Milestone 1

UCR Hotel Booking System:

**Hotel Homie**

Team 3

Michael Cole

Tracy Fong

Caleb Martinez

Zachary Sells

Sam Sepasi

**Table of Contents**

1. Introduction …………………………………………………………………………...…3
2. System Overview………………………………………………………………………..3
3. Design/System Architecture…………………………………………………....……3-4
4. Requirement Specification…………………………………………………………...4-5
5. Milestone 1 New Features…………………………………………………..……....6-7
6. System Flow………………………………………………………………………....7-11
7. Operating Manual ……………………………………………………...……….11-24
8. Testing……………………………………………………………………………….….24
9. Glossary…………………………………………………………………………..…….25
10. **Introduction**

The purpose of this document is to a provide detailed description for the design and use of our system. We made this system for our CS180 Introduction to Software Engineering class project. The system is an online hotel reservation system that allows users to search for hotels and make reservations. To accomplish this, we used the EAN API for the hotel information and pictures, Ruby on Rails for the system architecture, and HTML/CSS, Bootstrap for the user interface design.

We added to our original system with some additional features. These features were added after Milestone 0. Users are now able to refine their search based on price, rating, and amenities. They can also sort their search results by price, average rating, and distance. There is now also a map that displays the location of the hotel. Users also have their default payment information that they can save and reuse. The details are discussed in section 5, Milestone 1: New Features.

1. **System Overview**

Hotel Homie is an online hotel reservation system that allows users to search for hotels and make reservations. They can search based on the name of the hotel, city, state, and/or point of interest. After milestone 1, users can now filter results based on Price, Rating, and Amenities. Based on the input from the user, Hotel Homie will display a list of available hotels that the user can reserve. This list can be sorted by price, rating, and distance. With an account, users can view, make, and pay for a hotel reservation. Without an account, users can only search and view the available hotels. Users are also able to cancel or edit their reservation and make reviews about hotels they have reserved previously.

The system uses Ruby on Rails for its application framework and is currently available to the public in the following link: <https://salty-anchorage-1656.herokuapp.com/>

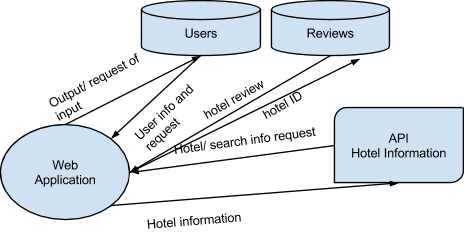
1. **Design**

These are the tools and technology that we used for this system:

* GitHub
* Bootstrap
* HTML5
* CSS
* JS
* Ruby on Rails v 4.2.1
* EAN API
* Leaflet v 0.7.3

For this project we used Ruby on Rails to create our back-end infrastructure, but most of our data came from an API. This API was free and available to all for hotel and flight reservations (EAN API). This API made our data retrieving and storing tasks much easier and faster. We retrieved most of our data using the API commands through our Ruby code. We were able to request a summary of a specific hotel by sending a request with Rails code and in response, we received the requested information. With the information we received from the EAP API, we embedded it into the HTML files for the UI using Ruby on Rails.

We created a database in Rails to store the user reviews of the hotels they have reserved before. We indexed the reviews by the hotel’s unique IDs given by the API and stored the review information in the corresponding element of our index. We retrieved the reviews also by Rails code and requests. The user’s profile information is also saved in a similar manner, where each user has their own unique ID as the primary key.



*Diagram of the system’s architecture.*

1. **Requirement Specification**

*Milestone 1:*

1. In the page of each hotel, there should be a map showing the location of

the hotel.

2. Users can filter search results by features of hotels such as star level, price,

free wifi, free parking etc. For example, users may only want hotels with free parking and free wifi or hotels under $200 per night.

3. Users can sort search results by distance, price or average rate, this sorting is done by requesting the API for the desired sorting option and then sorting it on our server side to assure that we have the most accurate sorting done.

4. Users can enter their personal information and default payment information.

When they reserve a hotel, they can choose to use the default information to make reservation and payment.

*Milestone 0:*

1. Users are able to log in their account or create new account. To sign up an account, a username and password is needed. Only logged in users can reserve a hotel. There

should also be a page for users to see their profile and change password.

2. Users can search hotels by hotel name, point of interest or city and also input check-in date and check-out date.

3. There should be a page to show a list of hotel as search results. In the list, Each hotel

should show its name, city, price, average rate and number of reviews, star level and a

thumbnail of the hotel’s picture.

4. Each hotel should have a page showing information of the hotel. The information will

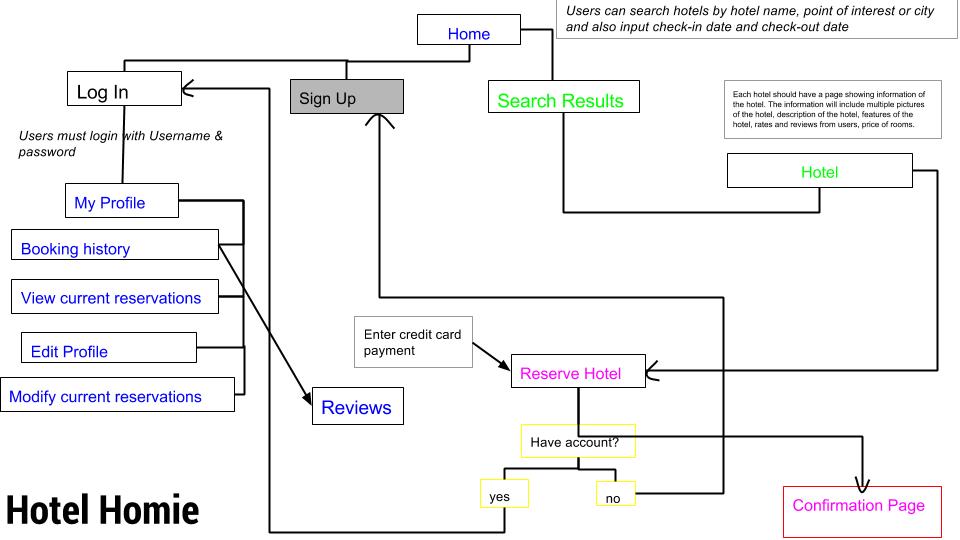
include multiple pictures of the hotel, description of the hotel, features of the hotel, rates

and reviews from users, price of rooms.

5. Users can reserve rooms and enter payment information and confirm the reservation.

6. Users can review their reservation and booking history. They are able to cancel or make changes to their upcoming reservation. A reservation will be automatically moved to history. Users can rate and write review to the hotels in their booking history.

