, the property manager can reject the reservation. The customer is able to cancel their reservation at any point up until they make payment. After this point, there is a strict no refund policy. Following the approval of the reservation by the property manager, the customer will then be able to make a transaction. The transaction can be made online with a credit card or alternately, cash can be paid upon arrival. A unique transaction ID, timestamp and amount will describe the transaction and it will enable confirmation of the reservation. Given the confirmation, the property manager will adjust the reservation status accordingly. Each property is described by a cost and unique property ID. There are three different kinds of properties:

- LuxuryVilla (with ensuite)
- Cabin (with its own hot tub)
- StarGazeTent (includes a private firepit)

Each reservation must be managed by the property manager. Upon the completion of a short stay, the property manager will adjust the reservation status to indicate that the customers and their guests have checked out. Customers will be able to rate their stay at the property (with a unique rating id) by grading their experience on a scale from 1 to 5. Average feedback received from the customers for a property is to be considered the property's overall rating, and it can be viewed by future customers to assist with their booking decisions. The OPBS mustn't have any overlap between reservations in the same property. This issue is mitigated by enforcing the constraint that each combination of property ID and reservation date are to be unique. This is handled by enforcing a uniqueness constraint at the application level (not addressed in SQL) that ensures no two separate reservations can share the same Property ID with overlapping dates. In other words, this will prevent double bookings of any new booking for a particular Property ID which clashes with an existing booking for the same Property ID within its date range.

Our group agreed upon working within this domain because we felt that it was a good example in which course content could be illustrated and that we could dial up the complexity with relative ease should there be a need to do so (for example, with additional properties, property manager or catering arrangements). Additionally, with the boom of services like Airbnb, it is clear that there is a real-world application for DBMS of this type, and we wanted to learn about what was involved in such a system and the various approaches that one could take to solve the associated problems. By using the OPBS, all stakeholders in the booking experience stand to benefit.

The proposed OPBS will add value by overcoming some of the issues in the existing booking systems wherein customers typically pay in order to confirm their reservations. Using the OPBS, a customer is able to reserve their desired property and delay payment (and therefore also the right to cancel) for any point up until the check-in time of the reservation until which point the reservation will have a status of 'pending'. Business owners and the property manager benefit from the certainty of avoiding overbooking their rental properties and by having quick access to essential information such

as booking frequency, the number of returning customers and the average length of stay. Finally, the efficient booking process means that the business is able to offer its booking services 24/7, and little manpower is required for arduous administrative tasks.

The proposed system will allow a user to interact with data and do the following:

- enable customers to make and cancel reservations; however, the OPBS has a strict no-refund policy in place, so customers will not be entitled to a refund should they cancel their reservation.
- observe the booking schedule of each property.
- allow customers to rate their stay at the property.
- see which property is the most popular in terms of ratings.
- find a set of customers who are the "top spenders".
- calculate the average number of days that each property type is typically reserved.
- allow customers to make special requests for their reservations.
- give the property manager the ability to generate bookings that customers can then reserve.
- enable the manager the authority to accept/reject reservations and aggregate the total number of bookings over a given period.
- ensure that reservations are only confirmed upon the transfer of funds.

We have built the system primarily using MySQL and php. The Graphical User Interface front end development has been processed through html, css with the use of bootstrap. All data has been stored on the server side by using phpmyadmin. Additionally, we have taken steps to ensure customer privacy and security by using encryption protocols in such as way that owners of the site cannot see the user generated passwords.

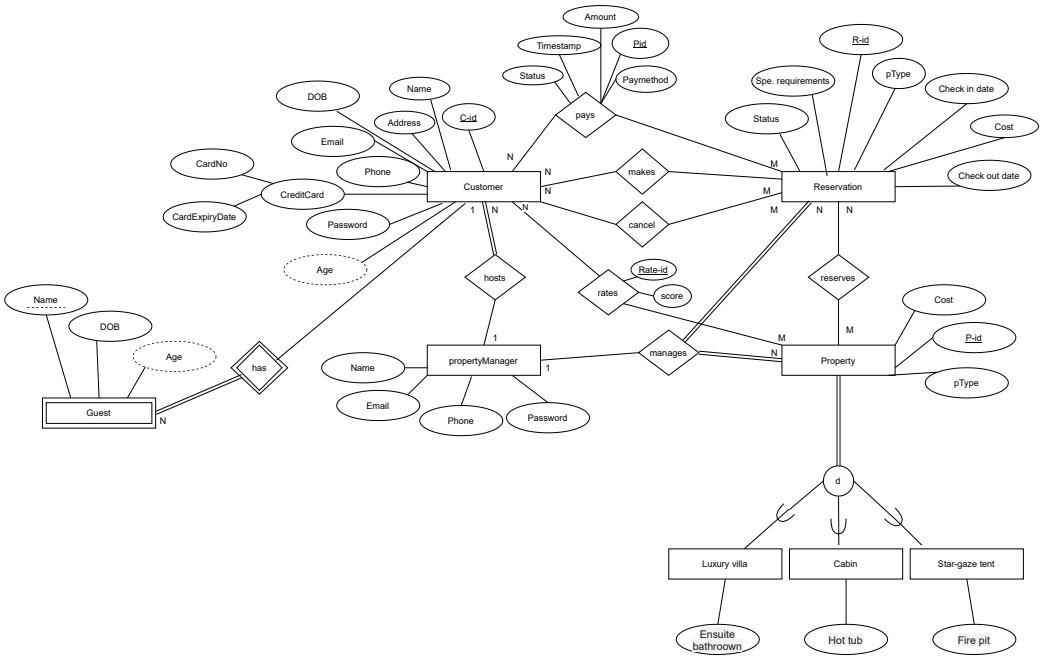

Figure 1: The Original ER Diagram for the Online Property Booking System (OPBS)

