Project Description

MyBnB is a home sharing service platform like AirBnb in which users can list or book residences around the world. Recognizing its mass audience, MyBnB is planned to be a full-stack application with web and mobile support.
Considering course objectives and limited time-frame, this project (its source code) aims to provide the HTTP server (Java) and database (MySQL) portion of MyBnB with minimal functionality given by the project requirements. Similarly, Java SQL adapters are used with raw MySQL queries to enable the developer to strenthen his understanding on the database transactions. No frameworks or ORMs are used and most if not all resulting tables from the SQL queries are returned directly as HTTP response (i.e. most operations including but not limited to sorting, filtering, etc., are done through SQL). Nevertheless, the server code is written as systematically as possible to keep a consistent high code quality.

Source Code and Setup

The source code can be found in https://github.com/pasaaliaslan/mybnb. 'README.md' provides all the necessary information (or references the documents with the specific information), including the setup process on a local machine, server API documentation, and blueprints of the database system (ER diagrams, schemas, assumptions, etc.).
After one completes the setup process, he can run HTTP queries on terminal or (more simply) on Postman. Each endpoint is well documented under './doc/apis'

Assumptions

ER Diagram

Schemas and Constraints

Limitations and Areas of Improvement

1. Country, Subcountry, and City tables are prepopulated to prevent multiple rows for the same entry (e.g., 'toronto' and 'Toronto' most likely refer to same city). Prepopulation is possible because the dataset is already provided and a subject to little or change in the long run (i.e., country/subcountry/city names change infrequently if not never.). Similarly, PostalCode field could have been prepopulated to further improve consistency, especially when the algorithm for price suggestion is considered.

2. There is a chance for a coordinate value to not match with the address. There is no way to prevent this issue on the database layer, but external APIs can be utilized to ensure coordinate values match the address.

3. Error handling on the server level is minimal, putting responsibility to the potential frontend level of the application. Server provides some helper viewsets to assist frontend in this regard. For instance, Country viewset yields the full list of countries stored in the Country table of the database, so that the

frontend could serve this list as options (like a dropdown component) and prevent users from inputting invalid values.

4. There is no authentication implemented in the server. Hence, anyone can call any endpoint, which is not ideal when the host/user hierarchy is considered. Simple token/session can be implemented to solve endpoint access issues entirely.

5. Key properties cannot be updated in MyBnB. For instance, as 'username' is key of 'User' table, one cannot simply change his/her username. There are various reasons for this design choice. First of all, as all updates are applied on non-key properties, it is guaranteed to not update any foreign references (assuming design follows normal forms). Thus, the update occurs only in the original table and nowhere else. Also, foreign references can break among different tables and rows if key properties could be updated. For instance, assume we have users A and B who lives in the same address. Hence, User A and User B both refer to same row in the 'Address' table through 'livesIn' table. Let User A change his address (i.e. key in 'Address' table). As A and B were refering the same row, the system would change the address of B too, which is not the case.