



UNSW
THE UNIVERSITY OF NEW SOUTH WALES

COMP9900 - Project Final Report Accommodation Web Portal

19T2

Group Name: NSW BNB

Qian Cheng z5149155@ad.unsw.edu.au - Scrum Master

Meiyan Pan z5147046@ad.unsw.edu.au - Developer

Paul Yi z5180174@ad.unsw.edu.au - Developer

Yizheng Ying z5141180@ad.unsw.edu.au - Developer

Submission Date:11-08-2019

1.Introduction	3
2.Background	4
3.Main Component	6
i.User Management Module	6
-User Registration	6
-User Login/Logout	6
-User Management	7
ii.Search Accommodation	10
-Search	10
-View Results	11
-Detailed Accommodation Page	11
iii.Accommodation Booking	12
iv.Accommodation Review	13
v.Recommendation System	14
-Home Page Recommendation: KNN	15
-Accommodation Detail Page Recommendation:Cosine Matrix	18
vi.Mobile App	20
-Andriod	20
4.Conclusion	22
5.Appendix	23
6.Reference	27

1.Introduction

Nowadays, many people consume the vast majority of their information via the internet. Due to the power and convenience of the web, people are increasingly inclined to complete tasks online. For this reason, businesses have also been forced to adapt and provide their services online as well. Thus, tasks such as booking flights, accommodation, transportation, activities, etc. that may have been completed over the phone or in person in the past, can now all be completed online.

Online accommodation booking services such as Trivago and Airbnb make it very easy for travelers to compare different hotels and homes and make informed decisions about where to stay during their holidays and business trips. These companies generally offer mobile applications as well, making their services available to customers wherever they may be.

We improve upon existing services by implementing a machine learning system to recommend accommodations to users based on their previous booking history and review data. We also provide helpful feedback to hosts by classifying reviews on their properties into “good” and “bad” reviews based on guest feedback and sentiment analysis on guests’ reviews. This provides hosts with a quick and easy interface to see what common complaints/compliments are for their properties so that they can see where they need to improve, and where they can continue providing excellent service.

2. Background

The main aim of our project is to create a web application that will enable:

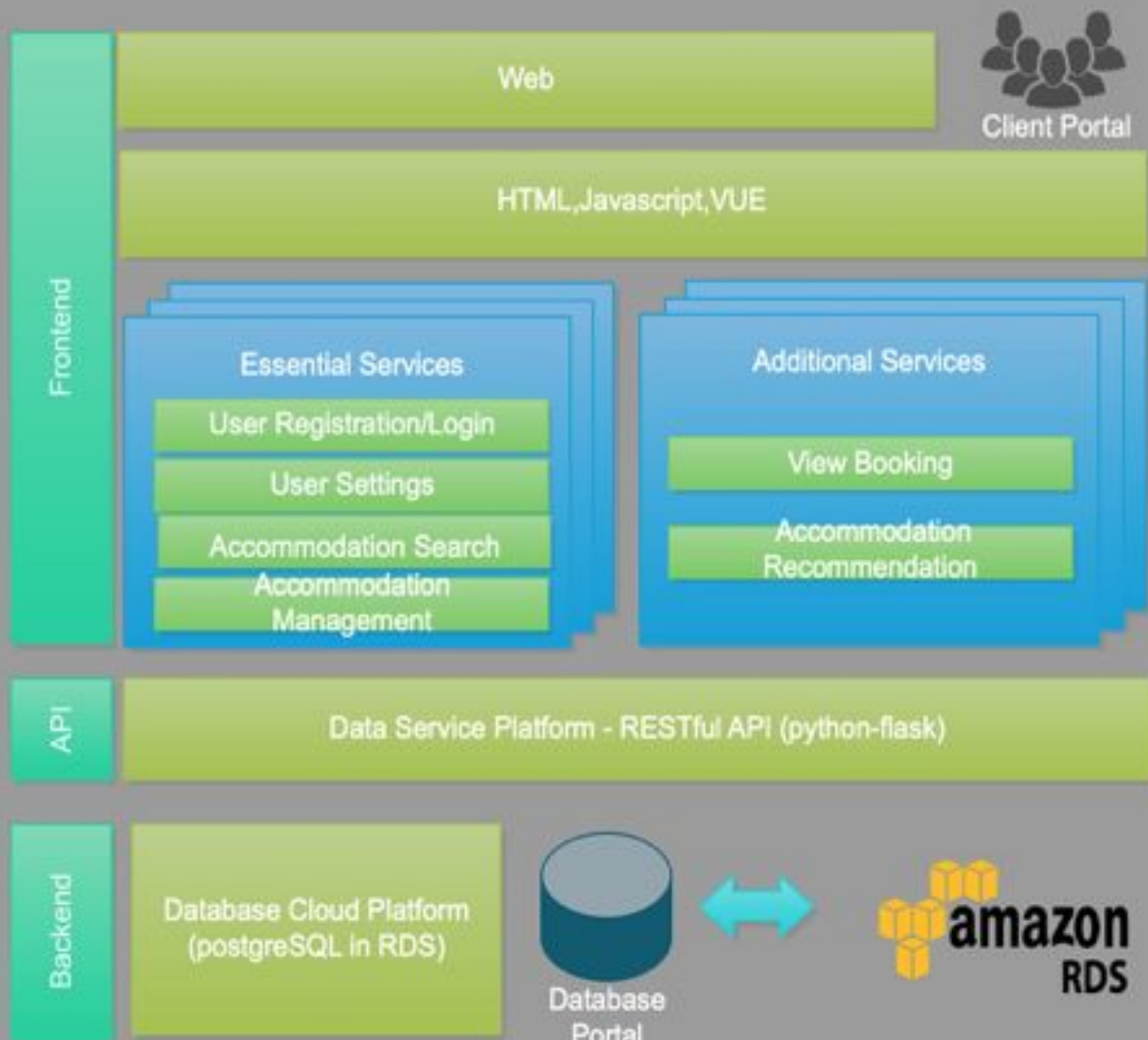
1. Users to search through properties that meet their preferences
2. Users to book the property that they decide is the best fit for their needs
3. Users to leave reviews for properties at which they have stayed
4. Hosts to upload properties and manage their properties and bookings
5. Us to recommend properties to users based on their previous reservations and ratings

The application will be limited to properties located in New South Wales. Thus, we decided to call our application NSW BNB. We will build our application with two types of users in mind:

1. The guests, who will be able to search through and book various properties by providing the dates of their travel and certain criteria such as number of guests, bedrooms, price, etc.
2. The hosts, who will be able to put their homes up on the website so that guests may rent them.

The following pages indicate our system architecture diagram and ER diagram which describes our database schema.