2 The Updated ER Diagram

Figure 2 shows the updated ER diagram for the system. The following are the modifications made:

- (C-id, R-id, Pid, P-id, Rate-id) attributes are renamed to (CustomerID, ReservationID, PaymentID, PropertyID, RateID).
- Adding ManagerID attribute as an identifier for the property manager relation.
- Changing the relationship type between Reservation and Property to have total participation from Reservation to Property where Reservation cannot exist without Property and each reservation must have a property.
- Removing property type attribute from Property relation as it was found to be a redundant attribute.

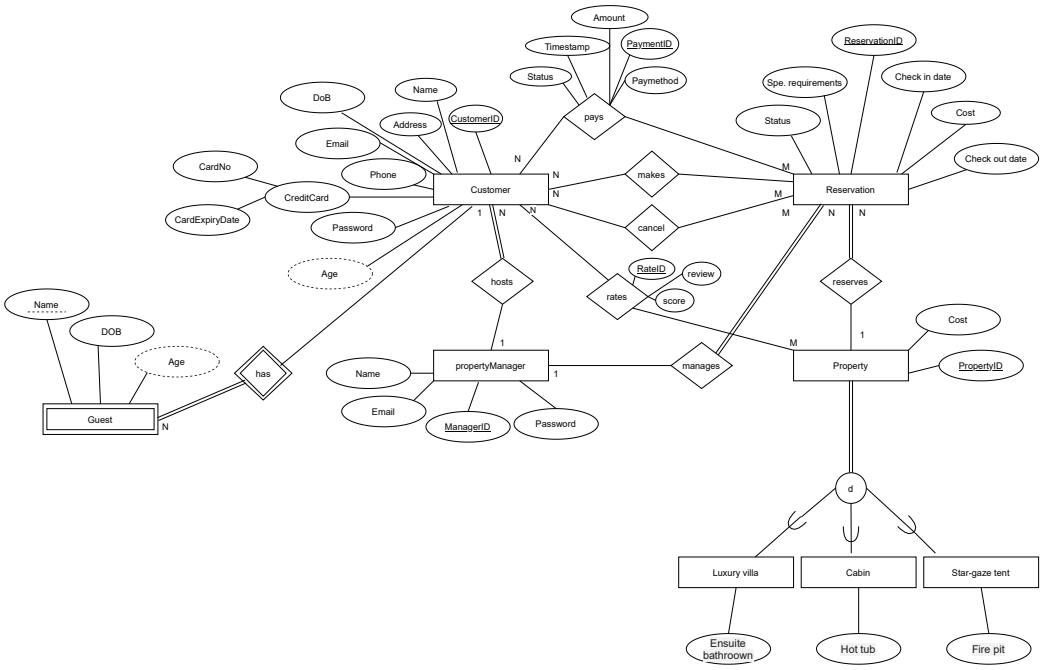

Figure 2: The Updated ER Diagram for the Online Property Booking System (OPBS)

- Adding the 'review' attribute to the Rates table to enable customers to comment on the quality of their experience at a specific property.

3 The Schema

- Customer(CustomerID:integer, Name:varchar, Phone:varchar, Email:varchar, Password:varchar, Address:varchar, CreditCardNo:varchar, CreditCard-Expiry:varchar, **ManagerID:integer**)
- Guest(**CustomerID:integer**, Name:varchar, DoB:date)
- Reservation(ReservationID:integer, Check in date:date, Check out date:date, SpecialRequirements:varchar, Status:varchar, Cost:integer, **PropertyID:integer**, **ManagerID:integer**)
- Property(PropertyID:integer, Cost:integer)

- LuxuryVilla(**PropertyID:integer**, Ensuitebathroown:varchar)
- Cabin(**PropertyID:integer**, Hottub:varchar)
- StarGazeTent(**PropertyID:integer**, Firepit:varchar)
- PropertyManager(ManagerID:integer, Name:varchar, Password:varchar, Email:varchar)
- Rates(RateID:integer, **PropertyID:integer**, **CustomerID:integer**, Score:integer, Review:varchar)
- MakesReservation(**CustomerID:integer**, **ReservationID:integer**)
- CancelReservation(**CustomerID:integer**, **ReservationID:integer**)
- Pays(PaymentID:integer, **CustomerID:integer**, **ReservationID:integer**, PaymentMethod:varchar, Amount:varchar, Timestamp:Timestamp, Status:varchar)

Foreign keys::

