Search4House

Yusuf Amer, Christina Nguyen, Arman Sandher
Team 35

CS 157A Sec 01

# 1 Project Description

## 1.1 Overview

This project will be a database application that is hosted for the housing industry to become easier to manage, and provide a simplified process for finding, arranging, and dealing with these properties. Its purpose is to create a single interface that allows the best interaction experience between the renter and the rentee.

### 1.1.1

The goal of this project is to build a community of housing-seekers and building owners to be able to find each other and match together to find a balance between those who need rooms and others who are looking to rent out spaces and get some extra cash. The demand for such an application is strong within the area surrounding SJSU, with so many students posting on several different social applications in order to find what would be the perfect fit for them. However, using such methods is not as effective as having a single web app that could do all of the research and matching done for a user, minimizing time, struggles, and energy for the users.

### 1.1.2

Currently, it is not easy to accomplish trustworthy research when it comes to finding housing accommodation, especially in Downtown San Jose, CA, where many San Jose State University students spend long durations of time seeking even just a bed to use for a night, instead of commuting far away, or paying thousands of dollars for a full apartment or even studio in the area. Our motivation is our own struggles going through this hunting party to find the best match for location, price, roommates, and lease length. As the economy in the Bay Area's Silicon Valley grows, so does the costs of living, which includes rent. Taking this into account with school tuition, and other living fees and expenses, having such an app could truly help many students save money in space that they may not be using.

### 1.1.3

The stakeholders for such a project idea would obviously be the students of that local university. These are the users that would get the primary benefit from having an application like this around and distributed to the neighborhoods within walking distance of the school. They would be finding the cheapest rent costs nearby and saving significant amounts of money that will add up after several months of discounted rent, minimizing some struggles related to housing and decreasing possible homelessness. The other users that would make sense to have as common stakeholders are the rentees, or the people with extra space to rent out, whether it's just a bed or area on the floor in a room to sleep in, to full rooms in houses, to apartments in a building, to a whole mansion. The range is very wide in order to accommodate people with different housing requirements, and access based on needs and available income. The owners of these places would be able to get more income on the side, through renting out to anyone starting from a single roommate to a large

number of people willing to live together to lower the rent paid by each renter, and still provide the homeowner good income.

Less common stakeholders could be neighbors of those renting out their spaces, because sometimes having so many roommates in a single space as neighbors could be very loud or messy or just uncomfortable. These people would need to adjust to the new number of people living near them, or at the very least have to come to an understanding or compromise about some rules which may involve noise levels, having friends over, and keeping parking spaces available. Another group that may also be forgotten about often times is the families of the students who are moving into housing locations on this application. Their emotions do play an aspect in how happy the students are living with strange people rather than back home with family, for the cost of commuting plus vs the rent of the apartment or room. Having worried or unsupportive family members could make the experience so much worse and harder than it should be if not all parties have settled with the issue.

**1.1.4**

The application domain is the housing industry in the neighborhoods in the areas near big city colleges and other places with condensed populations. This industry currently has many old school organizations successfully working in this department, and there are newer methods that are similar applications, but these either do not provide all of the options in an area, so one may not be able to find the best deal if it's not posted onto the application, or that the greatest deals found are all from scammers, and that there is nothing realistic about how many matches are made through the application. Our workaround is to look through other authenticated housing listings and set up a reference post on our web app as well, and also make sure to have all rentees' posted listings be authenticated and proved as a true place with accurate numbers, statistics, and rent costs.

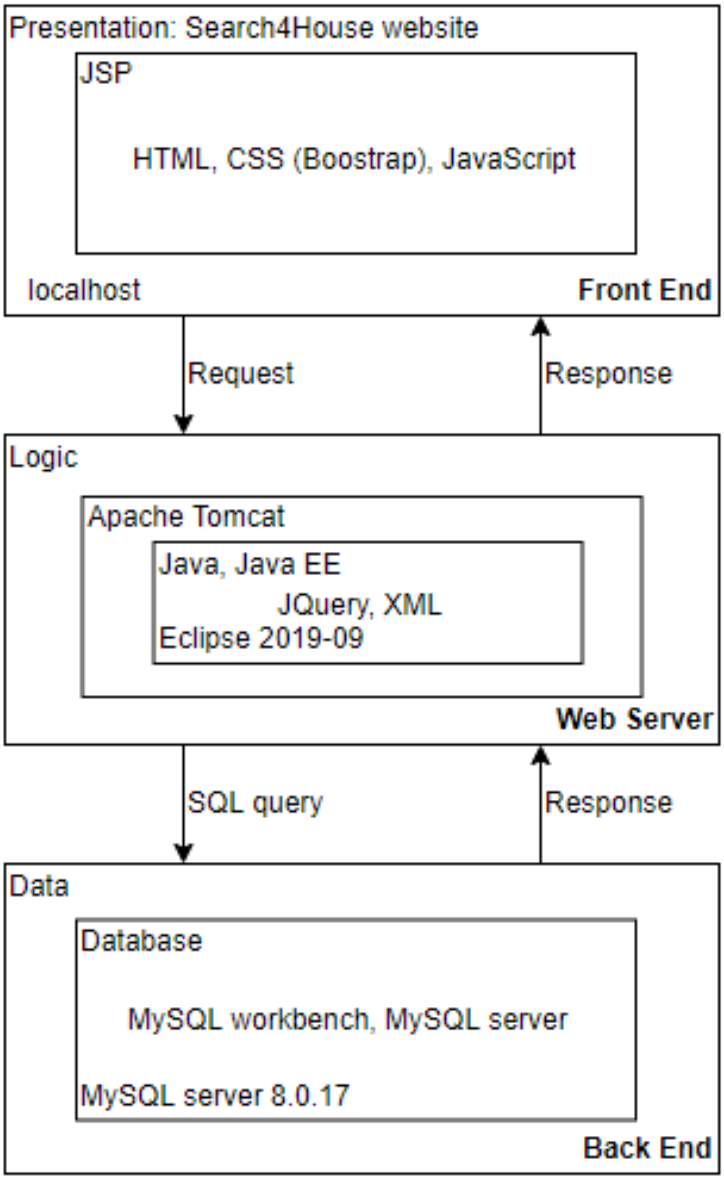
**1.1.5**

As mentioned all throughout the description, this project will provide the best deals for students to find housing nearby to their colleges, and the homeowners in the area would be able to rent out parts of their houses, full rooms, or share apartments for some extra income while making use of unused space. A typical user of this application would either be someone looking to make some money and rent out a space, or someone, usually a college student, looking for housing near a university, or in a large city. For example, a rentee could be someone who owns a large house and just simply has too much space and is looking for someone to live with them and possibly be friends with, or at least provide a little extra income for. It could also be someone renting out an apartment but would like to contribute some help to the community, so they provide some housing for anyone in need. Another possibility is that a bunch of college students are rooming together, but are focusing on school, so to afford rent they need even more roommates, and manage to split it with even more

people and for each to pay less. On the other hand, a renter could be a homeless person who just got a job, but does not have enough money saved to get their own place, or an international student who needs nearby housing for the time period of their student visa, or just a teenager looking to move out of their parents' or family's house and start living their life independently and on their own. These are all potential combinations for users to be matched up for housing with our application.