Web Accommodation Portal

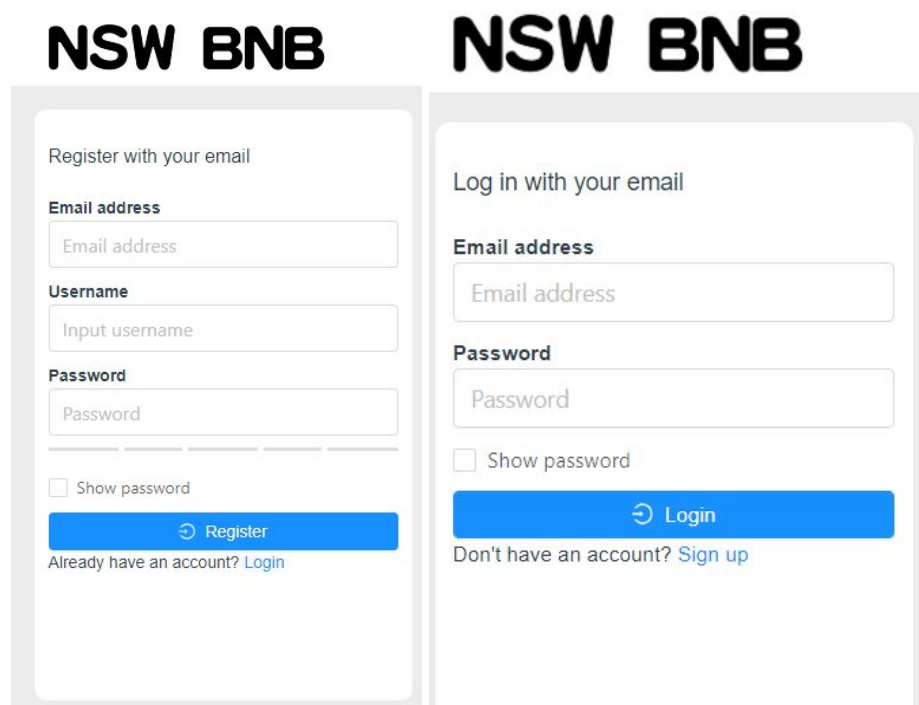


3.Main Component

i.User Management Module

-User Registration

The User Registration interface requests users to input their details such as username, email, password and optional features like phone number and gender into our user management system. The system will check the uniqueness of username, email, and phone number and will return error messages if any of them already exist in the database. The python package Werkzeug is used to securely hash passwords so that we can store the password hashes instead of plaintext passwords. User information will be stored in the database and users will be notified the registration is successful with a message.



The image displays two side-by-side user interface panels for a web application, both featuring the 'NSW BNB' logo at the top. The left panel is titled 'Register with your email' and contains three input fields: 'Email address', 'Username', and 'Password'. Below these fields is a checkbox labeled 'Show password' and a blue 'Register' button. At the bottom, it says 'Already have an account? [Login](#)'. The right panel is titled 'Log in with your email' and contains two input fields: 'Email address' and 'Password'. Below these is a checkbox labeled 'Show password' and a blue 'Login' button. At the bottom, it says 'Don't have an account? [Sign up](#)'.

User login/register interface

-User Login/Logout

The User Login interface allows users to log in to our application so that they may proceed to perform certain actions only available to authenticated and authorized users, such as booking a property or leaving a review. The user is required to input their email address and password. The system will then check if the email address has a match in the database, then hash the password to see if it matches the stored password hash. If they are both consistent with the database, a token will be generated and stored in the database. This token will also be returned to the frontend and stored in the cookies for future use.

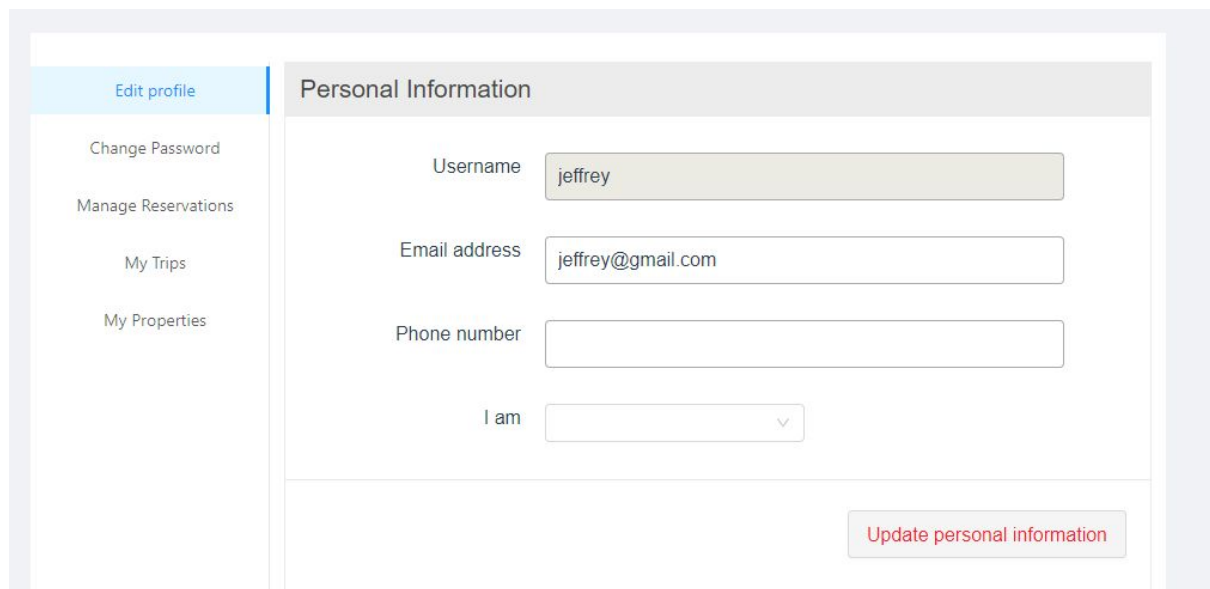
The User Logout interface allows users to log out securely by simply clicking the logout button. The frontend will send the logout request with the token belonging to the user. The backend will then check if the token matches what is stored in the database and also if the token is still valid (not expired). The token will then be revoked and the user will be logged out and redirected to the homepage.

Having to send the username and the password with every request is inconvenient and can be seen as a security risk. To achieve our security goals, we decided to use a token based authentication and authorization. We used Python's secrets module to generate cryptographically strong random strings as the security tokens. Each token also has an expiration time. When checking the validity of a token, the system will check if the token has expired by comparing the expiration time to the current local time. To revoke a token, the system will set the expiration time to one second before the current time.

-User Management

Our User Management System contains five sub functionalities :Profile Management, Password change, Reservation Management, Property Management and Trip Management. A valid token is always required as input to any action related to the User Management System, which means this system is only accessible to authorised users.

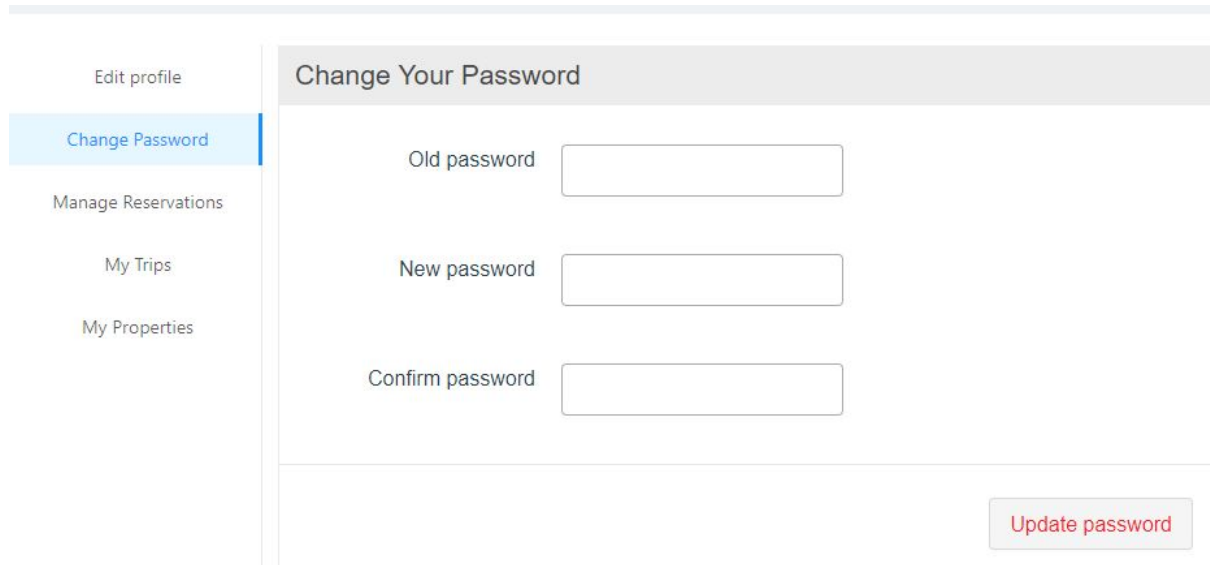
The “User Profile Management” page allows users to update their email, phone number, and gender. The uniqueness of email and phone numbers will be checked. Errors will be raised if the entered email or phone number already exists in the database.



