Hersh Patel
Saaketh Krosuri
Tanvi Parikh
Madison Ling

# Hulton - Online Hotel Reservation System

---

***Background:***

We used the following YouTube series to build our website ground up in Python / HTML, using the Flask / Jinja frameworks:

- Step 1 (Setup / Getting Started):
  - https://www.youtube.com/watch?v=zRwy8gtgJ1A&t=
- Step 2 (User Registration):
  - https://www.youtube.com/watch?v=addnlzdSQs4
- Step 3 (Login & Access Control):
  - https://www.youtube.com/watch?v=QEMtSUxtUDY
- Step 4 (Dashboard):
  - https://www.youtube.com/watch?v=EgnyWxKFwjs

The website itself currently runs locally, as we did not transfer it to AWS.

On the other hand, our database currently is hosted on Amazon Web Services in the RDS Platform. We used MySQLWorkbench to create an instance and connect to the AWS servers in order to create tables and populate them with data.

---

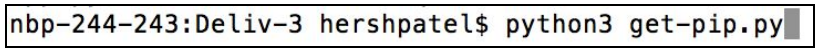
***Team Responsibilities:***

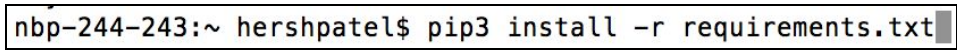
- Hersh Patel
  - Overall, Hersh did most of the front-end website design, including setting up the flask framework and collecting user-entered data. The reason for this is because he had the most knowledge about website development and has been doing it for many years. Hersh also detailed out the functionalities needed on the website so that his group members could setup and test the database, and write the queries.
- Saaketh Krosuri
  - Saaketh worked with Hersh to detail out the SQL Queries that needed to be run in order to display and collect the right content on the website. He also worked with Tanvi to write all of the queries for the website's functionalities, namely the reservations and the reviews. Finally, he also produced the queries to collect information as necessary for the customer analytics page, and the queries to populate and clear database information.

- Tanvi Parikh
    - Tanvi set up the database on AWS and ensured that all primary and foreign keys existed, along with implementing other functionalities such as auto_incrementation. She also worked with Saaketh to write queries for the website, namely with reservations and analytics. She also tested inserting data into the various databases to see if everything was implemented correctly, and worked with Madison on the documentation.
- Madison Ling
    - Madison thoroughly tested the website and the database to make sure that all of the errors and bugs would be fixed. She also worked with Tanvi and Saaketh to test the SQL Queries and ensure that data was being outputted and inserted correctly. Lastly, Madison worked heavily on the project's documentation. She took the screenshots, detailed out the queries, and facilitated the writing process.

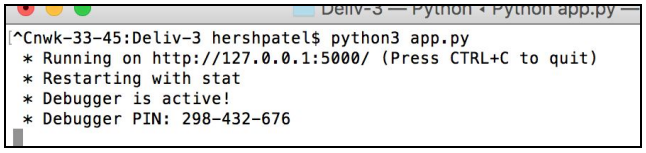
---

***<u>How to Use Our Platform (Locally):</u>***

1. **NOTE: this method of installing and using the software worked with MacOS
2. Download all of our project files, unzip them, and open up command line.
    a. Then, use the `cd` command to open the folder with all of our project files
3. Next, install all of the respective libraries and frameworks.
   **NOTE:** Must have `Python3` installed to run our code, as the Flask framework is not supported by `Python2.7`.
    a. Install `Python3` → [https://www.python.org/download/releases/3.0/](https://www.python.org/download/releases/3.0/)
    b. Install `pip3`, by running `python3 get-pip.py`
        - 

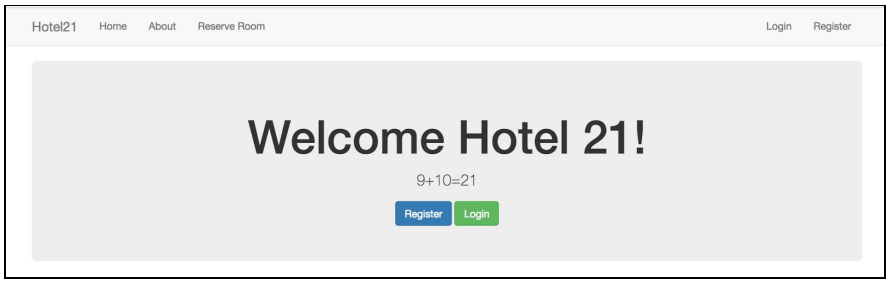
        - The 'get-pip.py' file is already in the project files folder
        - pip is a packagement management system for python
    c. Run `pip3 install -r requirements.txt` to install the respective packages we use (Flask, Flask-MySQLDB, Flask-WTF, Passlib)
        - 

4. Run '`python3 app.py`'.
    a. This will initiate a server and deploy our code locally for you to see




5. Take careful note of the first line that comes up (Running on `http://`…)

    a. That specific http link is what you can enter into your browser (preferably Google Chrome) to see our website platform live
6. Once you open up the website on your browser, the screen home page should pop up (like the one below)




7. Now, just register yourself and get familiar with our website!
    a. See ***Screenshots*** below for more details
8. ***<u>IMPORTANT INFORMATION</u>***
    a. Admin → username = 'admin' and password = '123'
    b. Hersh → username 'hershpatel' and password = '123'
    c. Tanvi → username 'tanviparikh' and password = '123'
    d. Saaketh → username 'saakethkrosuri' and password = '123'
    e. Madison → username 'madisonling' and password = '123'

---

***<u>Project - Assumptions:</u>***

- Project Guideline Assumptions:
    - Our entire website, including functionalities and database, was designed based on the original project background document on sakai:
        - https://drive.google.com/file/d/0B-Q-qLH-M-uFRVk1U3I5b3pGQTA/view?usp=sharing
- You can only make a reservation for one hotel at a time. However, you can select multiple rooms to be in your reservation. Assumed user input is generally okay (no duplicates for breakfast) and that credit card info is not needed as per discussion with TA.
- You are allowed to write reviews for the same thing multiple times
    - This assumption is based off of other product websites where customers can write multiple reviews about a certain item/product

---

***<u>Project Testing:</u>***

- Before doing anything, we sat down as a group and mapped out the entire project, functionally. We listed out, in detail, what kinds of things we needed to have on the website and how the website would interact with the SQL Database.

    - This included writing out pseudo-queries that would output or insert a specified amount of data and how the outputted table or newly inserted data should look.
    - We then used this documentation to fully test our project.
- The two main parts of testing our project were the database and the website.
    - **Database:** Before even inserting any of our queries into the website, we ran all of them in MySQLWorkbench to ensure each query we wanted to run did exactly what it was supposed to. Moreover, prior to this, we had to test that the database queries (CREATE, INSERT, DROP IF EXISTS) ran properly in creating the tables that we outlined in our E-R diagram from Project Deliverable 1. This included outputting a specified table, calculating a value, and inserting values correctly. We also had to pre-populate the database with a lot of 'fake-data' (as specified below in populate queries) in order to test outputs on the website.
    - **Website:**
        - Pre-Database Integration → Before even integrating out database into the website, we had to make sure that the foundation for it existed. Otherwise, we could potentially come across other unforeseeable problems that would have wasted time. More specifically, we made sure that certain pages were only accessible by certain users, that any data the user entered was validated, and that all pages worked together properly.
        - Post-Database Integration → After integrating the database into our website, we had to make sure that we were able to take data from any SELECT query we ran and display it to the user properly, with the use of CSS. This resulted in creating a general table output format. We then had to ensure that when running an INSERT query, all data being inserted was within the domain constraints of that specific table.

---

***<u>Website - Code Overview:</u>***

- **General:**
    - `App.py` → Contains all of our code for the flask framework and website functionalities. Namely, it is where we generate forms for each web page, run SQL queries on our database, and collect/manipulate data the user entered
    - `templates/` → This directory contains all of our front-end code in HTML. We used the Jinja template engine to generate content in each web page and bootstrap to format the content.
- `App.py`:
    - General Functionalities:
        - def index() → Renders the 'index.html' template to display the homepage of the website

        - def about() → Renders the ‘about.html’ template to display the about page of the website

    - User Registration:
        - class RegisterForm() → this form holds the data collected on the ‘register.html’ page in order to insert a new customer into the mysql database
        - def register() → this method renders the ‘register.html’ page to collect data the user entered, writes an insert sql query to insert all data into the database, and then redirects the user to the login page
            - Does not allow the user to create a username that does not exist
            - Encrypts the user’s password with sha256 before inserting into the database
    - User Login:
        - def login() → looks up the username and password the user entered in the database and if corect, redirects the user to the dashboard page
            - Creates a session for the user by logging him/her into the website
            - Outputs username does not exist if username does not exist
            - Outputs incorrect password if password does not match username
    - Dashboard:
        - def dashboard() → renders the ‘dashboard.html’ page where the user can see information about his/her profile, current reservations, upcoming reservations, past reservations, and past reviews
            - SELECTs user data from the ‘customer’ table in the database
            - SELECTs current, upcoming, and past reservation data from the database depending on the check-in and check-out dates of the reservation
            - SELECTs reviews written by your username and outputs them
    - Reservation System:
        - def search_room() → renders ‘1_search_room.html’ which is the first page of the reservation system so that you can enter in your search parameters for available hotels
            - SELECTs all available hotel country/location combos and prints them out for the user to choose from
            - Generates form fields for country, state, check-in date, check-out date, and number of rooms
        - def pick_room() → renders ‘2_pick_room.html’ which is the second page of the reservation system that takes in data from page 1 and displays available hotel rooms

            - SELECTs all available hotel rooms per hotel given all search parameters from page 1
            - Generates form fields for Hotel ID and Room Numbers
                - # of room number fields depends on how many rooms the user requested on page 1
        - def pick_amenities() → renders ‘3_pick_amenities.html’ which is the third page of the reservation system that takes in all data from pages 1 and 2 and displays all available breakfast and service options for the selected hotel
            - SELECTs all available breakfast/service options for one given hotel
            - Generates form fields for all breakfasts and services and allows the user to choose which ones of each and how many they’d like
        - def summary() → renders ‘4_summary.html’ which is the fourth and final page of the reservation system; it displays all information about the user’s reservation
            - It simply prints out the Hotel Information, Check-In Date, Check-Out Date, Number of Days, Room Numbers, Breakfast Orders, Service Orders, and the Total Cost of the stay
            - INSERTs the entire reservation into the database upon clicking the ‘Make Reservation’ button
            - User is then redirected to the dashboard where he/she can see all of the reservations they’ve made
        - def view_reservation(id) → renders the ‘reservation.html’ page which allows the user to see all the details/information of a given invoice #
            - User can only view reservations that exist in the database and were created by him/her
            - The user can only access this page from the dashboard
            - SELECTs all Room, Breakfast, and Service information to display
            - SELECTs all Hotel and Reservation Detail Information to display
    - Review System:
        - def add_review_1() → renders ‘add_review_1.html’ which is the first page of the review system and allows users to choose an invoice to write a review about as well as the review type
            - SELECTs all past reservation information for the user to choose from and write a review about
            - Generates form fields for an Invoice # and Review Type

        - def add_review_2() → renders 'add_review_2.html' which is the second page of the review system and allows users to enter information about the review they'd like to give
            - SELECTs all information for a given invoice # a user inputted on page 1 and displays information about the entered Rooms, Breakfasts, or Services in that reservation
            - Generates form fields for the review title about, rating, and textComment
            - INSERTs the review into the database upon clicking the 'Write Review' button
            - User is then redirected to the dashboard where he/she can see her review and view it
        - def review(id) → renders the 'review.html' which allows the user to view all information about a given reviewID #
            - User can access reviews from either their dashboard, or the 'All Reviews' tab
            - Any user, logged in or not, can view any review they'd like. This is because reviews should be a public platform
            - SELECTs all review information including but not limited to the username of the reviewer, rating, review title, and review text
        - def reviews() → renders 'reviews.html' which allows the user to view all reviews written by all users
            - User can click on any of the reviews to reveal for in-detail information about that specific review
    - Customer Analytics:
        - User must be logged into the admin account in order to view the the customer statistics page
        - def analytics() → renders the 'analytics.html' page which allows the user to enter in a begin date, end date, and stats type to output various information about users and hotels
            - Four stats options:
                - Highest Rated Room Type Per Hotel
                - Highest Rated Breakfast Across All Hotels
                - Highest Rated Service Across All Hotels
                - Top 5 Customers ($$)
- `templates/`: All of the .html files follow a simple layout structure based off of the Jinja template engine and bootstrap .css files
    - `templates/includes/` → This directory contains all helper .html files which are used by each .html page

- _formhelpers.html → this file contains a macro which helps to display forms on various web pages and ensure that all data entered is validated
- _messages.html → this file is implemented by all web pages to display any success or error messages, success messages are flashed in green while error messages are flashed in red
  - 

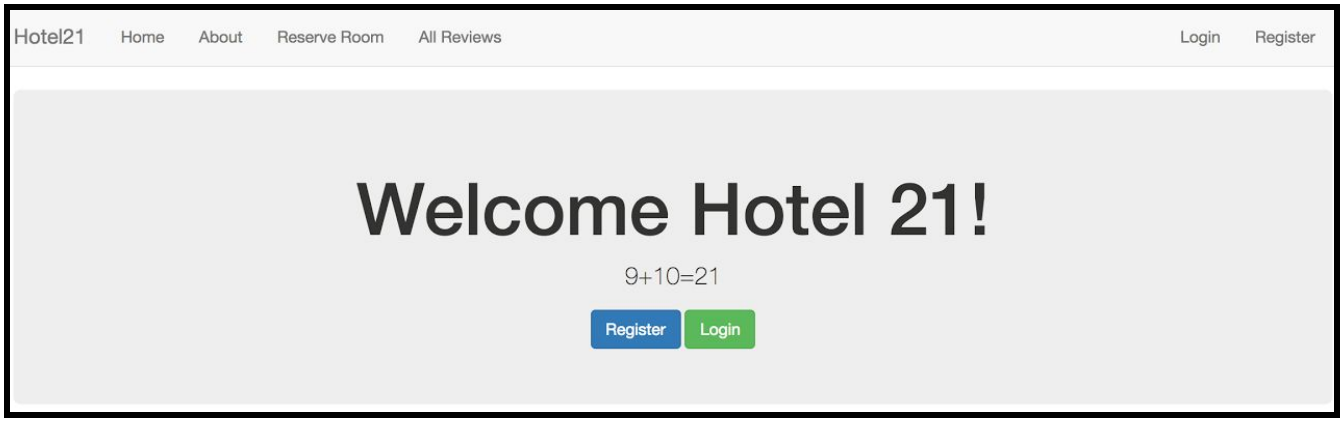
  - 

- _navbar.html → this file is implemented by all web pages to display the navigation bar at the top of the screen; it is helpful because if the navbar needs to be changed, then we only have to go to one file to change everything

