## URL to front-end pages:http://flip1.engr.oregonstate.edu:3468/

## Front and Backend Code: https://github.com/JackoBright/Nest-Quest/

## Reviews:

Review #10:

- *Does the UI utilize a SELECT for every table in the schema?* Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).

Yes, there are SELECT statements for all tables in the schema.

- *Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?*

Yes, there is a SELECT statement that filters out users by their first name.

- *Does the UI implement an INSERT for every table in the schema?* There should be UI input fields that correspond to each table and attribute in that table.

Yes, there are INSERT statements for all tables in the schema.

- *Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship?* For example, if there is an M:N relationship between *Orders* and *Products*, INSERTing a new order, should also INSERT row(s) into the intersection table *OrderDetails*. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.

Yes, all INSERT statements add corresponding FKs, and at least one M:N relationship that does this between Tenants and Properties.

- *Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship?* For example, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Yes, there is at least one DELETE statement and it is for the M:N relationship between Tenants and Properties.

- *Is there at least one UPDATE for any one entity?* For example, in the case of Products, can the attributes of an existing row/record be updated?

Yes, there is at least one UPDATE statement and it is for the M:N relationship between Tenants and Properties.

- *Is at least one 1:M relationship NULLable?* There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).

Yes, there is at least one 1:M relationship that is NULLable.

- *Do you have any other suggestions for the team to help with their HTML UI?* For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".

Everything looks great so far, the only suggestion I have is to add a way to distinguish between the different rows of the tables like a line or box.

## Review #11:

Hello Group 6. I see a select from each table dedicated towards showing each entity. I also see a selection meant to be used for each update. I also do see an insert for each entity. It also seems like you take into consideration regarding the foreign keys as you seem to add to Tenants when you add to TenantDirectory. That being said, I'm not super confident regarding how this would work, but wouldn't you technically have to also add it to Directory? Also wouldn't adding a Tenant involve inserting a new spot for TenantDirectory? I guess I would just make sure with someone else that you've added in consideration of the foreign keys correctly.

I see a delete option for each entity. Similar thought regarding insert and relationship of foreign keys. There also seems to be an update for each entity. I would consider whether you can make address and user_id NULL. If the user writes a review, will it be necessary for them to give their address? I guess I can see user_id being important though.

In terms of other suggestions, I found it a bit confusing trying to figure out which was your foreign keys within the Database Outline. In my opinion I would use some kind of symbol to illustrate that. I would also change the name "Tenant Directory" to something like "Tenants_Properties" so it's more clear that it's a intersection table between those two entities. That being said, I think you're DML for the most part looks good. Just took a little bit to understand how everything was labelled.

## Review #12:

**Does the UI utilize a SELECT for every table in the schema?**

Yes, the provided DML code includes SELECT queries for all tables in the schema.

**Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?**

Yes, there is a SELECT query that filters results based on the user's first name as an input.

**Does the UI implement an INSERT for every table in the schema?**

Yes, the provided DML code includes INSERT queries for all tables in the schema.

**Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship?**

Yes, the INSERT queries add the corresponding foreign key attributes, the schema provided shows a many-to-many relationship between Tenants and Properties, using foreign keys in Tenant Directory.

**Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship?**

The DML code includes DELETE queries for various tables.

**Is there at least one UPDATE for any one entity?**

Yes, the DML code includes UPDATE queries for multiple entities.

**Is at least one 1:M relationship NULLable?**

The provided documentation does not explicitly include an example of a nullable 1:M relationship. However, some foreign key columns could allow NULL values, making the relationship nullable.

**Do you have any other suggestions for the team to help with their HTML UI?**

HTML UI looks great, provides all the needed elements in a logical and easily navigable fashion.

Review #13:

- *Does the UI utilize a SELECT for every table in the schema?* Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).
    - Yes, the UI utilizes a SELECT query for all 7 entities in the schema.
- *Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?*
    - Yes. For example, there is a query which will selects a specific user given from the user/website.
- *Does the UI implement an INSERT for every table in the schema?* There should be UI input fields that correspond to each table and attribute in that table.
    - Yes, the UI implements an INSERT for every table in the schema with the form values corresponding to the attributes in respective tables.
- *Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship?* For example, if there is an M:N relationship between *Orders* and *Products*, INSERTing a new order, should also INSERT row(s) into the intersection table *OrderDetails*. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.