The screenshot displays the 'User Profile Management' interface. On the left is a sidebar with navigation links: 'Edit profile' (highlighted in blue), 'Change Password', 'Manage Reservations', 'My Trips', and 'My Properties'. The main content area is titled 'Personal Information' and contains a form with the following fields: 'Username' (pre-filled with 'jeffrey'), 'Email address' (pre-filled with 'jeffrey@gmail.com'), 'Phone number' (empty), and 'I am' (a dropdown menu). At the bottom right of the form is a red button labeled 'Update personal information'.

The user profile interface

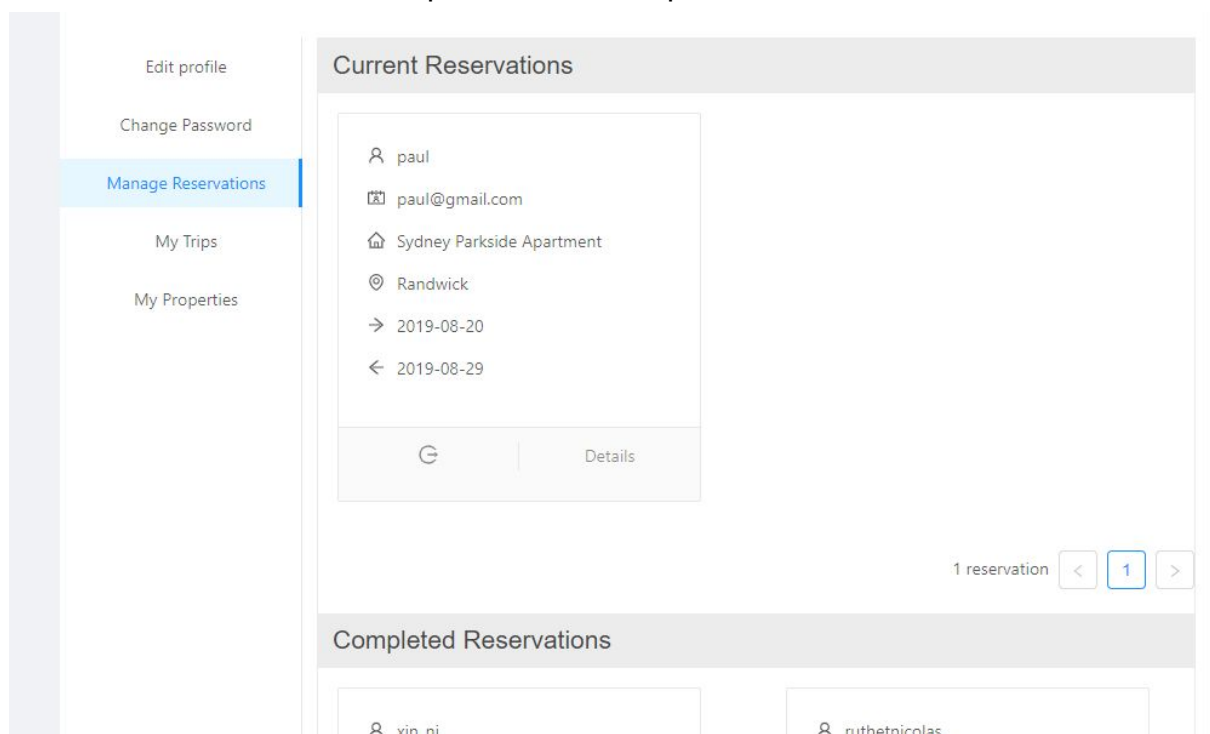
The “Change Password” page allows users to update their password. The Werkzeug library mentioned in the user registration will also be applied here to generate secure password hashes.



The screenshot shows a web interface for changing a password. On the left is a sidebar with links: 'Edit profile', 'Change Password' (highlighted in blue), 'Manage Reservations', 'My Trips', and 'My Properties'. The main content area is titled 'Change Your Password' and contains three input fields: 'Old password', 'New password', and 'Confirm password'. At the bottom right of the main area is a red button labeled 'Update password'.

The user password interface

The “Manage Reservations” page allows hosts to view and manage all their reservations, which is made up of current, completed, and cancelled reservations.