- Customer.ManagerID references PropertyManager.ManagerID
- Guest.CustomerID references Customer.CustomerID
- LuxuryVilla.PropertyID references Property.PropertyID
- Cabin.PropertyID references Property.PropertyID
- StarGazeTent.PropertyID references Property.PropertyID
- Reservation.PropertyID references Property.PropertyID
- Reservation.ManagerID references PropertyManager.ManagerID
- Rates.PropertyID references Property.PropertyID
- Rates.CustomerID references Customer.CustomerID
- MakesReservation.CustomerID references Customer.CustomerID
- MakesReservation.ReservationID references Reservation.ReservationID

- CancelReservation.CustomerID references Customer.CustomerID
- CancelReservation.ReservationID references Reservation.ReservationID
- Pays.CustomerID references Customer.CustomerID
- Pays.ReservationID references Reservation.ReservationID

The following are the checks for the schema:

- Pays.PaymentMethod in ('Cash','CreditCard')
- Rates.Score between 1 and 5

4 List of Functional Dependencies

The following is the list of functional dependencies for the previous schema:

- Relation: Customer
  - CustomerID → Name, Email, Password, Address, Phone, CreditCardNo, CreditCardExpiry
- Relation: Guest
  - CustomerID, Name → DoB
- Relation: Property
  - PropertyID → Cost
- Relation: Reservation
  - ReservationID → Cost, Check in date, Check out date, Status, SpecialRequirements
- Relation: PropertyManager
  - ManagerID → Name, Email, Password
- Relation: Pays

- PaymentID, ReservationID, CustomerID → Amount, PaymentMethod, Timestamp, Status

- Relation: Rates
  - RateID, CustomerID, PropertyID → Score, Review

5 The Normalised Schema

- Customer(CustomerID:integer, Name:varchar, Phone:varchar, Email:varchar, Password:varchar, Address:varchar, CreditCardNo:varchar, CreditCardExpiry:varchar, **ManagerID:integer**)
- Guest(**CustomerID:integer**, Name:varchar, DoB:date)
- Reservation(ReservationID:integer, Check in date:date, Check out date:date, SpecialRequirements:varchar, Status:varchar, Cost:integer, **PropertyID:integer**, **ManagerID:integer**)
- Property(PropertyID:integer, Cost:integer)
- LuxuryVilla(**PropertyID:integer**, Ensuitebathroown:varchar)
- Cabin(**PropertyID:integer**, Hottub:varchar)
- StarGazeTent(**PropertyID:integer**, Firepit:varchar)
- PropertyManager(ManagerID:integer, Name:varchar, Password:varchar, Email:varchar)
- Rates(RateID:integer, **PropertyID:integer**, **CustomerID:integer**, Score:integer, Review:varchar)
- MakesReservation(**CustomerID:integer**, **ReservationID:integer**)
- CancelReservation(**CustomerID:integer**, **ReservationID:integer**)
- Pays(PaymentID:integer, **CustomerID:integer**, **ReservationID:integer**, PaymentMethod:varchar,Amount:varchar, Timestamp:Timestamp, Status:varchar)

The following shows the foreign keys:

- Customer.ManagerID references PropertyManager.ManagerID
- Guest.CustomerID references Customer.CustomerID
- LuxuryVilla.PropertyID references Property.PropertyID
- Cabin.PropertyID references Property.PropertyID
- StarGazeTent.PropertyID references Property.PropertyID
- Rates.PropertyID references Property.PropertyID
- Rates.CustomerID references Customer.CustomerID
- Reservation.PropertyID references Property.PropertyID
- Reservation.ManagerID references PropertyManager.ManagerID
- MakesReservation.CustomerID references Customer.CustomerID
- MakesReservation.ReservationID references Reservation.ReservationID
- CancelReservation.CustomerID references Customer.CustomerID
- CancelReservation.ReservationID references Reservation.ReservationID
- Pays.CustomerID references Customer.CustomerID
- Pays.ReservationID references Reservation.ReservationID

6 The SQL Dump

Tables created and populated as follows:

```sql
CREATE TABLE `Cabin` (
  `PropertyID` int(11) NOT NULL,
  `Hottub` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `Cabin`
--

INSERT INTO `Cabin` (`PropertyID`, `Hottub`) VALUES
(2, NULL);

-- --------------------------------------------------------

--
-- Table structure for table `CancelReservation`
--

CREATE TABLE `CancelReservation` (
  `CustomerID` int(11) NOT NULL,
  `ReservationID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `CancelReservation`
--

INSERT INTO `CancelReservation` (`CustomerID`, `ReservationID`) VALUES
(1, 2),
(2, 3),
(3, 3),
(4, 3),
(5, 2);
```