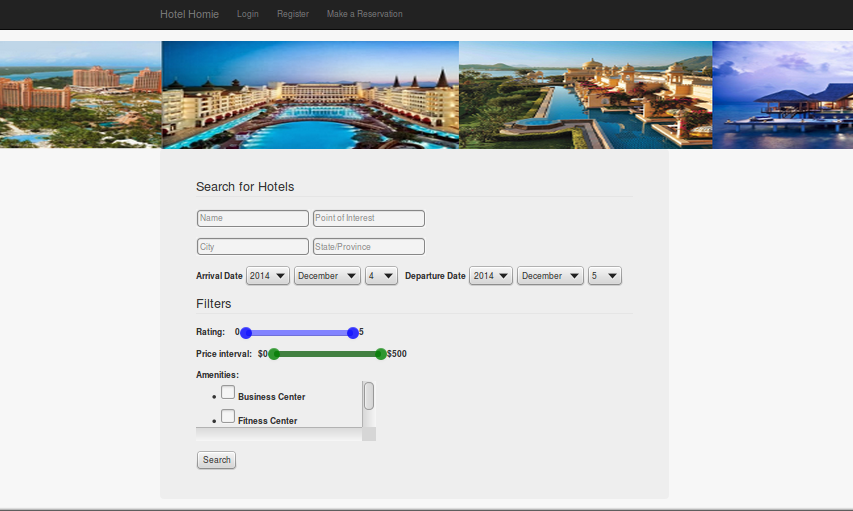
1. **Milestone 1: New Features**
2. On each hotel page, there is now a map displaying its location from the hotels latitude and longitude. This was implemented first by getting the new latitude and longitude data from the API and then storing it with the rest of the hotel information. The map itself is generated using an open source map called leaflet js. We pass in our latitude and longitude into the API and it generates a map of our given size at the location of the hotel and with a marker pointing to the hotel.
3. On the homepage, users can now filter search results by price, rating, and amenities. For example, users may only want to see hotels that cost less than 100$ per night, whose rating is between 4 and 5, and whose amenities include an indoor pool. With milestone 1, this is now possible. The price and rating filtering was implemented by using the API’s filtering methods. However, the amenity filtering required the use of a bit-mask. Each hotel has an associated amenities bit-mask which we used to compare against the filter. If the bit-masks matched, the hotel will be returned to the user.
4. Users can now sort hotel results by price, rating, and distance. After the initial search is done user is redirected to our getList page which shows them the hotel results based on a default sort which is ordered in best deals available at the moment. The user is then has the option to sort the results by price (low to high), rating (high to low) and distance (closest to initial location entered). The use can change the sorting using the buttons on top of the screen and these can be changed multiple times, ie. sort by price then rating and distance after.
5. Users can now save their personal credit card information to make hotel reservation quicker. On the user profile page, there is a button labeled “edit default information”. The user may click on this button to navigate to a form where they can enter and save their default payment information. After they have saved their default payment information, a button labeled “use default information” will appear at the bottom of the payment form whenever they make a reservation. They can auto-fill the form with their default information by clicking on this button.
6. **System Flow**



*The flow diagram above is a diagram that shows which page goes where depending on what the user does. Note that the color does not represent anything related to the actual flow.*

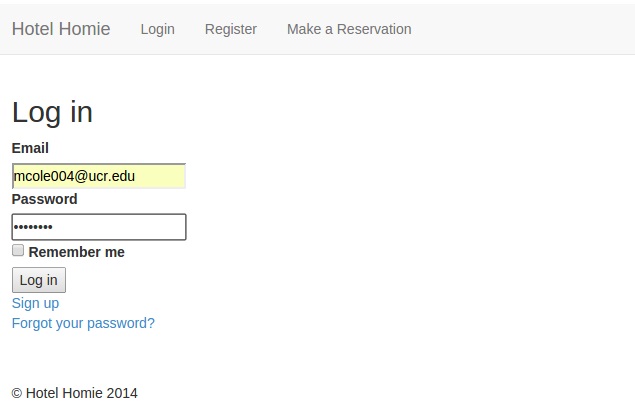
*Home page*

Users are able to search the hotel by name, point of interest, or city, and put in their desired check in check out page. They can also navigate to the login page or sign up for an account if they do not already have one. The home page was modified from the Rails default to redirect each page using encapsulate Ruby. We used Bootstrap for the styling of the pages. Users are now able to filter their searches by prices, ratings, and/or the amenities they want.



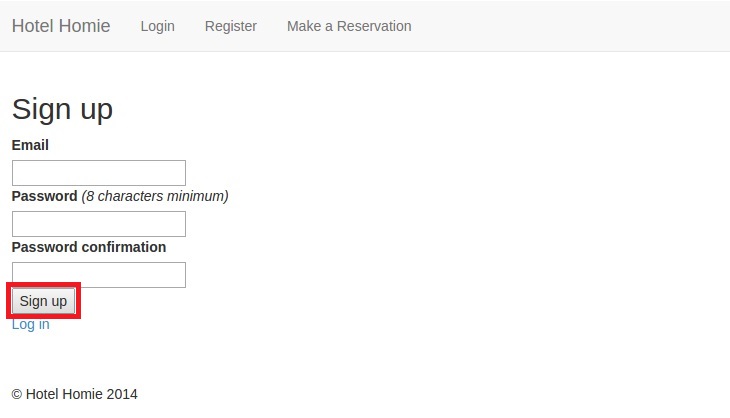
*Login Page*

Users must enter their username and password in order to login to the profile. On their profile, they can view their booking history, view their current reservations, or edit their profile. From the booking history, users can make reviews on hotels they have reserved before. This will take them to the review page that allows users to rate the hotel. The login page was made using the Ruby gem, Devise, which is an authentication tool that we used for the login page.



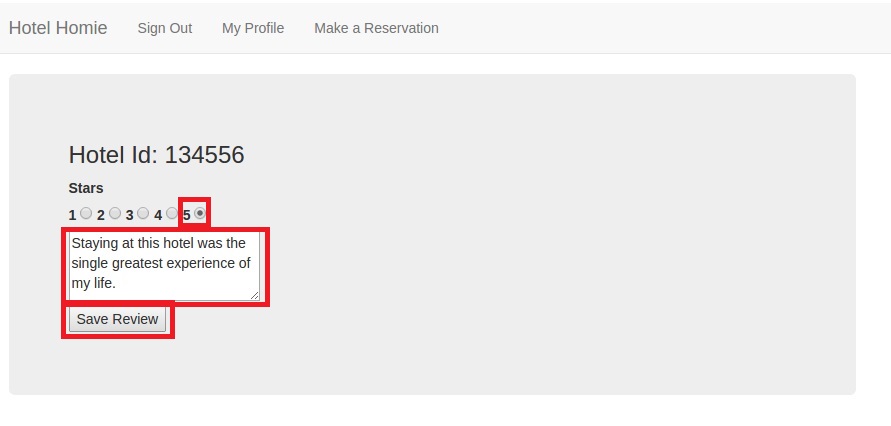
*Register/Sign Up Page*

All users need to sign up is an email and password. Passwords must be a minimum of 8 characters. This page was also made using the Ruby gem Devise.