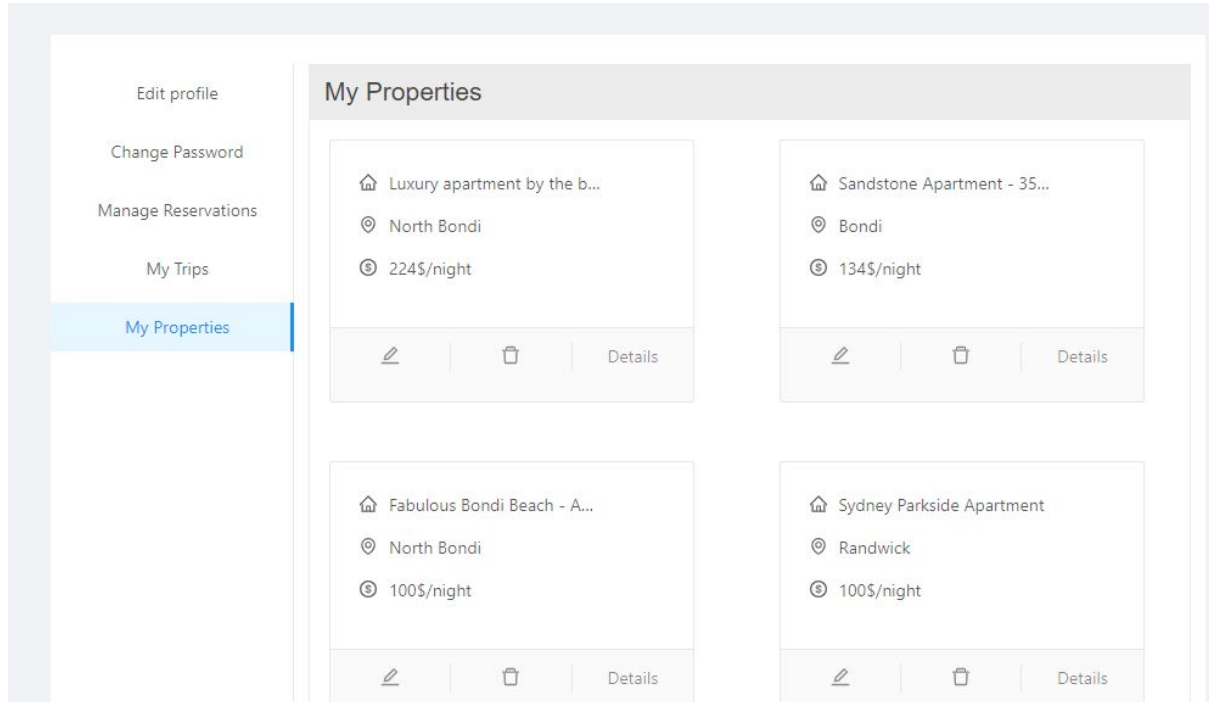


The screenshot shows a web interface for managing reservations. The sidebar on the left has links: 'Edit profile', 'Change Password', 'Manage Reservations' (highlighted in blue), 'My Trips', and 'My Properties'. The main content area is titled 'Current Reservations' and displays a reservation card for a user named 'paul'. The card shows the email 'paul@gmail.com', the property 'Sydney Parkside Apartment', the location 'Randwick', and the dates '2019-08-20' to '2019-08-29'. Below the card is a 'Details' button. At the bottom right of the main area, there is a pagination control showing '1 reservation' and a page number '1' in a blue box. Below the 'Current Reservations' section is a 'Completed Reservations' section, which currently shows no reservations.

The reservation management interface

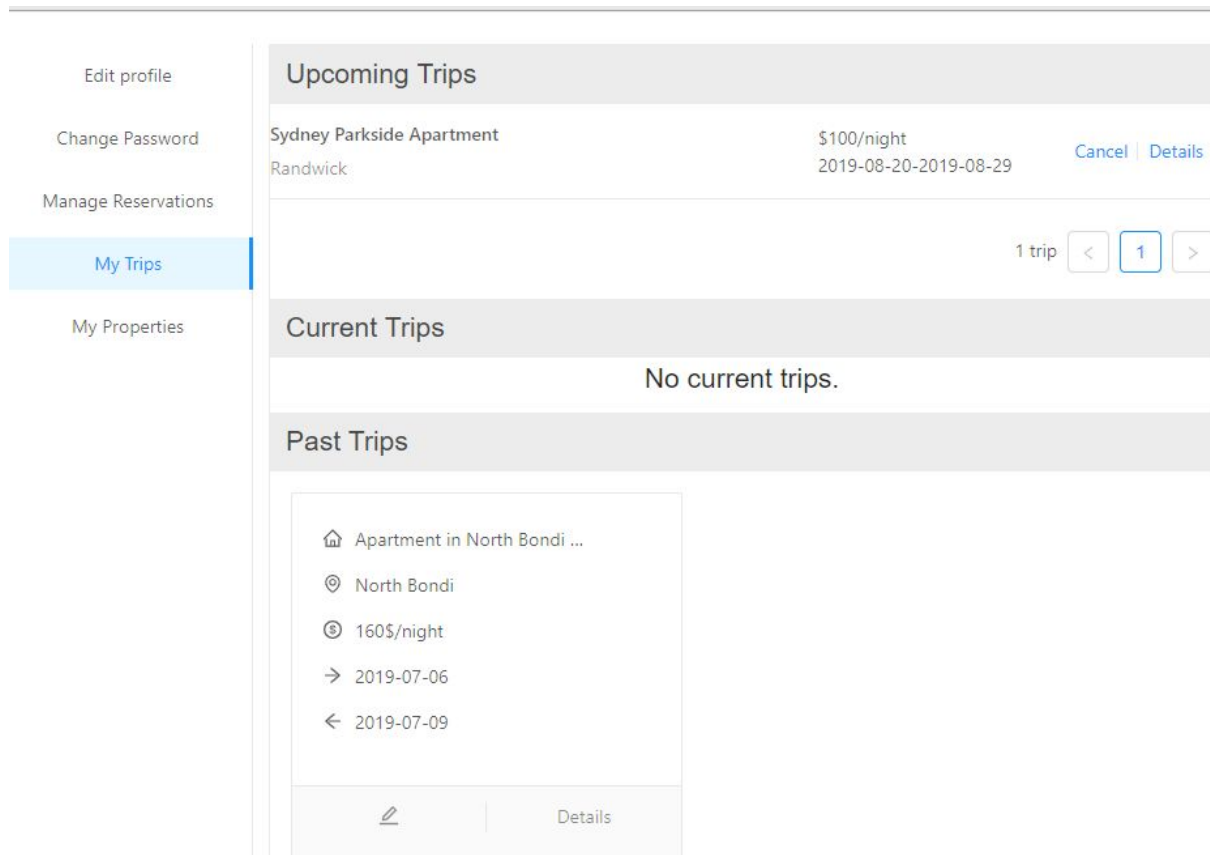
The “My Properties” page allows hosts to manage their registered properties. From this page, they will be able to do the following:

- Upload new properties
- View all properties that they have posted on the website
- View the status of their properties (available, booked, etc.)
- Edit the details and information of their properties
- Delete their properties from the website



The property management interface

The “My Tripes” page allows users to view and manage all the bookings they have made. Users are able to cancel the trip before the check in date if they satisfy the cancellation policy of the accommodation.



The trip management interface

ii. Search Accommodation

The “Search Bar” allow users to search through different properties located in NSW by many different criteria. After browsing through the results that have been filtered to match the criteria, users can open the detailed page of each accommodation. Users are not required to log in to access this subsystem.

-Search

Users will be able to use the search bar to select location, check in and check out date, range of price, number of guests, number of bedrooms, number of beds, number of bathrooms and other amenities like heating and washer for their dream accommodation. The calendar in the search bar will block the past dates and also ensure check out is after check in date. The system will first find all accommodations that meet user’s submitted criteria, then remove the accommodations which have been booked during the selected dates from the results.

Sydney Parkside Apartment

This flat has all the features you desire whilst travelling- comfortable bed, rain-shower head, big kitchen while being centrally located to Sydney city, beaches, great restaurants, entertainment and with beautiful Centennial park right on your doorstep. The Apartment features a new kitchen kitted out with everything you need to cook up a storm- induction cooktop, dishwasher, oven, pizza stone and fridge. The bedroom is big with comfy queen size bed and massive wardrobe space. We have installed a new bathroom with a wonderful rain-head shower and full sized bath and heated towel rack. Other features include bamboo floors, 6 sea- dining room, awesome couch (you won't want to get off it!), TV, stereo, DVD's and internal laundry with washing machine and portable washing line on the balcony. Location We are located in the Eastern Suburbs of Sydney on Alison Road in Kensington which is approx 4 kms to the Sydney CBD. The flat is centrally located and within walking distance

\$100

★★★★★

BEAUTIFUL COOGEE BEACH OCEAN VIEW

You can see the ocean from your balcony and it's only a ten minute walk down for a swim or a short bus ride into the city to enjoy all that Sydney has to offer! I do a lot of traveling so am happy to host you in my home & show you my part of the world. My apartment is a short stroll down to Coogee one of Sydney's most popular beaches, and only a 20/25min bus ride into the city & Sydney Harbour. Coogee itself is full of cafes, restaurants to suit all palates. It has a wonderful coastal walk, winding past Gordon's Bay which is great for...