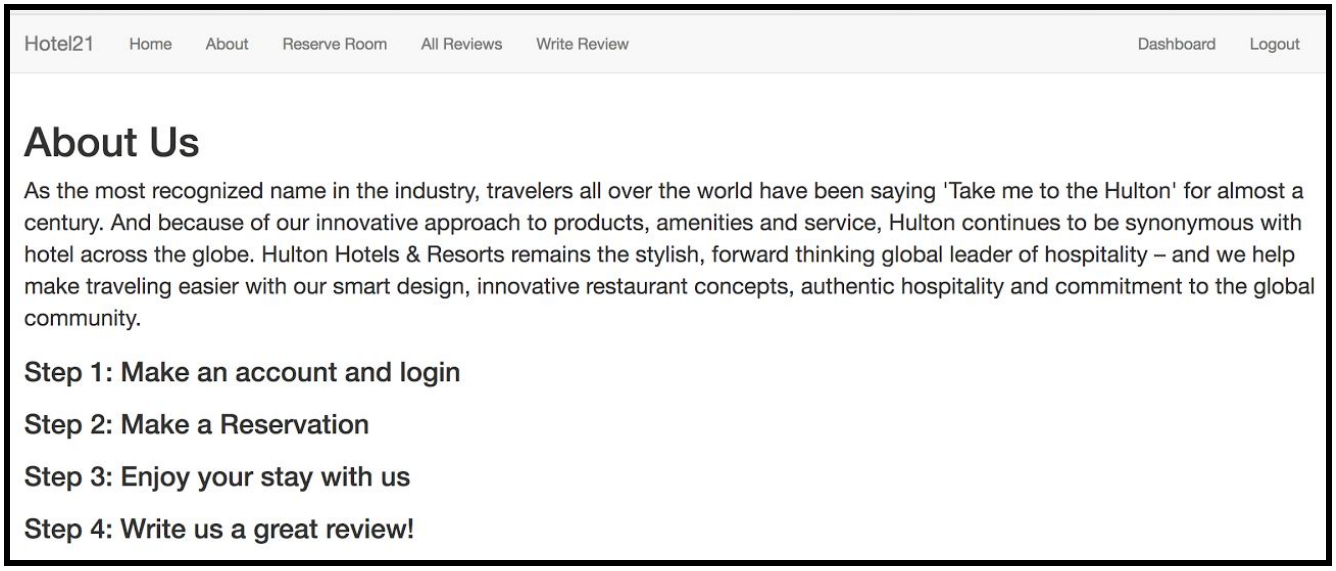
---

***<u>Website Screenshots:</u>***

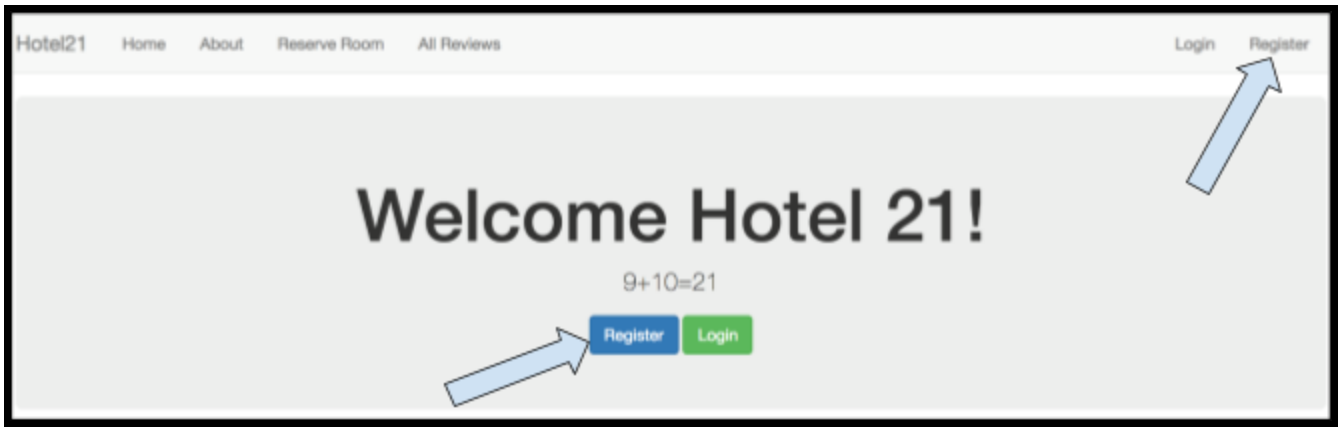
***Welcome Page: <index.html>***



***About Page: <about.html>***




***Registering an Account: <index.html>***



***<register.html>***

Click “Register” to create an account:


Fill in all information to create an account in the database system.

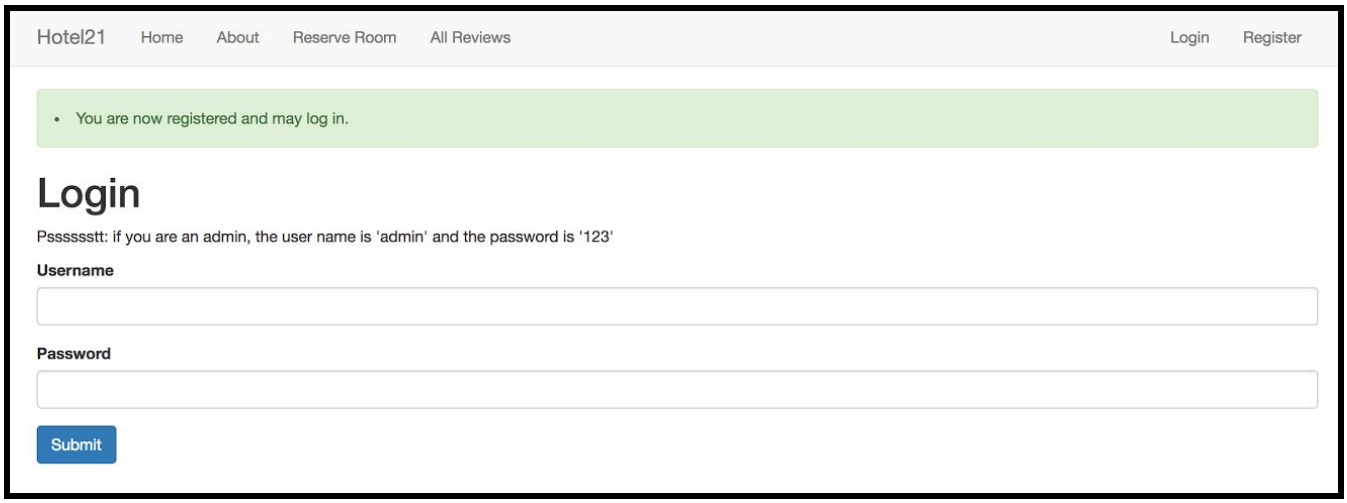
- ***NOTE: Password for ALL existing accounts is ‘123’***

Press “Submit” once completed entering information.

After registering in the database system, you will be able to Log In to the system.

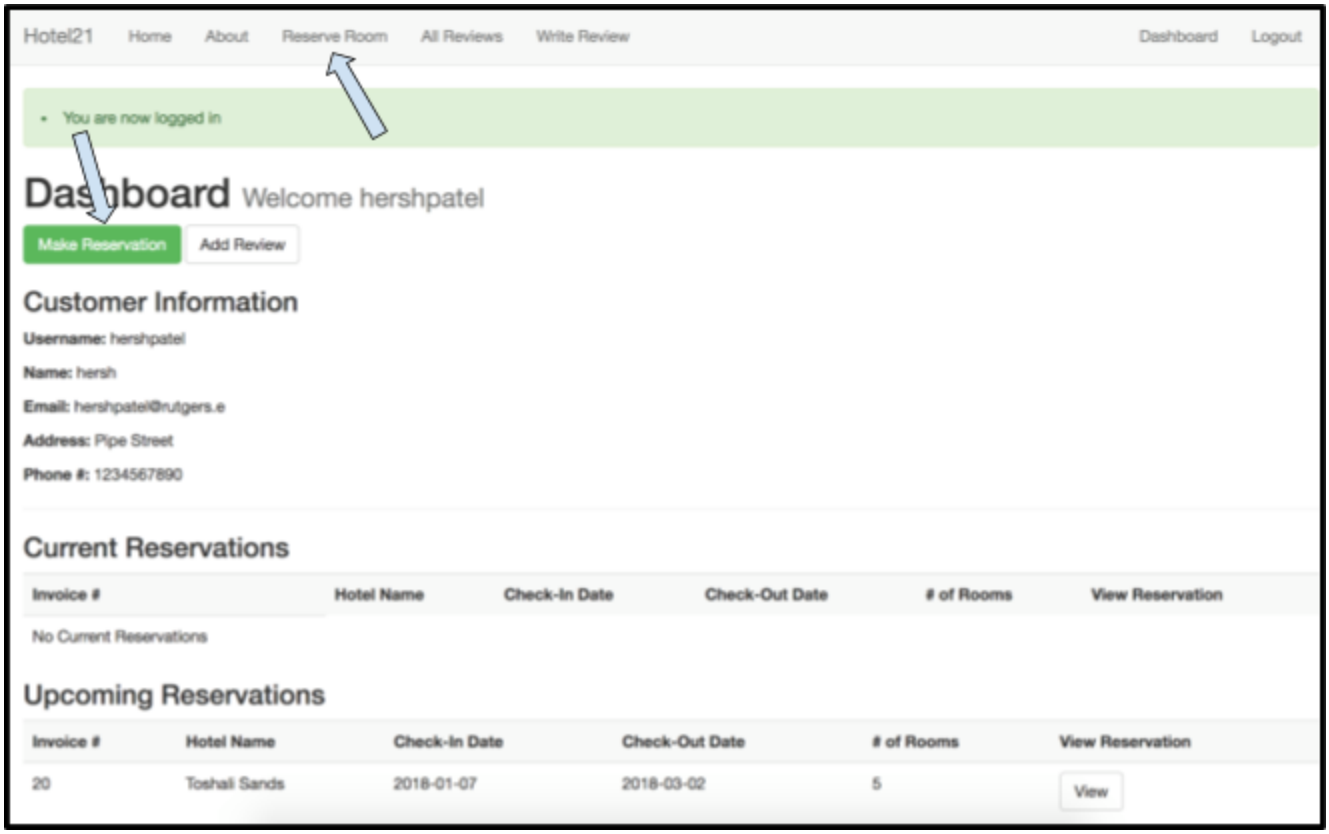
***Logging into an Account: <login.html>***

Log in with the account you created or an existing account to create reservations or view past stays.



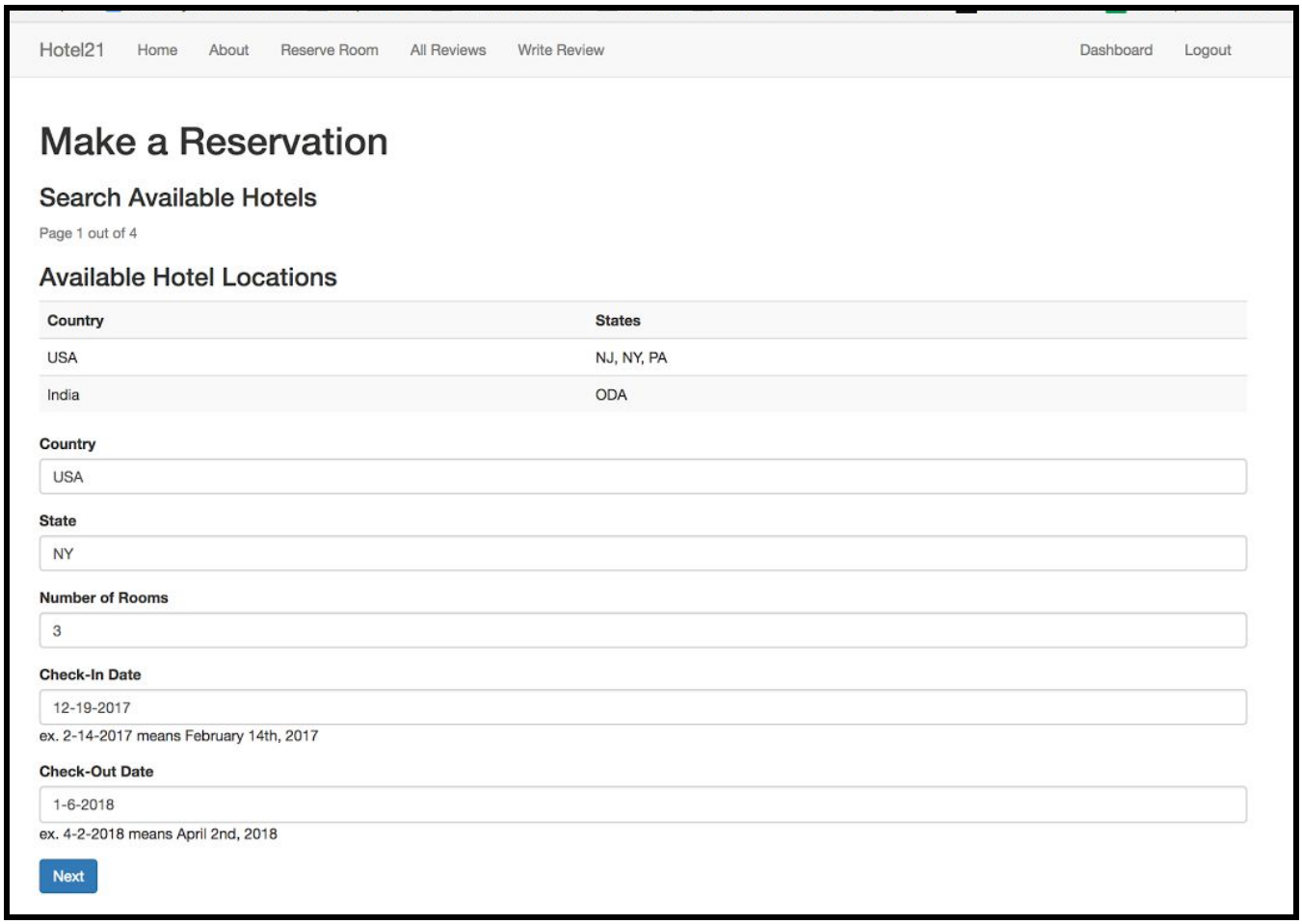

***Creating a Reservation: <dashboard.html>***

- Once logged in, you can view current, upcoming, or past reservations. You can also create a reservation or add a review to a past reservation.
- Click "Make Reservation" to create a reservation at a hotel.