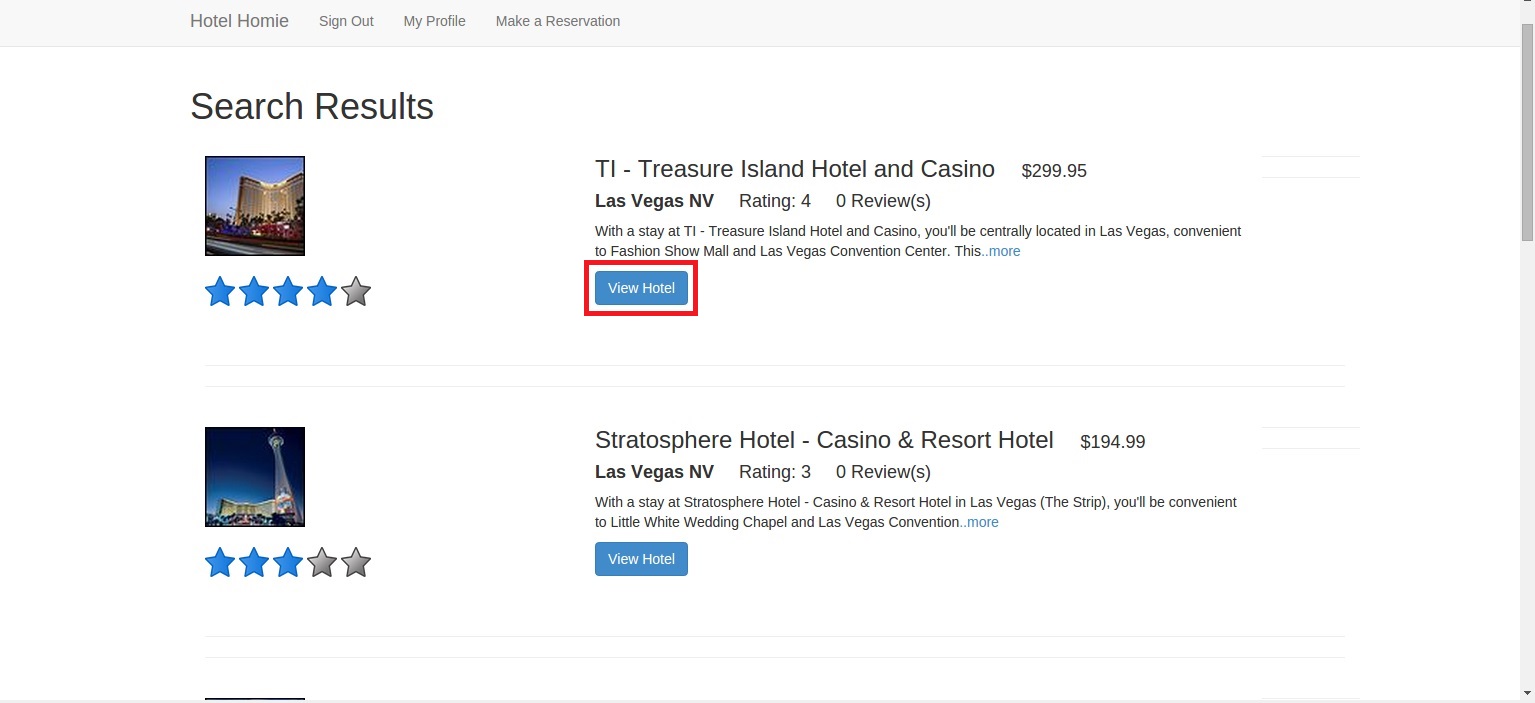


*Reviews*

Users can reviews on the hotels they have stayed at before. Total number of reviews from the user. Star level/Rating

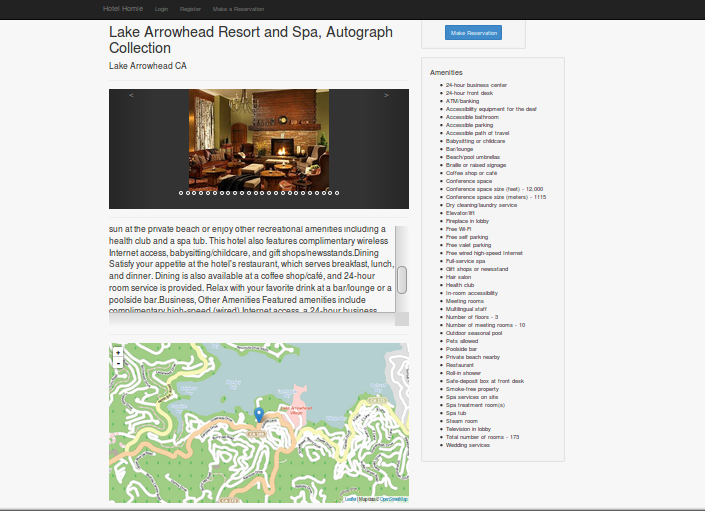


*Search Results Page*

After searching for their desired location/point of interest, users are taken to a results page the lists all the possible hotels with a thumbnail picture of each hotel, price, average rating, number of reviews, and a description of the hotel. This information we requested from the EAN API. 