Figure 3: Cabin and CancelReservation tables.

```
--
-- Table structure for table `Customer`
--

CREATE TABLE `Customer` (
  `CustomerID` int(11) NOT NULL,
  `Name` varchar(30) DEFAULT NULL,
  `Phone` varchar(13) DEFAULT NULL,
  `Email` varchar(30) DEFAULT NULL,
  `Password` varchar(30) DEFAULT NULL,
  `Address` varchar(50) DEFAULT NULL,
  `CreditCardNo` varchar(50) DEFAULT NULL,
  `CreditCardExpiry` varchar(50) DEFAULT NULL,
  `ManagerID` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `Customer`
--

INSERT INTO `Customer` (`CustomerID`, `Name`, `Phone`, `Email`, `Password`, `Address`, `CreditCardNo`, `CreditCardExpiry`, `ManagerID`) VALUES
(1, 'Tom B. Erichsen', '0432499200', 'e.b.e@gmail.com', 'abc1122', '1 Stavanger Street Norway', '1234567', '01/01/2025', 1),
(2, 'Ricky Martin', '0432499300', 'e.b.i@gmail.com', 'abc1142', '99 Roundabout Av', '7654321', '01/05/2024', 1),
(3, 'David Jones', '0432499400', 'e.b.d@gmail.com', 'abc1332', '21 Unicorn Way', '1357975', '05/01/2027', 1),
(4, 'Tom B. Erichsen', '0432499500', 'e.b.k@gmail.com', 'abc1632', '68 Montgomery Road', '2468086', '04/01/2023', 1),
(5, 'Tom B. Erichsen', '0432499600', 'e.s.e@gmail.com', 'abc1302', 'Pani Dam View', '12093498', '01/06/2024', 1);

-- --------------------------------------------------------
```

Figure 4: Customer table.

```sql
--
-- Table structure for table `Guest`
--

CREATE TABLE `Guest` (
  `CustomerID` int(11) NOT NULL,
  `Name` varchar(30) NOT NULL,
  `DoB` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `Guest`
--

INSERT INTO `Guest` (`CustomerID`, `Name`, `DoB`) VALUES
(1, 'Bobby', '1977-06-03'),
(2, 'Fisher', '1976-03-06'),
(3, 'Kristov', '1999-09-01'),
(4, 'Sean', '1996-01-02'),
(5, 'Leonard', '1986-12-01');

-- --------------------------------------------------------

--
-- Table structure for table `LuxuryVilla`
--

CREATE TABLE `LuxuryVilla` (
  `PropertyID` int(11) NOT NULL,
  `Ensuitebathroown` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `LuxuryVilla`
--

INSERT INTO `LuxuryVilla` (`PropertyID`, `Ensuitebathroown`) VALUES
(1, NULL);

-- --------------------------------------------------------
```

Figure 5: LuxuryVilla and Guest tables.

```sql
--
-- Table structure for table `MakesReservation`
--

CREATE TABLE `MakesReservation` (
  `CustomerID` int(11) NOT NULL,
  `ReservationID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `MakesReservation`
--

INSERT INTO `MakesReservation` (`CustomerID`, `ReservationID`) VALUES
(1, 2),
(2, 3),
(3, 3),
(4, 3),
(5, 2);

-- --------------------------------------------------------


--
-- Table structure for table `Pays`
--

CREATE TABLE `Pays` (
  `PaymentID` int(11) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `ReservationID` int(11) NOT NULL,
  `PaymentMethod` varchar(20) DEFAULT NULL CHECK (`PaymentMethod` in ('Cash','CreditCard')),
  `Amount` int(11) DEFAULT NULL,
  `Timestamp1` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `Status1` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `Pays`
--

INSERT INTO `Pays` (`PaymentID`, `CustomerID`, `ReservationID`, `PaymentMethod`, `Amount`, `Timestamp1`, `Status1`) VALUES
(1, 1, 1, 'CreditCard', 100, '2021-04-13 08:36:34', 'Confirmed'),
(2, 1, 2, 'Cash', 200, '2021-04-13 08:36:34', 'Confirmed'),
(3, 2, 3, 'CreditCard', 120, '2021-04-13 08:36:34', 'Confirmed'),
(4, 3, 4, 'Cash', 120, '2021-04-13 08:36:34', 'Confirmed'),
(5, 4, 5, 'CreditCard', 100, '2021-04-13 08:36:34', 'Confirmed');

-- --------------------------------------------------------
```

Figure 6: MakesReservation and Pays tables.

```sql
--
-- Table structure for table `Property`
--

CREATE TABLE `Property` (
  `PropertyID` int(11) NOT NULL,
  `Cost` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `Property`
--

INSERT INTO `Property` (`PropertyID`, `Cost`) VALUES
(1, 200),
(2, 120),
(3, 100);

-- --------------------------------------------------------

--
-- Table structure for table `PropertyManager`
--

CREATE TABLE `PropertyManager` (
  `ManagerID` int(11) NOT NULL,
  `Name` varchar(30) DEFAULT NULL,
  `Password` varchar(30) DEFAULT NULL,
  `Email` varchar(30) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `PropertyManager`
--

INSERT INTO `PropertyManager` (`ManagerID`, `Name`, `Password`, `Email`) VALUES
(1, 'admin', 'abc112233', 'admin@gmail.com');

-- --------------------------------------------------------
```