## 2 System Environment

### 2.1 Structure of the System:



## 2.2 Hardware/Software:

### 2.2.1 Apache Tomcat v8.5 Server:

Apache Tomcat, an open-source web server, allows users to run their web-based applications developed with Java on their local host servers. With the server, users can run Java Server Pages, JSP, files and Java servlets, Java programs.

### 2.2.2 Eclipse IDE for 2019-09:

An integrated development environment for Java Enterprise Edition (EE) used for web-development and web-applications. It includes the tools required to develop in both Java and JavaScript.

### 2.2.3 MacOS/Windows OS:

Our team will be using both Mac and Windows operating systems based on each members' preferences. Regardless of the OS, all members will be using the same software to develop the system.

## 2.3 RDBMS:

### 2.3.1 MySQL server 8.0.17:

Our team will be using MySQL to input, edit, and store our data. The server will be ran within our local host for running and testing during early development.

### 2.3.2 MySQL Workbench v8.0.17:

MySQL Workbench is a tool used for MySQL database development that allows users to visually view their database structure and values. Users can also generate and manage their database directly through the workbench.

## 2.4 Application languages:

### 2.4.1 Java 8:

Our team will be using Java for our main code. Our team is most familiar with this language and with MySQL Connector/J, it would be easy to connect our Java code to MySQL.

### 2.4.2 Java EE:

Java EE is built based on the regular Java programming language and also includes required tools for web development. Java EE allows for the 3-tier architecture in web development: presentation, logic, and data.

### 2.4.3 HTML:

HTML, Hypertext Markup Language, will be used for designing the basic formatting layout for our website.

### 2.4.4 CSS:

CSS, Cascading Style Sheets, will be used for designing the presentation version of our website. It will work along with HTML.

### 2.4.5 JavaScript:

JavaScript will be used for dynamic functions that directly communicates with HTML and user interactions and be able to communicate with the database.

### 2.4.6 XML:

XML, eXtensible Markup Language, will set rules and constraints on storing and sharing when publishing the data on the internet.

#### 2.4.7 SQL:

SQL is the language used for MySQL. This will be the primary language used for database management.

## 3 Functional Requirements

### 3.1 Users and their permissions:

The user will open their browser and type in our website into the address bar. Once they enter that, they should be on our website. From there, the user can do multiple things. There are a few other options the user can do upon entering the website, including logging into their own personal account. Users are limited to editing their own data and any data they have created. That person can be a guest, or a registered user. A guest only has the ability to read, while a user has the ability to read and write. The system is functional for those looking for housing, and those who are looking to put up a listing onto the website. The former has the ability to read, and the latter has the ability to write, but as long as they are users, they have both options open to them.

### 3.2 Functions, functional processes, and I/O:

#### 3.2.1a: Creating an Account:

If they do not already have an account, they would be able to do so by navigating to the "Log In" button on the navigation bar and selecting the "Register" button from there. All they need to input is a unique username and email along with their full name, phone number, and password to be able to create an account.

#### 3.2.1b: Returning User

If s/he is a returning user, all the user needs to do is click on the "Log In" button and it will redirect them to the page asking them to type in their username and password. On failure to do so will notify of an incorrect username and/or password. On success will lead them to their profile page. If they would like to log out of their account, there is a logout button also available in the navigation bar.

#### 3.2.2: Editing Profile:

If the user would like to edit their profile, they can do so by heading to the navigation bar and selecting "Profile." If the user is not logged in, it will redirect the user to the "Log In" page. Otherwise, the user can edit any information that they have access to and apply those changes. There are also some other options that they can view and edit while on this webpage such as any listing that they have added to a favorites list, or any listing that they have uploaded themselves.

#### 3.2.3a: Searching:

On the navigation bar, the user can click on the search bar and type in their desired zip code. From there, the user will be redirected to a page with all listings with that

zip code along with a side navigation bar that allows the user to filter by housing type: apartment, duplex, single room, shared room, and studio.

**3.2.4: View Listing:**

If the user has selected a housing option, they can expand on their desired listing by selecting on the listing photo which would redirect them to the page containing all the information on the listing. On that webpage, the person can find all the information pertaining to the listing as well as additional information like the contact information of the person who input the listing. The information that will show on the listing includes the pricing per month, the address, the number of bedrooms and bathrooms, the lease length, additional information, contact information, and a carousel of up to five photos.

**3.2.5: Favorite List:**

If multiple listings are found to be favorable, but the user still wants to look around, the user can click the heart button next to the listing to put it onto their favorites list to look at later. This list can be viewed by clicking the "favorites list" hyperlink at the navigation bar. By clicking this hyperlink, it will redirect the user to a different webpage that containing the top 5 favorited listings. From here, they can either view the listing by clicking on the visit button, which will take them to the listing page.

**3.2.6: Creating a Listing:**

If the user is looking to put up a listing onto the website, on the navigation bar, he/she is able to select "Add Listing." If the user is not already logged in, then it will redirect them to the "Log In" page to do so. If the user is already logged in, then it will redirect them to a different webpage that will prompt them for information for the listing such as pricing, number of bedrooms and bathrooms, contact information, location, 1 required and 4 optional photos, and any additional information which may include the number of people who will be living there, any extra amenities provided, etc. Once all the needed information has been successfully inputted, the user can press the submit button at the bottom of the page to put the listing in our database for our website to display or select the cancel button if the user changes his/her mind. The submit button will redirect the user to another webpage that shows all the details of the new listing and prompt the user to confirm the information or go back to re-edit it. Once confirmed, the user will be redirected to a view of the new listing. If the needed information was failed to be inputted, it the webpage will display an error stating to the user to input information in the required boxes.

**3.2.7: Editing a Listing:**

If the user would like to instead edit that they had put up, if they are logged in, they can head to their profile using the navigation bar. From there, they can click the button named "view your listings." This button will redirect the user to a page that will display a table showing all the listings a user has, if any. They can click any of the buttons listed next to their listings and that will take them to the listings page. At

the listing page, you can view all the information already on the listing. From here on the listing page, the user can press the edit button at the bottom of the page that will redirect them to a different page that will allow the user to edit the listing. When editing the listing, all prior saved values of the listing will be preloading into the form and the user can adjust each accordingly. Once the submit button is selected, the user will be redirected to a page with the updated values and will be redirected to a view of the newly updated listing.