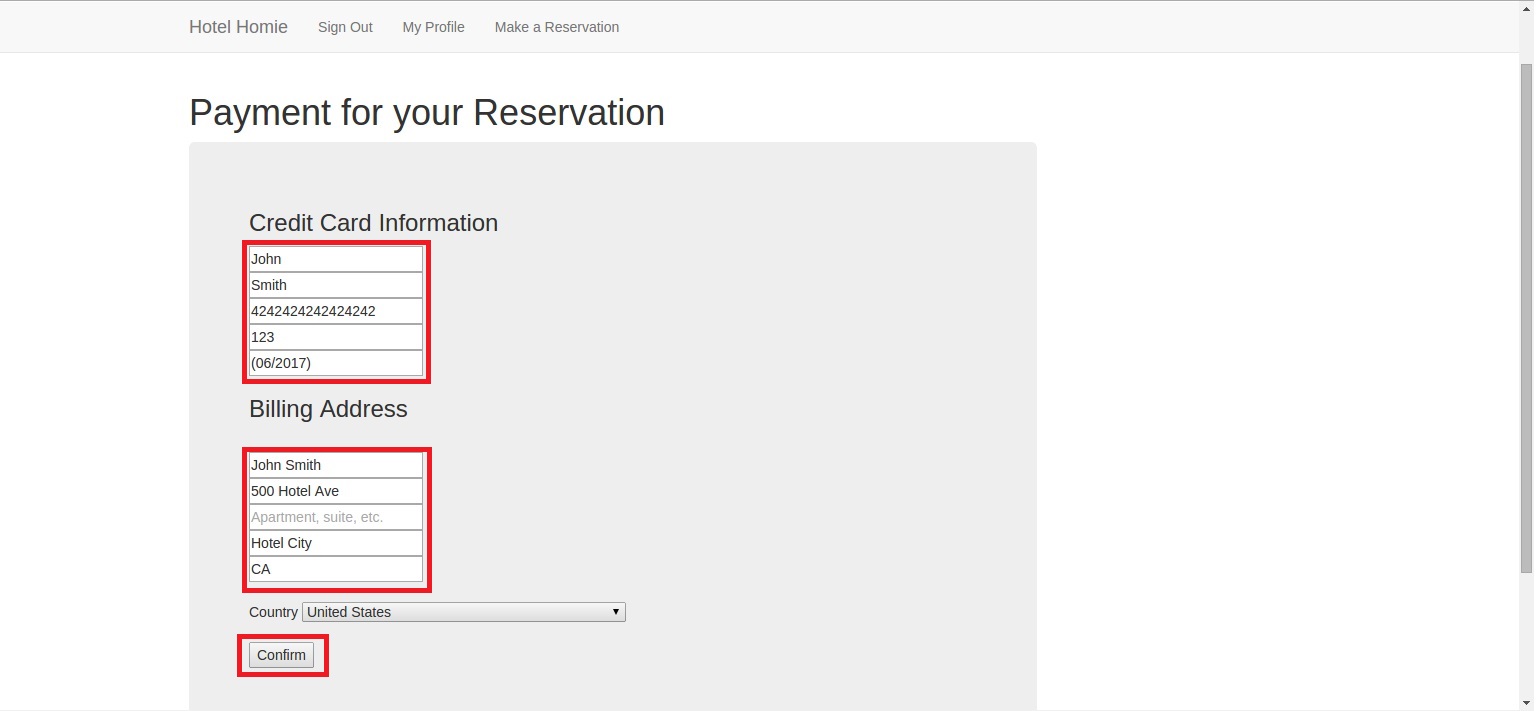
*Hotel Page*

Users can choose to see more information on a particular hotel from the list in the search results page. More information about the hotel can be found here such as a list of all the amenities on the right, user reviews on the hotel on the bottom, more pictures of the hotel, and a longer description of the hotel. The carousel for the pictures was made using Bootstrap. The hotel information and pictures are requested from the EAN API. The user reviews are currently self populated. They can also sort their search results based on distance, price, and rating.



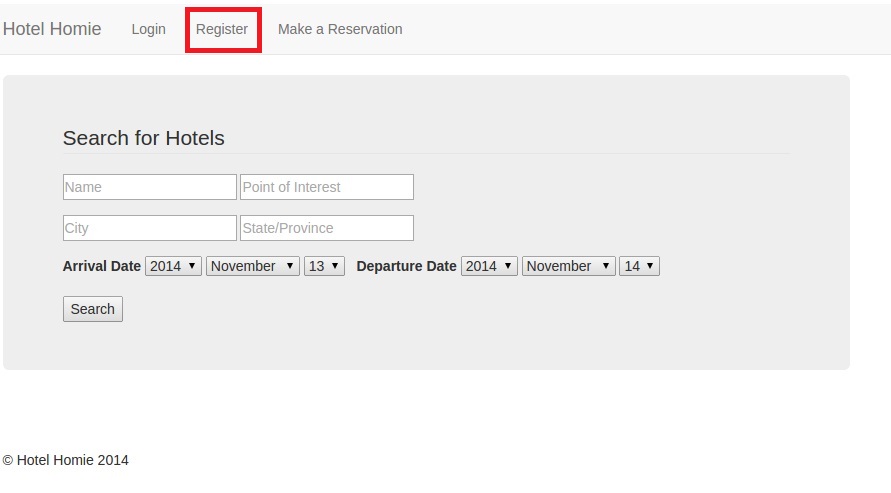
*Reservation page*

After the user selects their desired hotel, they can make a reservation. They must pay with a credit card. Their credit card information is entered into this page.



1. **Operating Manual**

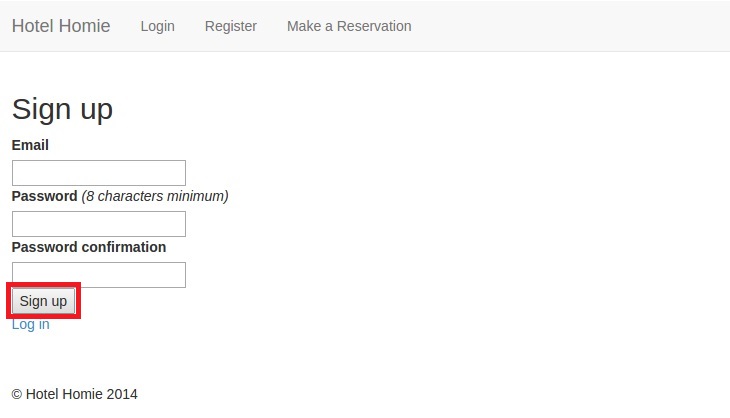
*Homepage*



There are only a few elements of UI on the homepage. The search form is used to search for hotels by Hotel Name, destination city and state. A point of interest such as the Eiffel Tower can also be used instead. When the user sets their arrival and departure date here, the data is automatically passed on so when they make a reservation, it knows which dates to reserve for.

The top navigation bar is present on every page and contains helpful links to navigate the site. When the user does not yet have an account, they can register from any page to get one by clicking the “Register” button on the bar. If they have an account but are not signed in, then they may click the “Login” button to be redirected to the login page.

*Register Page*



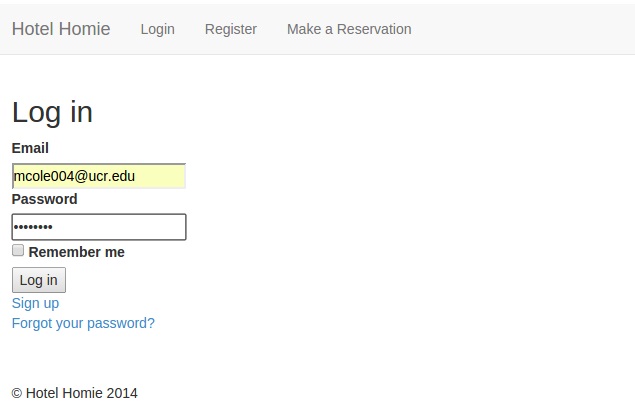
This page contains a form that users can use to register for use of the site. It requires a valid email and a password of at least 8 characters. There is also a link to the login page in case the user clicked “Register” on accident.

*Registration Error Page*



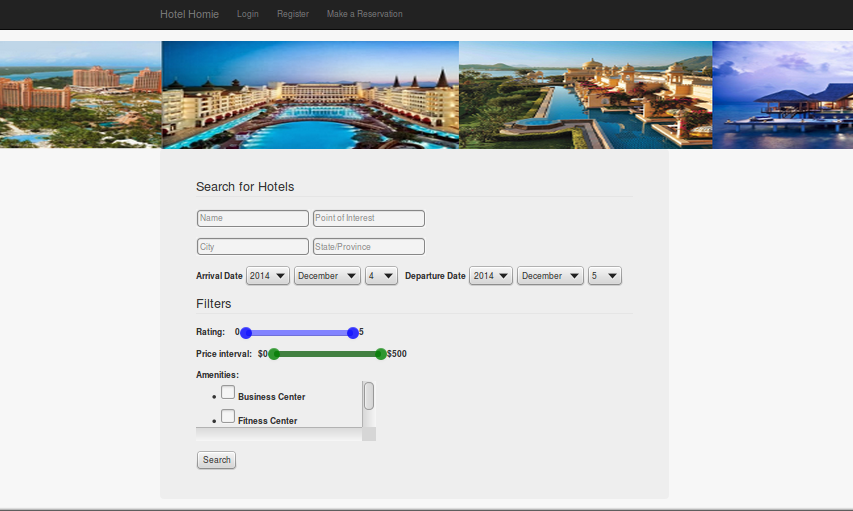
This is the same page as the Register page but is reached when a user inputs an invalid password or an invalid email. It’s function is the same as the Register page.

