Project Report

Title: Hotel Booking System

Website: https://www.booking.com/  
Project Members

Alex Babenko

Orion

Pujanpreet Kaur

Documenting the USE CASE

A diagram of a flowchart

Description automatically generated

The use-case diagram depicts how a user can book a stay at their desired hotel via booking.com. When the user opens the website, they must select their country/region and choose currency in which they would pay for their stay. After that they enter the destination at which they want to book a hotel. In the filter search, they can specify the duration of their stay, number of rooms they want to book, number of guests staying etc. The user can select the hotels that they like based on various factors listed in the property specifications (such as hotel rating, price range, amenities included etc.). Then the system shows the price and types of rooms available for the user to book from. The user must select at least one room to proceed with the reservation. The user provides the necessary booking details such as guest name, contact information etc. The system prompts the user to select the payment options which users can choose to pay for their stay. The users have option to pay at the property or pay online using credit/visa/MasterCard or apple/google pay. When the payment is done, the system sends the booking confirmation to the user email they provided. As a base for it was taken an API documentation of booking.com https://connect.booking.com/user\_guide/site/en-US/user\_guide.html

Gathering requirements

The requirements for the system to book hotels through booking.com are as follows:

The system must be able to keep booking information up to date on availability of rooms, new additions of rooms, hotels, properties etc. The properties listed on the website are created with contact details, fee policies and other information. The properties need to have an inventory of rooms to sell to guests. The room information must also include the details like room’s size, smoking or non-smoking and what amenities are included in the base price for the room. The combination of room, rate and policies for fees, cancellation, stay etc., are what the customer buys on the website booking.com. This is known as the product. These listings by hotels are then checked and verified by the system. Once verified, the customer can see the listings of the hotels on the website. The system then stores the listings of the hotel with the property information. The system must also provide a way for the users to select their desired location and specified dates and number of nights they want to stay based on the property information such as location, city name, postal code, country code. When the property has listed all the rooms available to sell to guests, the customer will see various room types and prices. Customers can then select the room type and number of rooms they want to book. Then the system is required to store the details of the customers that they entered which can then be used further for cancellation, access to booking in future etc. The system can use relational databases to store the properties information and the customers’ information. It must be able to support a reliable and scalable DBMS system such as MySQL, Oracle, PostgreSQL, etc. The details of booking transactions, accommodations booked, payment methods, amount, refunds issued, and cancellations processed for booking are needed to be stored in the system database as well.

Defining Entities

We have five entities for the booking system. They are customer, booking, payment, room, and hotel. Here is the detailed version of entities with their attributes:

CUSTOMER(Customer\_Id, f\_name, l\_name, email, phone, contact\_type)

BOOKING(booki\_id, *prod\_id, pay\_id,* date\_from, date\_to, total\_price, num\_of\_guests)

PAYMENT(Payment\_ID, *Customer\_ID,* pay\_method, date, amount, status)

PRODUCT(Product\_id, *hotel\_id, room\_id, rate\_id)*

RATEPLAN(Rate\_id, room\_nights, RN%)

ROOM(Room ID, Room number, room type, room size, max occupancy, availability, description)

AMENITIES(*Room number,* Amenity code)

HOTEL(Hotel id, *Legal entity id,* hotel name, country, city, state, Zip code, Street, Status, Currency code, Guest Room Quantity)

OONERSHIP(legal entity id, First name, last name, Email, Phone, Contact Profile type)

Conceptual Design

A diagram of a diagram

Description automatically generated with medium confidence

**Logical Design**

A diagram of a company

Description automatically generated

**Physical Design**

A diagram of a computer system

Description automatically generated with medium confidence

**SQL (Creating, Inserting, Updating, Deleting and Queries)**

SQL TABLES

CREATE TABLE Customer (

  cust\_id *INTEGER* PRIMARY KEY,

  f\_name *TEXT*,

  l\_name *TEXT*,

  email *TEXT*,

  phone *INTEGER*,

  contact\_type *TEXT*

);

CREATE TABLE PaymentInfo (

  pay\_id *INTEGER* PRIMARY KEY,

  cust\_id *INTEGER*,

  pay\_method *TEXT*,

  pay\_date *DATE*,

  amount *INTEGER*,

  status *TEXT*,

  FOREIGN KEY (cust\_id) REFERENCES Customer(cust\_id)