Figure 7: PropertyManager and Property tables.

```sql
-- Table structure for table `Rates`
--

CREATE TABLE `Rates` (
  `RateID` int(11) NOT NULL,
  `PropertyID` int(11) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `Score` int(11) DEFAULT NULL CHECK (`Score` between 0 and 5)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `Rates`
--

INSERT INTO `Rates` (`RateID`, `PropertyID`, `CustomerID`, `Score`) VALUES
(1, 1, 1, 4),
(2, 2, 2, 5),
(3, 3, 3, 3),
(4, 1, 4, 2),
(5, 3, 4, 5);

-- --------------------------------------------------------

--
-- Table structure for table `Reservation`
--

CREATE TABLE `Reservation` (
  `ReservationID` int(11) NOT NULL,
  `PropertyID` int(11) DEFAULT NULL,
  `ManagerID` int(11) DEFAULT NULL,
  `CheckInDate` date DEFAULT NULL,
  `CheckOutDate` date DEFAULT NULL,
  `SpecialRequirements` varchar(100) DEFAULT NULL,
  `Status` varchar(30) DEFAULT NULL,
  `Cost` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `Reservation`
--

INSERT INTO `Reservation` (`ReservationID`, `PropertyID`, `CheckInDate`, `CheckOutDate`, `SpecialRequirements`, `Status`, `Cost`) VALUES
(1, 1, '2021-05-20', '2021-05-21', 'WIFI', 'pending', 200),
(2, 1, '2021-05-22', '2021-05-23', 'WIFI', 'pending', 200),
(3, 2, '2021-05-24', '2021-05-26', 'queen sized bed', 'confirmed', 120),
(4, 3, '2021-05-27', '2021-05-30', 'WIFI', 'pending', 100),
(5, 2, '2021-06-10', '2021-06-11', 'WIFI', 'pending', 120);
```

Figure 8: Rates and Reservation tables.

```sql
--
-- Table structure for table `StarGazeTent`
--

CREATE TABLE `StarGazeTent` (
  `PropertyID` int(11) NOT NULL,
  `Firepit` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `StarGazeTent`
--

INSERT INTO `StarGazeTent` (`PropertyID`, `Firepit`) VALUES
(3, NULL);

--
-- Indexes for dumped tables
--

--
-- Indexes for table `Cabin`
--
ALTER TABLE `Cabin`
  ADD PRIMARY KEY (`PropertyID`),
  ADD KEY `PropertyID` (`PropertyID`);

--
-- Indexes for table `CancelReservation`
--
ALTER TABLE `CancelReservation`
  ADD PRIMARY KEY (`CustomerID`,`ReservationID`),
  ADD KEY `ReservationID` (`ReservationID`),
  ADD KEY `CustomerID` (`CustomerID`);

--
-- Indexes for table `Customer`
--
ALTER TABLE `Customer`
  ADD PRIMARY KEY (`CustomerID`),
  ADD KEY `ManagerID` (`ManagerID`);

--
-- Indexes for table `Guest`
--
ALTER TABLE `Guest`
  ADD PRIMARY KEY (`CustomerID`,`Name`),
  ADD KEY `CustomerID` (`CustomerID`);
```

Figure 9: Indexes for the created tables.

```sql
--
-- Indexes for table `LuxuryVilla`
--
ALTER TABLE `LuxuryVilla`
  ADD PRIMARY KEY (`PropertyID`),
  ADD KEY `PropertyID` (`PropertyID`);

--
-- Indexes for table `MakesReservation`
--
ALTER TABLE `MakesReservation`
  ADD PRIMARY KEY (`CustomerID`,`ReservationID`),
  ADD KEY `ReservationID` (`ReservationID`),
  ADD KEY `CustomerID` (`CustomerID`);

--
-- Indexes for table `Pays`
--
ALTER TABLE `Pays`
  ADD PRIMARY KEY (`PaymentID`,`CustomerID`,`ReservationID`),
  ADD KEY `CustomerID` (`CustomerID`),
  ADD KEY `ReservationID` (`ReservationID`);

--
-- Indexes for table `Property`
--
ALTER TABLE `Property`
  ADD PRIMARY KEY (`PropertyID`);

--
-- Indexes for table `PropertyManager`
--
ALTER TABLE `PropertyManager`
  ADD PRIMARY KEY (`ManagerID`);

--
-- Indexes for table `Rates`
--
ALTER TABLE `Rates`
  ADD PRIMARY KEY (`RateID`,`PropertyID`,`CustomerID`),
  ADD KEY `CustomerID` (`CustomerID`),
  ADD KEY `PropertyID` (`PropertyID`);
```

Figure 10: Constraints for the created tables.

```sql
--
-- Indexes for table `Reservation`
--
ALTER TABLE `Reservation`
  ADD PRIMARY KEY (`ReservationID`),
  ADD KEY `PropertyID` (`PropertyID`);
  ADD KEY `ManagerID` (`ManagerID`);

--
-- Indexes for table `StarGazeTent`
--
ALTER TABLE `StarGazeTent`
  ADD PRIMARY KEY (`PropertyID`),
  ADD KEY `PropertyID` (`PropertyID`);

--
-- Constraints for dumped tables
--


--
-- Constraints for table `Cabin`
--
ALTER TABLE `Cabin`
  ADD CONSTRAINT `cabin_ibfk_1` FOREIGN KEY (`PropertyID`) REFERENCES `Property` (`PropertyID`);

--
-- Constraints for table `CancelReservation`
--
ALTER TABLE `CancelReservation`
  ADD CONSTRAINT `cancelreservation_ibfk_1` FOREIGN KEY (`CustomerID`) REFERENCES `Customer` (`CustomerID`);
  ADD CONSTRAINT `cancelreservation_ibfk_2` FOREIGN KEY (`ReservationID`) REFERENCES `Reservation` (`ReservationID`);

--
-- Constraints for table `Customer`
--
ALTER TABLE `Customer`
  ADD CONSTRAINT `customer_ibfk_1` FOREIGN KEY (`ManagerID`) REFERENCES `PropertyManager` (`ManagerID`);

--
-- Constraints for table `Guest`
--
ALTER TABLE `Guest`
  ADD CONSTRAINT `guest_ibfk_1` FOREIGN KEY (`CustomerID`) REFERENCES `Customer` (`CustomerID`) ON DELETE CASCADE;
```