*Login Page*



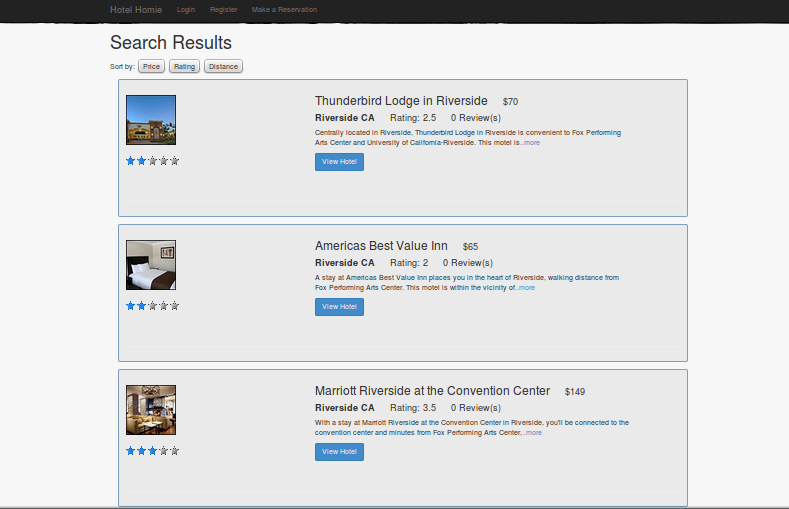
This page’s purpose is to provide a short form for a previously registered user to login using their email and password. It also contains a “Remember me” checkbox in case the user does not wish to sign in to the website every time they visit. There is also a link to the forgotten password page where the user can send themselves an email that will allow them to reset their password.

*Homepage -Search*



This is where the user conducts their search for hotels. This form takes the information the user gives, sends it to the API, and redirects the user to a results page showing the results of their search. The main fields are, Hotel Name, Point of Interest, City, State/Country, and Arrival/Departure dates. It also has a range slider for the prices and ratings. There is also a checklist for the desired amenities.

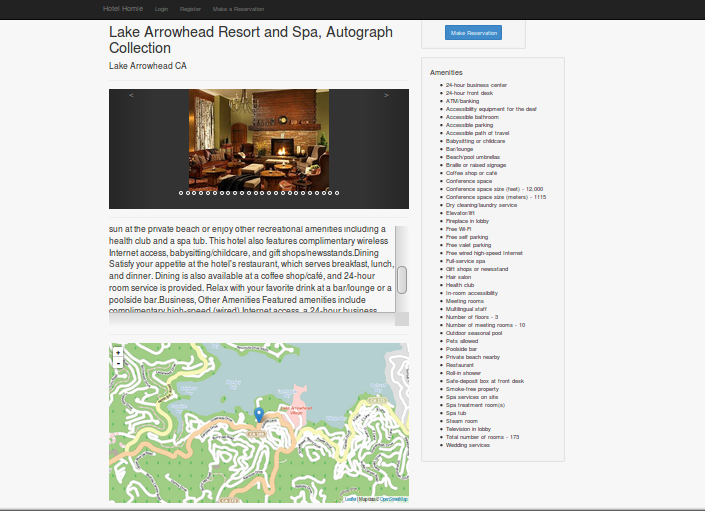
*Search Results*



Here is where the user is shown the results of their search. It is a list of 10 hotels all fitting the users search criteria. Each hotel results contains, Hotel Name, Price, Average Star Rating, current number of Reviews on our site, City, State, Brief Description, and a button to the Hotel Page. Clicking the more will expand the description of the hotel.

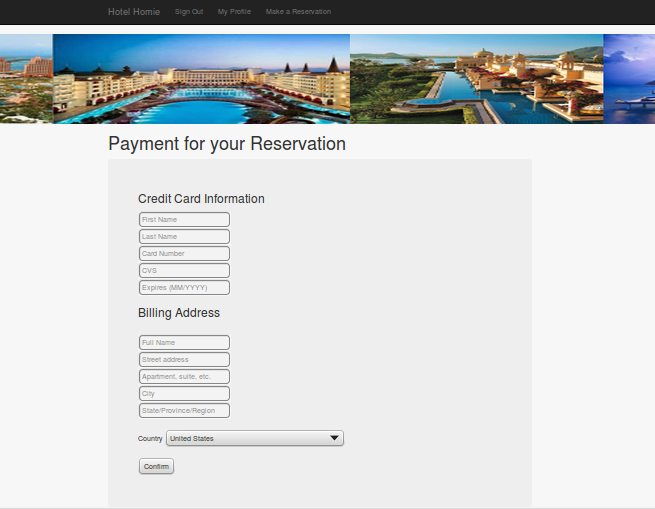
Users are now able to sort their search results by distance, price, or average rate. For distances, it will sort from shortest distance to farthest distance. For ratings, it will sort them by highest rating to lowest rating. For prices, it will sort by lowest price to highest price.

*Hotel Page*



The Hotel page contains a large amount of information about each hotel. It contains a gallery of various pictures from the hotel to show the User what the interior and exterior look like. Amenities that are offered are also listed on the right side to tell the User what the hotel has to offer. A brief description of the hotel is also given below the gallery of pictures to describe the hotel. In the upper right is the “Make a Reservation” button that the User can click to reserve a room at this hotel. Users can also view a map with the hotel’s location on it.