);

CREATE TABLE Booking (

  booki\_id *INTEGER* PRIMARY KEY,

  prod\_id *INTEGER*,

  pay\_id *INTEGER*,

  date\_from *DATE*,

  date\_to *DATE*,

  totall\_price *INTEGER*,

  num\_of\_guests *INTEGER*,

  FOREIGN KEY (prod\_id) REFERENCES Product(prod\_id),

  FOREIGN KEY (pay\_id) REFERENCES PaymentInfo(pay\_id)

);

CREATE TABLE Room (

  room\_id *INTEGER* PRIMARY KEY,

  room\_num *INTEGER*,

  room\_type *TEXT*,

  room\_size *INTEGER*,

  max\_occup *INTEGER*,

  availability *TEXT*,

  description *TEXT*

);

CREATE TABLE Amenities (

  room\_id *INTEGER*,

  Amen\_code *INTEGER*,

  FOREIGN KEY (room\_id) REFERENCES Room(room\_id)

);

CREATE TABLE Hotel (

  hotel\_id *INTEGER* PRIMARY KEY,

  legal\_entity\_id *INTEGER*,

  hotel\_name *TEXT*,

  country *TEXT*,

  city *TEXT*,

  state *TEXT*,

  ZIP\_code *INTEGER*,

  street\_addr *TEXT*,

  status *TEXT*,

  currency\_code *INTEGER*,

  room\_quantity *INTEGER*,

  FOREIGN KEY (legal\_entity\_id) REFERENCES ManagementInfo(legal\_entity\_id)

);

CREATE TABLE ManagementInfo (

  legal\_entity\_id *INTEGER* PRIMARY KEY,

  f\_name *TEXT*,

  l\_name *TEXT*,

  email *TEXT*,

  phone *INTEGER*,

  contact\_type *TEXT*

);

CREATE TABLE Product (

  prod\_id *INTEGER* PRIMARY KEY,

  hotel\_id *INTEGER*,

  room\_id *INTEGER*,

  rate\_id *INTEGER*,

  FOREIGN KEY (hotel\_id) REFERENCES Hotel(hotel\_id),

  FOREIGN KEY (room\_id) REFERENCES Room(room\_id),

  FOREIGN KEY (rate\_id) REFERENCES RatePlan(rate\_id)

);

CREATE TABLE RatePlan (

  rate\_id *INTEGER*,

  room\_nights *INTEGER*,

  RN *INTEGER*

);

SQL INSERT

-- Populate Customer table

INSERT INTO Customer (f\_name, l\_name, email, phone, contact\_type)

VALUES ('John', 'Doe', 'john@example.com', 1234567890, 'email'),

       ('Jane', 'Smith', 'jane@example.com', 9876543210, 'phone');

-- Populate ManagementInfo table

INSERT INTO ManagementInfo (f\_name, l\_name, email, phone, contact\_type)

VALUES ('Manager', 'One', 'manager1@example.com', 1234567890, 'email'),

       ('Manager', 'Two', 'manager2@example.com', 9876543210, 'phone');

-- Populate Hotel table

INSERT INTO Hotel (legal\_entity\_id, hotel\_name, country, city, state, ZIP\_code, street\_addr, status, currency\_code, room\_quantity)

VALUES (1, 'Example Hotel', 'USA', 'New York', 'NY', 10001, '123 Main St', 'active', 1, 50),

       (2, 'Test Hotel', 'Canada', 'Toronto', 'ON', 'M5V 2V4', '456 Maple Ave', 'active', 2, 100);

-- Populate Room table

INSERT INTO Room (room\_num, room\_type, room\_size, max\_occup, availability, description)

VALUES (101, 'Standard', 250, 2, 'available', 'Standard room with basic amenities'),

       (201, 'Suite', 500, 4, 'available', 'Luxurious suite with a view');

-- Populate Amenities table

INSERT INTO Amenities (room\_id, Amen\_code)

VALUES (1, 101),

       (2, 201);

-- Populate PaymentInfo table

INSERT INTO PaymentInfo (cust\_id, pay\_method, pay\_date, amount, status)