Figure 11: Constraints for the created tables.

```sql
--
-- Constraints for table `LuxuryVilla`
--
ALTER TABLE `LuxuryVilla`
  ADD CONSTRAINT `luxuryvilla_ibfk_1` FOREIGN KEY (`PropertyID`) REFERENCES `Property` (`PropertyID`);

--
-- Constraints for table `MakesReservation`
--
ALTER TABLE `MakesReservation`
  ADD CONSTRAINT `makesreservation_ibfk_1` FOREIGN KEY (`CustomerID`) REFERENCES `Customer` (`CustomerID`);
  ADD CONSTRAINT `makesreservation_ibfk_2` FOREIGN KEY (`ReservationID`) REFERENCES `Reservation` (`ReservationID`);

--
-- Constraints for table `Pays`
--
ALTER TABLE `Pays`
  ADD CONSTRAINT `pays_ibfk_1` FOREIGN KEY (`CustomerID`) REFERENCES `Customer` (`CustomerID`);
  ADD CONSTRAINT `pays_ibfk_2` FOREIGN KEY (`ReservationID`) REFERENCES `Reservation` (`ReservationID`);

--
-- Constraints for table `Rates`
--
ALTER TABLE `Rates`
  ADD CONSTRAINT `rates_ibfk_1` FOREIGN KEY (`CustomerID`) REFERENCES `Customer` (`CustomerID`),
  ADD CONSTRAINT `rates_ibfk_2` FOREIGN KEY (`PropertyID`) REFERENCES `Property` (`PropertyID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `Reservation`
--
ALTER TABLE `Reservation`
  ADD CONSTRAINT `reservation_ibfk_1` FOREIGN KEY (`PropertyID`) REFERENCES `Property` (`PropertyID`);
  ADD CONSTRAINT `reservation_ibfk_2` FOREIGN KEY (`ManagerID`) REFERENCES `PropertyManager` (`ManagerID`);

--
-- Constraints for table `StarGazeTent`
--
ALTER TABLE `StarGazeTent`
  ADD CONSTRAINT `stargazetent_ibfk_1` FOREIGN KEY (`PropertyID`) REFERENCES `Property` (`PropertyID`);
COMMIT;
```

Figure 12: Constraints for the created tables.

7 Screenshots of the Platform

To book a property, a user is required to create an account on the website. The account contains their details, such as; an email address and password, name, phone number, address, and date of birth (NOTE: The user can create an account with a given email address only once). This can be seen in Figure 13 below. Once the user has created an account and logged in successfully (see Figure 14 and Figure 15), they are presented with pictures of various properties that are offered, they can access a contact page which contains management contact details (see Figure 15 and Figure 16) and they can reserve a selected property. The property manager can login and can monitor the list of customers and their reservation status. This can be seen in Figure 17 and Figure 18, where the manager can update existing customers details or delete customers from the list.

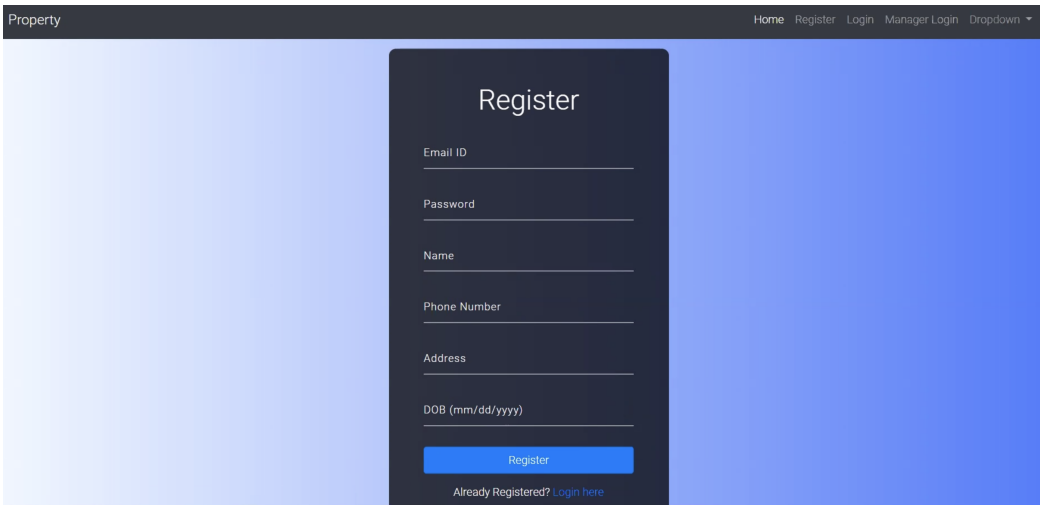

Figure 13: Customer registration page.