## 4 Non-functional Issues

### 4.1 Graphical User Interface (GUI):

The website will have some basic components that are available on every page, like Home, Profile, Favorites, and Add buttons. These will always be available at the top of each page, also along with a Search Bar. If there is no user logged in, there will also be a Log In button populated along with the rest of them, and if there is already someone logged in, then a Log Out button will populate instead. The Home page will also have a Register button for new users to create accounts for themselves. The home page will also have a short introduction about the web application, and a carousel of some of the listings already currently posted to demo some photos. The Register page has several fields for the user to fill out, including username, full name, password, email address, and phone number. The Log In page will also have username and password fields for after a user already has an account created. These credentials will be shared among the other pages that are navigated to. The Log Out button will show you a confirmation page, then send you back to the Log In page after 3 seconds. The Search Bar on the top also allows anyone to look at listings within a zip code, whether logged in or not.

The Profile page will give you options to Edit your profile, View your listings, and will also show you details about your profile. When editing a Profile, one cannot change a username, because that is a unique identifier. Seeing your own listings will also Allow you to Edit details about them. Adding a Listing is a page with default user's contact information at the top, and then some listing information to be filled out like Type, Address, Price, Number of Bedrooms, Number of Bathrooms, Length of Lease, Photos, and Additional Info. These are the same details that can later be edited by the user that posted the listing, and then a Confirm Changes button will save and ensure those changes are applied.

The Search feature shows you several listings, and groups them by type. The left side of the page will also have shortcuts that will allow you to scroll quickly to where the next Type of listing starts. Each listing will show a Photo, Address, Price, and some more information. These can be selected for a fuller view of an individual Listing. When viewing a Listing in this mode, the Contact Information shows up, and that is how users get connected together and are able to find a match and complete their first search4House journey.

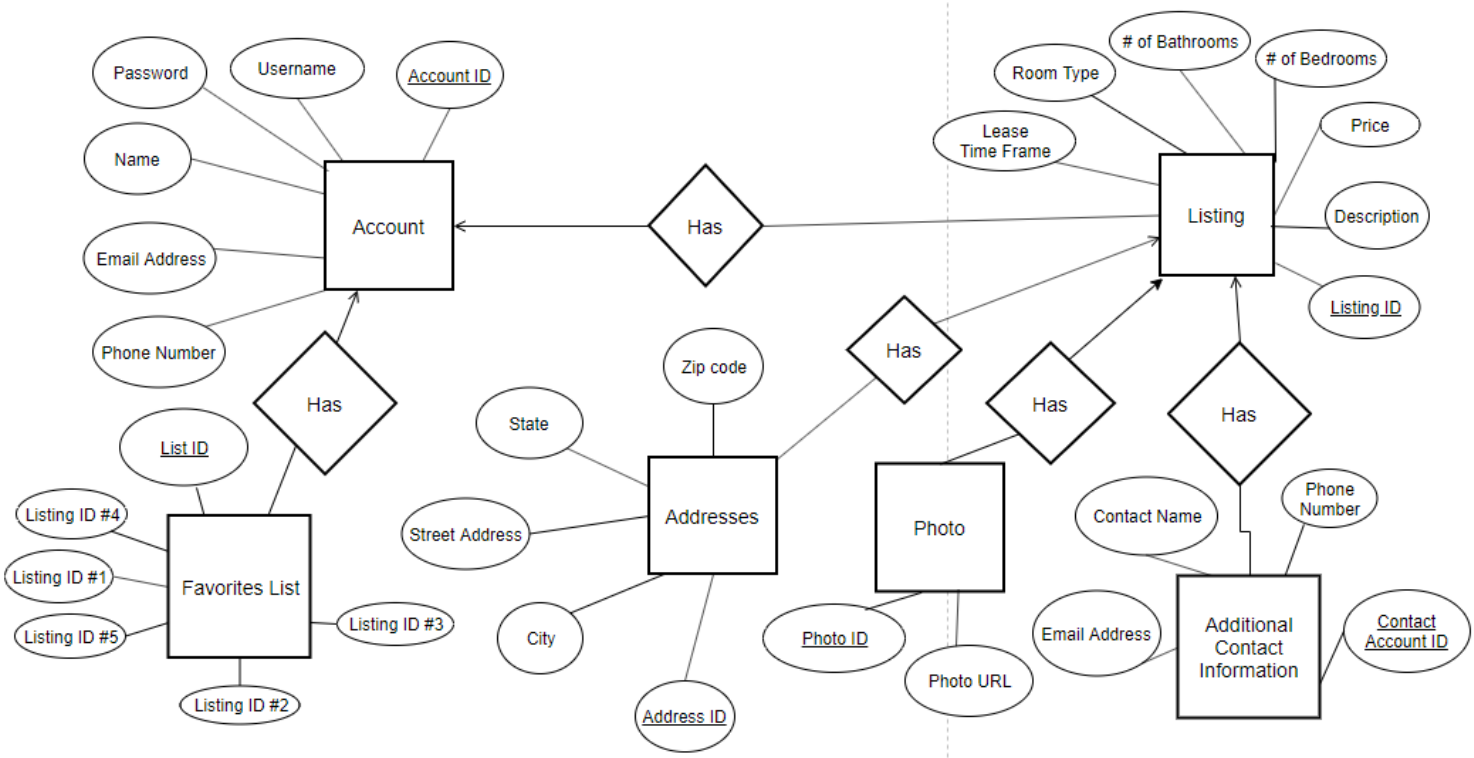
### 4.2 Security:

Each user will require an email and password in order to login and access listings that only they have created and their own personal account information. The email and password will be stored in the server's database and no users will be able to have access to view this.

### 4.3 Access Control:

Users will have access to their house listings and information such as description and pricing corresponding to it. Because they have access to this, users will be allowed to modify their listings through editing, deleting, or adding more listings. Users will be able to access their account information; this includes their name, contact information, and profile picture. However, users will not have direct access to the database in which their accounts and listings reside in. If they wish to make any changes to these, they will send a request to the database. The logic tier will read the changes and make changes to the database accordingly.

## 5 Entity Relationship Diagram (ERD)




## 5.1 Entities and their Attributes, Dependencies, Relationships and Constraints

### 5.1.1: Entities:

- Accounts(accID, username, password, fullName, email, phoneNum)
  - The Account entity set will be used for each user, whether a leaser or a renter. A unique Account ID will be used as the primary key for this entity set.
  - The username is the user's username, fullName is the user's full name, email is the user's email address, and phoneNum is the user's phone number.
- Listings(listingID, roomType, price, leaseTimeframe, roomNum, bathroomNum, description)
  - The Listing entity set will be used for each post by a leaser for renters to be able to scroll through and search for a good fit. Every post will have its own Listing ID, which may be reused when a posting opens up again after a lease ends.
  - The roomType is the listing's type of room (Apartment, Duplex, Shared Room, Single Room, or Studio), price is the listing's price per month, leaseTimeframe is the listing's minimum leasing time, roomNum is the number of rooms the listing has, bathroomNum is the number of bathrooms the listing has, and description is any additional information that the user wishes to input such as a more in-dept description.

