# CSCC43 AirBnB Project User Manual

William Chiu
1006998493

Aug 8, 2023
Prof. Nick Koudas
CSCC43: Introduction to Databases
GitHub Repo Link: https://github.com/williamc99/AirBnB_ProjectC43

## Prerequisites

- Project was developed/tested using Java 19, some functions may not work with older versions of Java
- MySQL Connector for Java
- Stanford CoreNLP 4.5.4 (used for most popular nouns in reports)

## Flow of Application

*Note: Please take a look at the source code as it is well commented and explains many of the functions and individual code steps*

The app will start by prompting the user for a choice of option. Please note that for important props where the app asks the user for important information, there is error checking, however not all prompts are error checked (this was one of my assumptions for the user being competent). The availability calendar works like this: by default, we assume the host is available to rent all year round. If they are unavailable on some days, then they have the option to enter this. All the days that are not booked/unavailable are considered available to rent days. When the user enters one of the 11 options from the main menu, the respective handler function will be run:

### Create a New User (Handler Function 1)

This function prompts the user for information about a user and stores it in the Users table. There is error checking for the username prompt, since there cannot be duplicate usernames (an SQL query is made to check if the username already exists). It also makes sure that the user is at least 18 years old.

### Book a Listing (Handler Function 2)

This function first asks the user how they would like to search for a listing. There are 3 search methods: searching by longitude/latitude, searching by postal code, or searching by exact address. The search by distance method makes use of the Haversine Function, which is used to calculate distance between two geographical points. Then, it will ask the user if they want to enter filters. The filters are: amenities, price range, and time range. It then lists the listings for the user. It makes sure that if the user wants to book, they must have a credit card on file. They also cannot book

their own listings. Finally, they enter a start/end date and the booking is created in the Bookings table.

### Cancel a booking (Handler Function 3)

The function asks a user for their username and bookingID so that the program knows if the user is the host or renter of a listing. Then the user can choose to cancel the booking if they want. Note that renters can not cancel other renter's bookings and hosts can cancel anyone's bookings.

### Create a new listing (Handler Function 4) [Price Algorithm]

- This function asks the user for all the required information to create a new listing
- Listing type can only either be 'House', 'Apartment', 'Guesthouse', and 'Hotel' (just like the actual AirBnB website)
- Longitude/Latitude values are only up to 6 decimal places
- There are 24 amenities, and it uses the same ones from the AirBnB website
- The **Price Recommendation algorithm** works like this:
    - Depending on which Listing type was entered, it will set the base price. So House is $300, Apartment is $200, Guesthouse is $250, and Hotel is $175 (loosely based on the average price of these types on AirBnB website)
    - Any amenities added will add a percentage to the base price.
    - Wifi, AC, Heating, TV, Pool, Hot Tub, Gym, Indoor Fireplace, Beachfront, and Waterfront are most expensive/important so they add 4% of the total base price per expensive amenity (i.e # of expensive amenities * (0.04 * base price))
    - Kitchen, Washer, Dryer, Dedicated Workspace, Crib, BBQ Grill, and Breakfast are deemed features but not as expensive/important like the previous, so they add 3% of the total base price per feature amenity
    - Hair Dryer, Iron, Free parking, EV charger, Smoking allowed, Smoke alarm, and carbon monoxide alarm are deemed as other amenities and they add 2% of the total base price per other amenity
    - Finally, you take this amenity price + the original base price and that is what the application recommends as a good listing price
- The user is asked if they want to enter unavailability dates
- By default, it is assumed that the user is available to rent for all the days in a year
- If they are not available on certain days, then they can enter it here or in the "Edit a Listing" method

- Listing is then created in Listings table

### Remove a listing (Handler Function 5)

- Makes sure user is the owner of the listing
- Deletes the listing at their request
- All related bookings and reviews are also deleted through the foreign key ON DELETE CASCADE in the schema

### Edit a listing (Handler Function 6)

- Have a choice of editing price or unavailability
- If price, it will ask user for a price and a price change effect date
- All bookings before the change effect date and all bookings 7 days after the price change effect date will not be affected by the price change
- All bookings after that will have their price updated
- Unavailability changes work the same as they did in “Create a Listing”, however now, you cannot change the unavailability of a date if it is already booked (they must cancel the booking first)

### Write a review (Handler Function 7)

- User can choose to write or view reviews
- Users can only write reviews for listings that they have rented in the past 2 years or renters that have rented from them in the past 2 years

### Delete a user (Handler Function 8)

- User is prompted to login and asked if they want to delete their account
- If they do, all related listings, bookings, and reviews will be also be deleted
- This is because it wouldn’t make sense to reference a listing, booking, or review from a user that doesn’t exist anymore
- The related listings, bookings, and reviews are deleted using the schema’s foreign key ON DELETE CASCADE

## User History (Handler Function 9)

- Shows the listing and/or booking history of a user

## Reports (Handler Function 10)

- Please look at the source code to see the SQL queries that were used
- Note that the Stanford CoreNLP 4.5.4 library was used for Report 7, which is the most popular noun phrases per listing report
- The library has functions/classes/models that will analyze the text, and determine if it is a noun
- I then add all the noun phrases to a HashMap and record the number of occurrences
- Note that strictly noun phrases will be included in the HashMap, as CoreNLP doesn't include adjectives, pronouns, connecting words, etc.
- Note that it is also case sensitive

## Sync Database with Today's Date (Handler Function 11)

- Since the app is only online when I run it, that means that the database, namely the Bookings is not always up to date
- What I mean is that if today's date is past a booking's end date, it will not be updated automatically
- This is why I included this function; when this function is run, it will get today's date and make sure that all bookings that had their status as "booked" will be updated to "completed" if today's date is past their booking end date

I also have a list of helper functions but they are pretty self explanatory and there are comments detailing what they do and how they do it. Please look at the source code for more details.

## Limitations / Possible Improvements

- Error checking on all user prompts
- Use more reusable code (i.e make more use of classes/methods, etc)
- Incorporate software design principles/concepts
- Would be more user friendly if it was using an interactive GUI
- CoreNLP is slow at generating noun results
- Would want to incorporate more support for a much wider range of listing types