GitHub Repo

# C43 Final Project

## Project Report

Minjun Kim

Mustafa Musitapa

GitHub Repo

## Description

A complete database system that supports the operations of a platform like to the well-known home-sharing service, Airbnb, is what the MyBnB project seeks to create and build. The database will make it easier to handle user profiles, comments, ratings, bookings, user profiles, and numerous reports in an effective manner. MyBnB aims to give hosts and renters a simple and pleasurable experience while listing, searching, and booking lodgings by building a comprehensive and user-friendly database.

## Purpose

The main goal of the MyBnB project is to create a comprehensive database system that takes into account the demands and difficulties of a home-sharing network. The project's goal is to develop a setting where users can easily post their rental units, look for lodging, make reservations, and communicate with one another through comments and ratings. The platform will be able to provide a dependable and user-centric service thanks to the database design and implementation, which will guarantee data accuracy, availability, and performance.

## Key Objectives

Effective Listing Management: MyBnB will give hosts the resources they need to build and take care of real estate listings. Listing categories, geographic coordinates, calendars of availability, and rental price administration are all included in this.

Management of User Profiles: The system will let users establish and manage their profiles, which will include crucial data including name, address, date of birth, occupation, and payment information.

Booking and Reserving: Users of MyBnB will be able to look for available properties within a given area and time frame, make reservations, and get confirmation. Depending on their availability, hosts will have the choice to accept or refuse reservations.

User reviews and comments are welcome for both listings and personal encounters. Additionally, hosts can comment on tenants, promoting an honest and open community.

Reporting and Search: The database will allow you to do searches using a variety of parameters, including location, price range, amenities, and availability dates. The technology will also produce comprehensive reports with booking data broken down by location, nation, host ranks, and more.

GitHub Repo

## Conceptual Solutions

The MyBnB project introduces innovative solutions to complex challenges in the realm of accommodation management:

**Geographical Precision**: Leveraging latitude and longitude coordinates with Google Maps API for accurate location tracking allows users to find listings with pinpoint accuracy.

**NLP Noun Phrase Extraction**: The system can automatically recognize and extract noun phrases from user-generated comments thanks to CoreNLP's natural language processing capabilities. Key details and opinions shared by customers regarding their accommodations, such as particular amenities, features, and elements of the resort, are frequently condensed into noun phrases.

**Advanced Availability Calendar**: The system's availability calendar ensures hosts can efficiently manage their property's rental schedule and update prices based on availability.

**Intelligent Recommendation**: MyBnB's host toolkit incorporates data analysis to suggest optimal pricing and amenities for listings, enhancing host success and revenue potential.

## Justification

The necessity for a dependable, user-friendly, and feature-rich accommodation management platform justifies the database design and execution of the MyBnB project. The project seeks to offer a smooth experience for both hosts and renters by addressing the intricate processes of listing development, booking, user engagement, and reporting. The platform will succeed and expand as a result of the system's user-friendly interface, comprehensive data management, and powerful search capabilities, which will satisfy the changing needs of the home-sharing business.

In summary, the MyBnB project aims to provide an extensive and user-centric accommodation management system that makes use of cutting-edge database architecture and implementation. The project seeks to deliver a dependable and effective platform for hosts and renters, boosting their experiences in the home-sharing sector by tackling the difficulties of property listing, booking, user engagement, and reporting.

GitHub Repo

## Assumptions:

Legal Age Requirement: Assumed that users must be at least 18 years old to create an account, based on the reference to legal age (at least 18 years of age).

Cancellation Policy: Assumed that both hosts and renters have the ability to cancel bookings, but the specific cancellation policy details were not provided.

Rating and Commenting Window: Assumed that users can only rate and comment on a listing or host if they have recently completed a stay, but the exact time frame for "recently" was not defined.

Search by Latitude and Longitude: Assumed that the search functionality based on latitude and longitude calculates the distance using a predefined algorithm (e.g., Haversine formula).

Amenity Suggestions: Assumed that amenity suggestions provided to hosts are based on the characteristics of the listing, the preferences of similar listings, or other relevant factors. The specific algorithm for suggesting amenities and estimating revenue increase was not provided.

Noun Phrase Extraction Criteria: Assumed that the noun phrase extraction for comment analysis focuses on key phrases that provide insights into the user experience, sentiments, and features of the listing. The threshold for considering a noun phrase as significant was not defined.

User Authentication and Security: Assumed that the system employs appropriate user authentication and data security measures, such as password hashing and encryption, to protect sensitive user information.