***<1_search_room.html>***

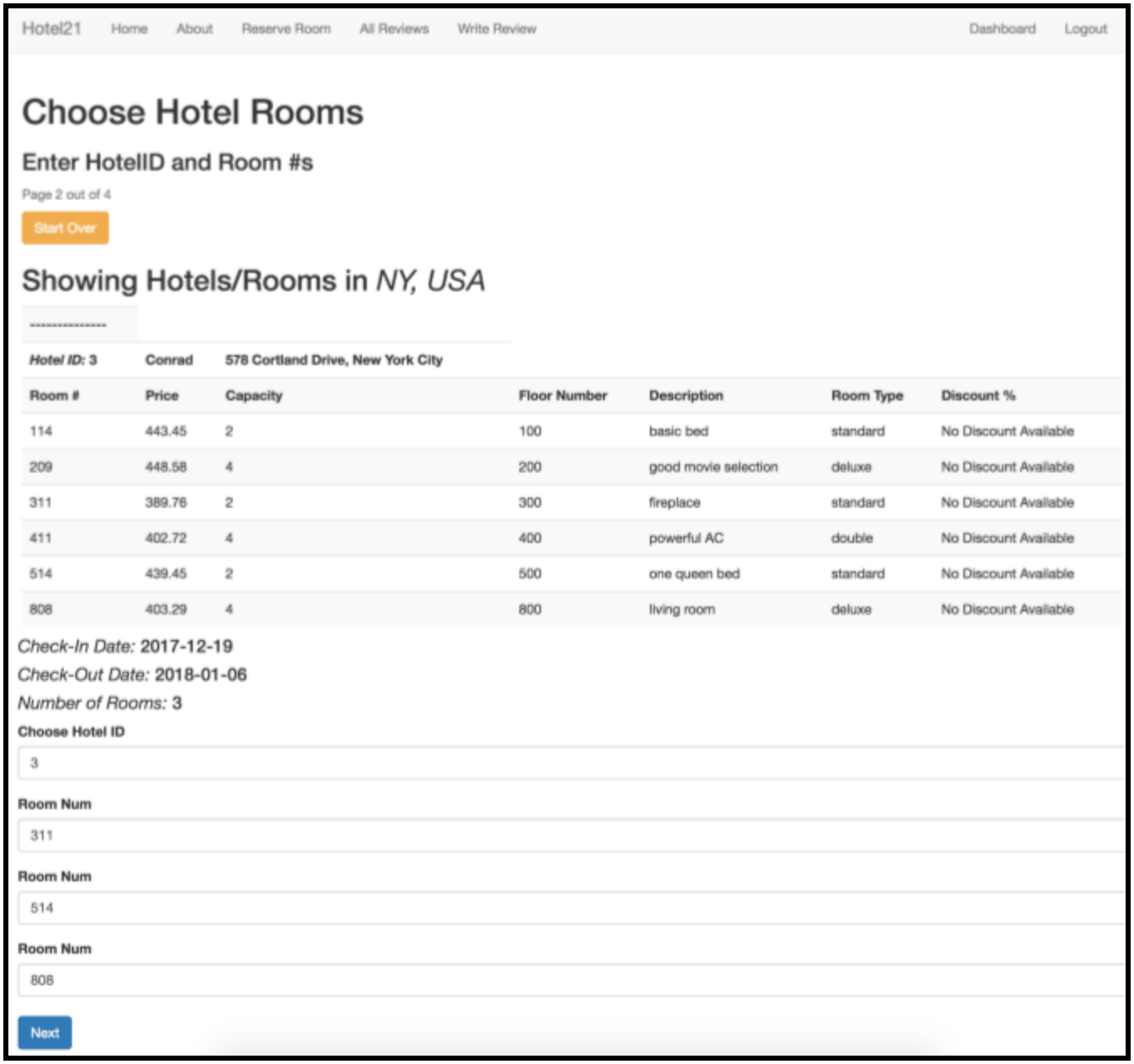
When creating a reservation, you must choose a country and state that is available or you will encounter an error trying to book a hotel that does not exist. Below “USA” and “NY” are selected. As with other inputs below, the selected data must match exactly character-by-character to what is stored in the database. Typing in “USA ” with a space would trigger an error.



***<2_pick_room.html>***

The rooms available from the hotel location you choose will be displayed. The room number, price, capacity, floor number, description, type of room, and available discount will be shown.

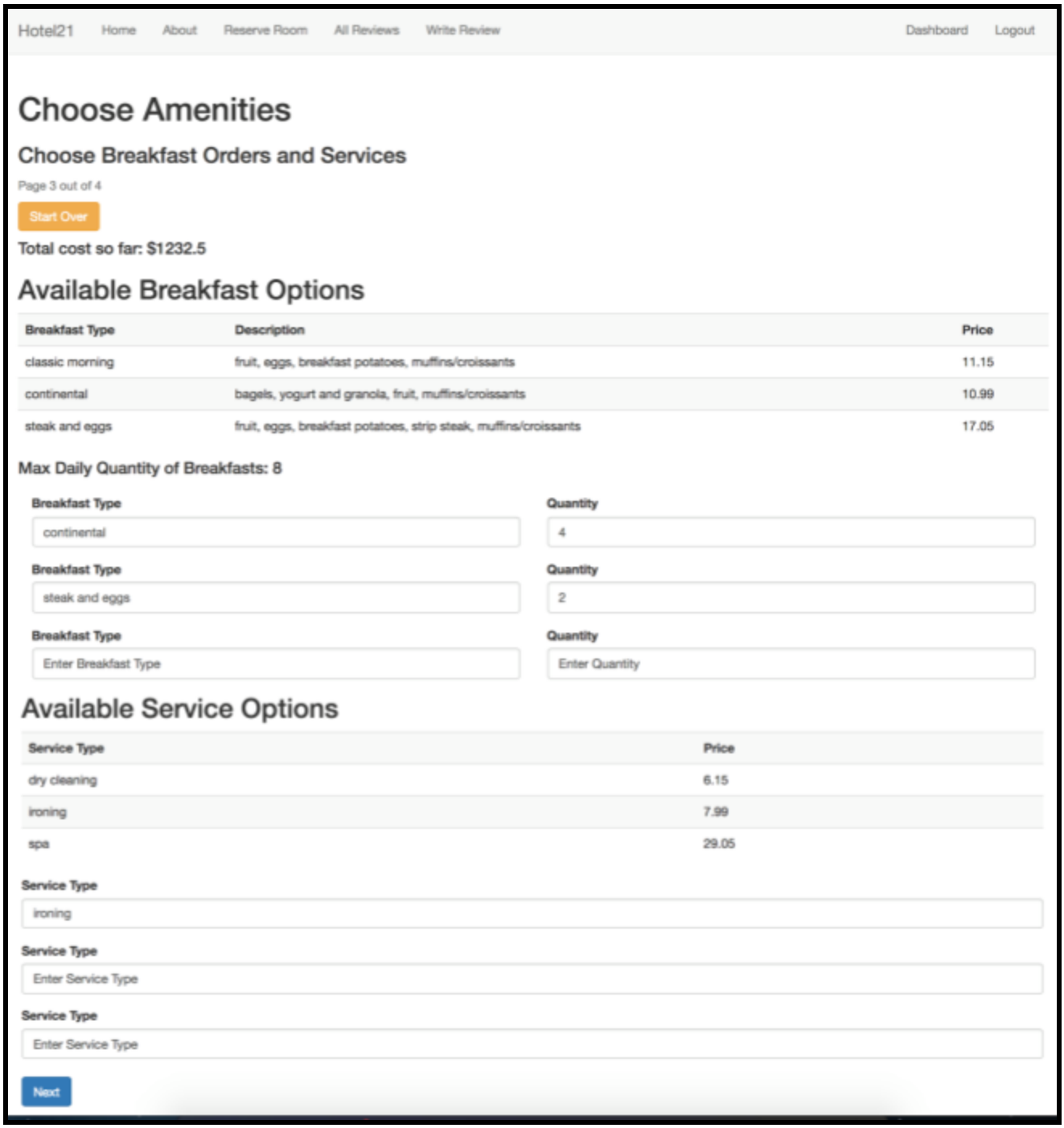
- Choose which rooms you would like to book from the available selection and enter those room numbers at the bottom of the page to reserve the rooms.
- After choosing the rooms you want to reserve, you will be able to choose the amenities you want in addition to your reservation.



***<3_pick_amenities.html>***

You will be able to see the breakfast options and service options of the hotel you are booking for your stay.

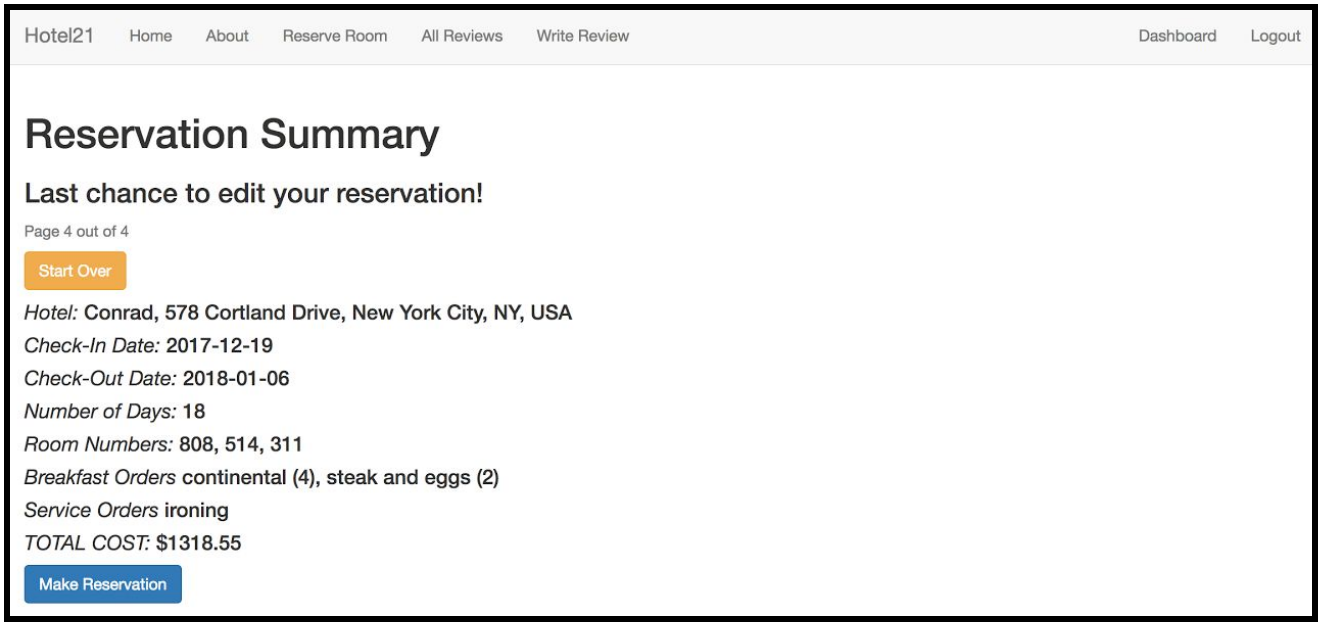
- Choose the breakfast type(s), quantity, and service(s) you want added to your reservation. Click next when you are satisfied with your decision.



***<4_summary.html>***

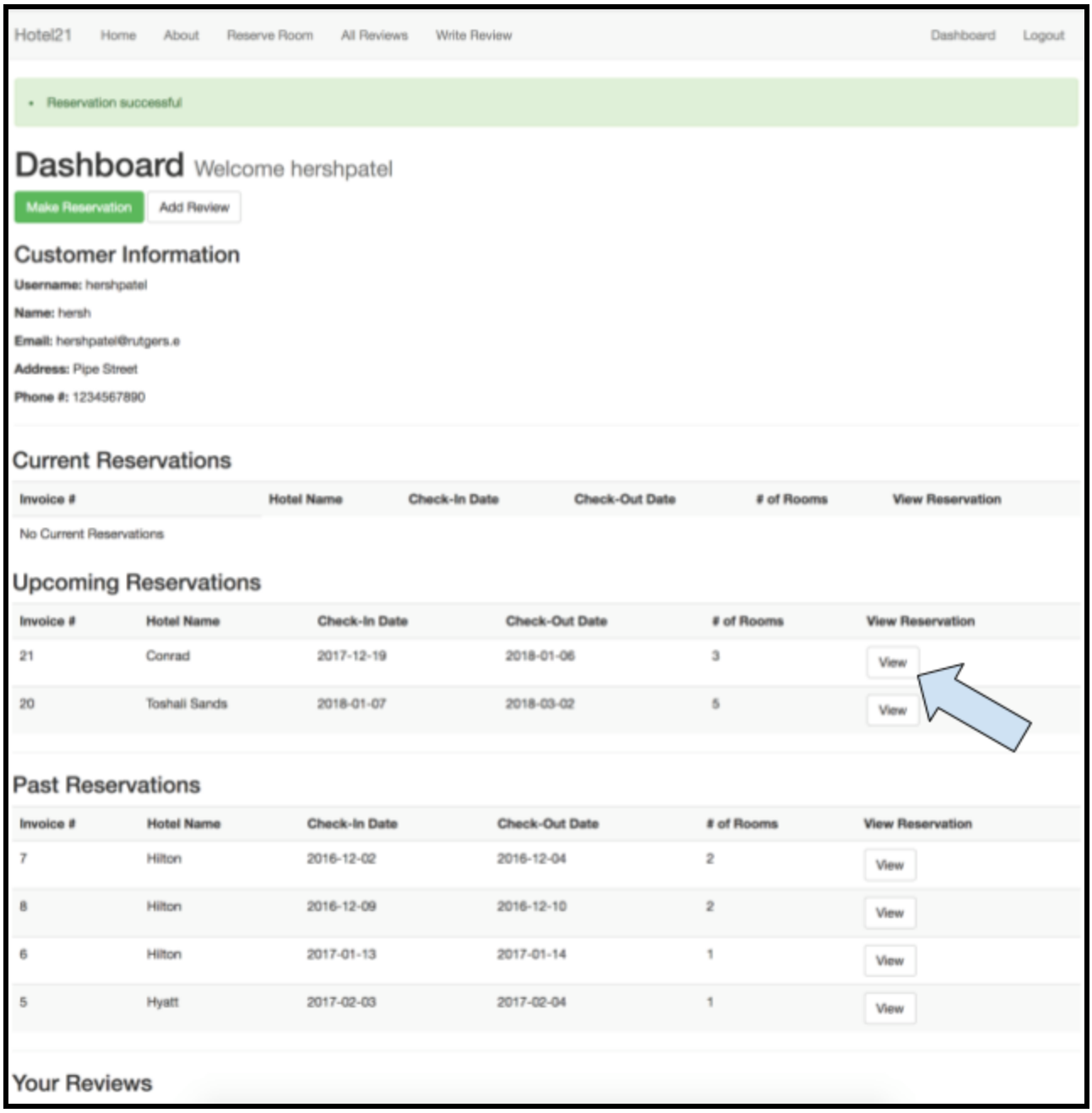
After choosing breakfasts and services, you will be able to review your reservation choices including the hotel, dates of stay, room numbers, breakfast orders, service orders, and total cost of the reservation.

- Upon reviewing your reservation, you may start over if unsatisfied. Otherwise you can click “Make Reservation” to complete booking your stay.