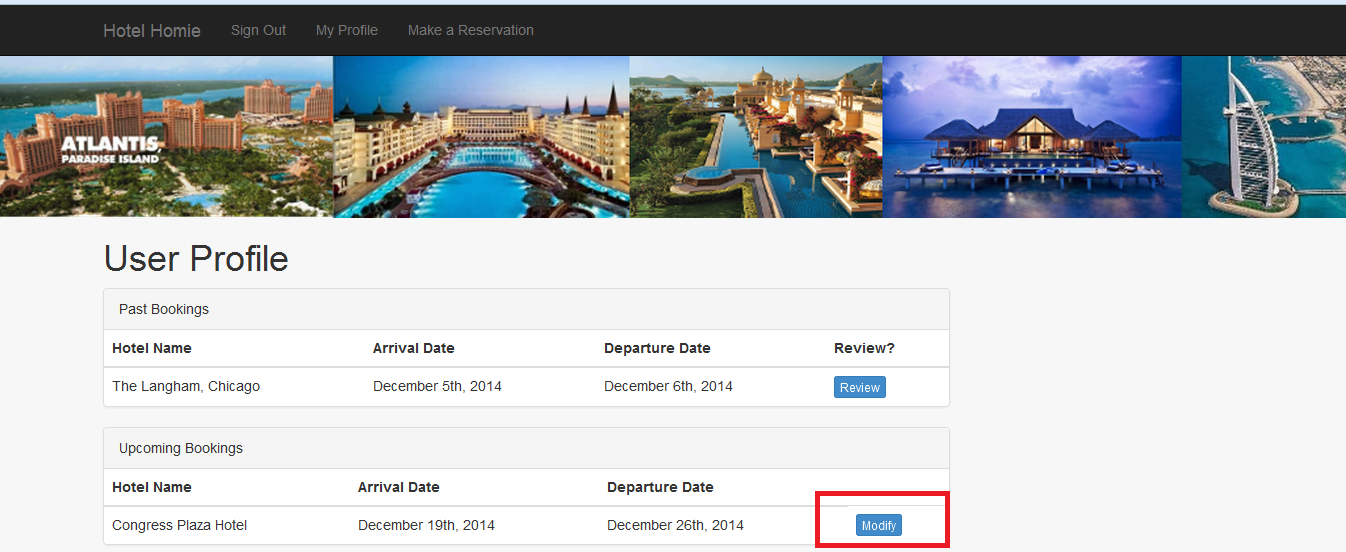
*Reservation Credit Card Page*

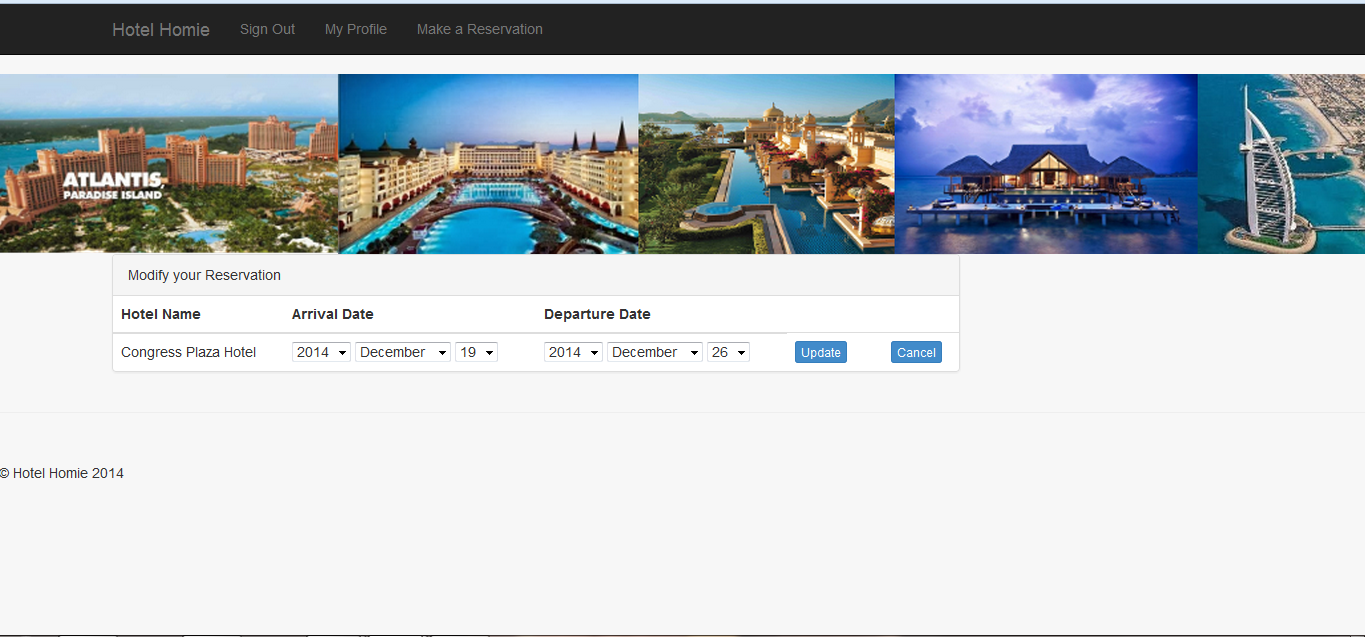


Here is where the User enters all their credit card information as well as their billing address. The card is checked for validity when the confirm button is pressed to make the reservation, and if the credit card number is not valid then the reservation will not go through. They can now save their default credit card information and use that for future reservations.

*Modify Reservations Page*

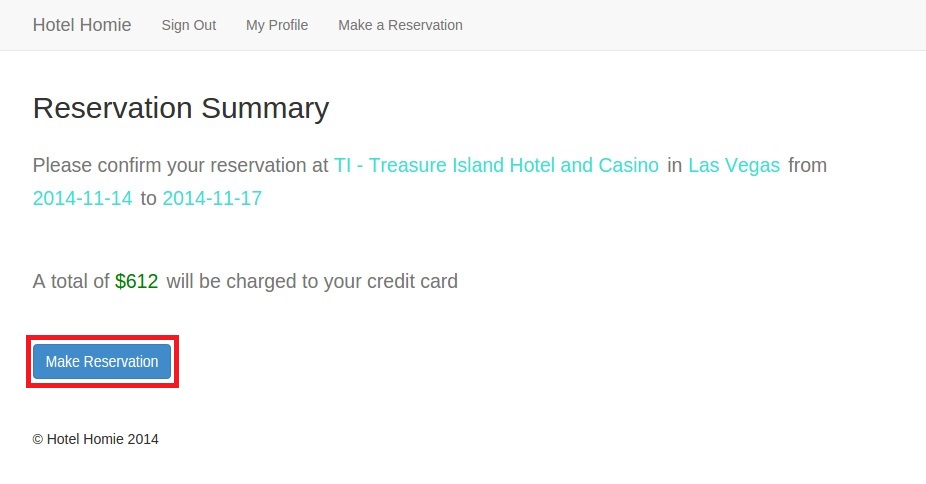
They can also modify their reservation(s) or cancel it.



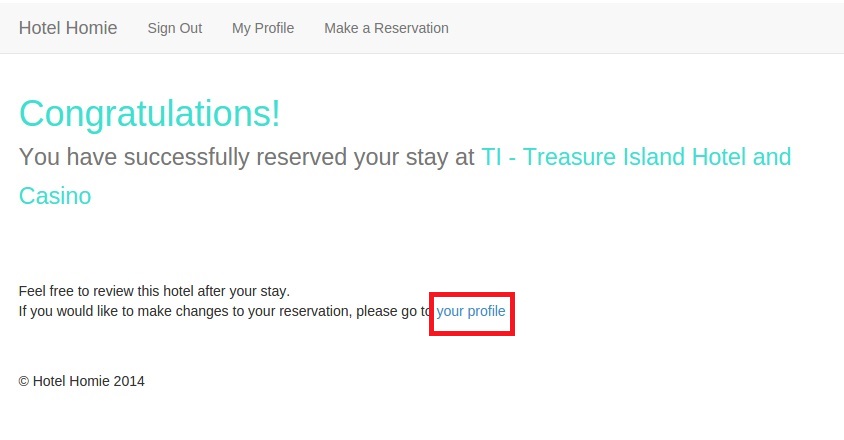


*Reservation Confirmation Page*

This is the final page before the reservation is finalized and saved to the database. It gives the total amount that will be charged to the credit card as well as stating the days that the reservation is good for. Clicked the “Make Reservation” button sends the User to the congratulations page.