More filters

price

bedrooms

bed

bath

Amenities

☐ Kitchen
 ☐ Heating
☐ Tv
 ☐ Hangers
☐ Iron
 ☐ Washer
☐ Dryer
 ☐ Hot water

Search accommodation

-View Results

Users will be able to view the results of their searches on the “View Results” page. This page will have some images and basic details such as name, price, rating, a brief description, and an image. Users will be able to sort the results based on criteria such as price and rating. Users will also be able to select a property that they are interested in to view additional details on the “Detailed Accommodation Page”.

-Detailed Accommodation Page

The detailed accommodation page contains more descriptions such as all the amenities the property provides, more images, and also reviews and ratings left by previous guests. If the user decides that they would like to book the property, they will be able to do that from the right sidebar of this page as well.

Sydney Parkside Apartment

Randwick

☐ Entire house
 2 guests 1 bedroom 1 bed 1 bath

Description

\$100/night

Check-in

2019-08-09

Check-out

2019-08-10

Accommodation detail page

iii.Accommodation Booking

The accommodation booking system can be accessed through the right sidebar of the detailed accommodation page. Users can update the check in and check out date and select the actual number of guests. The unavailable dates will be blocked and the system will prevent users from booking if their number of guests exceeds the maximum number of guests the property can accommodate. By clicking “Book”, a “booking information page” including the total price of the order will be shown. Users can click “Agree and continue” to confirm the order and will be able to view the order in the “My Trips” page.

Randwick

🏠 Entire house

2 guests 1 bedroom 1 bed 1 bath

Description

This flat has all the features you desire whilst travelling- comfortable bed, rain-shower head, big kitchen while being centrally located to Sydney city, beaches, great restaurants,entertainment and with beautiful Centennial park right on your doorstep. The Apartment The apartment features a new kitchen kitted out with everything you need to cook up a storm- induction cooktop, dishwasher, oven, pizza stone and fridge. The bedroom is big with comfy queen size bed and massive wardrobe space. We have installed a new bathroom with a wonderful rain-head shower and full sized bath and heated towel rack. Other features include bamboo floors, 6 sea- dining room, awesome couch (you won't want to get off it!), TV, stereo, DVD's and internal laundry with washing machine and portable washing line on the balcony. Location We are located in the Eastern Suburbs of Sydney on Alison Road in Kensington which is approx 4 kms to the Sydney CBD. The flat is centrally located and within walking distance

Amenities

tv

wifi

free parking on premises

internet

kitchen

🔍 More

\$100/night

Check-in

2019-08-09

Check-out

2019-08-10

Guests

1

✓ Book

You won't be charged yet

Enter dates and number of guests to see the total trip price

Accommodation booking

Booking Information

Sydney Parkside Apartment

Randwick

1 guests

price

100/night

check-in

2019-08-09

check-out

2019-08-10

Total(AUD)

\$ 200

Agree and continue

Booking payment

iv.Accommodation Review

After checking out, users can choose to rate and write a review for the accommodation. Their reviews cannot be modified once submitted. The request will be sent with a token and the reservation id to check if the user is allowed to leave a review. The system will recalculate the overall rating of the accommodation and update it in the database and save the new review records as well. The ratings and previous reviews can be found in the detailed accommodation page and the latest reviews will be shown at the top. The system divides the review into “good reviews” and “bad reviews” based on the ratings left by the user and the polarity score generated by sentiment analysis. The reviews with an average rating of 6/10 or less and a polarity score smaller than 0.3 are regarded as “bad reviews”. Otherwise, they are considered as “good reviews”. The system applied the Python package TextBlob to perform sentiment analysis. The principle is to capture the emotional words and compare it to the library provided by TextBlob to check if the word is considered positive or negative. The “deny words” like “not”, “couldn’t” and adverbs of degree will also be checked to determine the final polarity of the phrases. The sentiment property returns a polarity score which is a float within the range [-1.0, 1.0].

Rating

Accuracy

★★★★★

Location

★★★★★

Communication

★★★★★

Check-in

★★★★★

Cleanliness

★★★★★

Value

★★★★★

Reviews

All reviews

Good reviews

Bad reviews

15 reviews

rita

I stayed at Louise apartment for 2 weeks cause it is close to UNSW. It is convenient to visit my son who studies there. The apartment is same as the pictures shown in website, it is very clean with well equipped kitchen. I feel like at home and both me and my husband are satisfied staying here.

★★★★★

ross

\$100/night

Check-in

2019-08-09

Check-out

2019-08-10

Guests

1

✓ Book

You won't be charged yet

Enter dates and number of guests to see the total trip price

Accommodation review

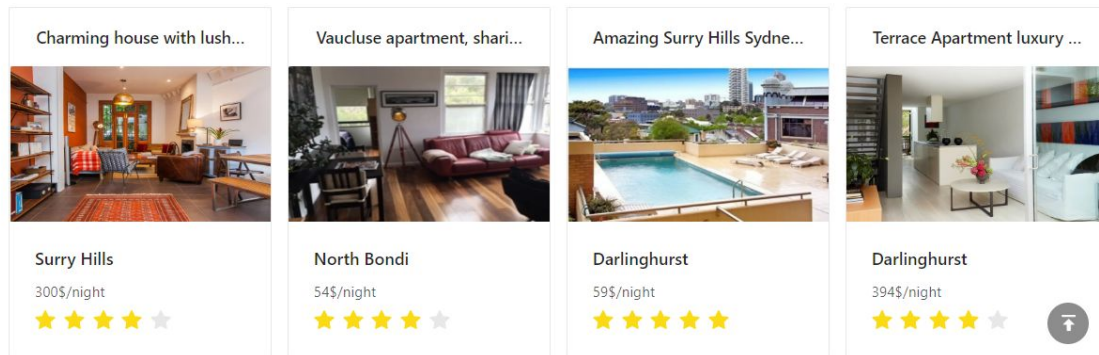
v.Recommendation System

The online accommodation booking service has become an extremely competitive space where businesses try to attract users' attention with recommending, comparing, and sharing properties. In our project, we built a personalized user-oriented and item-based recommendation system to improve the user experience for our users.

In the process of developing a recommendation system, we tried several different machine learning algorithms including decision trees, neural networks and Q-learning. We finally decided to use k-nearest neighbors (KNN) method and Cosine Matrix to build our system because both of these algorithms have good space and time complexity, and thus perform very well.

The output of our recommendation algorithm is four properties which we show to users at the bottom of the homepage and on the accommodation detail page.

You may like

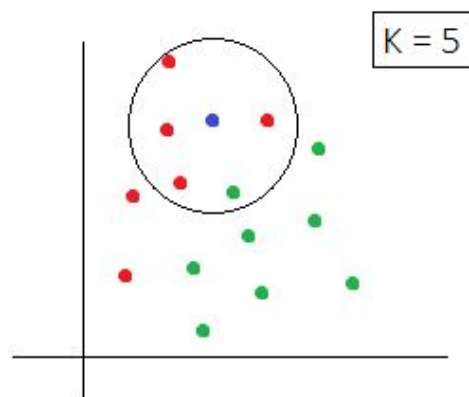


The recommendation results in the bottom of the home page

The following is a description of the algorithms we selected:

-Home Page Recommendation: KNN

The KNN algorithm is one of the simplest classification algorithms and it is one of the most widely used. KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.



we set $k=5$ to get the top 5 nearest neighbors(including the original point itself)

In our project, for logged-in users, we retrieve their previous review and ratings data to calculate the distance to the most similar accommodations by using KNN. For anonymous users who are just browsing without having logged in, we assert `user_id = 0` and recommend top four popular accommodations for them. We also make sure that all the recommended accommodations have an average rating of at least 8/10.