- Favorites(listID, listing1ID, listing2ID, listing3ID, listing4ID, listing5ID)
  - The Favorites List entity set will be used as an entity set for each Account entity set, where a user prioritizes favorite listings posted. The list will have an id tied to it as the primary key and will have 5 attributes where the 5 photoIDs will be saved.
  - The listing1ID is the listing of the firstly added listing into the user's favorite list, etc.
- Photos(photoID, photoURL)
  - The Photo entity set will be used for the Listing entity set because the entity sets will need to provide pictures to help in identification. Each photo uploaded will have a unique Photo ID used as its primary key.
  - The photoURL is the URL of the photo not including the ".jpg" ending. This will be taken care of within the code.
- AdditionalContacts(contactAccID, phoneNum, email, name)
  - The Contact Information entity set will be used as an entity set for each Listing entity set, where a leaser can choose to add several other people to contact for inquiries or more information about the Listing posted.
  - The phoneNum is the additional contact's phone number, the email is the email address, and the name is his/her name.
- Addresses(addrID, streetAddress, city, state, zipCode)
  - The Address entity will be used as an entity for each Listing Entity. Each listing entity will have 1 address, but multiple addresses can be used for different listings, such as different rooms for a single house.
  - The streetAddress is the street address, city is the city, state is the state, and zipCode is the zip code.

**5.1.2: Relationships:**

- Accounts_Listings(listing_id, user_id)
  - Accounts can have 1 or more Listings in a one-to-many relationship. The listing_id is the id of the listing which will be the primary key because a listing can be owned by only one user.
- Listing_Address(listing_id, addr_id)
  - Listings can have exactly 1 Addresses in a one-to-many relationship. The listing_id is the id of the listing which will be the primary key because a listing can be owned by only one user.
- Listings_AdditionalContact(listing_id, addContact_id)
  - Listings can have exactly 1 Additional Contact in a one-to-many relationship. The listing_id is the id of the listing which will be the primary key because in our application, there should only be only contact to minimize confusion for users.

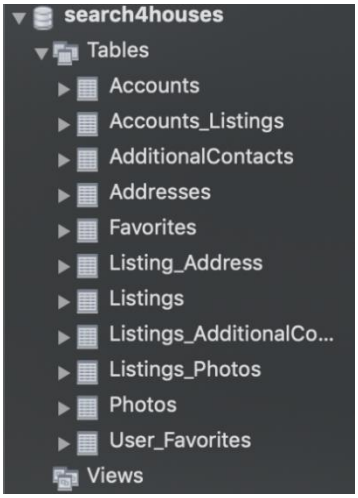
- Listings_Photos(photo_id, listing_id)
  - Listings can have 1 or more Photos in a one-to-many relationship. The photo_id is the id of the photo which will be the primary key because a photo should be unique to each different listing.
- User_Favorites(user_id, list_id)
  - Users can have exactly 1 Favorites list in a one-to-many relationship. The user_id is the id of the user which will be the primary key because each user should only have one favorites list.

**5.1.3: Nontrivial Dependencies:**

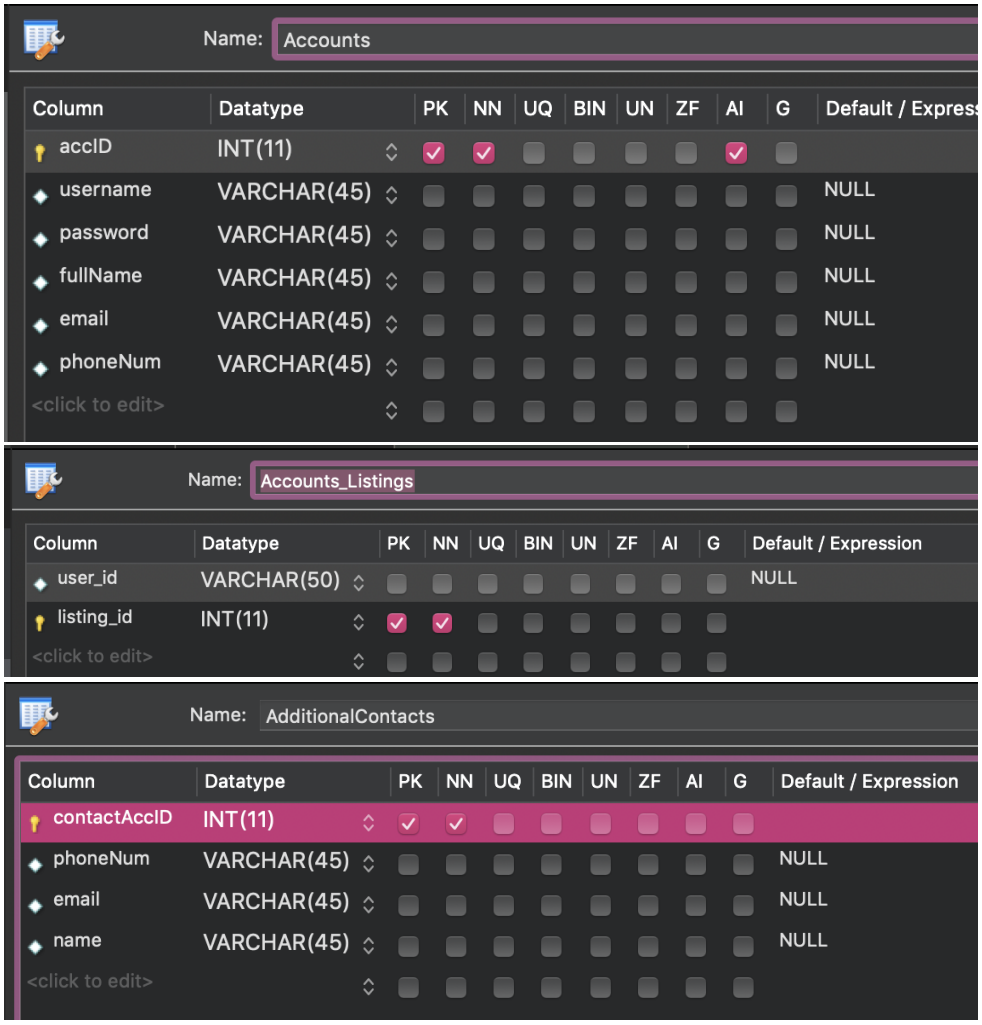
Our nontrivial dependencies are:

- accounts
  - accId → username
  - accId → password
  - accId →fullName
  - accId → email
  - accId → phonNum
- additionalcontacts
  - contactAccID → phoneNum
  - contactAccID → email
  - contactAccID → name
- addresses
  - addrID → streetAddress
  - addrID → city
  - addrID → state
  - addrID → zipCode
- favorites
  - listID → listing1ID
  - listID → listing2ID
  - listID → listing3ID
  - listID → listing4ID
  - listID → listing5ID
- listings
  - listingID → roomType
  - listingID → price
  - listingID → leaseTimeFrame
  - listingID → roomNum
  - listingID → bathroomNum
  - listingID → description

## 5.2 Schemas




### 5.3.1 Tables



Name: Addresses

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| addrID | INT(11) | ✓ | ✓ | | | | | | | |
| streetAddress | VARCHAR(45) | | | | | | | | | NULL |
| city | VARCHAR(45) | | | | | | | | | NULL |
| state | VARCHAR(45) | | | | | | | | | NULL |
| zipCode | VARCHAR(45) | | | | | | | | | NULL |
| <click to edit> | | | | | | | | | | |

Name: Favorites

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| listID | INT(11) | ✓ | ✓ | | | | | | | |
| listing1ID | VARCHAR(45) | | | | | | | | | NULL |
| listing2ID | VARCHAR(45) | | | | | | | | | NULL |
| listing3ID | VARCHAR(45) | | | | | | | | | NULL |
| listing4ID | VARCHAR(45) | | | | | | | | | NULL |
| listing5ID | VARCHAR(45) | | | | | | | | | NULL |
| <click to edit> | | | | | | | | | | |

Name: Listing_Address

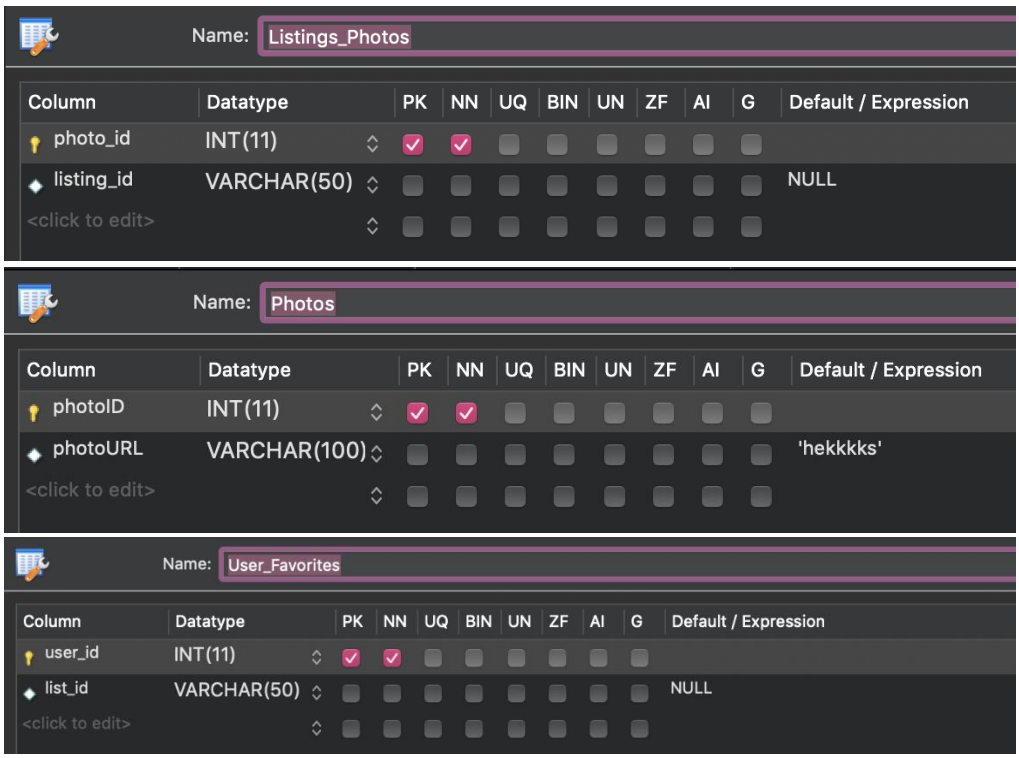
| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| listing_id | INT(11) | ✓ | ✓ | | | | | | | |
| addr_id | VARCHAR(50) | | | | | | | | | NULL |
| <click to edit> | | | | | | | | | | |

Name: Listings

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| listingID | INT(11) | ✓ | ✓ | | | | | | | |
| roomType | VARCHAR(50) | | | | | | | | | 'House' |
| price | INT(100) | | | | | | | | | '99' |
| leaseTimefra... | INT(100) | | | | | | | | | '99' |
| roomNum | INT(100) | | | | | | | | | '1' |
| bathroomNum | INT(100) | | | | | | | | | '1' |
| description | VARCHAR(100) | | | | | | | | | 'No additional information available.' |
| <click to edit> | | | | | | | | | | |

Name: Listings_AdditionalContact

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| listing_id | INT(50) | ✓ | ✓ | | | ✓ | | ✓ | | |
| addContact_id | VARCHAR(45) | | | | | | | | | NULL |
| <click to edit> | | | | | | | | | | |




# 6 Implementation

## 6.1 Explanation of Database System

We had a database that was implemented using MySQL. All of the tables and their information here. The JSPs that we have are created using JavaEE. Depending on the page, the contents of the JSPs are a mix of HTML, Java, CSS, SQL, and XML. The pages use SQL queries to take information from our database to use as a for multiple different purposes. It uses the HTML, Java, CSS, and XML in the page and the information from the database to generate our webpages and make changes on an internet browser.

Our Log-in page displays a page on the browser and the user inputs values for username and password. The page then takes those values and uses a SQL query to try to find an account with those specified values in the account table of the database. If an account is found, then it leads the user to the profile page. If not, it asks for the user to try again.

