Student Name: Ali Nada

Student Number: 1007018585

Project Documentation

Description of project and problems encountered:

The purpose of this project is to replicate the basic features of Airbnb (see user manual on page 5 for more information on the basic features). This project has a text based interface that can be run from the console.

There were a few problems encountered while making this project. First, because projects such as this one require time to make a robust implementation, a few assumptions had to be made (see assumptions below), for example, assuming that all user input will be valid and therefore no validation measures were taken. Second, it was very important to format queries correctly in the java code before executing them. For example, when dealing with a MySQL VARCHAR, if the goal was to insert a varchar into some table, then the statement would have to be formatted as follows:

> Statement.executeUpdate("INSERT INTO some_table(column_name) VALUES ('var_name')")

And Not:

> Statement.executeUpdate("INSERT INTO some_table(column_name) VALUES (var_name)")

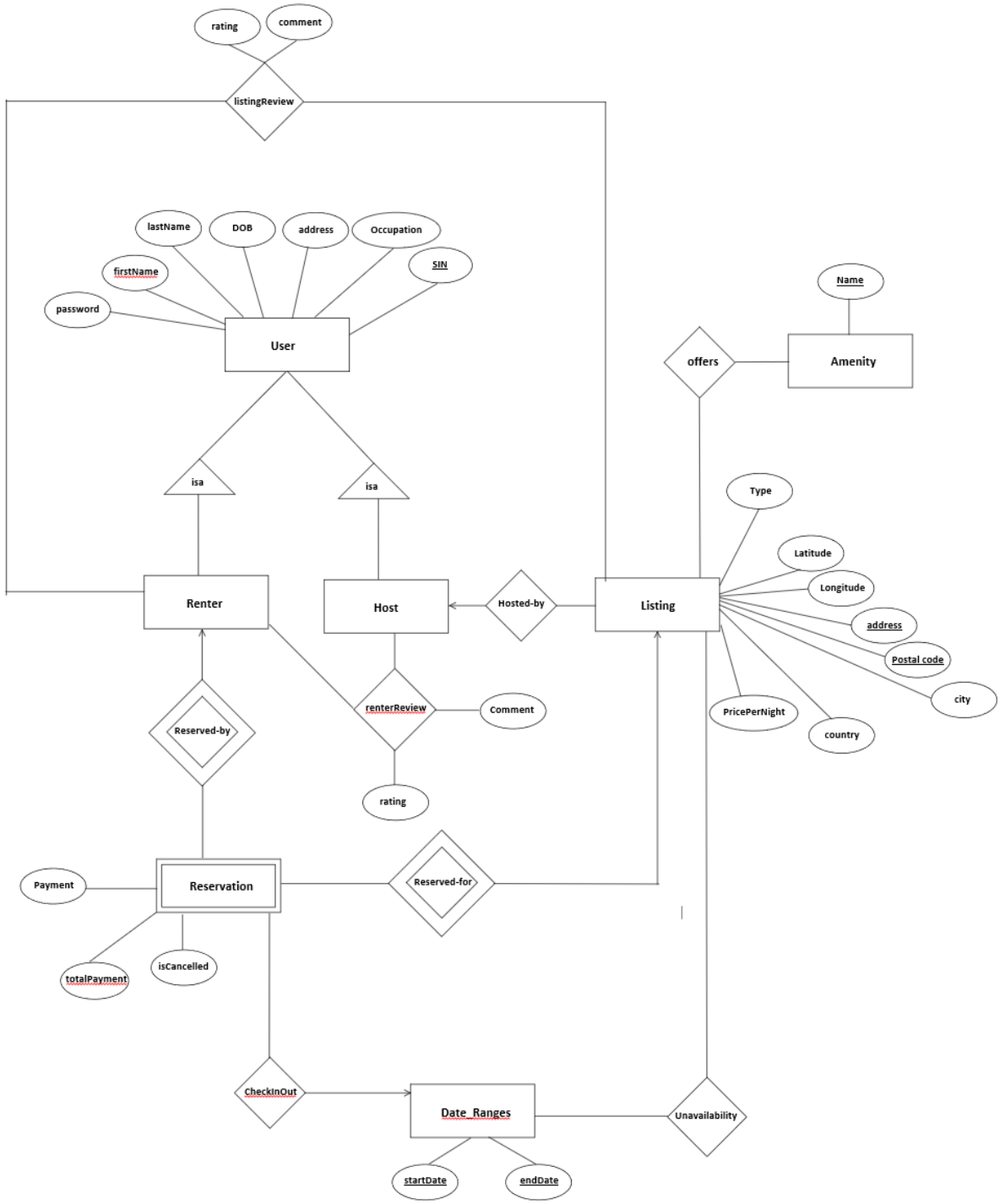
Notice the main difference between these two statements is that the second one does not have quotation marks around var_name. If not careful, this could lead to bugs that were very hard to find because it is difficult to detect a subtle difference such as this. Therefore, each operation, query and report were always separated into many individual components before combining them in order top pinpoint where the errors were coming from in case they occur.