Comments and Reviews: Assumed that comments and reviews provided by users are stored and associated with the corresponding listings or hosts for future reference and analysis.

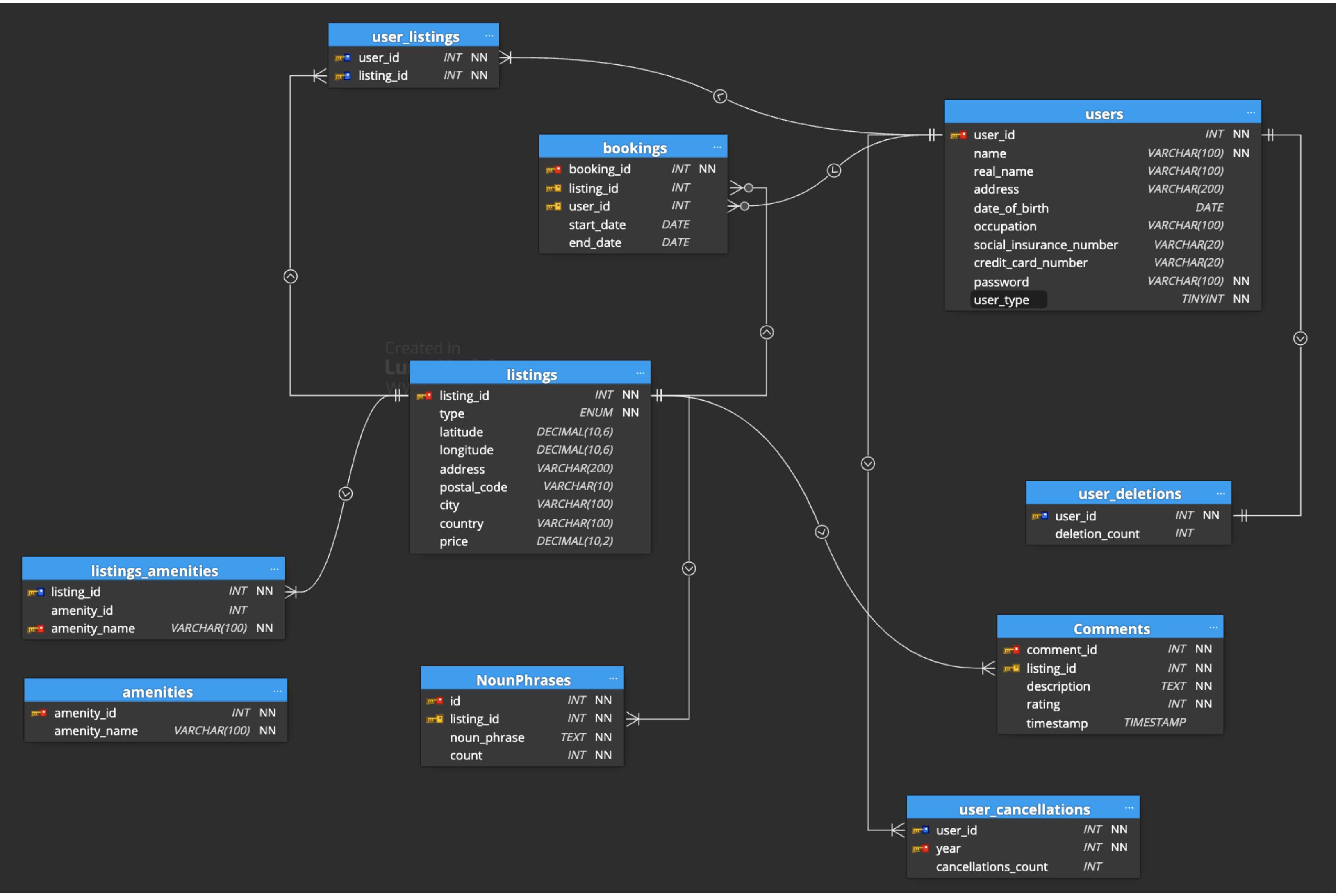
Interaction Limits: Assumed that interactions between hosts and renters (e.g., comments, ratings) are subject to certain limitations, but the specific limits were not defined.

Database Implementation: Assumed that the database implementation uses Java embedded SQL with MySQL, as specified in the project description.