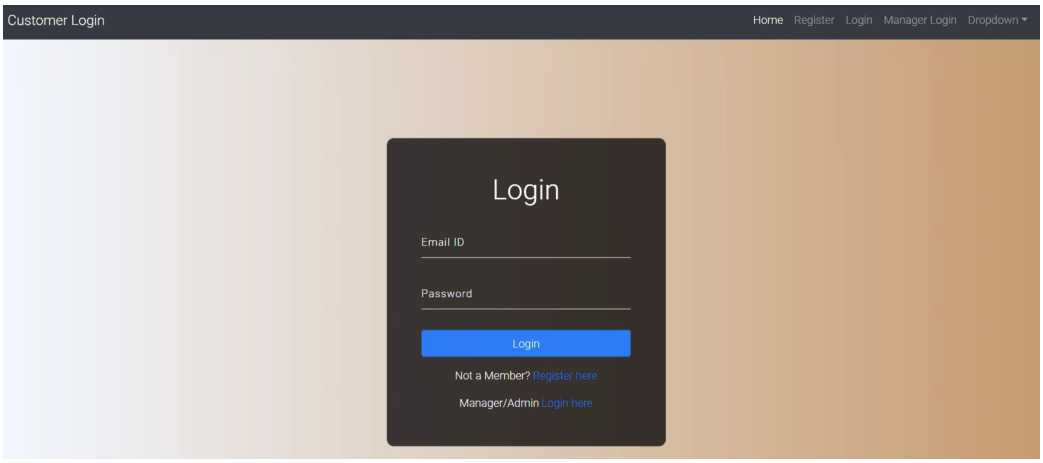

Figure 14: Customer login page.

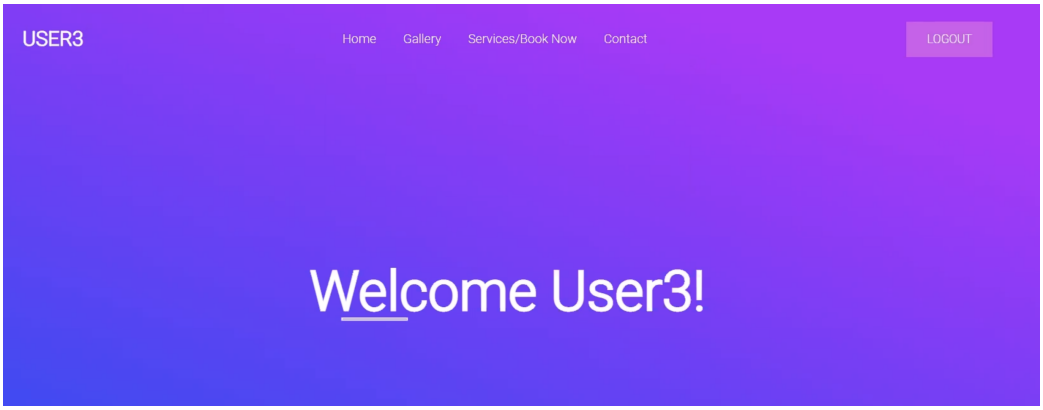

Figure 15: Customer successful login page.

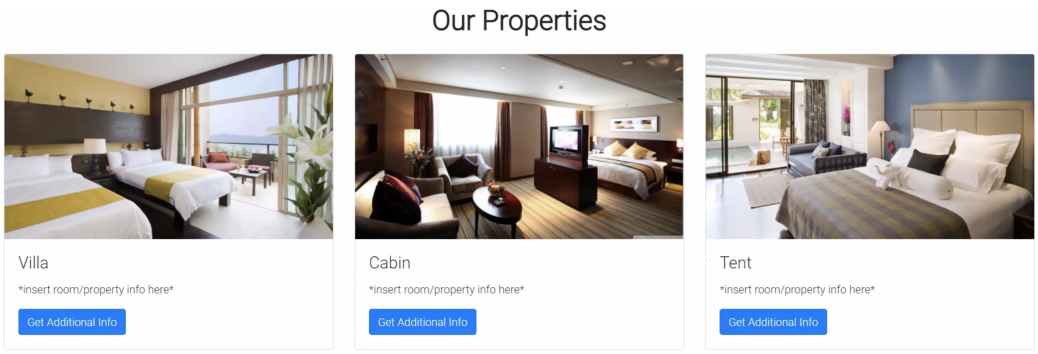

Figure 16: List of proprieties page.