Assumptions:

- All user input is valid. Any invalid input will yield undefined results. What is meant here by valid is the following:
    - The user will choose one of the options given in a prompt. So, if a prompt gives the option to either create a listing by inputting "cl" or delete a listing by inputting "dl", it is assumed the user will not input anything other than these two options

    - Any operations that require deleting/updating something from the database assume that these pieces of data do exist. So, if user would like to update a price of a listing, it is assumed the user had already created that listing beforehand
    - User input will follow the format specified. So, when inputting a date, the format will be in YYYY-MM-DD
- No two listings will have the same address and postal code. In the case of apartments in the same building, the apartment number will be included in the address
- A listing can only have one host and no co-hosts. While a practical implementation would have this feature, this restriction was placed to remove a layer of complexity from the project in this given time constraint
- A renter can comment on multiple different listings as long as they rented them, but for each listing, they can only comment once on that listing
- A host can comment on multiple renters as long as they rented from them, but for each renter, they can only comment once on that renter

ER diagram of the enterprise (page below):


rating
comment
listingReview
lastName
DOB
address
Occupation
SIN
firstName
password
User
Name
offers
Amenity
isa
isa
Type
Latitude
Longitude
address
Postal code
city
country
PricePerNight
Renter
Host
Hosted-by
Listing
Reserved-by
renterReview
Comment
rating
Payment
Reservation
Reserved-for
totalPayment
isCancelled
CheckInOut
Date_Ranges
Unavailability
startDate
endDate

Relational schema:

User(SIN, firstName, lastName, DOB, address, occupation, password)
Renter(SIN)
Host(SIN)
Listing(address, postalCode, city, country, type, latitude, longitude, pricePerNight)
Amenity(name)
DateRange(startDate, endDate)
RenterReview(hostSIN, renterSIN, comment, rating)
listingReview(renterSIN, listingAddress, listingPostalCode, comment, rating)
Reservation(renter-SIN, listing-address, listingPostalCode, checkIn, checkOut, payment, isCancelled, totalPayment)
Hosted-by(listing-address)
Offers(listing-address, listingPostalCode, amenity-name)
Check-in(renter-SIN, listing-address, listingPostalCode)
Check-out(renter-SIN, listing-address, listingPostalCode)
Unavailability(listing-address, listingPostalCode, startDate, endDate)

User manual:

Signing in / registering:

When starting up the program, the user is presented with two options:

```
Hi! Would you like to:
1. Register
2. Sign in
```

To Register, either input “1” or “register”. Similarly, to sign in, input either “2” or “sign in”.

Registering:

Upon choosing to register, the user will be prompted to enter their relevant information which will then be processed and registered into the database

```
SIN: 192837465
Password (press enter for no password): password
First Name: Ali
last Name: Nada
Username: a_NC43
Date of birth(YYYY-MM-DD): 2002-10-29
Address: 1265 Military Trail
Occupation: Student
Would you like to be host(Y/N): n
Would you like to be renter(Y/N): y
Registration Successfull!
```

Signing in:

Upon choosing to sign in, the user's SIN and password will be verified an they will be signed in:

```
SIN: 192837465
Password (press enter for no password): password
Welcome back Ali Nada!
```