The input to our algorithm is the user id of the user who is browsing our website:

```
def knnpredict(input_user_id):
```


The first step is to get our accommodation, users, and review data from database:

	id	num_guests	num_bedrooms	num_beds	num_bathrooms	price	rating	\
0	1	1	1	1	1.0	11000	8.8	
1	2	1	1	1	1.0	13000	9.8	
2	3	2	1	1	1.0	17400	9.5	
3	4	4	2	2	1.0	24400	8.6	
4	30	2	1	1	1.0	14000	9.1	

	num_reviews
0	321
1	60
2	128
3	25
4	86

Example review data after processing

When retrieving the user's data, we also check if they have a review history. If a user does not have enough reviews we will generate recommendations by selecting top rating accommodations:

```
56 reviews = df_review[df_review.guest_id.isin(input_user_id)]
```

After that we create a training model whose target value(y) is review_score(a user gives high review rating which not lower than 8 out of 10):

```
76 #create a dataframe with all training data except the target column
77 #check that the target variable has been removed
78 X = items_num[['id', 'num_guests', 'num_bedrooms', 'num_beds', 'num_bathrooms', 'price']]
79 #separate target values
80 y = items_num['review_score'].values
```

example of y(target values):

```
[0, 0, 0, 1, 0, 1]
```

A 0 represents the fact that the user did not give a high rating or that the user gave no review for that accommodation, while a 1 represents the opposite meaning.

Then we built a KNN classifier and fit the data into the model:

```
88 #split dataset into train and test data
89 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1, stratify=y)
90 # Create KNN classifier
91 knn = KNeighborsClassifier(n_neighbors = 5)
92 # Fit the classifier to the data
93 knn.fit(X_train,y_train)
```


We then used GridCV to improve the performance of our model:

```
107 #create new a knn model
108 knn2 = KNeighborsClassifier()
109 #create a dictionary of all values we want to test for n_neighbors
110 param_grid = {'n_neighbors': np.arange(1, 8)}
111 #use gridsearch to test all values for n_neighbors
112 knn_gscv = GridSearchCV(knn2, param_grid, cv=5)
113 #fit model to data
114 knn_gscv.fit(X, y)
115 #check top performing n_neighbors value
116 knn_gscv.best_params_
117 p = knn.predict(X)
```

We then check cv accuracy:

```
102 #train model with cv of 5
103 cv_scores = cross_val_score(knn_cv, X, y, cv=5)
104 #print each cv score (accuracy) and average them
105 print(cv_scores)
106 print('cv_scores mean:{}'.format(np.mean(cv_scores)))
```

```
[0.99137931 0.99137931 1.          1.          1.          ]
cv_scores mean:0.9965517241379309
```

Check predict accuracy on test dataset:

knn predict score 0.9961464354527938

Implemented our improved model to our dataset:

```
nbrs = KNeighborsClassifier(n_neighbors=5, algorithm="auto").fit(X,y)
distances, indices = nbrs.kneighbors(X)
```

We can now get the distances(Euclidean distance) and indices between our items:

Indices:

	self	top1	top2	top3	top4
0	0	49	41	16	66
1	1	83	162	159	38
2	2	77	237	245	253
3	3	52	25	36	184
4	4	150	155	183	39

Distances:

	self2	d1	d2	d3	d4
0	0.0	187.974067	267.016179	307.774268	311.658868
1	0.0	128.831246	167.107869	167.613961	192.919698
2	0.0	143.559082	258.058152	258.323615	269.232260
3	0.0	48.509381	500.982994	502.543889	535.935630
4	0.0	131.122233	152.269531	171.338875	192.211758

Merge our result by accommodation id:

- 'id' indicates accommodation id

- 'top1~4' indicates the top four nearest neighbor to that accommodation

- 'd1~4' indicates the distance of the top four nearest neighbor to that accommodation

	id	num_guests	num_bedrooms	num_beds	num_bathrooms	price	rating	\
0	1	1	1	1	1.0	11000	8.8	
1	2	1	1	1	1.0	13000	9.8	
2	3	2	1	1	1.0	17400	9.5	
3	4	4	2	2	1.0	24400	8.6	
4	30	2	1	1	1.0	14000	9.1	

	num_reviews	review_score	self	top1	top2	top3	top4	self2	d1	\
0	321	0.0	0	49	41	16	66	0.0	187.974067	
1	60	1.0	1	83	162	159	38	0.0	128.831246	
2	128	1.0	2	77	237	245	253	0.0	143.559082	
3	25	0.0	3	52	25	36	184	0.0	48.509381	
4	86	0.0	4	150	155	183	39	0.0	131.122233	

	d2	d3	d4
0	267.016179	307.774268	311.658868
1	167.107869	167.613961	192.919698
2	258.058152	258.323615	269.232260
3	500.982994	502.543889	535.935630
4	152.269531	171.338875	192.211758

Our output is a list which contains the accommodation id of the four recommendation results obtained based on the shortest KNN distance we calculated above:

```
152 knnpredict(12)
```

```
Out[9]: ['83', '162', '159', '38']
```

-Accommodation Detail Page Recommendation: Cosine Matrix

Cosine similarity is a metric used to measure how similar the items are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

To find similar accommodations, we use the accommodation's features to generate dataframes, then clean the data and normalize them. We convert the dataframe to a standardized array and then calculate the similarity between each element by using cosine matrix algorithms.

Our input is a specific accommodation id which we are browsing:

```
def predict(accommodation_id):
```

The first step is to get our accommodation data from database and convert data into dataframe by using the Python pandas library, then clean it:

```
-----
#import data from database
conn = psycopg2.connect("dbname='loki' user='loki' password='00000000' host='project9900-v2-sdyney.clnpnedcibyx.
cur = conn.cursor()
cur.execute("SELECT * from public.accommodation")
rows = cur.fetchall()
df = pd.DataFrame(rows)
df.fillna(0,inplace=True)

#Data clean
col = ['id','host_id','name','num_guests','num_bedrooms','num_beds','num_bathrooms','description','suburb','cit
'num_reviews','scores_accuracy','scores_check_in','scores_cleanliness','scores_communication','scores_location',
df.columns = col
df = df[df.is_deleted.isin([False])]
df = df[['id','num_guests','num_bedrooms','num_beds','num_bathrooms','price','rating','num_reviews']]

#Getting useful features for predicting
items_num = df
items_num_norm = items_num[['num_guests','num_bedrooms','num_beds','num_bathrooms','price','rating','num_r
```

Once we have cleaned our data and extracted the useful features, we use `standard_scaler` and `standard_matrix` to normalize data into array with a range:

```
37     #Use standard_scaler and standard_matrix to normalize data
38     standard_scaler = preprocessing.StandardScaler()
39     standard_matrix = standard_scaler.fit_transform(items_num)
40     print(standard matrix)
```

```
[[-1.72904638 -0.55632054 -0.68748939 ... -0.60749009 -0.68923139
 4.25368898]
 [-1.72303231 -0.55632054 -0.68748939 ... -0.51234263 0.61965546
 0.31232363]
 [-1.71701823 0.2388175 -0.68748939 ... -0.30301822 0.22698941
 1.33919276]
 ...
 [ 1.72303231 -0.55632054 -0.68748939 ... -0.60749009 -0.03478796
 -0.33702009]
 [ 1.72904638 -0.55632054 -0.68748939 ... -0.41719517 0.22698941
 0.52373786]
 [-1.68093379 0.2388175 0.35734937 ... -0.36962144 -0.54644373
 0.40292973]]
```