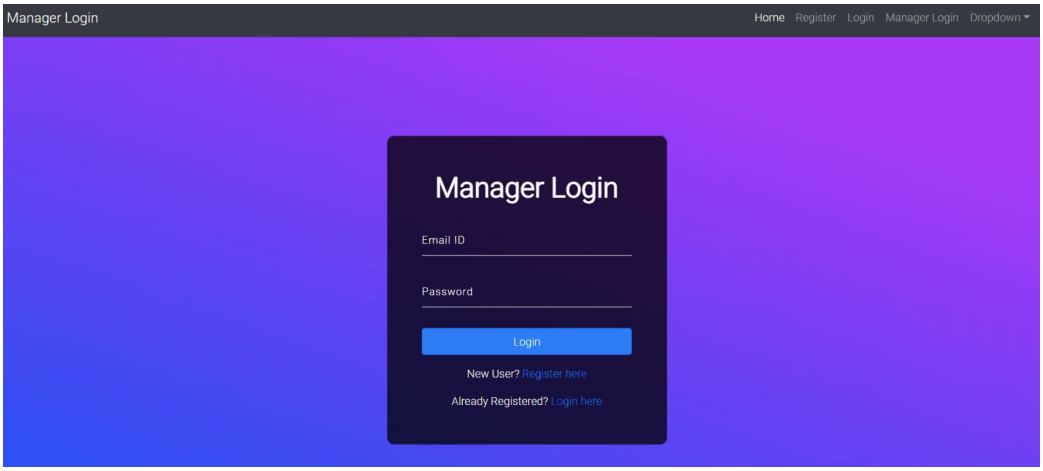

Figure 17: Property Manager login page.

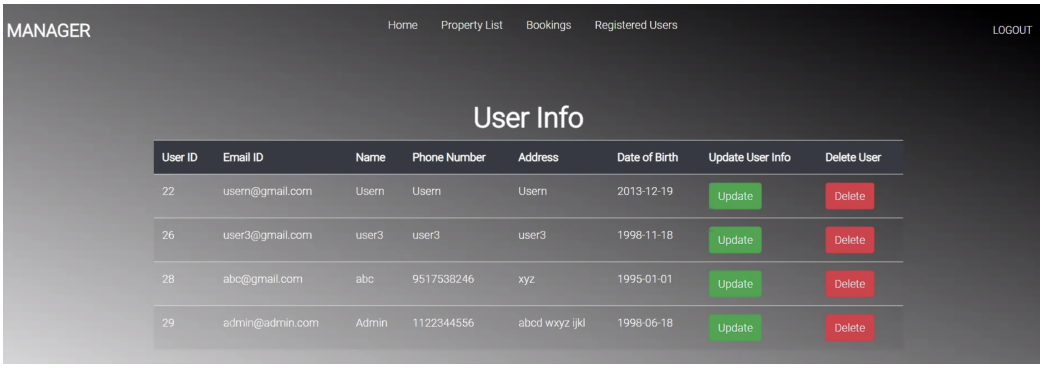

Figure 18: Registered Customers as seen by the Property Manager.

7.1 Some of the executed queries

```php
$email = mysqli_real_escape_string($conn, $_POST['email']);
$password = mysqli_real_escape_string($conn, $_POST['password']);
$name = mysqli_real_escape_string($conn, $_POST['name']);
$number = mysqli_real_escape_string($conn, $_POST['number']);
$address = mysqli_real_escape_string($conn, $_POST['address']);

$dob = date('Y-m-d', strtotime($_POST['date']));


$pass = password_hash($password, PASSWORD_BCRYPT);


$emailquery = "select * from registerinfo where email = '$email'";
$query = mysqli_query($conn, $emailquery);


$emailcount = mysqli_num_rows($query);

if($emailcount>0){

  echo "Email already exists";

} else {


    $insertquery = "insert into registerinfo (email, password, name, phone, address, dob)
    values ('$email','$pass','$name','$number','$address','$dob')";

    $iquery = mysqli_query($conn, $insertquery);
```

Figure 19: Snapshot of SQL code for insert query.

```php
<?php

include 'dbcon.php';
if(isset($_POST['done'])){




    $id = $_GET['id'];
    $email = $_POST['email'];
    $name = $_POST['name'];
    $pass = $_POST['password'];
    $number = $_POST['phone'];
    $address = $_POST['address'];
    $dob = date('Y-m-d', strtotime($_POST['dob']));


     $insertquery = " update registerinfo set id = $id, email = '$email', password = '$pass', name =  '$name', phone = '$number
         ', address = '$address', dob = '$dob' where id = $id";

    $query = mysqli_query($conn, $insertquery);

    header('location: users.php');

}

?>
```

Figure 20: Snapshot of SQL code for update query.

```php
<?php

$server = "localhost";
$user = "root";
$password = "";

$db = "registration";


$conn = mysqli_connect($server, $user, $password, $db);

if($conn){

}else {

    ?>
    <script>
            alert("No Existing Connection!");
    </script>
    <?php
}

$id = $_GET['id'];
$q = "DELETE FROM registerinfo where id = $id";

mysqli_query($conn, $q);

header('location:users.php');

?>
```

Figure 21: Snapshot of SQL code for delete query.

```php
<?php


session_start();
if(!isset($_SESSION['name'])){

    header('location:managerlogin.php');
}


$server = "localhost";
$user = "root";
$password = "";

$db = "registration";


$conn = mysqli_connect($server, $user, $password, $db);

if($conn){

}else {

    ?>
    <script>
            alert("No Connection!");
    </script>
    <?php
}


$q = "select * from registerinfo";
$query = mysqli_query($conn, $q);
```

Figure 22: Snapshot of SQL code for select query.