Jeremy Neilson CSCC43 Final Report:

Introduction:

This project was done alone, there was no help from any others throughout the process.

Copy this link to access the GitHub for this project:

https://github.com/JeremyNeilson/CSCC43/tree/main

The source code is within the CSCC43 folder of the repo, and contains two SQL script files in addition to all Java implementation files:

- CSCC43ProjectScript
- bookingScript

These scripts are designed to be run in the listed order in the MySQL Workbench and will create and populate the database with a suitable amount of data for demonstration.

The Java Build Path relies on three external jars:

- mysql-connector-java-8.0.29.jar
- opennlp-tools-2.3.0.jar
- opennlp-uima-2.3.0.jar

The first is so the project can connect to the database through JDBC, and the other two are for using OpenNLP for the Noun Parsing report.

To access report functionalities, there is a hardcoded account.

Email: root@root.com

Password: rootpassword

These details will bring you into the administration account that can run database reports.

Description of the Solution:

This project contains a text based JDBC and MySQL BnB service. Since JDBC is used, the project is obviously written in Java. This project

Each page the user can visit, for the most part, has its own hub file, which has a ‘runHub’ method that is run immediately after initialization to enter the user into that page. There are then additional methods to support the functionalities of each page.

In addition to those hub files, there are User, Listing, Booking and Payment classes that are used to store individual rows from the tables of the same names. H_review and u_review tables did not really need their own classes, as there was no need to store the information past first retrieval from a ResultSet, or after storing it in the database.

Finally, there are some miscellaneous files to help keep things more organized, like a login file, a printer interface that multiple different printer types inherit from (printers are for printing listings and bookings to the user), and a tuple class that is used in the Noun Parsing.

There were several problems encountered, most of which were simple debugging issues. Many SQL queries did not work as expected and needed review and rewriting.

Host Algorithm:

The host algorithm for this system does two things:

- Provides hosts a suggested price when scheduling availabilities for their listing that is tailored to the type of the listing. In addition to that, weekday scheduling mode will provide a suggested price for the type and the day of the week selected.
- When selecting available amenities, hosts can choose to receive suggestions for amenities for their listing, depending on the type and a few other factors that are asked about.

It is a rather simple algorithm, and I attribute that to my being alone for this project, and this algorithm was one of the last things to be added to the project.

User Manual:

To use this program, open the project in an IDE and run the project in the IDE's terminal. Navigation through pages is done by typing in one of the possible letters that correspond to each possible page you can navigate to. These letters are between square brackets.

For example: "Welcome to Jeremy's BnB service: [L]ogin, [C]reate Account, [E]xit Application"

The options when this text is displayed are 'L' or 'C' or 'E'. Case does not matter.

Aside from that, there are some pages with listed options where you must enter a number that corresponds to the entry in the list that you wish to view/work with/etc.

For example: "[0]: Total Bookings By City…"

In the report hub, to select the above report, you would enter '0' as your input.