After that we compute the cosine similarity matrix using the dummy ratings matrix:

```
41 #Compute the cosine similarity matrix using the dummy ratings matrix
42 cosine_sim = cosine_similarity(standard_matrix)
43 print(cosine_sim)
```

```
[[ 1.          0.51628059  0.81347728 ... -0.22899725  0.08062411
 0.45564575]
 [ 0.51628059  1.          0.83777818 ... -0.24280623 -0.17904836
 0.47150695]
 [ 0.81347728  0.83777818  1.          ... -0.38457376 -0.17784439
 0.57605196]
 ...
 [-0.22899725 -0.24280623 -0.38457376 ... 1.          0.91563833
 -0.73168111]
 [ 0.08062411 -0.17904836 -0.17784439 ... 0.91563833  1.
 -0.7026217 ]
 [ 0.45564575  0.47150695  0.57605196 ... -0.73168111 -0.7026217
 1.          ]]
```

Our output is a list which contains the accommodation ids of the four recommendation results obtained based on the cosine matrix:

```
64 predict(102)
```

```
Out[3]: [69, 281, 283, 66]
```

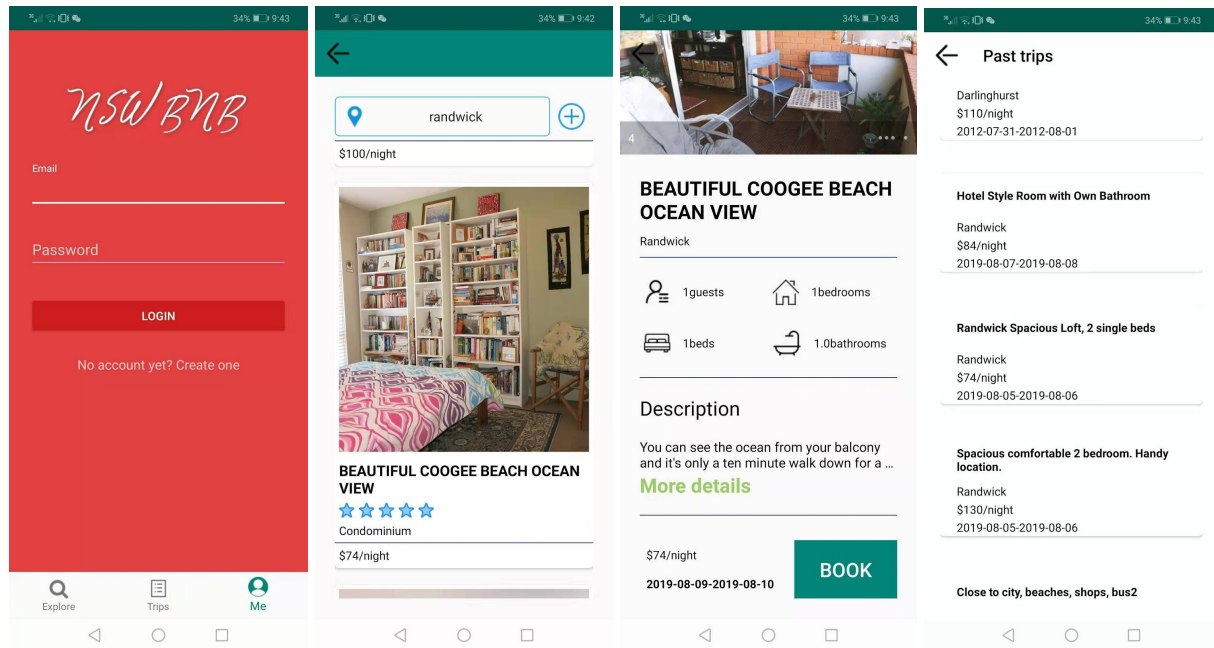
vi. Mobile App

-Android

In order to make it easier and more convenient for users to book accommodations while on the go, we felt that a mobile application was necessary. We decided to build an Android application as our team had past experience with Android. In the Android app, users can create an account and login just as they can on the website. Users can search for the accommodations they want with the filter conditions such as

location, check in and check out date and guest number. The accommodations that match the users' criteria will be shown on the results page with the accommodation image, price, and rating. The user can click the accommodation they are interested in to see more details on detail page. On this page, there is a booking button on the bottom of the page for the user to book. After confirming the booking, the user can see the booking information on the trip page. In general, the Android application supports most of the same functionality as our web application, but not all, as we started developing it later on.

In the Android app, we use xutils for network requests and glide for loading images.



General preview of the mobile version

4. Conclusion

Overall, we feel that we were successful in building a well structured and well tested application. We were able to finish all of the features that we wanted for our MVP. In addition, we were even able to complete tasks that went beyond our MVP such as building out our Android application. Our novel features such as the recommendation system worked very well. We feel that users will be able to use the recommendation system to find accommodations that they prefer quicker and easier than if using existing trip booking applications. For the future, we would like to improve our review sentiment analysis feature for hosts. It would be even more helpful to our hosts if we did not just group the reviews into “good” and “bad” reviews, but also showed what some common elements of each category were. That way if a popular property had hundreds of reviews, hosts could quickly see what about their property was popular and well-received, and what the common negative aspects were. We would also like to build a more feature complete Android application, and also start working on an IOS application in the future as well.

5. Appendix

i. Used techniques

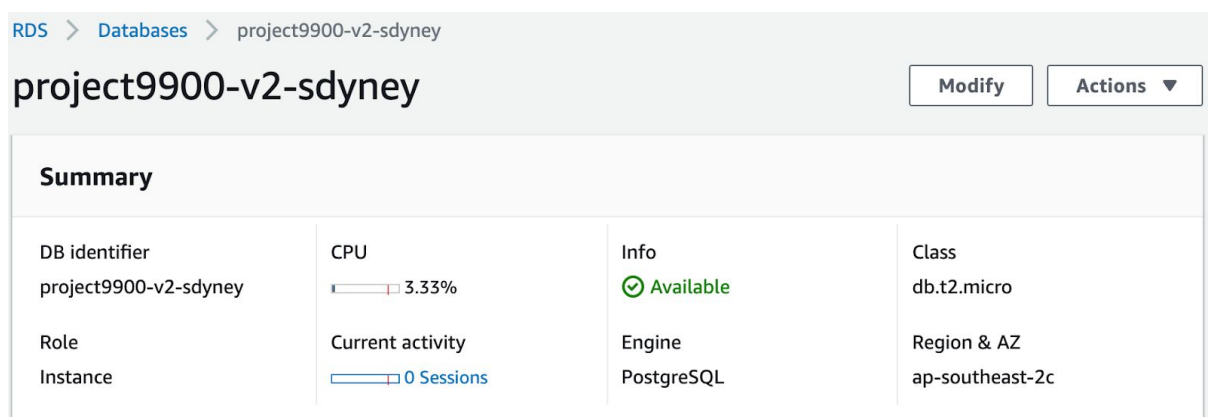
We used a variety of technologies, third party libraries, and frameworks in building our application. To build our REST API, we used Python and the Flask web framework. We felt that this was the logical choice for our team, as we were all familiar with the Python language, and most members of our team had experience using Flask in the past. Our API layer was designed so that it could support different types of frontends. For example, the primary interface to our application is a web frontend using the common web technologies such as HTML, CSS, and Javascript. In particular, we decided to use Vue.js as our frontend framework. We made this decision based on the fact that Vue.js is a very popular framework to build single page applications due to its great documentation, vibrant open source ecosystem, and ease of use for developers. The great documentation meant that even though none of us had prior experience using Vue.js, it was easy to jump in and be productive immediately. Also, due to its popularity, we felt that most of the common problems that we would encounter would be very easy to solve by a quick Google search. Lastly, the declarative style of programming that is used in Vue applications makes it very simple and easy to build applications very quickly.