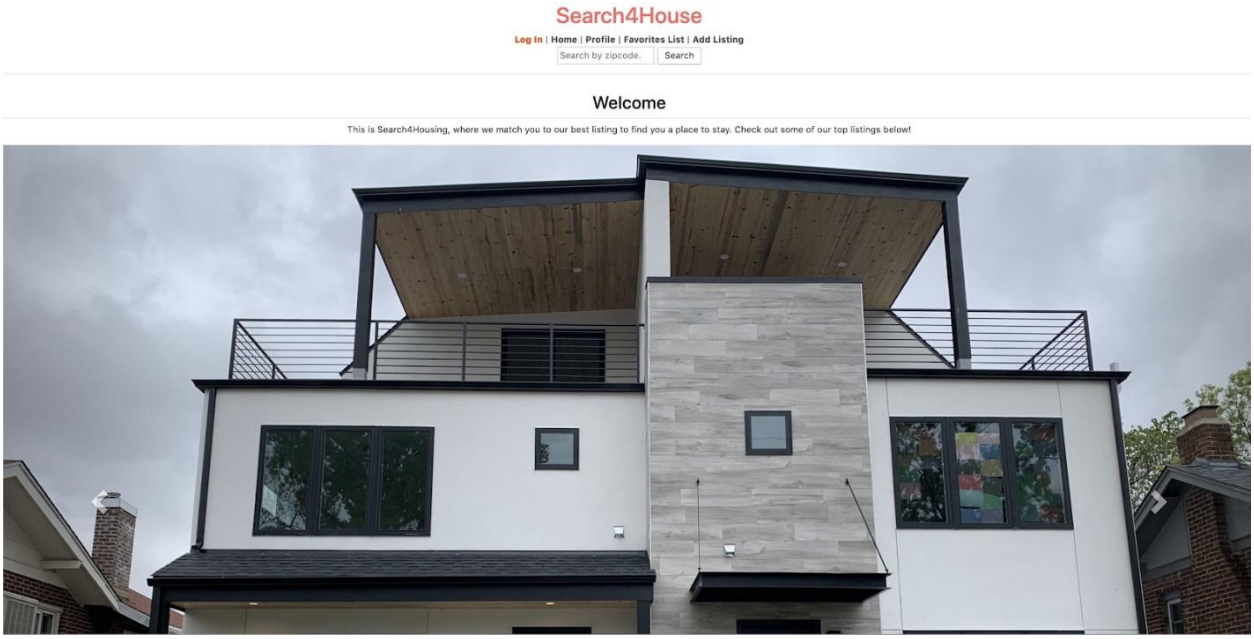
Similarly, the registration page asks for the user to fill out information to create an account. That information is used in the JSP to send a SQL query to the database and create a new account in the account table.

The edit profile page displays similar boxes to the registration page to allow the user to change some of the values in the table. Once the values have been submitted, the JSPs will send a SQL query to update the following changes.

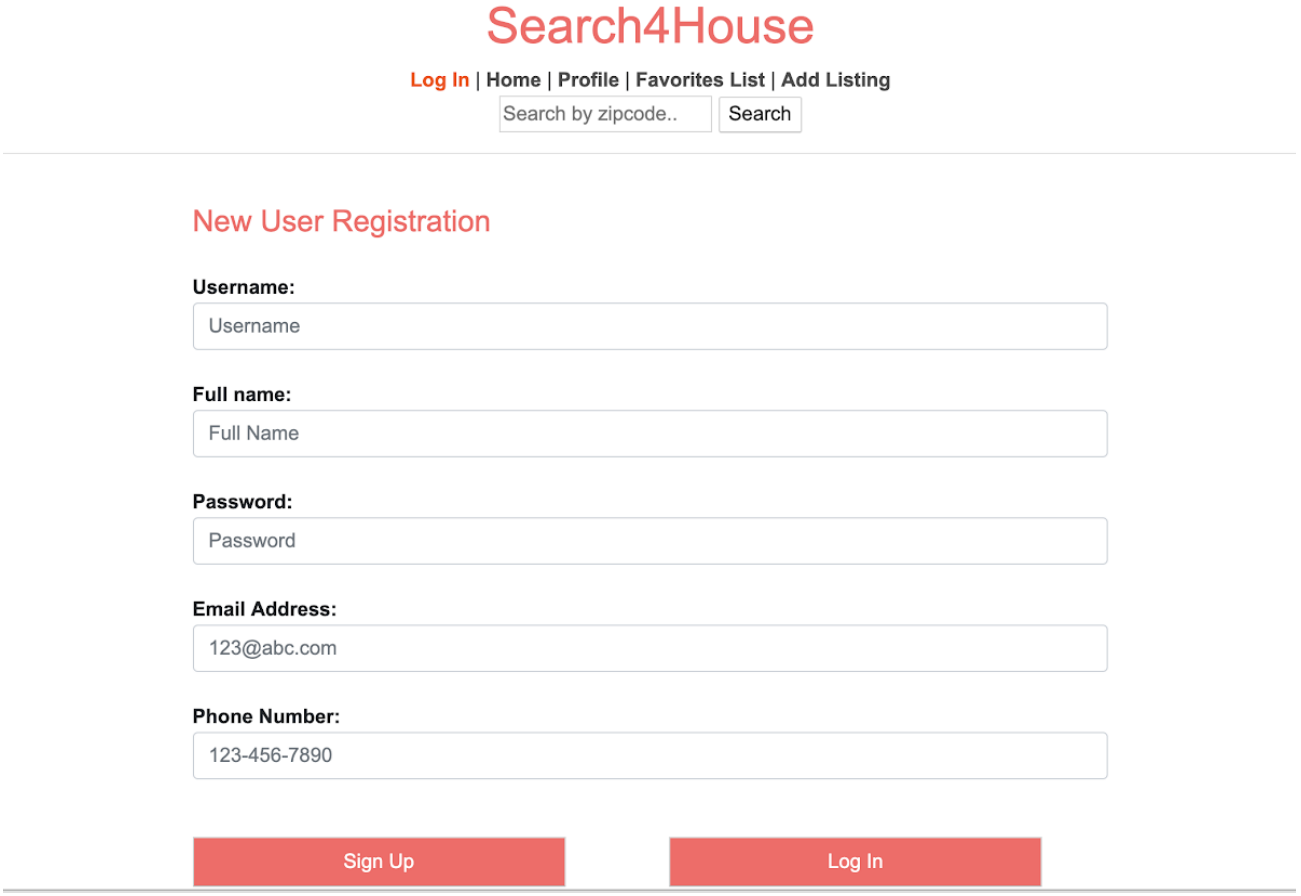
All of the other pages do similar things with the information provided by the user and the database to generate pages on the browser