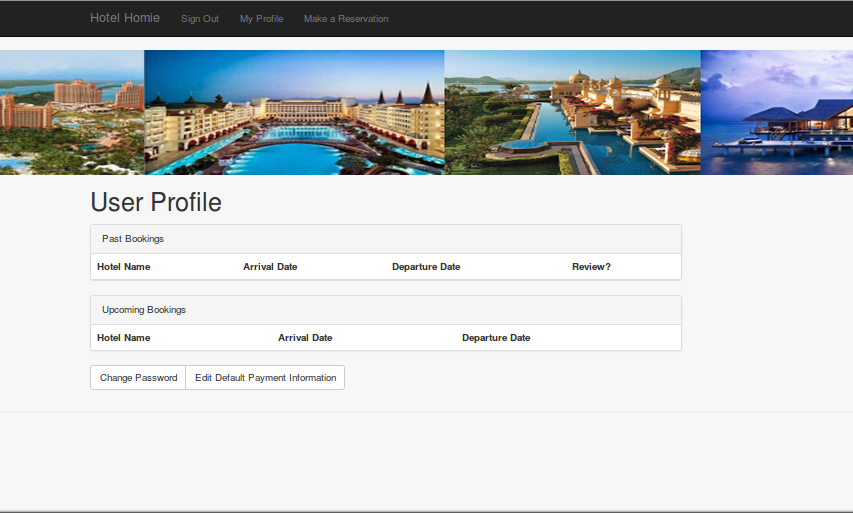


*Congratulations Page*



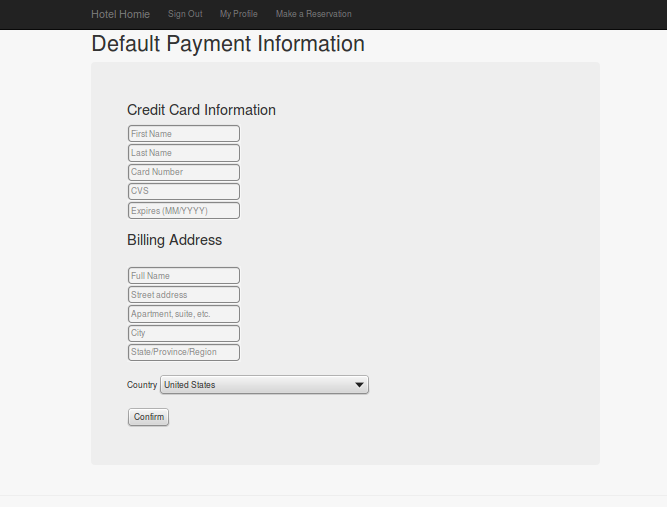
After successfully booking the hotel, this page will congratulate the User on a successful reservation and to redirect the user to the profile page if they choose to do so.

*Profile Page with Reservation*



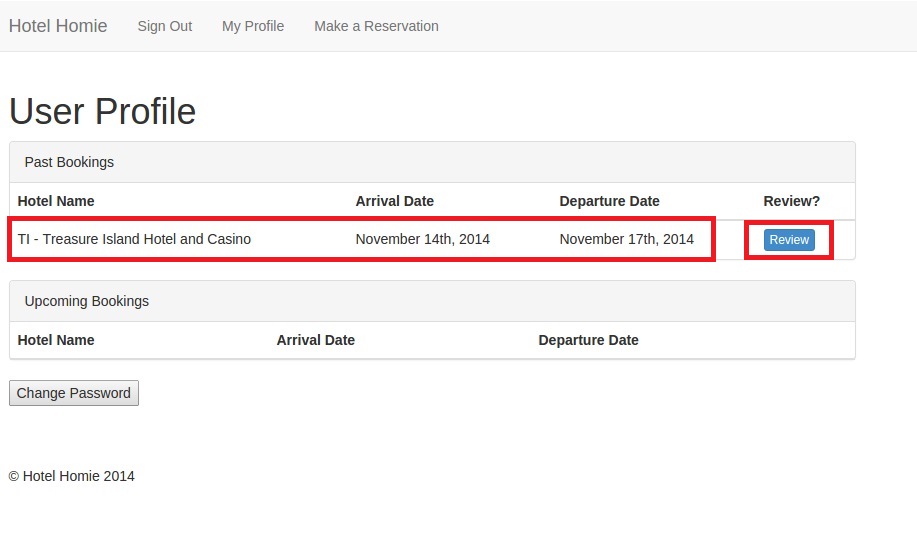
Their user profile page has been changed slightly because the reservation got added. In the table labeled “Upcoming Bookings,” the reservation the User made has appeared. When the date passes by, it will be moved to their “Past Bookings.”

*Default Payment Page*

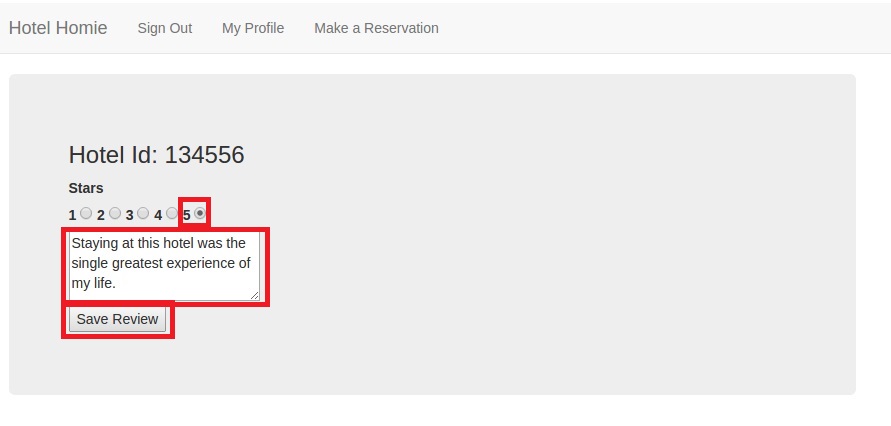


This page allows users to store credit card and billing information for ease of access when making reservations.