Options:

After a user signs in or registers, they will be show what their options. Each user will have a different view of which options they can choose depending on whether they are a renter, host, or both since there are some operations only accessible to hosts and renters:

Renter view:

```
What would you like to do:
- Book a Reservation (BR)
- Comment on past listing (LC)
- Search for listings close to a specific location (SL)
- Search for listings close to a specific postal code (SP)
- Search for a listing with a specific address (SS)
- Cancel a Reservation (CR)
- Run a report (RP)
- Switch account (SA)
- Logout (L)
```

Host view:

```
What would you like to do:
- Create a Listing (CL)
- Remove a Listing (RL)
- Update listing price (UP)
- Create an unavailable time range for a Listing (CU)
- Remove an unavailable time range for a Listing (RU)
- Comment on past renter (RC)
- Cancel a Reservation (CR)
- Run a report (RP)
- Switch account (SA)
- Logout (L)
```

A view for someone who is both a renter and a host would just show all the above options

Create a listing:

If a host chooses to make a reservation, the relevant information is collected, then verified and finally if it is valid (no duplicate address and postal code) then the listing is created. The host can either choose the option of a host toolkit to help with the pricing and amenities or just set those manually:

Without host toolkit:

```
Address: 123 Some Street

Postal Code: U6NG4C

City: Toronto

Country: Canada

Type of listing (house, apartment, guest house or hotel): guest house

Latitude: 43.458

Longitude: -72.821
Would you like the host toolkit to help you with pricing and amenities? (Y/N):
n
Price per night: 225
- Beach View
- Dedicated Workspace
- Dryer
- Garden View
- Kitchen
- Pets Allowed
- Pool
- Security Cameras
- TV
- Washer
- Wifi

Which amenities do you offer from the ones listed above? Separate values by commas NO SPACES(e.x. 'TV,Wifi'):
beach view,dryer,washer,kitchen
```

With host toolkit:

```
Address: 123 some street

Postal Code: U6NG4C

City: Toronto

Country: Canada

Type of listing (house, apartment, guest house or hotel): guest house

Latitude: 43.458

Longitude: -72.821
Would you like the host toolkit to help you with pricing and amenities? (Y/N):
y
The price will be set to: 100.0
The amenities will be set to: Wifi,TV,Washer,Dryer
You can always change them later
```

Remove a listing:

To remove a listing, the address and postal code will be collected and then the listing will subsequently be deleted:

```
Address: 123 some street

Postal Code: U6NG4C
```

Update a listing price:

The address and postalCode are collected, verified, and the listing price is updated

```
Address: 564 main st e

Postal Code: e8c1t5
What would you like the new price to be:
20
```

All other operations, queries, and reports follow the same pattern of collecting information, validating it, then creating/deleting/inserting data if the data does not violate any constraints

System limitations and possibilities for improvement:

In this project, there are numerous things that can be improved.

- **User validation:** First and foremost, user input needs to be validated. It is more realistic that many users do not always follow guidelines of the options provided and so it is up to the programmer to make sure the user input will not break the program
- **Frameworks:** While a text-based interface is simple and easy to implement, using a framework such as spring to create a GUI will make much more user-friendly interface with an enhanced experience, such as showing pictures of listings.
- **Encryption:** to make the application more secure, it is important to encrypt important information such as passwords and payment information
- **Using a third party maps API:** to get accurate distances when doing a distance search it would be a good idea to use an API that can calculate accurate distances that take into account polar flattening and other factors that affect distance between locations
- **User SQL schema:** the way the schema was designed, there is the user table, and the host and renter tables which refer to SINs in the user table, this can lead to very slow operations when verifying that a user is a renter or a host, or when joining tables. This

can be mitigated by simply creating one user table and giving anyone the capability of being either a user or a renter without storing their SINS in a separate table

- **Creating a unique key for listings:** the way the schema was designed, a key for a listing would be the combination of its listing and postal code. While this is not necessarily a bad choice of key, simply creating a listing ID would significantly shorten the amount of information that needs to be stored, since there are also foreign key references to a listing from many other tables (reservation, offers, listingReview, etc.)