***Reviewing Reservations: <dashboard.html>***

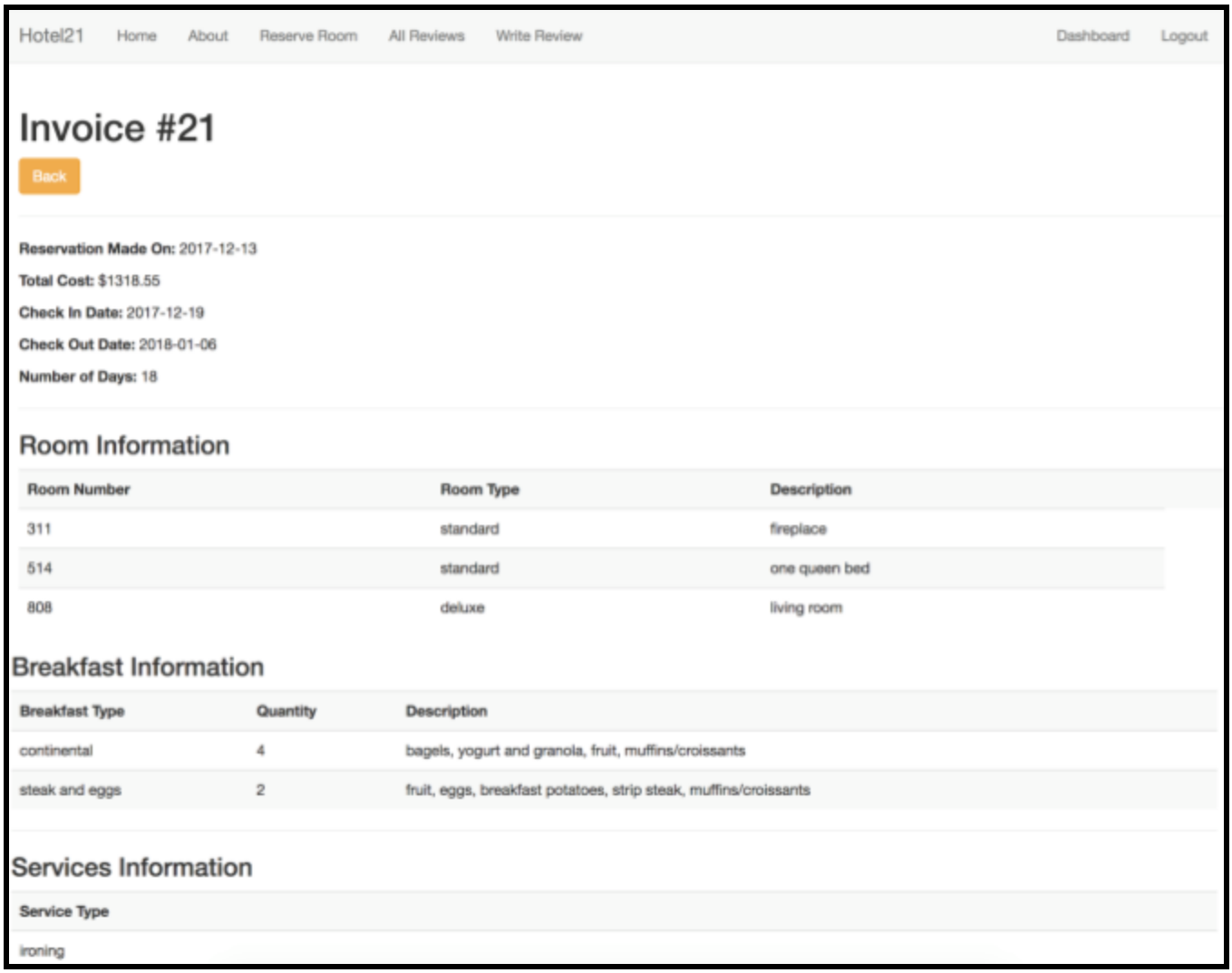
After completing booking your reservation, you can view your upcoming reservation back on your account dashboard by clicking the “View” button.



***<reservation.html>***

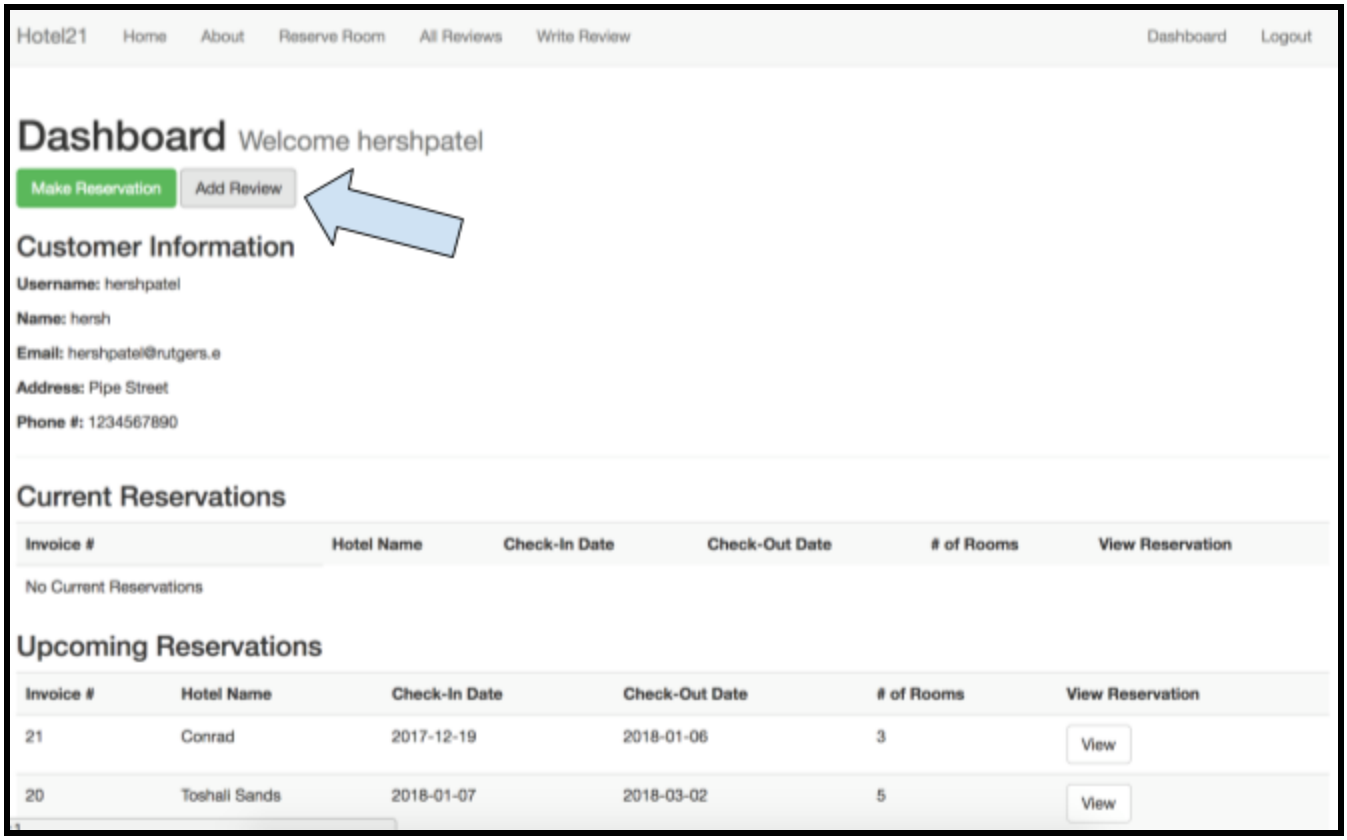
Review your reservation information including when the reservation was created, total cost, and dates of reservation. The invoice will also include room, breakfast, and services information.

- Click “Back” to return to Dashboard.



***Creating a Review: <dashboard.html>***

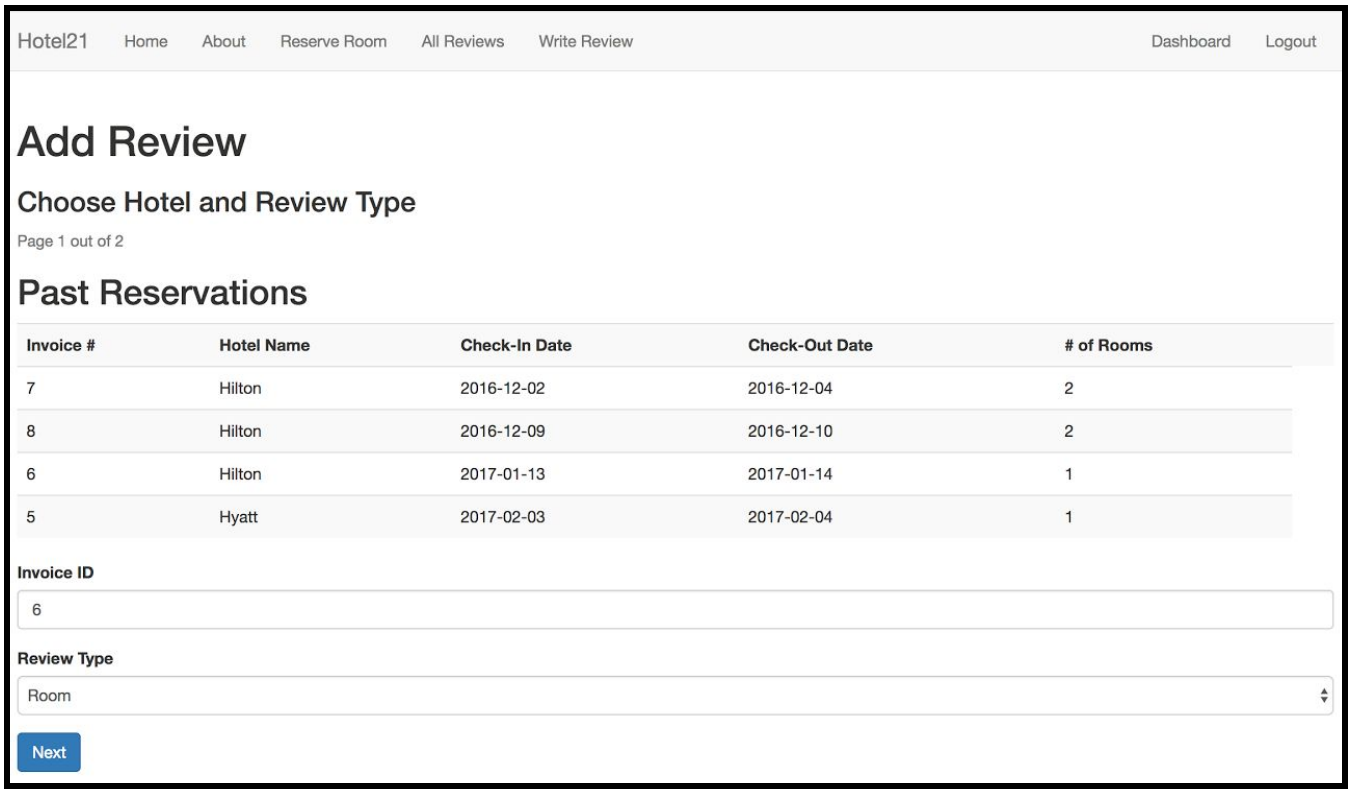
To add a review to a past reservation, click “Add Review.”



***<add_review_1.html>***

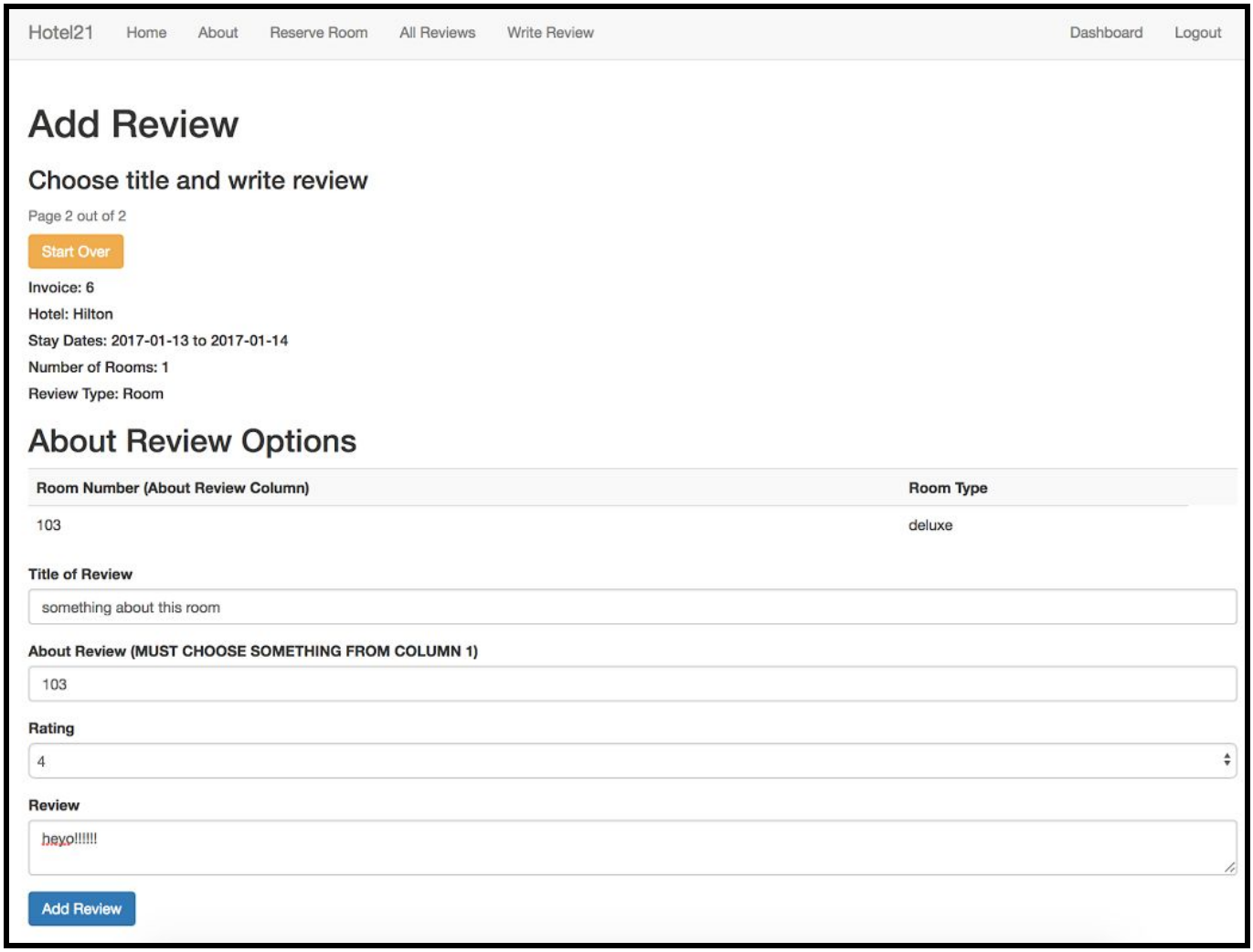
Choose what past reservation you want to review by invoice number and review type (i.e. room, service, breakfast).

- Include a review title, room number, rating, and review description for your review.