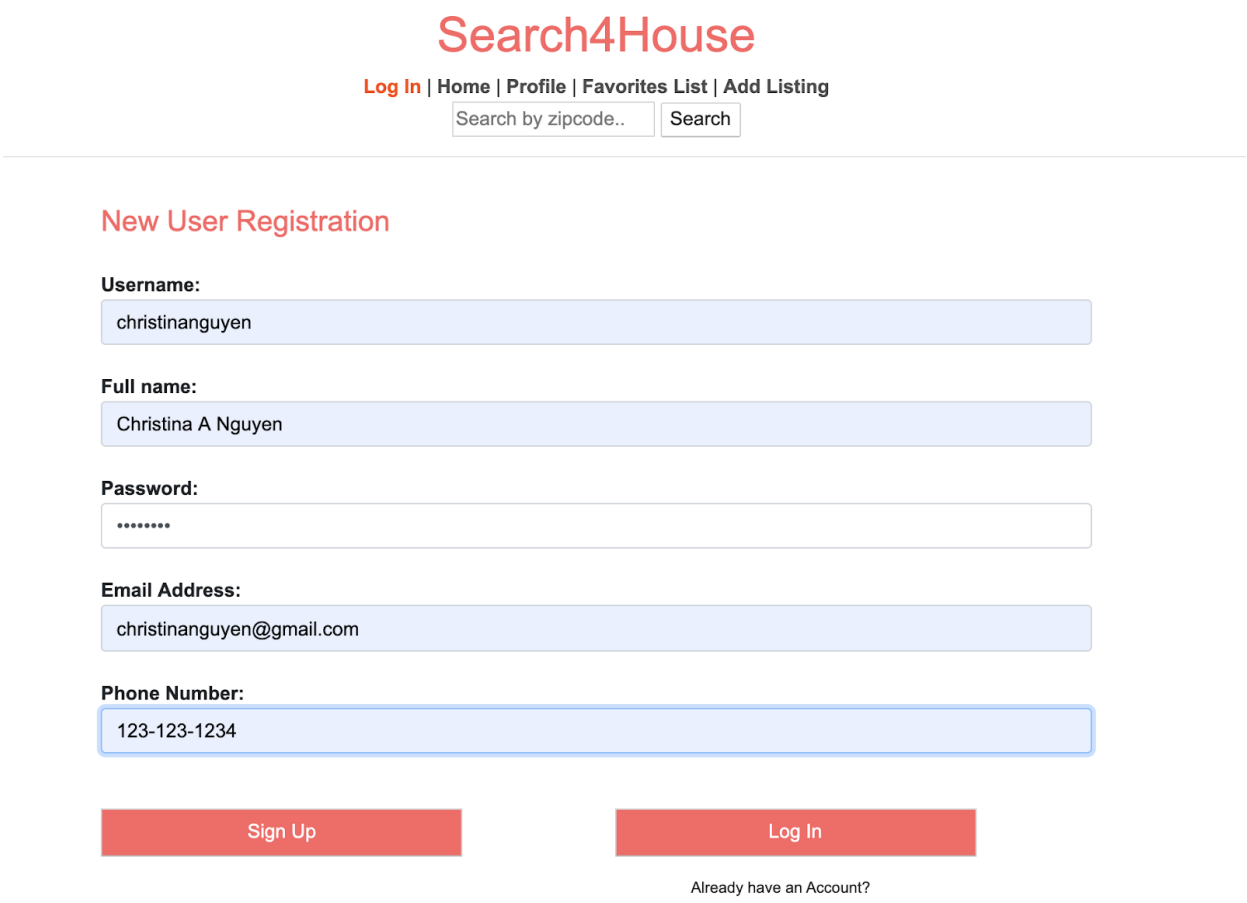
## 6.2 Screenshots

### 6.2.1: Home Page:




### 6.2.2: Register:






This Register page uses the following SQL queries:

stmt2.executeUpdate("INSERT INTO search4houses.Accounts (accID, username, password, fullName, email, phoneNum) VALUES('"+newUserID+"','"+username+"','"+password+"','"+name+"','"+email+"','"+phoneNum+"');");

to insert the values into the accounts table.

stmt5.executeUpdate("INSERT INTO search4houses.Favorites (listID) VALUES ("+newUserID+");");

stmt6.executeUpdate("INSERT INTO search4houses.User_Favorites (user_id, list_id) VALUES ("+newUserID+", "+newUserID+");");

To update the favorites and the relationship between favorites and accounts with the new user so that the user has a favorites list.