As a single page application, much of the logic that would traditionally be handled on the server is actually handled on the client. This includes routing, deciding what data to load and render, etc. The only time a full page load occurs when the client requests the homepage. After that, the frontend only requests the data that it needs from the server and only updates the client as it needs to. This fits perfectly with the design of our REST API. Since we need to support multiple types of clients, we don't send HTML for every page request, since that would only be compatible with a browser. Instead, we only send JSON data to the client. Thus, when we decided to build an Android mobile application, we did not have to make a single change to our Flask application.

A key characteristic of a REST API like ours is that it is stateless. This means that all session state is stored on the client side. Traditionally web applications stored session state on the server. More recently, RESTful API's have increased in popularity. One of the main advantages of keeping the server stateless is that it becomes incredibly easy to horizontally scale the API layer of the application. Imagine that we have deployed our application using Amazon Web Services. We can configure our architecture such that we simply spin up extra EC2 instances as traffic to our website increases. Say it's the holiday season and many customers are booking accommodations. There may be large spikes in traffic, but it will be handled gracefully and transparently. This problem is significantly more difficult to handle in a traditional web application.

A consequence of building a RESTful API is that we can't simply depend on the session capabilities offered out of the box by Flask in order to log in users and manage their sessions, as these assume that session state is stored on the server. Thus, we will implement a token based authentication system which will eliminate the need to store session information on the server. When the client receives a token upon login, it will need to send this token along with all requests that require a logged in user.

Lastly, our Flask application also communicates with our PostgreSQL database. Our data is very structured and has a consistent schema. Thus, an RDBMS such as PostgreSQL was a better fit than a NoSQL database such as MongoDB. When we deploy our application, we can deploy our database to a service such as RDS in Amazon Web Services, which is a well-known cloud platform and makes it easy to manage and scale our database, and ensure that it is secure and highly available.



The screenshot shows the AWS RDS console interface. At the top, there's a breadcrumb trail: 'RDS > Databases > project9900-v2-sdyney'. Below this, the instance name 'project9900-v2-sdyney' is displayed in large text, with 'Modify' and 'Actions' buttons to its right. A 'Summary' section follows, containing a table with four columns: DB identifier, CPU, Info, and Class. The DB identifier is 'project9900-v2-sdyney'. The CPU usage is shown as a bar chart at 3.33%. The Info column shows a green checkmark and the word 'Available'. The Class is 'db.t2.micro'. Below this, another row shows 'Role Instance', 'Current activity' with a bar chart at '0 Sessions', 'Engine PostgreSQL', and 'Region & AZ ap-southeast-2c'.

Summary			
DB identifier project9900-v2-sdyney	CPU <div><div></div></div> 3.33%	Info ✔ Available	Class db.t2.micro
Role Instance	Current activity <div><div></div></div> 0 Sessions	Engine PostgreSQL	Region & AZ ap-southeast-2c

Database built on AWS platform

ii. Installation manual

In order to run our project, you need the following tools installed on your machine:

- Python3
- pip3
- PostgreSQL
- npm

If you have the above tools installed, you can then proceed to clone the project and install the dependencies by running the following commands (these commands will work on Mac or Linux):

- git clone <https://github.com/unsw-cse-comp3900-9900/capstone-project-nsw-bnb.git>

- `cd capstone-project-nsw-bnb`
- `python3 -m venv venv` (this creates a virtual environment where the dependencies will be installed)
- `source venv/bin/activate` (this activates the virtual environment you created in the last step)
- `pip install -r requirements.txt` (this will install all of the project dependencies into the virtual environment)

We did our best to build our project according to best practices. Thus, we did not hard-code sensitive details such as database connection details and passwords directly into our codebase. Instead, when our application starts, it reads this sensitive data from the environment. Therefore, in order to successfully run our application, you need to create a file called `“.env”` in the project’s root directory. This is a file that is not checked into version control, and it contains the environment variables that will be loaded into the application. The contents of this file should be the following:

- `FLASK_APP=app`
`FLASK_ENV=development`
`DATABASE_URL=postgresql://loki:00000000@project9900-v2-sdyney.clnpne`
`dcibyx.ap-southeast-2.rds.amazonaws.com:5433/loki`

Now we are ready to run the Flask application. This can be done by running the following command:

- `flask run`

This application will automatically connect to our PostgreSQL database running in AWS.

In order to run the frontend for our project, `cd` into the frontend directory. From here, run the following commands:

- `npm install` (this will install the frontend dependencies)
- `npm run dev` (serves the frontend on <http://localhost:8080>)

Now you should be able to navigate to <http://localhost:8080> in your browser and see the web application.

iii. Source code structure

The source code can be broadly divided up into the code for the Flask application, the Vue.js application, the Android application, and the Recommendation system. These are in directories called `app`, `frontend`, `android`, and `RecommendationSystem`, respectively.

The Flask application was built according to the Application Factory Pattern. This allowed us to write our application in a modular way. Routes are neatly split up in different files according to functionality and the resource it operates on. For example, all of the functionality related to registering, logging in, and logging out users is in a file called `auth.py`, and all functionality related to searching is in a file called `search.py`. If there was some new functionality required, all that is required is to

make another file in the same directory, write the logic for the routes in there, then register that route on the application in the file called `app/__init__.py`. Another great advantage of the Application Factory Pattern is that it makes it very easy to write tests. The application factory is just a function that creates the Flask application. We can supply different conditions to this function to create different types of applications. Thus, in a test environment, we can create a test application with different configurations and use this in our integration tests.

All of the `vue.js` code for the frontend is in the directory named `frontend`. Most of the logic and view code is in files in the directory `frontend/src/views`. There are some smaller components that are used in the view files that are stored in `frontend/src/components`. New pages can be created by creating a new view file in the views directory and registering that view with the router, which is in the file `frontend/src/router`.

All of the Android app code is in the directory named `android`. The logic code is in files in the directory `android/app/src/main/java/com/example/yyz/nswbnnb_android`. The view code is in the directory `android/app/src/main/res/layout`. All of the view code is contained in XML files. The `app-release.apk` in the directory `android/app/release` can be used to install the app on the mobile phone.

All of the recommendation system code is in the directory named `RecommendSystem`. One file in this directory is `project_recommendation.py`. This file contains the code for the Cosine Matrix algorithm. Another file, `knnpredict.py`, contains the logic for the KNN algorithm. Lastly the file called `sentiment.py` contains the code that conducts the sentiment analysis on the review data. There is some test data that we used in the test code in the CSV files.

6.Reference

[1]PostgreSQL database adapter for Python 2019, Psycopg Team, accessed 14 June 2019 <<http://initd.org/psycopg/docs/usage.html>>

[2]The Ant Design Specification 2019, Ant Design Team,accessed 15 June 2019 <<https://vue.ant.design/docs/vue/introduce>>