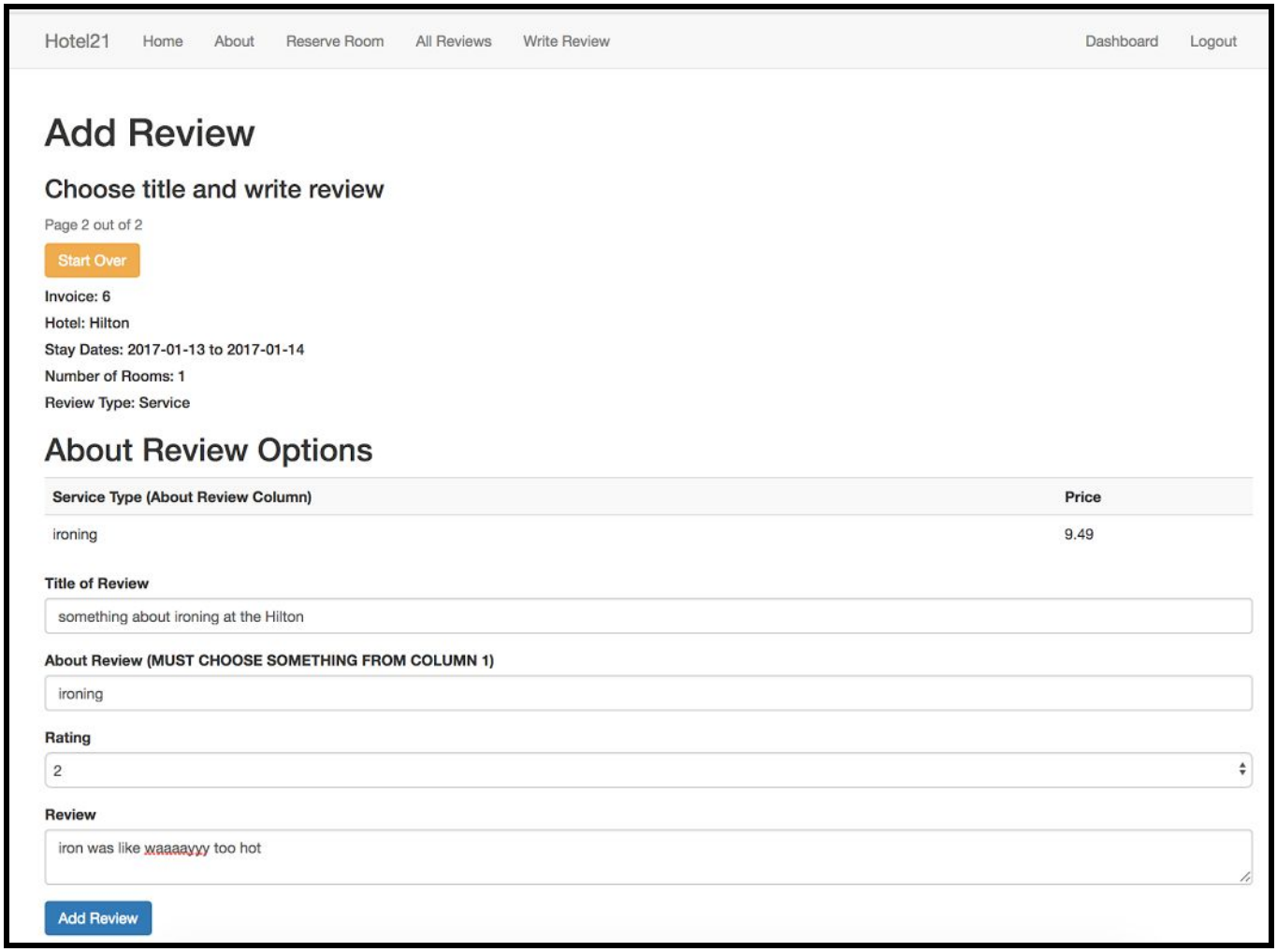
***Write Room Review: <add_review_1.html>***

The following screenshot depicts an example of the user adding a room review.



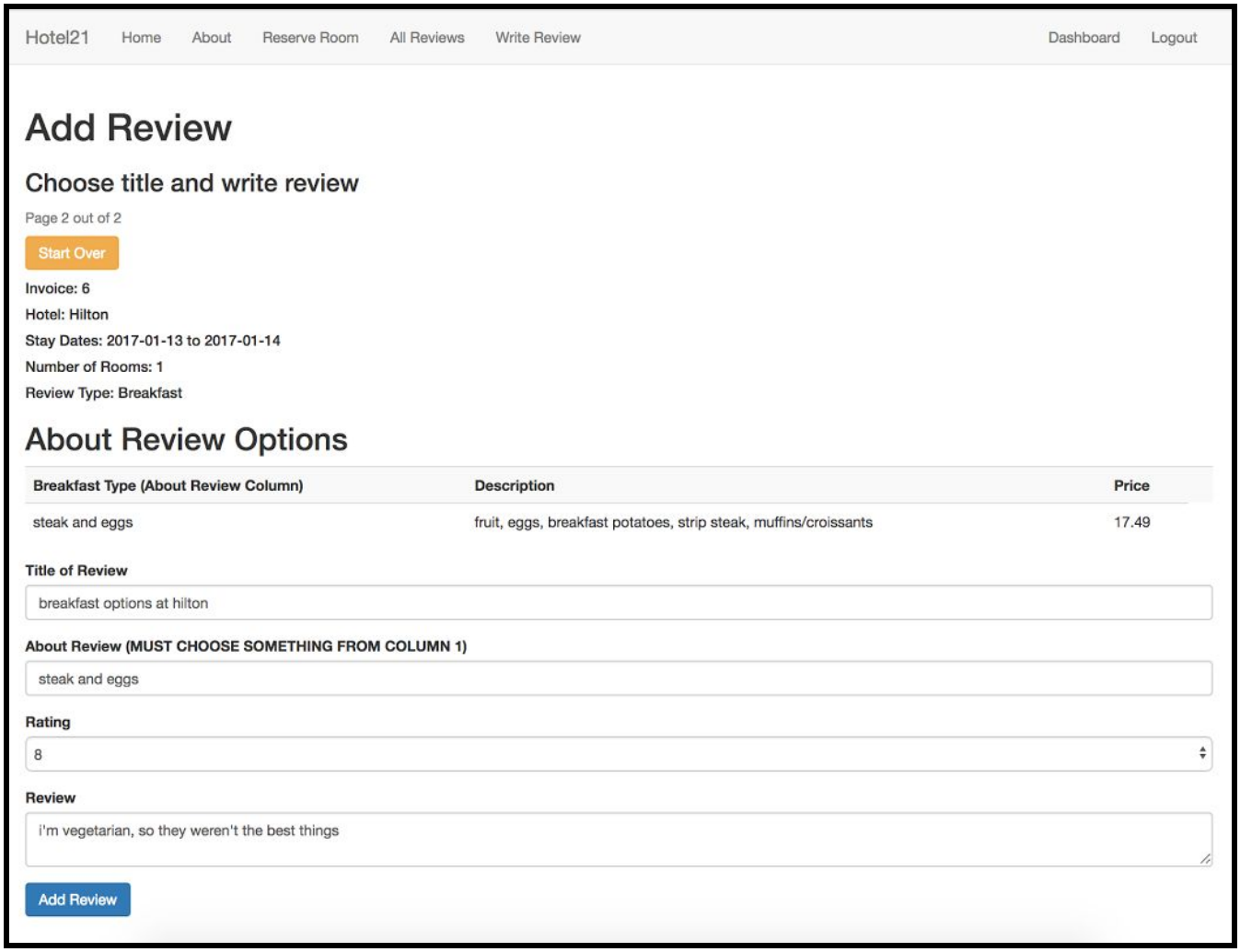
***Write Service Review: <add_review_2.html>***

The following screenshot depicts an example of the user adding a service review.



***Write Breakfast Review: <add_review_2.html>***

The following screenshot depicts an example of the user adding a breakfast review.




After adding the review you want to a past reservation, you will be brought back to your Dashboard where you can see a list of your past reviews.

- You are able to review your past reviews by clicking “View.”

Hotel21 Home About Reserve Room All Reviews Write Review Dashboard Logout

- Review Added

# Dashboard Welcome hershpatel

Make Reservation | Add Review

## Customer Information

**Username:** hershpatel

**Name:** hersh

**Email:** hershpatel@rutgers.e

**Address:** Pipe Street

**Phone #:** 1234567890

## Current Reservations

| Invoice # | Hotel Name | Check-In Date | Check-Out Date | # of Rooms | View Reservation |
|---|---|---|---|---|---|
| No Current Reservations | | | | | |

## Upcoming Reservations

| Invoice # | Hotel Name | Check-In Date | Check-Out Date | # of Rooms | View Reservation |
|---|---|---|---|---|---|
| 21 | Conrad | 2017-12-19 | 2018-01-06 | 3 | View |
| 7 | Hilton | 2016-12-02 | 2016-12-04 | 2 | View |
| 8 | Hilton | 2016-12-09 | 2016-12-10 | 2 | View |
| 6 | Hilton | 2017-01-13 | 2017-01-14 | 1 | View |
| 5 | Hyatt | 2017-02-03 | 2017-02-04 | 1 | View |

## Your Reviews

| ID | Title | Rating | Author | Review Type | View Review |
|---|---|---|---|---|---|
| 9 | Title 9 | 4 | hershpatel | Breakfast | View |
| 10 | Title 10 | 6 | hershpatel | Breakfast | View |
| 11 | Title 11 | 5 | hershpatel | Breakfast | View |
| 12 | Title 12 | 7 | hershpatel | Breakfast | View |
| 13 | Title 13 | 10 | hershpatel | Breakfast | View |
| 14 | Title 14 | 7 | hershpatel | Breakfast | View |
| 15 | Title 15 | 8 | hershpatel | Breakfast | View |
| 16 | Title 16 | 5 | hershpatel | Breakfast | View |
| 17 | Title 17 | 6 | hershpatel | Breakfast | View |
| 18 | Title 18 | 6 | hershpatel | Breakfast | View |
| 38 | Title 38 | 8 | hershpatel | Service | View |
| 39 | Title 39 | 6 | hershpatel | Service | View |

***View Service Review: <review.html>***

Hotel21 Home About Reserve Room All Reviews Write Review Dashboard Logout

# Spa Review @ Hilton

**Written By:** hershpatel

**Service Type:** spa

**Rating:** 5

**Review:** it was mild

***View Room Review: <review.html>***

Hotel21 Home About Reserve Room All Reviews Write Review Dashboard Logout

# something about this room

**Written By:** hershpatel

**Hotel:** Hilton, 78 Candylane, Hoboken, NJ, USA

**Room Type:** deluxe

**Room Description:** beautiful view

**Rating:** 4

**Review:** heyo!!!!!!

***View Breakfast Review: <review.html>***

Hotel21 Home About Reserve Room All Reviews Write Review Dashboard Logout

# breakfast options at hilton

**Written By:** hershpatel

**Breakfast Type:** steak and eggs

**Rating:** 8

**Review:** i'm vegetarian, so they weren't the best things

***<reviews.html>***

This page allows you to view all reviews by all the users:


- After going through all the functionalities of a user's account, we can go back to the dashboard and log out.

Hotel21 Home About Reserve Room All Reviews Write Review Dashboard Logout

# Dashboard Welcome hershpatel

Make Reservation Add Review

## Customer Information

**Username:** hershpatel

**Name:** hersh

**Email:** hershpatel@rutgers.e

**Address:** Pipe Street

**Phone #:** 1234567890

## Current Reservations

| Invoice # | Hotel Name | Check-In Date | Check-Out Date | # of Rooms | View Reservation |
|---|---|---|---|---|---|
| No Current Reservations | | | | | |

## Upcoming Reservations

| Invoice # | Hotel Name | Check-In Date | Check-Out Date | # of Rooms | View Reservation |
|---|---|---|---|---|---|
| 21 | Conrad | 2017-12-19 | 2018-01-06 | 3 | View |
| 20 | Toshali Sands | 2018-01-07 | 2018-03-02 | 5 | View |

***<def logout()>***

After logging out you will be brought to the login page.

Hotel21 Home About Reserve Room All Reviews Login Register

- You are now logged out

# Login

Psssssstt: if you are an admin, the user name is 'admin' and the password is '123'