    - Yes, each INSERT also adds the corresponding FK attributes.
- *Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship?* For example, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.
    - Yes, there are DELETE queries for the required entities and only deletes the corresponding rows, nothing extra. e.g. the delete query for reviews does not delete the user nor the property itself.
- *Is there at least one UPDATE for any one entity?* For example, in the case of Products, can the attributes of an existing row/record be updated?
    - Yes, there are UPDATE queries for the required entities.
- *Is at least one 1:M relationship NULLable?* There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).
    - No, I don't think so (although I might be misunderstanding this question). Although there is at least one 1:M relationship, I don't think it is NULLable. For example, between Information Requests and Seekers. If I were to try to remove the user_id (i.e. make it Null) it would not let me since user_id is specified as "NOT NULL" in the database outline.
- *Do you have any other suggestions for the team to help with their HTML UI?* For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".
    - Yes, I think it would be a good idea to replace the column names with ones that gives more context and are more user friendly.
    - Also maybe adding more columns to give more context. For example, in the Tenants tab in the website, you could add the first name and last name relating to the user_id to provide more context of who the owner or tenant is.

## Revision Plan:

1. Rename the Tenant Directory table to Tenants_Properties
    a. Since it is just an intersection table, it should follow naming conventions
    b. Updated it in all the SQL files as well

2. Add table styling to the website
    a. Makes the tables easier to read

3. Add foreign key symbol to outline: ^

## Upgrades to the Draft version

1. Added address dropdown to tenant page.
2. Added submit buttons to creation pages
3. Removed traces of Photo attributes from DML, DDL, and HTML

## Revised Project Design:

## Title: NestQuest

## Team Members: Sankalp Patil and Michael Molineaux

## Overview:

Our project is to develop a web application that allows students at Oregon State University to find roommates and places to live. It also allows landlords to post information about their properties to attract tenants. Our database will be a relational database tasked with storing and listing information about vacant rooms in Corvallis, Oregon. This application is intended for students at Oregon State University. There are approximately 40,000 students, most of which live in apartments or townhomes off campus. There are over 400 apartments, townhomes, houses, and other properties where students reside off campus. We estimate that the average size of apartments will be about 1200, with the majority of apartments being good quality. As such, our database needs to have the storage capacity and flexibility to accommodate a large variety of living situations. Users need to be able to insert, delete, and modify certain information from this database given their current living situation and goals. This database will contain information such as names, addresses, ratings, and preferences - such as non-smoking, pets, and noise levels. Additionally, people will be classified into two categories. Owners are people who currently reside in a Property and are looking for roommates. Seekers are people who are looking for a place to live.

## Database Outline:

Primary keys have a star*
Foreign keys have a carat^

- **Users**: Holds the data of anyone who uses the application
  - user_id*^: int, auto_increment
  - fname: varchar
  - lname: varchar
  - email: varchar
  - phone: varchar
  - smoking: varchar
  - pets: varchar
  - gender: varchar
  - age: varchar
  - Relationships:
    - A 1: 0 or M relationship between Users and Reviews (A User can write many Reviews).

        - A 0 or 1 relationship between Users and Tenants (If a User exists, they can either be a Tenant or not a Tenant.)
        - A 0 or 1 relationship between Users and Seeker(If a User exists, they can either be a Seeker or not a Seeker.)

- **Tenants**: Holds the data of people residing in a Property who are looking for roommates
    - user_id*^: int
    - role: varchar
    - Relationships:
        - A 1: 1 or M relationship between Tenants and Tenant Directory (Many Tenants can be living in many Properties).
        - A 1:1 relationship between Tenants and Users (If a Tenant exists, they must be a User)

- **Seekers:** Holds data of people currently looking for roommates and a place to live.
    - user_id*^: int, auto_increment
    - price_upper: int
    - price_lower: int
    - Relationships:
        - A 1: 0 or M relationship between Seekers and Information request (A Seeker can send out multiple Information requests).
        - A 1:1 relationship between Seekers and Users (If a Seeker exists, they must be a User)