GitHub Repo

# ER Diagram

https://github.com/Mustafa0503/CSCC43/blob/master/ERdiagram.png



GitHub Repo

## Relation Schema and Keys

**users**:
user_id (Primary Key)
name
real_name
address
date_of_birth
occupation
social_insurance_number
credit_card_number
password
User_type

**listings**:
listing_id (Primary Key)
type
latitude
longitude
address
postal_code
city
country
price

**Comments**:
comment_id (Primary Key)
listing_id (Foreign Key referencing listings)
description
rating
Timestamp

**NounPhrases**:
id (Primary Key)
listing_id (Foreign Key referencing listings)
noun_phrase
count

GitHub Repo

**user_listings**:
user_id (Foreign Key referencing users)
listing_id (Foreign Key referencing listings)
Primary key: {user_id, listing_id}

**listings_amenities**:
listing_id (Foreign Key referencing listings)
amenity_id (Foreign Key referencing amenities)
amenity_name
Primary key: {listing_id, amenity_name}

**amenities**:
amenity_id (Primary Key)
amenity_name

**bookings**:
booking_id (Primary Key)
listing_id (Foreign Key referencing listings)
user_id (Foreign Key referencing users)
start_date
end_date

**user_cancellations**:
user_id (Foreign Key referencing users)
year
cancellations_count
Primary Key: {user_id, year}

**user_deletions**:
user_id (Primary key, Foreign Key referencing users)
deletion_count

GitHub Repo

## DDL Statements:

```
DROP TABLE IF EXISTS NounPhrases;
DROP TABLE IF EXISTS Comments;
DROP TABLE IF EXISTS user_listings;
DROP TABLE IF EXISTS bookings;
DROP TABLE IF EXISTS listings_amenities;
DROP TABLE IF EXISTS listings;
DROP TABLE IF EXISTS user_cancellations;
DROP TABLE IF EXISTS user_deletions;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS amenities;


CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    real_name VARCHAR(100),
    address VARCHAR(200),
    date_of_birth DATE,
    occupation VARCHAR(100),
    social_insurance_number VARCHAR(20),
    credit_card_number VARCHAR(20),
    password VARCHAR(100) NOT NULL,
    user_type TINYINT NOT NULL DEFAULT 0
);
```

GitHub Repo

```sql
CREATE TABLE listings (
    listing_id INT AUTO_INCREMENT PRIMARY KEY,
    type ENUM('house', 'apartment', 'room') NOT NULL,
    latitude DECIMAL(10, 6),
    longitude DECIMAL(10, 6),
    address VARCHAR(200),
    postal_code VARCHAR(10),
    city VARCHAR(100),
    country VARCHAR(100),
    price DECIMAL(10, 2)
);


CREATE TABLE Comments (
    comment_id INT AUTO_INCREMENT PRIMARY KEY,
    listing_id INT NOT NULL,
    description TEXT NOT NULL,
    rating INT NOT NULL  CHECK (rating >= 1 AND
    rating  <= 5),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (listing_id)
    REFERENCES listings(listing_id)
);

CREATE TABLE NounPhrases (
    id INT AUTO_INCREMENT PRIMARY KEY,
    listing_id INT NOT NULL,
    noun_phrase TEXT NOT NULL,
    count INT NOT NULL,
    FOREIGN KEY (listing_id)
    REFERENCES listings(listing_id)
);
```

GitHub Repo

```
CREATE TABLE user_listings (
user_id INT,
listing_id INT,
    PRIMARY KEY (user_id, listing_id),
    FOREIGN KEY (user_id) REFERENCES users (user_id),
    FOREIGN KEY (listing_id) REFERENCES
    listings (listing_id)
);


CREATE TABLE listings_amenities (
       listing_id INT,
       amenity_id INT,
       amenity_name VARCHAR(100) NOT NULL,
       PRIMARY KEY (listing_id, amenity_name),
       FOREIGN KEY (listing_id) REFERENCES listings
(listing_id)
);


CREATE TABLE amenities (
    amenity_id INT PRIMARY KEY AUTO_INCREMENT,
    amenity_name VARCHAR(100) NOT NULL
);

CREATE TABLE bookings (
     booking_id INT AUTO_INCREMENT PRIMARY KEY,
     listing_id INT,
     user_id INT,
     start_date DATE,
     end_date DATE,
     FOREIGN KEY (listing_id)
     REFERENCES listings(listing_id),
     FOREIGN KEY (user_id) REFERENCES users(user_id));
```

```
CREATE TABLE user_cancellations (
       user_id INT,
       year INT,
       cancellations_count INT,
       PRIMARY KEY (user_id, year),
       FOREIGN KEY (user_id) REFERENCES
       users (user_id)
);

CREATE TABLE user_deletions (
       user_id INT,
       deletion_count INT,
       PRIMARY KEY (user_id),
       FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

**Source Code:** Github     SQL file