**Username**

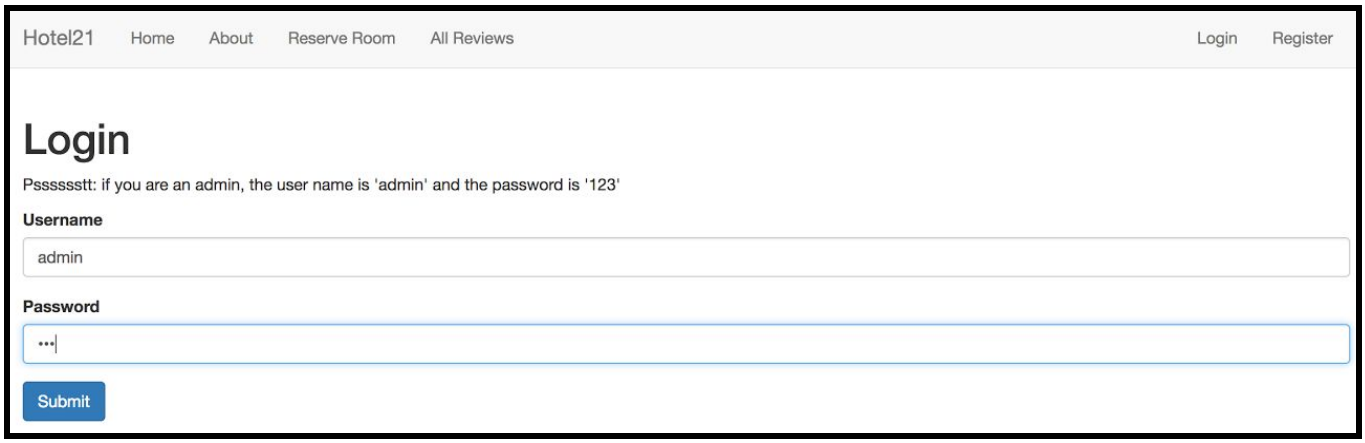
**Password**

Submit

***Customer Analytics:***

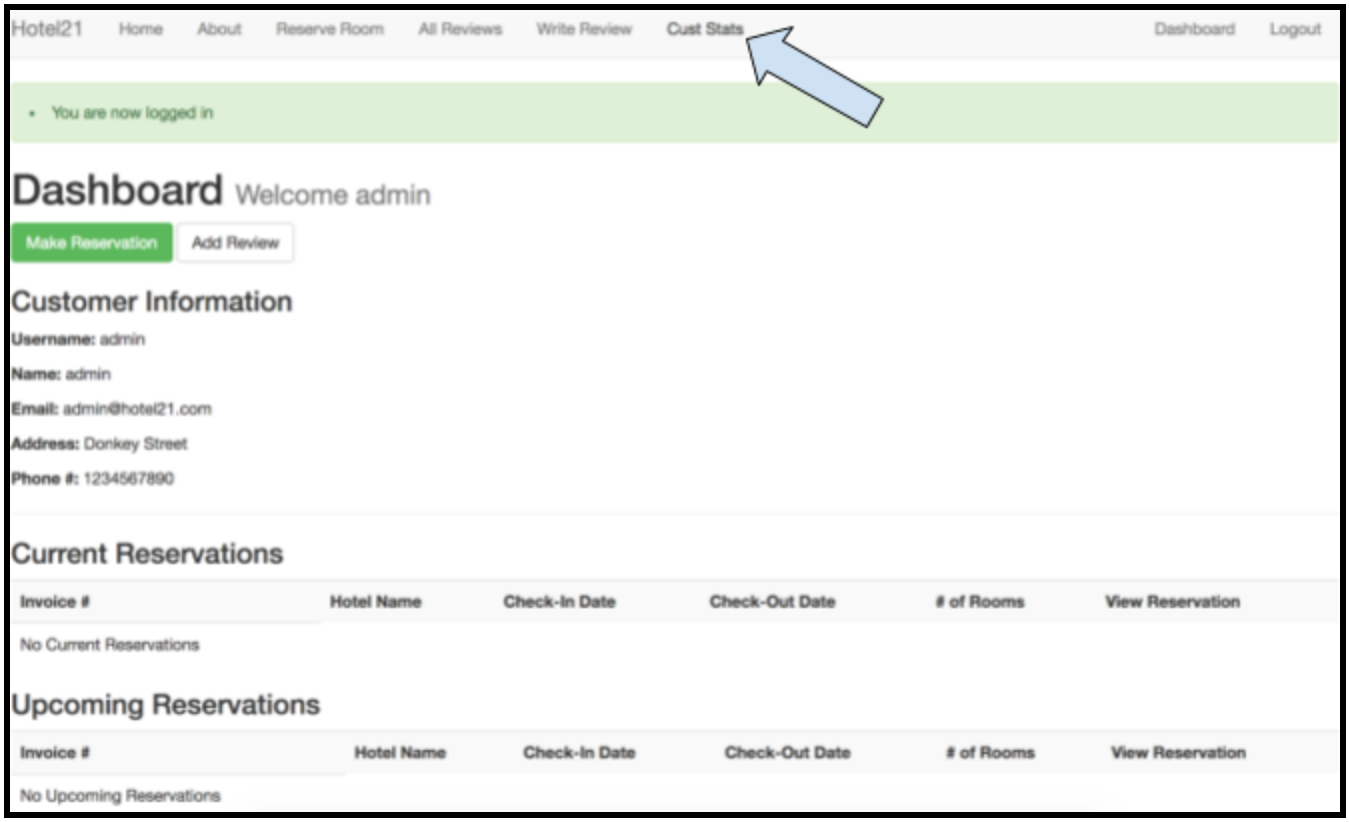
***<login.html>***

Log in to Admin Account to view all customer statistics.




***<dashboard.html>***

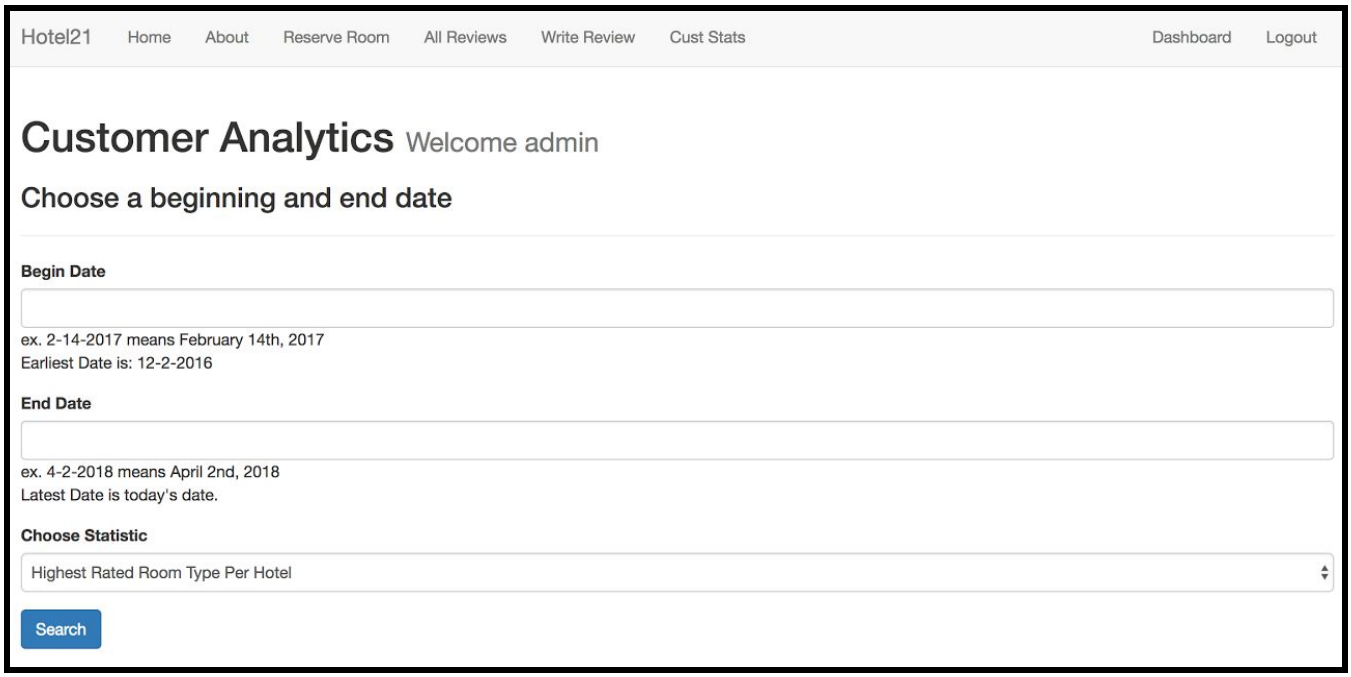
Upon logging in, you will go to the Admin Dashboard and have the option to view customer statistics



***<analytics.html>***

On the Customer Analytics page, you will be able to enter a range of dates and find the statistics for 4 different types of customer statistics. They work just as specified by the documentation linked in the Project Assumptions section.

1. Highest Rated Room Type Per Hotel
2. Highest Rated Breakfast Across All Hotels
3. Highest Rated Service Across All Hotels
4. Top 5 Customers ($$)



**Begin Date**

12-2-2016

ex. 2-14-2017 means February 14th, 2017
Earliest Date is: 12-2-2016

**End Date**

12-12-2017

ex. 4-2-2018 means April 2nd, 2018
Latest Date is today's date.

**Choose Statistic**

Highest Rated Room Type Per Hotel

Search

## Highest Rated Room Type Per Hotel

| Hotel ID | Room Type | Average Rating |
|---|---|---|
| 1 | suite | 6.0000 |
| 2 | standard | 7.0000 |
| 3 | deluxe | 4.0000 |
| 4 | double | 9.5000 |
| 5 | deluxe | 8.0000 |

Hotel21 Home About Reserve Room All Reviews Write Review Cust Stats Dashboard Logout

# Customer Analytics Welcome admin

## Choose a beginning and end date

**Begin Date**

12-2-2016

ex. 2-14-2017 means February 14th, 2017
Earliest Date is: 12-2-2016

**End Date**

12-12-2017

ex. 4-2-2018 means April 2nd, 2018
Latest Date is today's date.

**Choose Statistic**

Highest Rated Breakfast Across All Hotels

Search

## Highest Rated Breakfast Across All Hotels

| Breakfast Type | Highest Average Rating |
|---|---|
| classic morning | 7.2857 |

Hotel21 Home About Reserve Room All Reviews Write Review Cust Stats Dashboard Logout

# Customer Analytics Welcome admin

## Choose a beginning and end date

**Begin Date**

12-2-2016

ex. 2-14-2017 means February 14th, 2017
Earliest Date is: 12-2-2016

**End Date**

12-12-2017

ex. 4-2-2018 means April 2nd, 2018
Latest Date is today's date.

**Choose Statistic**

Highest Rated Service Across All Hotels

Search

## Highest Rated Breakfast Across All Hotels

| Service Type | Highest Average Rating |
|---|---|
| ironing | 6.1111 |

## Choose a beginning and end date

**Begin Date**

12-2-2016

ex. 2-14-2017 means February 14th, 2017
Earliest Date is: 12-2-2016

**End Date**

12-12-2017

ex. 4-2-2018 means April 2nd, 2018
Latest Date is today's date.

**Choose Statistic**

Top 5 Customers ($$)

Search

## Top 5 Customers ($$)

| CID | Name | Total Spent |
|---|---|---|
| 2 | hersh | $629.0 |
| 4 | madison | $572.0 |
| 1 | tanvi | $414.0 |
| 3 | saaketh | $393.0 |

***MySQL Database:***

Created a database "HotelDB" that utilizes multiple CREATE TABLE queries to store hotel data and to aid in adding various functionalities to the website design. The *hotel.sql* file also includes DROP IF and other such functions to make setting up the database schema easier.

*Hotel.sql - Hotel Database Queries:*

- The 'hotel' table holds identifying information about Hotels, such as name, id, and location. It is used to assign a hotel to a given reservation. It is also used to keep track of which reservations have been made, and output the available hotels for given dates (when joined with other tables such as 'reserves' and 'myroom').
    - CREATE TABLE `hotel` (
      `hotelID` int NOT NULL auto_increment,
      `hotel_name` varchar(50) DEFAULT NULL,
      `country` varchar(30) DEFAULT NULL,
      `state` varchar(30) DEFAULT NULL,
      `city` varchar(50) DEFAULT NULL,
      `address` varchar(50) DEFAULT NULL,
      `zipcode` int DEFAULT 1,
      PRIMARY KEY (`hotelID`))

- The 'HotelPhoneNumbers' table is created to hold multiple phone numbers for each hotel, which is why it has to be in a separate table apart from the above 'hotel' table.
    - CREATE TABLE `HotelPhoneNumbers` (
      `hotelID` int NOT NULL DEFAULT 1,
      `phone_no` varchar(30) NOT NULL DEFAULT ' ',
      PRIMARY KEY (`hotelID`, `phone_no`),
      FOREIGN KEY (`hotelID`) REFERENCES `hotel`(`hotelID`))
- The 'myroom' table is created in order to assign a room to a reservation in the reservation system, as well as hold information and specific details regarding the room booked (e.g. the amount of people that can stay in the room and what floor it's on)
    - CREATE TABLE `myroom` (
      `room_num` INT NOT NULL DEFAULT 1,
      `hotelID` INT NOT NULL DEFAULT 1,
      `price` FLOAT DEFAULT NULL,
      `capacity` INT DEFAULT NULL,
      `floor_no` INT NOT NULL DEFAULT 1,
      `description` varchar(250) DEFAULT NULL,
      `room_type` ENUM('standard','double','deluxe','suite'),
      PRIMARY KEY (`room_num`, `hotelID`),
      FOREIGN KEY (`hotelID`) REFERENCES `hotel`(`hotelID`))

- The ‘room_discount’ table is created in order to keep track of which rooms have how much discount during given dates. This information is displayed when the customer looks at room/hotel availability -- for the customer’s given dates, if a room is available, this table will be used to output whether or not there is a discount offered during that period.
    - CREATE TABLE `room_discount` (
      `room_no` INT NOT NULL DEFAULT 1,
      `hotelID` INT NOT NULL DEFAULT 1,
      `discount` FLOAT NOT NULL DEFAULT 1,
      `sdate` DATE NOT NULL DEFAULT '2008-7-04',
      `edate` DATE NOT NULL DEFAULT '2008-7-04',
      PRIMARY KEY (`hotelID`,`room_no`, `sdate`,`edate`,`discount`),
      FOREIGN KEY (`room_no`, `hotelID`) REFERENCES `myroom`(`room_num`, `hotelID`)

- The ‘customer’ table holds information regarding the customer and the customer’s account on the reservation website, such as username, password, and personal info. It also automatically assigns each customer a unique CID, which is used to identify the customer when it comes to credit cards, reviews, and reservations. This table is populated by the website.
    - CREATE TABLE `customer` (
      `CID` int NOT NULL AUTO_INCREMENT,
      `username` varchar(20) DEFAULT ' ',
      `password` varchar (100) DEFAULT NULL,
      `name` varchar(20) DEFAULT NULL,
      `email` varchar(20) DEFAULT NULL,
      `address` varchar(50) DEFAULT NULL,
      `PHONE_NUM` int DEFAULT NULL,
- The ‘reservation’ table holds information regarding the reservation, the customer who made the reservation, the amount the customer spent, and the date the reservation was made. The table also automatically assigns an ‘invoiceNo’ for each reservation, which is used as a number to uniquely identify each reservation. This unique identifier comes in handy later on, when reservations have to be accessed later for breakfast and service reviews as well as for analytics.
    - CREATE TABLE `reservation` (
      `invoiceNo` INT NOT NULL auto_increment,
      `CID` INT NOT NULL DEFAULT 1,
      `totalAmt` INT NOT NULL DEFAULT 1,
      `resDate` DATE NOT NULL DEFAULT '2008-7-04',
      PRIMARY KEY (`invoiceNo`),
      FOREIGN KEY (`CID`) REFERENCES `customer`(`CID`)

- The ‘reserves’ table holds information regarding the reservation and the room reserved, include the reservation dates. The table also assigns an ‘invoiceNo’ for each reservation, which is used as a reservation number to uniquely identify each reservation. This unique identifier comes in handy later on, when reservations have to be accessed later for breakfast and service reviews as well as for analytics.
    - CREATE TABLE `reserves` (
      `invoiceNo` INT NOT NULL DEFAULT 1,
      `room_num` INT NOT NULL DEFAULT 1,
      `hotelID` INT NOT NULL DEFAULT 1,
      `noOfDays` INT DEFAULT NULL,
      `inDate` DATE NOT NULL DEFAULT '2008-7-04',
      `outDate` DATE NOT NULL DEFAULT '2008-7-04',
      PRIMARY KEY (`invoiceNo`,`room_num`,`hotelID`),
      FOREIGN KEY (`invoiceNo`) REFERENCES `reservation`(`invoiceNo`),
      FOREIGN KEY (`room_num`,`hotelID`) REFERENCES `myroom`(`room_num`,`hotelID`))
- The ‘breakfast’ table holds the type and price of the various breakfast choices that each hotel offers. The options would have to be inputted by the hotel owners, and these inputs are displayed later on for customers when they choose their breakfast options. We also have a ‘service’ table that serves a very similar function.
    - CREATE TABLE `breakfast` (
      `bType` varchar(20) NOT NULL DEFAULT ' ',
      `hotelID` INT NOT NULL DEFAULT 1,
      `description` varchar(256) DEFAULT NULL,
      `bPrice` float DEFAULT NULL,
      PRIMARY KEY (`bType`,`hotelID`),
      FOREIGN KEY (`hotelID`) REFERENCES `hotel`(`hotelID`))
- The ‘services’ table holds the type and price of the various service choices that each hotel offers. The options would have to be inputted by the hotel owners, and these inputs are displayed later on for customers when they choose their service options.
    - CREATE TABLE `services` (
      `sType` varchar(20) NOT NULL DEFAULT ' ',
      `hotelID` INT NOT NULL DEFAULT 1,
      `sCost` float DEFAULT NULL,
      PRIMARY KEY (`sType`,`hotelID`),
      FOREIGN KEY (`hotelID`) REFERENCES `hotel`(`hotelID`))
- The ‘review’ table holds reviews regarding the room, breakfast, or service of the hotel. Each customer can write a review and rate the room/breakfast/service. This table is also used later to run analytics by the hotel owner (e.g. to see which services were highest ratest). The table also generates a ‘reviewID’ which auto increments for each review.
    - CREATE TABLE `review` (
      `reviewID` int NOT NULL auto_increment,
      `CID` int NOT NULL DEFAULT 1,
      `rating` int DEFAULT NULL,
      `title` varchar (100) DEFAULT NULL,
      `textcomment` varchar(1028) DEFAULT NULL,
      `review_type` varchar(10) DEFAULT NULL,
      PRIMARY KEY (`reviewID`),
      FOREIGN KEY (`CID`) REFERENCES `customer`(`CID`))