- **Information Requests:** Hold data about requests for Information about Properties sent by Seekers
    - user_id*^: int, not NULL, unique
    - date_contacted: DATE, not NULL
    - address*: varchar, not NULL
    - Relationships:
        - A 1:1 relationship to Seeker(One Information request must have one Seeker)
        - A 1:1 relationship to Property(One Information request can be about only one Property)

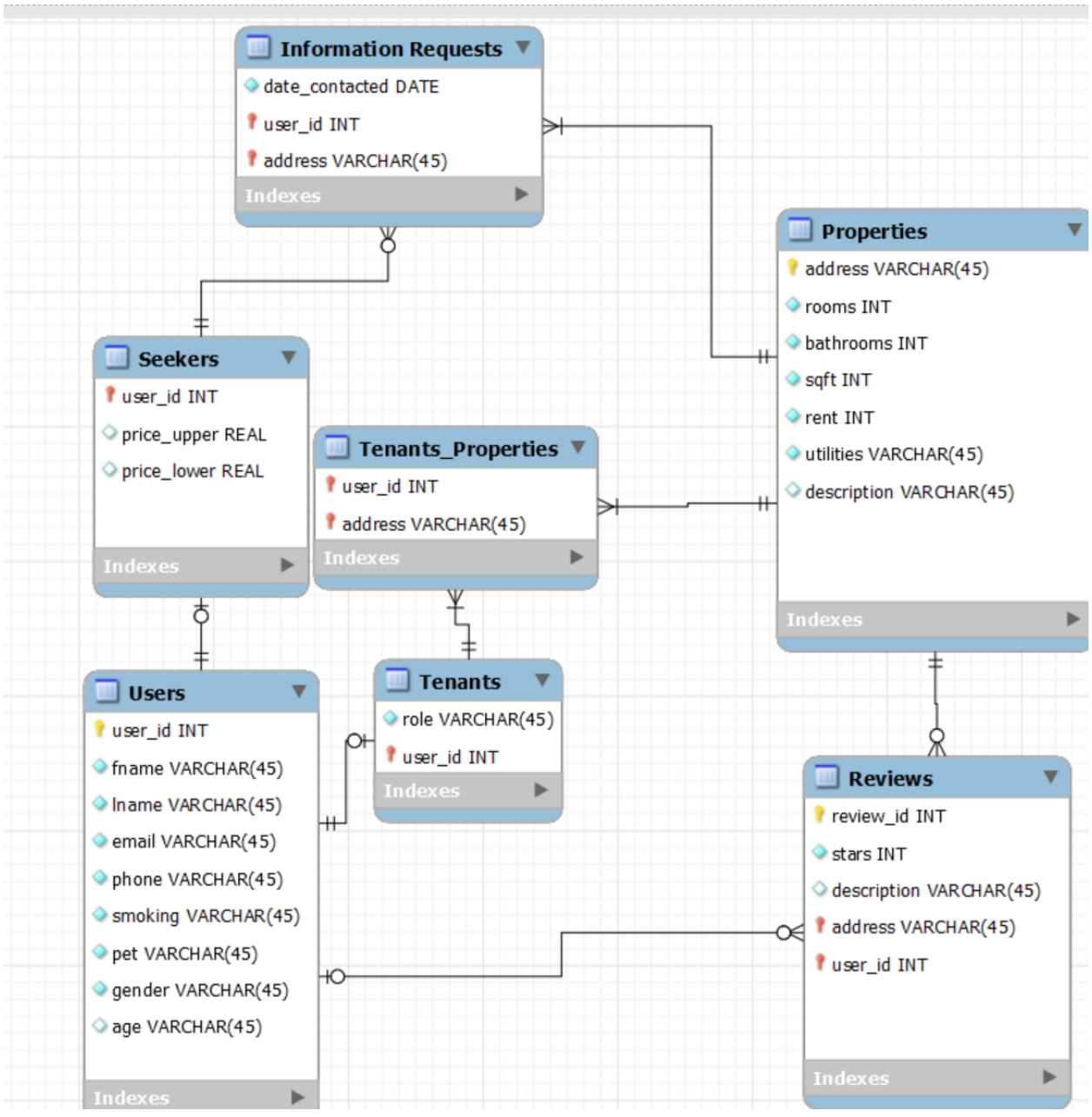
- **Properties:** The Properties that Tenants are living at where openings are available
    - address*: varchar, not NULL, unique
    - rooms: int, not NULL
    - bathrooms: int, not NULL
    - sq_ft: varchar, not NULL
    - rent: varchar, not NULL