*Profile Page Review*

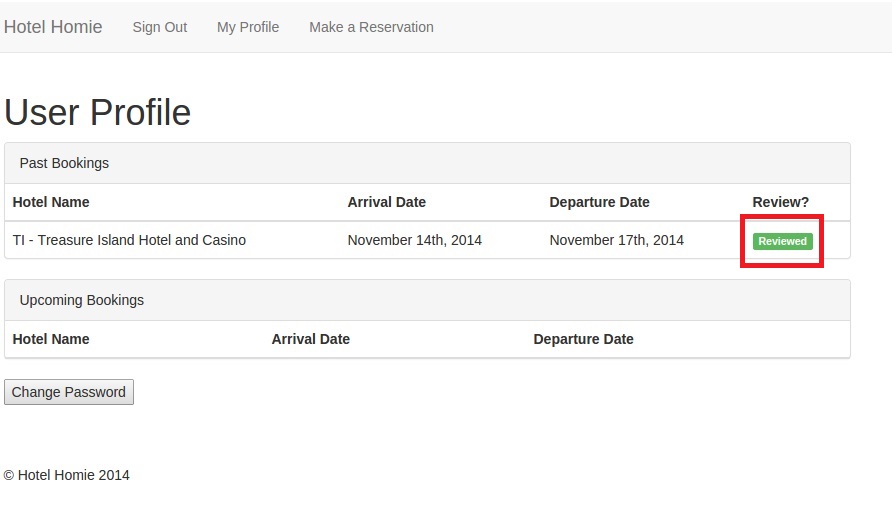


Once the date has passed, the reservation information gets moved to the “Past Bookings” and a review button appears to the right of it. The User can click on this to review the hotel.

*Leave a Review Page*

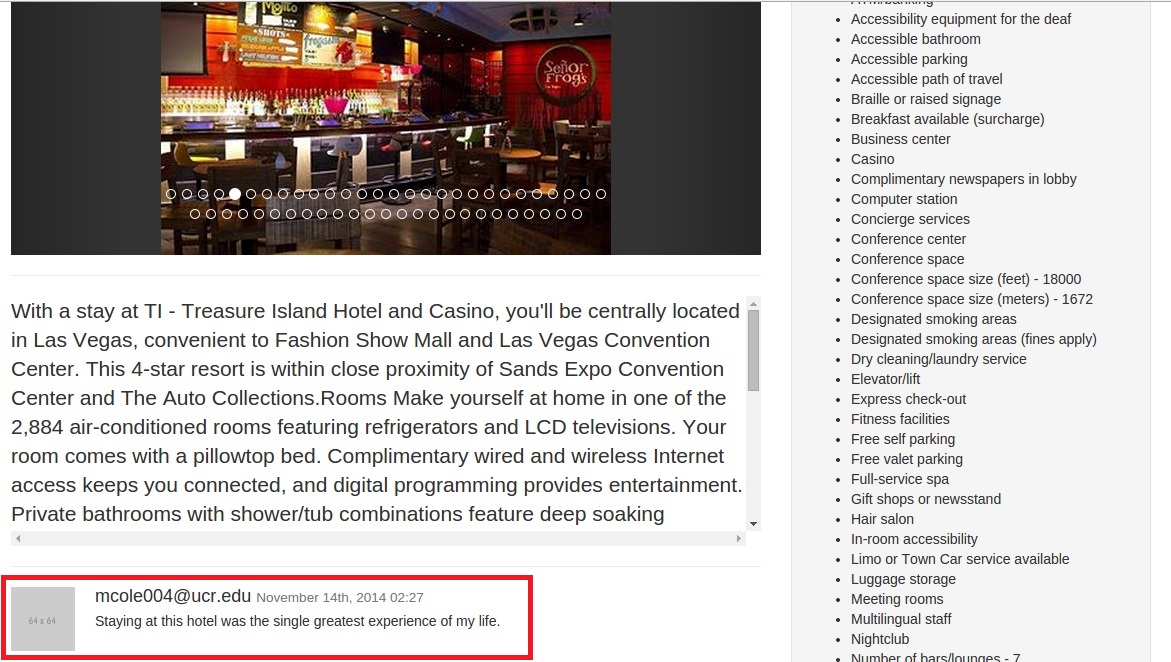
Here the User gives the hotel a star rating from one star to five stars, and they write their review in the textbox below. Clicking the “Save Review” button will save their review to the database and whenever a User visits the hotel page for that hotel, they will see the review.

*Profile Page After Review*



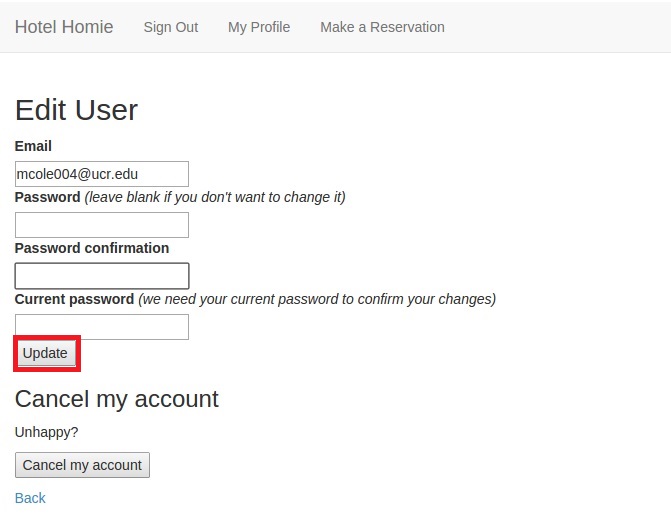
The past booking has been updated to show that the User reviewed that hotel and to stop them from leaving multiple reviews for a single visit.

*Hotel Page with Review*



Here is the updated hotel page that has the review just below the description of the hotel. It has the review text as well as the user email who left the review and the date and time they left it. All other Users are able to see this review on this particular hotels page.

*Change Password Page*



Anytime from the user profile page, the User can click the “Change Password” link to come to this page. This page allows the user to change their old password by confirming it, and then entering their new password twice. It will not work if the new password fields are left blank.

1. **Testing**

Our testing was mostly done by manually checking the different components of site: user signup and sign in, searching, viewing, reviewing, and reservation/ payment. We tested these components with varying inputs, valid and invalid and tried to look for any bugs or false behavior. After finally completing all milestone requirements, we checked the functionality of our website by trying all possible solutions and looking for any errors. Furthermore, we will check the security of our payment page by ensuring rigorous back-end code and PCI compliant procedure. This is done by using the devise gem that has a great reputation in the RoR community.

1. **Glossary**
2. Bootstrap - is a collection of CSS, fonts and js code made available to the web designer to use and edit to make the site’s look professional.
3. database - organized collection of information usually stored in tables
4. EAN API- Expedia Affiliate Network(EAN) Application Programming Interface (API)
5. GitHub - a Git repository hosting service that manages and stores revisions of projects
6. HTML - Hypertext Markup Language: standardized system for tagging elements on web documents
7. JS - JavaScript: dynamic programming language mostly used in web browsers
8. PCI compliant - Payment Card Industry data security standards for all merchants level that accepts credit card payments offline or online
9. Ruby on Rails (RoR)- open source web framework running on the Ruby programming language