- The ‘room_review’ table holds information regarding the review and the room that is being evaluated. It uses the foreign key ‘reviewID’ from the ‘review’ table to identify which review by which customer, and holds information such as the room and hotel.
    - CREATE TABLE `room_review` (
      `reviewID` int NOT NULL DEFAULT 1,
      `room_num` int DEFAULT NULL,
      `hotelID` int DEFAULT NULL,
      PRIMARY KEY (`reviewID`),
      FOREIGN KEY (`reviewID`) REFERENCES `review`(`reviewID`),
      FOREIGN KEY (`room_num`,`hotelID`) REFERENCES `myroom`(`room_num`,`hotelID`))
- The ‘breakfast_review’ table holds information regarding the review and the breakfast that is being assessed. It acts similar to the room_review table, with reviewID and breakfast/hotel information.
    - CREATE TABLE `breakfast_review` (
      `reviewID` int NOT NULL DEFAULT 1,
      `bType` varchar(20) DEFAULT NULL,
      `hotelID` int DEFAULT NULL,
      PRIMARY KEY (`reviewID`),
      FOREIGN KEY (`reviewID`) REFERENCES `review`(`reviewID`),
      FOREIGN KEY (`bType`,`hotelID`) REFERENCES `breakfast`(`bType`,`hotelID`))
- The ‘service_review’ table holds information regarding the review and the service that is being assessed. It acts similar to the room_review and breakfast_review tables, with reviewID and breakfast/hotel information.
    - CREATE TABLE `service_review` (
      `reviewID` int NOT NULL DEFAULT 1,
      `sType` varchar(20) DEFAULT NULL,
      `hotelID` int DEFAULT NULL,
      PRIMARY KEY (`reviewID`),
      FOREIGN KEY (`reviewID`) REFERENCES `review`(`reviewID`),
      FOREIGN KEY (`sType`,`hotelID`) REFERENCES `services`(`sType`,`hotelID`))
- The ‘includes’ table connects the reservation table and breakfast that is offered by the hotel. It holds information on the type of breakfast and the number of breakfasts that were ordered when creating the reservation, and references the invoiceNo from the reservation table as well as the hotelID from the hotel table.
    - CREATE TABLE `includes` (
      `invoiceNo` INT NOT NULL DEFAULT 1,
      `num_of_breakfasts` INT NOT NULL DEFAULT 1,
      `hotelID` INT NOT NULL DEFAULT 1,
      `bType` varchar(20) NOT NULL DEFAULT '',
      PRIMARY KEY (`invoiceNo`,`bType`,`hotelID`),
      FOREIGN KEY (`invoiceNo`) REFERENCES `reservation`(`invoiceNo`),
      FOREIGN KEY (`bType`,`hotelID`) REFERENCES `breakfast`(`bType`,`hotelID`))

- The ‘contains’ table connects the reservation table and the service that is provided by the hotel. It holds information on the type of service chosen when creating the reservation, acting similar to the above ‘includes’ table.
    - ```
      CREATE TABLE `contains` (
      `invoiceNo` INT NOT NULL DEFAULT 1,
      `num_of_services` INT NOT NULL DEFAULT 1,
      `hotelID` INT NOT NULL DEFAULT 1,
      `sType` varchar(20) NOT NULL DEFAULT ' ',
      PRIMARY KEY (`invoiceNo`,`sType`,`hotelID`),
      FOREIGN KEY (`invoiceNo`) REFERENCES `reservation`(`invoiceNo`),
      FOREIGN KEY (`sType`,`hotelID`) REFERENCES `services`(`sType`,`hotelID`))
      ```
- The ‘creditcard’ table holds information regarding the customer’s credit card that is being used for the hotel reservation. It references CID from the customer table, indicating which customer the card belongs to, as well as the invoiceNo from the reservation table, indicating which reservation the card was used for. This was not used, after conversation with TA.
    - ```
      CREATE TABLE `creditcard` (
      `CNumber` INT NOT NULL DEFAULT 1,
      `CID` INT NOT NULL DEFAULT 1,
      `InvoiceNo` INT NOT NULL DEFAULT 1,
      `Name` varchar(30) DEFAULT NULL,
      `ExpDate` DATE DEFAULT NULL,
      `Type` varchar (20) DEFAULT NULL,
      `SecCode` INT NOT NULL DEFAULT 1,
      `BillingAddr` varchar(50) DEFAULT NULL,
      PRIMARY KEY (`CNumber`, `CID`, `InvoiceNo`),
      FOREIGN KEY (`CID`) REFERENCES `customer`(`CID`),
      FOREIGN KEY (`InvoiceNo`) REFERENCES `reservation`(`invoiceNo`))
      ```

*Clearing Database Queries:*

- All the below queries clear the database and reset the auto-incrementers to 1. This was useful when we wanted to reset the database -- without this, everytime we ran the insert, there would be multiple duplicate values in the database.
    - ```
      DELETE FROM `creditcard`
      WHERE `InvoiceNo` >=1;
      DELETE FROM `includes`
      WHERE `invoiceNo` >=1;
      DELETE FROM `contains`
      WHERE `invoiceNo` >=1;
      DELETE FROM `service_review`
      WHERE `reviewID` >=1;
      DELETE FROM `breakfast_review`
      WHERE `reviewID` >=1;
      DELETE FROM `room_review`
      WHERE `reviewID` >=1;
      DELETE FROM `review`
      WHERE `reviewID` >=1;
      ALTER TABLE `review`
      AUTO_INCREMENT = 1 ;
      DELETE FROM `services`
      WHERE `hotelID` >=1;
      ```
      ```

```
DELETE FROM `breakfast`
WHERE `hotelID` >=1;
DELETE FROM `reserves`
WHERE `invoiceNo` >=1;
DELETE FROM `reservation`
WHERE `invoiceNo` >=1;
ALTER TABLE `reservation`
AUTO_INCREMENT = 1;
DELETE FROM `room_discount`
WHERE `hotelID` >=1;
DELETE FROM `myroom`
WHERE `hotelID` >=1;
DELETE FROM `HotelPhoneNumbers`
WHERE `hotelID` >=1;
DELETE FROM `hotel`
WHERE `hotelID` >=1;
ALTER TABLE `hotel` AUTO_INCREMENT = 1 ;
```

*Populating Database Queries:*

- Below are **samples** of insert queries which we used to populate the database once we created the tables.
- Populate 'hotel' with values that fit table parameters
  - INSERT INTO `hotel` (`hotel_name`, `country`, `state`, `city`, `address`, `zipcode`)
    VALUES ('Hyatt','USA','NJ','New Brunswick','102 Woodland Place', 10248),
    ('Hilton','USA','NJ','Hoboken','78 Candylane', 89567),
    ('Conrad','USA','NY','New York City','578 Cortland Drive', 25368),
    ('Marriot','USA','PA','Philadelphia','32 Chocolate Avenue', 37465),
    ('Westin','USA','PA','Philadelphia','198 Obama Highway', 78354),
    ('Toshali Sands','India','ODA','Puri','Konark Marine Drive', 752002);
- Populate 'HotelPhoneNumbers' with values that fit table parameters. Accounted for each hotel having multiple phone numbers.
  - INSERT INTO `HotelPhoneNumbers` VALUES
    (1,'7325678987'),(2,'9085476235'),(3,'6465672345'),(4,'8016785395'),(4,'6198234278'),(5,'7513984783'),
    (5,'9346752837'),(5,'2035837465'),(6,'7324987043'),(6,'8072438062');
- Populate 'myroom' with values that fit table parameters. While they're not all displayed here to save space, we have accounted for the requirement that each hotel should have at least 10 rooms.
  - INSERT INTO `myroom` VALUES
    (209, 2, 114.87, 4, 200, 'jacuzzi', 'deluxe'),
    (404, 2, 135.04, 4, 400, 'two queens', 'double'),
    (719, 2, 122.25, 4, 700, 'adults and kids rooms', 'suite'),
    (999, 2, 555.04, 4, 900, 'two queens', 'double'),
- Populate 'room_discount' with values that fit table parameters. Similarly, we have only displayed a portion to save space.
  - INSERT INTO `room_discount` VALUES
    (103, 2, 10, '2018-1-13','2018-2-27'),
    (202, 2, 20, '2018-1-02','2018-4-25'),
    (809, 2, 10, '2017-12-11','2018-12-29'),
    (302, 2, 30, '2017-12-11','2018-3-29'),
    (302, 3, 10, '2017-12-09','2018-3-11'),

- Populate ‘reservation’ with values that fit table parameters. The totalAmt’s were calculated using separate queries.
    - INSERT INTO `reservation` (`CID`, `totalAmt`, `resDate`)VALUES
      (1,309.2199979,NOW()), (1,308.8599973,NOW()), (1,446.3199957,NOW()),(1,261.2999991,NOW()),
      (2,221.039999,NOW()), (2,219.5499986,NOW()), (2,917.9399953,NOW()),(2,570.3099985,NOW()),
      (3,658.4099988,NOW()), (3,468.6999986,NOW()), (3,1301.829996,NOW()),(1529.959997,NOW()),
      (4,533.5099975,NOW()), (4,394.7800005,NOW()),
      (4,737.5700066,NOW()),(4,827.3300002,NOW()),(4,287.4099992,NOW()),(4,712.709996,NOW());
- Populate ‘reserves’ with values that fit table parameters. We have only displayed a portion here to conserve space.
    - INSERT INTO `reserves` VALUES
      (1, 101, 1, 2, '2016-12-18','2016-12-21'),
      (2, 102, 1, 2, '2017-1-01','2017-1-03'),
      (3, 105, 1, 3, '2016-12-15','2016-12-19'),
      (4, 406, 1, 1, '2017-1-14','2017-1-15'),
- Populate ‘breakfast’ with values that fit table parameters. We have only displayed a portion here to conserve space.
    - INSERT INTO `breakfast` VALUES
      ('continental', 1, 'bagels, yogurt and granola, fruit, muffins/croissants', 11.79),
      ('classic morning', 1, 'fruit, eggs, breakfast potatoes, muffins/croissants', 12.69),
      ('steak and eggs', 1, 'fruit, eggs, breakfast potatoes, strip steak, muffins/croissants', 16.89),
- Populate ‘services’ with values that fit table parameters.
    - INSERT INTO `services` VALUES
      ('ironing', 1, 8.79), ('dry cleaning', 1, 7.69), ('spa', 1, 28.89), ('ironing', 2, 9.49),
      ('dry cleaning', 2, 8.29), ('spa', 2, 27.49), ('ironing', 3, 7.99), ('dry cleaning', 3, 6.15),
      ('spa', 3, 29.05), ('ironing', 4, 7.49), ('dry cleaning', 4, 7.75), ('spa', 4, 29.45),
      ('ironing', 5, 7.99), ('dry cleaning', 5, 6.99), ('spa', 5, 32.65), ('ironing', 6, 7.94),
      ('dry cleaning', 6, 4.99), ('spa', 6, 12.65);
- Populate ‘review’ with values that fit table parameters. We have only displayed a portion here to conserve space.
    - INSERT INTO `review` (`CID`, `rating`, `title`, `textcomment`, `review_type`) VALUES
      (1, 1, 'Title 1', 'This was bad - 1/10! I like eating cheese.', 'Breakfast'),
      (1, 5, 'Title 2', 'This was good - 5/10. I like eating sunflower seeds.', 'Breakfast'),
      (1, 7, 'Title 3', 'This was good - 7/10. I like eating eel.', 'Breakfast'),
      (1, 6, 'Title 4', 'This was good - 6/10. I like eating arugula.', 'Breakfast'),
      (1, 10, 'Title 5', 'This was great - 10/10! I like eating almond butter.', 'Breakfast'),
      (1, 10, 'Title 6', 'This was great - 10/10! I like eating rum.', 'Breakfast'),
      (1, 5, 'Title 56', 'This was good - 5/10. I like eating turducken.', 'Room'),
- Populate ‘room_review’ with values that fit table parameters. We have only displayed a portion here to conserve space. These reviews correspond to reviews with ID’s from 41-60.
    - INSERT INTO `room_review` VALUES
      (41, 101, 1), (42, 103, 2), (43, 808, 3), (44, 104, 5),
      (45, 202, 2), (46, 407, 4), (47, 102, 1), (48, 118, 5),

- Populate ‘breakfast_review’ with values that fit table parameters. We have only displayed a portion here to conserve space. These reviews correspond to reviews with ID’s from 1-20.
    - INSERT INTO `breakfast_review` VALUES
      (1, 'continental', 1), (2, 'continental', 2),
      (3, 'continental', 3), (4, 'continental', 4),
      (5, 'continental', 5), (6, 'classic morning', 1),
      (7, 'classic morning', 1), (8, 'classic morning', 2),
- Populate ‘service_review’ with values that fit table parameters. We have only displayed a portion here to conserve space. These reviews correspond to reviews with ID’s from 21-40.
    - INSERT INTO `service_review` VALUES
      (21, 'spa', 1), (22, 'spa', 2), (23, 'spa', 3), (24, 'spa', 4),
      (25, 'spa', 5), (26, 'spa', 5), (27, 'dry cleaning', 1),
      (28, 'dry cleaning', 2), (29, 'dry cleaning', 2),
- Populate ‘contains’ with values that fit table parameters
    - INSERT INTO `contains` VALUES
      (1, 2, 1, 'spa'), (1, 3, 1, 'dry cleaning'), (1, 4, 1, 'ironing'),
      (2, 2, 1, 'dry cleaning'), (3, 1, 1, 'ironing'), (4, 5, 1, 'spa'),
      (5, 6, 1, 'dry cleaning'), (6, 7, 2, 'ironing'), (7, 8, 2, 'spa'),
      (8, 2, 2, 'dry cleaning'),(9, 1, 2, 'ironing'), (10, 4, 3, 'spa'),
      (11, 5, 3, 'ironing'), (12, 6, 3, 'dry cleaning'), (13, 8, 3, 'spa'),
      (14, 11, 4, 'ironing'), (15, 3, 4, 'ironing'), (16, 4, 5, 'spa'),
      (17, 1, 6, 'dry cleaning'), (18, 2, 6, 'ironing');
- Populate ‘includes’ with values that fit table parameters. We have only displayed a portion here to conserve space.
    - INSERT INTO `includes` VALUES
      (1, 2, 1, 'classic morning'), (1, 1, 1, 'steak and eggs'),
      (1, 2, 1, 'continental'), (2, 5, 1, 'classic morning'),
      (3, 6, 1, 'steak and eggs'), (4, 9, 1, 'continental'),
      (5, 1, 1, 'classic morning'), (6, 4, 2, 'steak and eggs'),