- utilities: varchar, not NULL
- description: varchar
- Relationships:
  - A 1: 0 or M relationship to Information Request (One Property can have multiple Information requests)
  - A 1: 0 or M relationship to Tenant Directory (One Property can house multiple Tenants)
  - A 1: 0 or M relationship to Reviews (One Property can have none or multiple Reviews)

- **Reviews:** These are Reviews written by Users of the app about Properties
  - review_id*: int, auto_increment, not NULL
  - user_id^: int, not NULL
  - address^: varchar, not NULL
  - stars: int, not NULL
  - description: varchar
  - Relationships:
    - A 1:1 relationship with Users (Any given review can only be written by one User)
    - A 1:1 relationship to Properties (A review can only be related to one Property)

- **Tenants_Properties:** This is an intersection table between Tenants and Properties
  - user_id*^ int, not NULL (Note, the pk is the combination of the two)
  - address*^ varchar
  - Relationships:
    - A 1:1 relationship to Tenants
    - A 1:1 relationship to Properties

## Entity Relationship Diagram:



## Sample Data

| Users | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| user_id | fname | lname | email | phone | smoking | pet | gender | age |
| 1 | Rob | Michaels | rob@gmail.com | 1234567890 | yes | no | male | 22 |
| 2 | Amy | Stevenson | amy@outlook.com | 987654321 | no | no | female | 20 |
| 3 | Jim | Davis | jim@oregonstate.edu | 5413214567 | no | yes | male | 25 |
| 4 | Jordan | Brooks | jordan@gmail.com | 3219876540 | yes | yes | female | 20 |
| 5 | Bill | Bobert | bill@oregonstste.edu | 4436758766 | no | no | male | 24 |
| 6 | Jace | Smith | jace@yahoo.com | 4336751369 | no | yes | male | 21 |

| Tenants | |
|---|---|
| user_id | role |
| 4 | owner |
| 5 | tenant |
| 6 | tenant |

| Seekers | | |
|---|---|---|
| user_id | price_upper | price_lower |
| 1 | 750 | 0 |
| 2 | 950 | 300 |
| 3 | 600 | 0 |

| Properties | | | | | | |
|---|---|---|---|---|---|---|
| address | rooms | bathrooms | sqft | rent | utilities | description |
| 12345 Berry st | 3 | 3 | 1550 | 750 | water, sewage, garbage, washer/dryer | Townhome 2.2 miles from campus |
| 23465 west ave | 2 | 2 | 1450 | 850 | water, sewage, garbage, washer/dryer | Apartment room located 5 minutes from campus |
| 876 robin blvd | 4 | 3 | 1650 | 600 | electricity, water, sewage, garbage, washer/dryer | nice place |
| 23819 bald mtn rd | 4 | 4 | 1800 | 650 | electricity, water, sewage, garbage, washer/dryer | House from the 80s but in great condition. |

| Information Request | | |
|---|---|---|
| date_contacted | user_id | address |
| 12-11-2022 | 1 | 12345 Berry st |
| 09-28-2022 | 2 | 12345 Berry st |
| 02-24-2023 | 3 | 23819 bald mtn rd |

| Tenant_Properties | |
|---|---|
| user_id | address |
| 4 | 12345 Berry st |
| 4 | 23465 west ave |
| 5 | 876 robin blvd |
| 6 | 23819 bald mtn rd |

| Reviews | | | | |
|---|---|---|---|---|
| review_id | user_id | address | stars | description |
| 1 | 1 | 23819 bald mtn rd | 4 | great place! The location is nice |
| 2 | 2 | 876 robin blvd | 4 | Included electricity is a huge plus. |
| 3 | 2 | 23465 west ave | 3 | The rooms have a weird smell and the carpets are discolored |