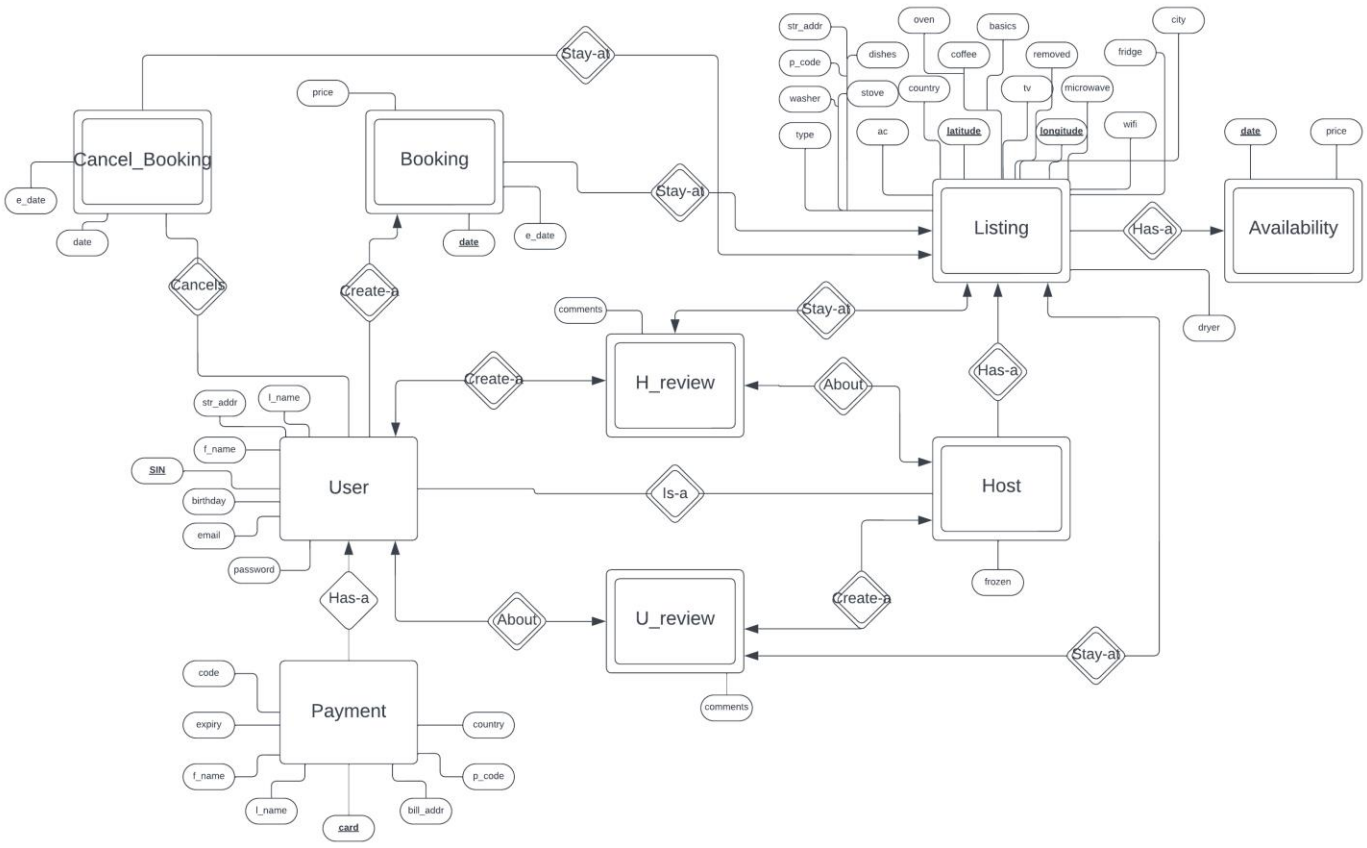
Assumptions:

While the system can handle a user removing their address and then adding it again, it is not really designed for that, and the functionality of doing so isn't perfect. We therefore assume that if a user removes their listing, they will not want to add it again.

Relation Schema and Keys:

user(sin, f_name, l_name, str_addr, occuptation, email, password, birthday, c_card)
host(h_sin, frozen)
listing(latitude, longitude, host, type, str_addr, p_code, city, country, washer, dryer, tv, ac, wifi, stove, oven, basics, dishes, fridge, coffee, microwave, parking, removed)
availability(l_latitude, l_longitude, date, price)
booking(l_latitude, l_longitude, date, user, e_date, price)
cancel_booking(l_latitude, l_longitude, date, user, e_date)
payment(card, f_name, l_name, bill_addr, p_code, country, expiry, code)
h_review(host, stars, user, comments, latitude, longitude)
u_review(user, stars, host, comments, latitude, longitude)

ER Diagram:



DDL Statements:

User

```
CREATE TABLE `user` (
  `SIN` varchar(9) NOT NULL,
  `f_name` varchar(25) DEFAULT NULL,
  `l_name` varchar(25) DEFAULT NULL,
  `str_addr` varchar(50) DEFAULT NULL,
  `occupation` varchar(50) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  `password` varchar(50) DEFAULT NULL,
  `birthday` varchar(10) DEFAULT NULL,
  `c_card` varchar(19) DEFAULT NULL,
  PRIMARY KEY (`SIN`)
)
```

Host:

```
CREATE TABLE `host` (
  `h_SIN` varchar(9) DEFAULT NULL,
  `frozen` bit(1) DEFAULT NULL,
  KEY `h_SIN` (`h_SIN`),
  CONSTRAINT `host_ibfk_1` FOREIGN KEY (`h_SIN`)
REFERENCES `user` (`SIN`)
)
```

Payment:

```
CREATE TABLE `payment` (
  `card` varchar(19) NOT NULL,
  `f_name` varchar(25) DEFAULT NULL,
  `l_name` varchar(50) DEFAULT NULL,
```

Availability:

```
CREATE TABLE `availability` (
  `l_latitude` decimal(10,6) NOT NULL,
  `l_longitude` decimal(10,6) NOT NULL,
  `date` date NOT NULL,
```

```
  `bill_addr` varchar(50) DEFAULT NULL,
  `p_code` varchar(7) DEFAULT NULL,
  `country` varchar(56) DEFAULT NULL,
  `expiry` date DEFAULT NULL,
  `code` varchar(3) DEFAULT NULL,
  PRIMARY KEY (`card`)
)
```

Booking:

```
CREATE TABLE `booking` (
  `l_latitude` decimal(10,6) NOT NULL,
  `l_longitude` decimal(10,6) NOT NULL,
  `date` date NOT NULL,
  `user` varchar(9) NOT NULL,
  `e_date` date DEFAULT NULL,
  `price` float DEFAULT NULL,
  PRIMARY KEY
(`l_latitude`,`l_longitude`,`date`,`user`),
  KEY `user` (`user`),
  CONSTRAINT `booking_ibfk_2` FOREIGN KEY
(`user`) REFERENCES `user` (`SIN`)
)
```

H_review:

```
CREATE TABLE `h_review` (
  `host` varchar(9) NOT NULL,
  `stars` int DEFAULT NULL,
  `user` varchar(9) NOT NULL,
  `comments` varchar(500) DEFAULT NULL,
  `l_latitude` decimal(10,6) DEFAULT NULL,
  `l_longitude` decimal(10,6) DEFAULT NULL,
  PRIMARY KEY (`user`,`host`),
  KEY `host` (`host`),
  KEY `l_latitude` (`l_latitude`,`l_longitude`),
  CONSTRAINT `h_review_ibfk_1` FOREIGN KEY
(`user`) REFERENCES `user` (`SIN`),
  CONSTRAINT `h_review_ibfk_2` FOREIGN KEY
(`host`) REFERENCES `host` (`h_SIN`),
  CONSTRAINT `h_review_ibfk_3` FOREIGN KEY
(`l_latitude`, `l_longitude`) REFERENCES `listing`
(`latitude`, `longitude`)
)
```

Listing:

```
CREATE TABLE `listing` (
```

```
  `price` float DEFAULT NULL,
  PRIMARY KEY (`l_latitude`,`l_longitude`,`date`)
)
```

Cancel_booking:

```
CREATE TABLE `cancel_booking` (
  `l_latitude` decimal(10,6) DEFAULT NULL,
  `l_longitude` decimal(10,6) DEFAULT NULL,
  `date` date DEFAULT NULL,
  `user` varchar(9) DEFAULT NULL,
  `e_date` date DEFAULT NULL,
  `canceller` bit(1) DEFAULT NULL
)
```