*Populating Customer Database:*

- Only was used for initial database setup - thereafter, website handled this information
- Populate database with customer information; include values that fit table parameters
    - INSERT into customer (`username`, `password`, `name`, `email`, `address`, `PHONE_NUM`)
      VALUES
      ('tanviparikh', '$5$rounds=535000$Qz0XvJE8ANPjDgHS$7n2eKORfMtvxuOc6ghI13rKTpJpYTiUAkhtmWesd7G/', 'Tanvi', 'tanvi', 'boi', '1234567890'), ('hershpatel', '$5$rounds=535000$pATprEP/pcwTAZKO$CXGD8gBoNMNkI3eXZBVKS5iVMzmK6iKQj1MO4Viil.B', 'Hersh', 'hersh', 'yoyo', '1234567890'), ('saakethkrosuri', '$5$rounds=535000$e0FGSezaaidJF3w.$IjS6FKTVVt6zcG11yV8NcpPQZcqeEuwWD7Rz8xwzI3/', 'Saaketh', 'saaketh', '345', '1234567890'), ('madisonling', '$5$rounds=535000$D5QxFikU4BQMH0jp$ffRJoGp03haGV.SXgcLKiYbMsidUPLqXe6pkP2zLzs4', 'Madison Ling', 'mady', 'ergaerg', '1234567890'), ('admin', '$5$rounds=535000$xfS/wbqA4kMTOzj3$Cehq1Z3pGFVxkhSOuHxtnerRZVXFABe6bHVJ4OJOXt7', 'admin', 'admin', 'admin', '1234567890');

*Reservation Queries:*

- Query to output ‘hotel’ and ‘myroom’ information given check in and check out dates. We had to join multiple tables for this query, as the necessary information was spread out over several tables. ‘myroom’ contained the room information, ‘hotel’ contained the specific hotel and hotel location information. This was to find the room count per hotel.
    - SELECT DISTINCT r.hotelID, h.hotel_name, h.country, h.state, h.city, h.address, h.zipcode, ph.phone_no, COUNT(r.room_num) AS roomCount
      FROM hotel AS h
      LEFT JOIN myroom AS r
      ON h.hotelID = r.hotelID
      LEFT JOIN reserves AS re
      ON r.hotelID = re.hotelID
      AND r.room_num = re.room_num
      AND (re.inDate <= '2017-4-08' /*insert user input date here*/ OR re.outDate >= '2019-4-10'/*insert user input date here*/)
      LEFT JOIN room_discount AS rd
      ON r.hotelID = rd.hotelID
      AND r.room_num = rd.room_no
      AND (rd.sdate >= '2017-4-08' /*insert user input date here*/ AND rd.edate <= '2019-4-10'/*insert user input date here*/)
      INNER JOIN HotelPhoneNumbers AS ph
      ON h.hotelID = ph.hotelID
      WHERE re.invoiceNo IS NULL
      AND h.country = 'USA' /*insert user input date here*/ AND h.state = 'NJ' /*insert user input date here*/
      GROUP BY r.hotelID
      HAVING COUNT(r.room_num)>0
      ORDER BY r.hotelID, r.room_num;
- Query to output hotel and room information (including room discount if possible) given check in and check out dates. The room discount told the customer whether or not a discount would be offered during those dates. We also considered having a separate query to output discount dates for specific hotels, but we were told this would not be necessary.
    - SELECT DISTINCT r.hotelID, r.room_num, h.hotel_name, h.country, h.state, h.city, h.address, h.zipcode, r.price, r.capacity, r.floor_no, r.description, r.room_type, ph.phone_no, IFNULL(rd.discount, 'No Discount in period') AS DiscountPct
      FROM hotel AS h
      LEFT JOIN myroom AS r
      ON h.hotelID = r.hotelID
      LEFT JOIN reserves AS re
      ON r.hotelID = re.hotelID
      AND r.room_num = re.room_num
      AND (re.inDate <= '2017-4-08' /*insert user input date here*/ OR re.outDate >= '2019-4-10'/*insert user input date here*/)
      LEFT JOIN room_discount AS rd
      ON r.hotelID = rd.hotelID
      AND r.room_num = rd.room_no
      AND (rd.sdate >= '2017-4-08' /*insert user input date here*/ AND rd.edate <= '2019-4-10'/*insert user input date here*/)
      INNER JOIN HotelPhoneNumbers AS ph
      ON h.hotelID = ph.hotelID
      WHERE re.invoiceNo IS NULL
      AND h.country = 'USA' /*insert user input date here*/ AND h.state = 'NJ' /*insert user input date here*/
      ORDER BY r.hotelID, r.room_num;

- Query to insert reservation values into database. These are separate from the other insert queries above, since for each input there would be information inserted into multiple tables, not just one. There is also information that needs to be extracted from other tables, such as CID and InvoiceNo. There are also certain values that are calculated rather than manually inputted by the user, such as total amount.
    - INSERT INTO `customer`(`username`, `password`, `name`, `email`, `address`, `PHONE_NUM`)VALUE(/*auto CID*/'tanvi','123','Tanvi','tanvi@rutgers','rpo way','999123');
      INSERT INTO `reservation`(`CID`, `totalAmt`, `resDate`)
      VALUES(/*auto InvoiceNo*/ /*CID*/ 6, /*totalAmt*/ 121.12, NOW());
    - INSERT INTO `reserves`(`invoiceNo`, `room_num`, `hotelID`, `noOfDays`, `inDate`, `outDate`)
      VALUES(/*get invoiceNO from reservation*/(SELECT MAX(invoiceNo) AS invNo FROM reservation), /*user input room numer*/101, /*hotelID query*/(SELECT h.hotelID FROM hotel AS h WHERE h.hotel_name = 'Hyatt'/*userinput*/ AND h.country = 'USA'/*userinput*/ AND h.state = 'NJ'/*userinput*/ AND h.city = 'New Brunswick'/*userinput*/), /*calc noOfDays*/(DATEDIFF('2017-12-21', '2017-12-18')), '2017-12-18','2017-12-21'/*these dates should be user input*/);
    - INSERT INTO `includes`(`invoiceNo`, `hotelID`,`bType`, `num_of_breakfasts`)
      VALUES(/*get invoiceNO from reservation*/(SELECT MAX(invoiceNo) AS invNo FROM reservation), /*hotelID query*/(SELECT h.hotelID FROM hotel AS h WHERE h.hotel_name = 'Hyatt'/*userinput*/ AND h.country = 'USA'/*userinput*/ AND h.state = 'NJ'/*userinput*/ AND h.city = 'New Brunswick'/*userinput*/), /*userinput bType*/'steak and eggs', /*userinput*/ 5);
    - INSERT INTO `contains`(`invoiceNo`, `hotelID`,`sType`, `num_of_services`)
      VALUES(/*get invoiceNO from reservation*/(SELECT MAX(invoiceNo) AS invNo FROM reservation), /*hotelID query*/(SELECT h.hotelID FROM hotel AS h WHERE h.hotel_name = 'Hyatt'/*userinput*/ AND h.country = 'USA'/*userinput*/ AND h.state = 'NJ'/*userinput*/ AND h.city = 'New Brunswick'/*userinput*/), /*userinput sType*/'spa', /*userinput*/ 6);
- Query to output service options given hotelID. Uses information from multiple tables, since service information is located in ‘services’ and hotel information is stored in ‘hotel’
    - SELECT h.hotelID, S.stype, S.sCost
      FROM hotel AS h, services AS S
      WHERE h.hotelID = S.hotelID AND (h.hotelID = 6 /*insert user input hotelID here*/);
- Query to output breakfast options given hotelID. Uses information from multiple tables, since bfast information is located in ‘breakfast’ and hotel information is stored in ‘hotel’
    - SELECT h.hotelID, B.btype, B.description, B.bprice
      FROM hotel AS h, breakfast AS B
      WHERE h.hotelID = B.hotelID AND (h.hotelID = 1 /*insert user input hotelID here*/);

*Review Queries:*

- Query outputs invoice number, all hotels, and number of rooms of each past reservation
    - SELECT DISTINCT reserves.invoiceNo, hotel.hotel_name, reserves.inDate, reserves.outDate,
      COUNT(reserves.room_num) AS numRooms
      FROM reserves, reservation, hotel
      WHERE reserves.invoiceNo = reservation.invoiceNo AND reserves.hotelID = hotel.hotelID
      AND CID = '2'/*insert user CID here*/ AND reserves.outDate < NOW() AND reserves.inDate < NOW()
      GROUP BY reserves.invoiceNo, hotel.hotel_name, reserves.inDate, reserves.outDate
      ORDER BY reserves.inDate;
- Query to insert room reviews into corresponding tables. User enters invoice number, and chooses type of review.

- INSERT INTO `review` (`CID`, `rating`, `textcomment`, `review_type`)
  VALUES (/*input CID*/ CID, /*user rating*/ 10, /*user comment*/ 'was dope','Room');
  INSERT INTO `room_review` VALUES ((SELECT MAX(reviewID) AS revID FROM review), /*user room_num*/ 101, /*user hotelID*/ 1);

- Query to insert breakfast reviews into corresponding tables. User enters invoice number, and chooses type of review.
  - INSERT INTO `review` (`CID`, `rating`, `textcomment`, `review_type`)
    VALUES (/*input CID*/ CID, /*user rating*/ 9, /*user comment*/ 'was great','Breakfast');
    INSERT INTO `breakfast_review` VALUES ((SELECT MAX(reviewID) AS revID FROM review), /*user b_type*/ 'steak and eggs', /*user hotelID*/ 2);
- Query to insert service reviews into corresponding tables. User enters invoice number, and chooses type of review.
  - INSERT INTO `review` (`CID`, `rating`, `textcomment`, `review_type`)
    VALUES (/*input CID*/ CID, /*user rating*/ 2, /*user comment*/ 'was terrible','Service');
    INSERT INTO `service_review` VALUES ((SELECT MAX(reviewID) AS revID FROM review), /*user s_type*/ 'spa', /*user hotelID*/ 3);
- Query to list type of breakfast user ordered. User enters invoice number, and chooses type of review.
  - SELECT DISTINCT reserves.hotelID, hotel.hotel_name, reserves.invoiceNo, reserves.inDate, reserves.outDate, includes.bType, breakfast.bPrice, breakfast.description
    FROM reserves, includes, reservation, breakfast, hotel
    WHERE reserves.hotelID = hotel.hotelID AND reserves.hotelID = includes.hotelID AND reserves.invoiceNo = reservation.invoiceNo
    AND includes.invoiceNo = reserves.invoiceNo AND breakfast.bType = includes.bType AND breakfast.hotelID = reserves.hotelID
    AND reserves.invoiceNo = '1'/*insert user Invoice Number here*/;
- Query lists types of services user has used. User enters invoice number, and chooses type of review.
  - SELECT DISTINCT reserves.hotelID, hotel.hotel_name, reserves.invoiceNo, reserves.inDate, reserves.outDate, `contains`.sType, services.sCost
    FROM reserves, `contains`, review, reservation, services, hotel
    WHERE reserves.hotelID = hotel.hotelID AND reserves.hotelID = `contains`.hotelID AND reservation.CID = review.CID
    AND reserves.invoiceNo = reservation.invoiceNo
    AND `contains`.invoiceNo = reserves.invoiceNo AND services.sType = `contains`.sType AND services.hotelID = reserves.hotelID
    AND reserves.invoiceNo = '1'/*insert user Invoice Number here*/
    AND review.review_type = 'Service';
- Query lists types of rooms user has stayed at. User enters invoice number, and chooses type of review.
  - SELECT DISTINCT reserves.hotelID, hotel.hotel_name, reserves.invoiceNo, reserves.inDate, reserves.outDate, myroom.room_num, myroom.room_type
    FROM reserves, review, reservation, myroom, hotel
    WHERE reserves.hotelID = hotel.hotelID AND reserves.hotelID = myroom.hotelID AND reservation.CID = review.CID
    AND reserves.invoiceNo = reservation.invoiceNo
    AND reserves.room_num = myroom.room_num
    AND reserves.invoiceNo = '1'/*insert user Invoice Number here*/
    AND review.review_type = 'Room';

*Statistic Queries:*

- Given check in and check out dates, query outputs highest rated breakfast type for each hotel.
    - SELECT br.bType, ROUND(SUM(re.rating)/COUNT(b.bType),3) AS Average
      FROM breakfast AS b, review AS re, breakfast_review AS br, reserves AS res
      WHERE re.reviewID = br.reviewID AND br.btype = b.bType AND (res.inDate >= '2017-4-08' /*insert user input date here*/ AND res.outDate <= '2019-4-10'/*insert user input date here*/)
      GROUP BY br.btype
      ORDER BY SUM(re.rating)/COUNT(b.btype) DESC
      LIMIT 1 ;
- Given check in and check out dates, query outputs highest rated service type for each hotel.
    - SELECT s.sType, ROUND(SUM(re.rating)/COUNT(sr.sType),3) AS Average
      FROM services AS s, review AS re, service_review AS sr, reserves AS res
      WHERE re.reviewID = sr.reviewID AND sr.sType = s.sType AND (res.inDate >= '2017-4-08' /*insert user input date here*/ AND res.outDate <= '2019-4-10'/*insert user input date here*/)
      GROUP BY s.sType
      ORDER BY SUM(re.rating)/COUNT(sr.sType) DESC
      LIMIT 1 ;
- Given check in and check out dates, query outputs highest rated service type across all hotels
    - SELECT r.hotelID, r.room_type, ROUND(SUM(re.rating)/COUNT(r.room_num),3) AS Average
      FROM myroom AS r, reserves AS res, review AS re, room_review AS rr
      WHERE re.reviewID = rr.reviewID AND rr.room_num = r.room_num AND rr.hotelID = r.hotelID AND r.hotelID = 1 /*insert hotelID here*/ AND (res.inDate >= '2017-4-08' /*insert user input date here*/ AND res.outDate <= '2019-4-10'/*insert user input date here*/)
      GROUP BY r.hotelID, r.room_type
      ORDER BY SUM(re.rating)/COUNT(r.room_num) DESC
      LIMIT 1 ;
- Given check in and check out dates, query outputs 5 best customers in terms of money spent on reservations
    - SELECT reserves.invoiceNo, reserves.room_num, reserves.hotelID, SUM(breakfast.bPrice*includes.num_of_breakfasts)
      FROM reserves, breakfast, includes
      WHERE reserves.hotelID = breakfast.hotelID AND includes.bType = breakfast.bType
      GROUP BY reserves.invoiceNo;