VALUES (1, 'Credit Card', '2024-02-25', 200, 'paid'),

       (2, 'PayPal', '2024-02-26', 300, 'paid');

-- Populate Product table

INSERT INTO Product (hotel\_id, room\_id, rate\_id)

VALUES (1, 1, 1),

       (2, 2, 2);

-- Populate RatePlan table

INSERT INTO RatePlan (rate\_id, room\_nights, RN)

VALUES (1, 1, 100),

       (2, 2, 200);

-- Populate Booking table

INSERT INTO Booking (prod\_id, pay\_id, date\_from, date\_to, totall\_price, num\_of\_guests)

VALUES (1, 1, '2024-03-01', '2024-03-05', 1000, 2),

       (2, 2, '2024-04-01', '2024-04-10', 2000, 4);

SQL DELETE

DELETE FROM PaymentInfo

WHERE pay\_method = 'PayPal';

DELETE FROM RatePlan

WHERE RN >= 150;

DELETE FROM Customer

WHERE id = 2;

SQL UPDATE

UPDATE Customer

SET f\_name = Johan, l\_name = Libert

WHERE cust\_id = 1;

UPDATE Amenities

SET Amen\_code = 777

WHERE room\_id = 2;

UPDATE PaymentInfo

SET amount = 350, status = 'unpaid'

WHERE cust\_id = 2;

SQL QUERY

-- A select query that extracts all fields in a table.

SELECT \* FROM Customer;

-- A select query that extracts specific fields in a table.

SELECT f\_name, l\_name, email FROM Customer;

-- A select query that extracts fields from two or more tables and uses a condition to filter query results.

SELECT b.date\_from, b.date\_to, p.pay\_method, p.status, b.totall\_price

FROM Booking b

JOIN PaymentInfo p ON b.pay\_id  = p.pay\_id

WHERE b.totall\_price > 1500;

-- A select query that extracts fields from two or more tables and uses a pattern search as a filter.

SELECT r.room\_id, r.room\_type, r.description, a.amen\_code

FROM Room r

JOIN Amenities a ON r.room\_id = a.room\_id

WHERE r.description like '%Luxuri%' AND a.amen\_code = 201;

-- Two select queries that demonstrates the use of group by with aggregate functions

-- Example 1: Total amount of payments per payment method

SELECT pay\_method, SUM(amount) AS total\_amount

FROM PaymentInfo

GROUP BY pay\_method;

-- Example 2: Average number of guests per booking

SELECT AVG(num\_of\_guests) AS avg\_guests

FROM Booking;

-- A select query that demonstrates the use of group by with having

clause and sorts the data in a descending format.

SELECT booki\_id, date\_from, date\_to

FROM Booking

GROUP BY booki\_id

HAVING totall\_price >= 2

ORDER BY totall\_price DESC;

-- A select query that utilizes a subquery

SELECT f\_name, l\_name, email

FROM Customer

WHERE cust\_id IN (

    SELECT cust\_id

    FROM Booking

    JOIN Product ON Booking.prod\_id = Product.prod\_id

    JOIN Hotel ON Product.hotel\_id = Hotel.hotel\_id

    WHERE hotel\_name = 'Example Hotel'

);

-- h. Other queries discussed in the class

SELECT pay\_method FROM PaymentInfo

MINUS

SELECT pay\_method FROM PaymentInfo;

Screenshots of the application where queries were used.

1. Insert Query in add screen

A screenshot of a computer

Description automatically generated

2. Update Query in Edit Screen

A computer screen shot of a black screen

Description automatically generated

3. Delete Query In Delete Screen

A computer screen with a black background

Description automatically generated

4. Select Query in Input Search Screen

A screenshot of a computer

Description automatically generated

5. Select Query in Output search screen

A screenshot of a computer

Description automatically generated

Code snippets of the application where the queries were utilized.

1. Insert Query in add option

A computer screen with text on it

Description automatically generated

2. Update query in edit option

A computer screen shot of code

Description automatically generated

3. Delete Query in delete option

A building with many windows at night

Description automatically generated

4. Select query to get column name options in dropdown list

A building with many windows

Description automatically generated

5. Select query to get table name grid display

A screen shot of a computer

Description automatically generated

6. Select query to show output in select option

A computer screen shot of code

Description automatically generated