U_review:

```
CREATE TABLE `u_review` (
  `user` varchar(9) NOT NULL,
  `stars` int DEFAULT NULL,
  `host` varchar(9) NOT NULL,
  `comments` varchar(500) DEFAULT NULL,
  `l_latitude` decimal(10,6) DEFAULT NULL,
  `l_longitude` decimal(10,6) DEFAULT NULL,
  PRIMARY KEY (`user`,`host`),
  KEY `host` (`host`),
  KEY `l_latitude` (`l_latitude`,`l_longitude`),
  CONSTRAINT `u_review_ibfk_1` FOREIGN KEY
(`user`) REFERENCES `user` (`SIN`),
  CONSTRAINT `u_review_ibfk_2` FOREIGN KEY
(`host`) REFERENCES `host` (`h_SIN`),
  CONSTRAINT `u_review_ibfk_3` FOREIGN KEY
(`l_latitude`, `l_longitude`) REFERENCES `listing`
(`latitude`, `longitude`)
)
```

```
  `latitude` decimal(10,6) NOT NULL,
  `longitude` decimal(10,6) NOT NULL,
  `host` varchar(9) DEFAULT NULL,
  `type` varchar(25) DEFAULT NULL,
  `str_addr` varchar(50) DEFAULT NULL,
  `p_code` varchar(7) DEFAULT NULL,
  `city` varchar(20) DEFAULT NULL,
  `country` varchar(30) DEFAULT NULL,
  `washer` bit(1) DEFAULT NULL,
  `dryer` bit(1) DEFAULT NULL,
  `tv` bit(1) DEFAULT NULL,
  `ac` bit(1) DEFAULT NULL,
  `wifi` bit(1) DEFAULT NULL,
  `stove` bit(1) DEFAULT NULL,
  `oven` bit(1) DEFAULT NULL,
  `basics` bit(1) DEFAULT NULL,
  `dishes` bit(1) DEFAULT NULL,
  `fridge` bit(1) DEFAULT NULL,
  `coffee` bit(1) DEFAULT NULL,
  `microwave` bit(1) DEFAULT NULL,
  `parking` int DEFAULT NULL,
  `removed` bit(1) DEFAULT NULL,
  PRIMARY KEY (`latitude`,`longitude`),
  KEY `host` (`host`),
  CONSTRAINT `listing_ibfk_1` FOREIGN KEY (`host`) REFERENCES `host` (`h_SIN`)
)
```

System Limitations and Potential Improvements

I believe this project fulfills all the necessary requirements that are asked for. However, there are certain limitations that do not matter for the scope of this project but could be improved to make this system work more fluidly and be more user-friendly in future.

1. **The ability to delete or edit a review you left for a host/guest**: This is not asked for in the handout, but like listings and bookings, being able to remove or edit a comment made about a host has obvious benefits for the user, like if a guest stays with a host at the same listing more than once, they may wish to change their review if the experience was better/worse.
2. **Duplication of code:** The design of the system required a lot of changes throughout development, and that led to a design in the end that is not particularly well-suited for expansion. If one wanted to continue developing this system, refactoring the existing system would likely prove to be one of the most useful things to start with.
3. **Additional features:** Obviously, this system is not as feature rich as the actual Airbnb system is, and potential future work could involve adding a chat system or other useful features not required by the handout.
4. **Reworking Availabilities:** As of right now, assigning availabilities for listings can be done in two ways: By weekday and manually. Weekdays are much faster and assign a price and availability for each day of the

week the user desires and saves availabilities for the next year like that. Manual adds availabilities for single days, which is much slower. This is obviously not the most effective method, but it was beyond my scope to find a solution that works in a text-based interface.

5. **Changing Keys:** Certain keys for certain tables probably could have been different and deserve to be reworked. For example, the key for the user table is their SIN, but an account should in reality be tied to the uniqueness of their email. This doesn't matter for the scope of this project, but the same person should technically be able to have multiple accounts, they just must use separate emails.