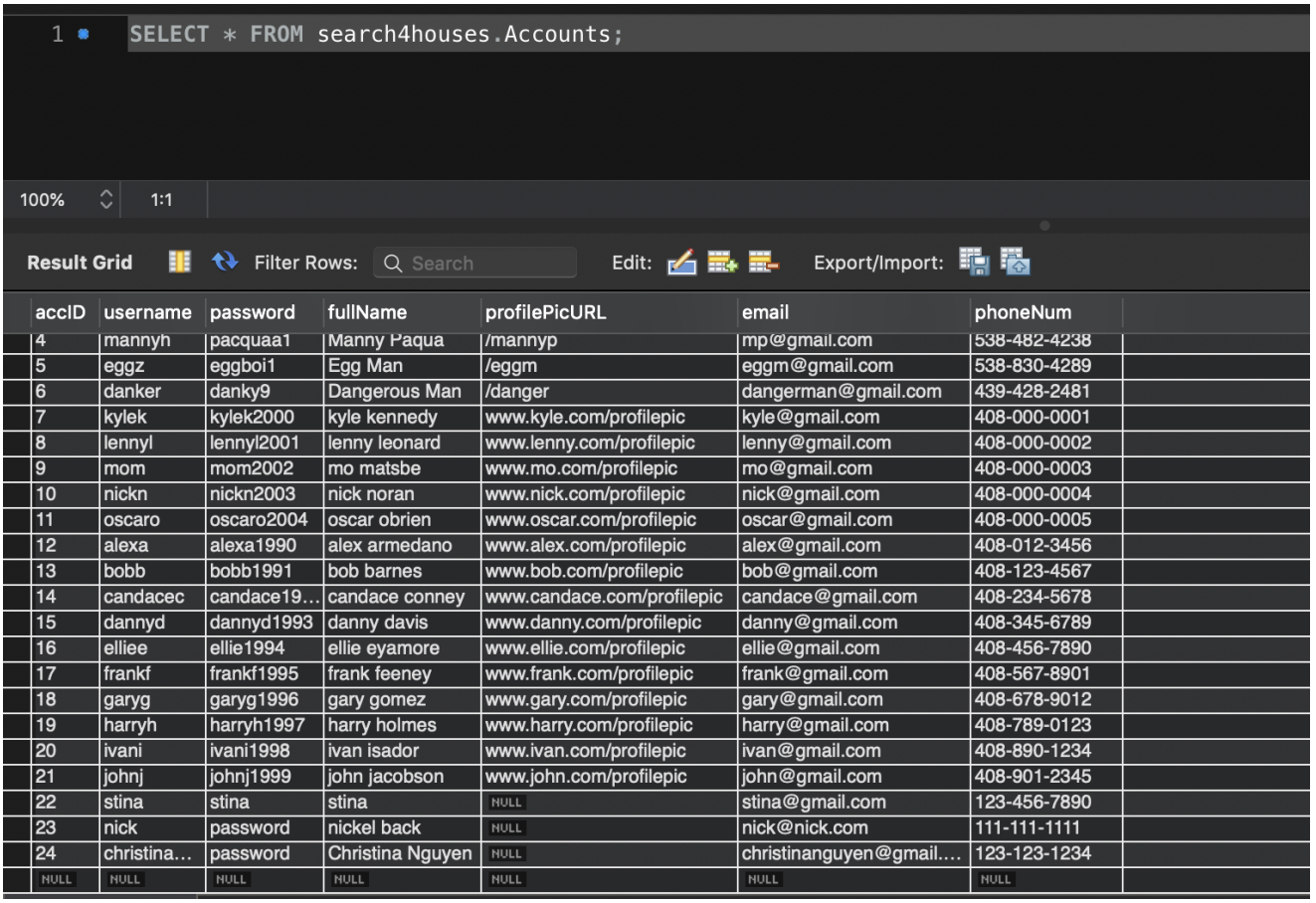
# Search4House

**Home | Profile | Favorites List | Add Listing | Log Out**

Search by zipcode.. Search

## Registration Success!

You will be redirected in 3 seconds...



```
1  SELECT * FROM search4houses.Accounts;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| accID | username | password | fullName | profilePicURL | email | phoneNum |
|---|---|---|---|---|---|---|
| 4 | mannyh | pacquaa1 | Manny Paqua | /mannyp | mp@gmail.com | 538-482-4238 |
| 5 | eggz | eggboi1 | Egg Man | /eggm | eggm@gmail.com | 538-830-4289 |
| 6 | danker | danky9 | Dangerous Man | /danger | dangerman@gmail.com | 439-428-2481 |
| 7 | kylek | kylek2000 | kyle kennedy | www.kyle.com/profilepic | kyle@gmail.com | 408-000-0001 |
| 8 | lennyl | lennyl2001 | lenny leonard | www.lenny.com/profilepic | lenny@gmail.com | 408-000-0002 |
| 9 | mom | mom2002 | mo matsbe | www.mo.com/profilepic | mo@gmail.com | 408-000-0003 |
| 10 | nickn | nickn2003 | nick noran | www.nick.com/profilepic | nick@gmail.com | 408-000-0004 |
| 11 | oscaro | oscaro2004 | oscar obrien | www.oscar.com/profilepic | oscar@gmail.com | 408-000-0005 |
| 12 | alexa | alexa1990 | alex armedano | www.alex.com/profilepic | alex@gmail.com | 408-012-3456 |
| 13 | bobb | bobb1991 | bob barnes | www.bob.com/profilepic | bob@gmail.com | 408-123-4567 |
| 14 | candacec | candace19... | candace conney | www.candace.com/profilepic | candace@gmail.com | 408-234-5678 |
| 15 | dannyd | dannyd1993 | danny davis | www.danny.com/profilepic | danny@gmail.com | 408-345-6789 |
| 16 | elliee | ellie1994 | ellie eyamore | www.ellie.com/profilepic | ellie@gmail.com | 408-456-7890 |
| 17 | frankf | frankf1995 | frank feeney | www.frank.com/profilepic | frank@gmail.com | 408-567-8901 |
| 18 | garyg | garyg1996 | gary gomez | www.gary.com/profilepic | gary@gmail.com | 408-678-9012 |
| 19 | harryh | harryh1997 | harry holmes | www.harry.com/profilepic | harry@gmail.com | 408-789-0123 |
| 20 | ivani | ivani1998 | ivan isador | www.ivan.com/profilepic | ivan@gmail.com | 408-890-1234 |
| 21 | johnj | johnj1999 | john jacobson | www.john.com/profilepic | john@gmail.com | 408-901-2345 |
| 22 | stina | stina | stina | NULL | stina@gmail.com | 123-456-7890 |
| 23 | nick | password | nickel back | NULL | nick@nick.com | 111-111-1111 |
| 24 | christina... | password | Christina Nguyen | NULL | christinanguyen@gmail.... | 123-123-1234 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

```
1 • SELECT * FROM search4houses.Favorites;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| listID | listing1ID | listing2ID | listing3ID | listing4ID | listing5ID |
|---|---|---|---|---|---|
| 1 | 14 | 33 | NULL | NULL | NULL |
| 2 | 2 | 9 | 9 | 9 | 14 |
| 3 | 2 | 9 | 9 | 9 | 14 |
| 4 | 2 | 9 | 9 | 9 | 14 |
| 5 | 2 | 9 | 9 | 9 | 14 |
| 6 | 2 | 9 | 9 | 9 | 14 |
| 7 | 2 | 9 | 9 | 9 | 14 |
| 8 | 2 | 9 | 9 | 9 | 14 |
| 9 | 2 | 9 | 9 | 9 | 14 |
| 10 | 2 | 9 | 9 | 9 | 14 |
| 11 | 2 | 9 | 9 | 9 | 14 |
| 12 | 2 | 9 | 9 | 9 | 14 |
| 13 | 2 | 9 | 9 | 9 | 14 |
| 14 | 2 | 9 | 9 | 9 | 14 |
| 15 | 2 | 9 | 9 | 9 | 14 |
| 16 | 2 | 9 | 9 | 9 | 14 |
| 17 | 2 | 9 | 9 | 9 | 14 |
| 21 | 2 | 9 | 9 | 9 | 14 |
| 22 | NULL | NULL | NULL | NULL | NULL |
| 23 | NULL | NULL | NULL | NULL | NULL |
| 24 | NULL | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL |

Favorites 1

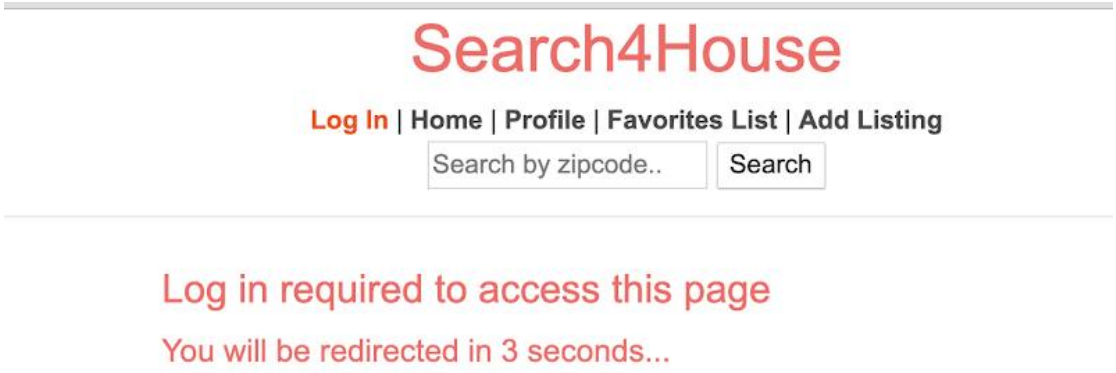
```
1 • SELECT * FROM search4houses.User_Favorites;
```

100% 1:1

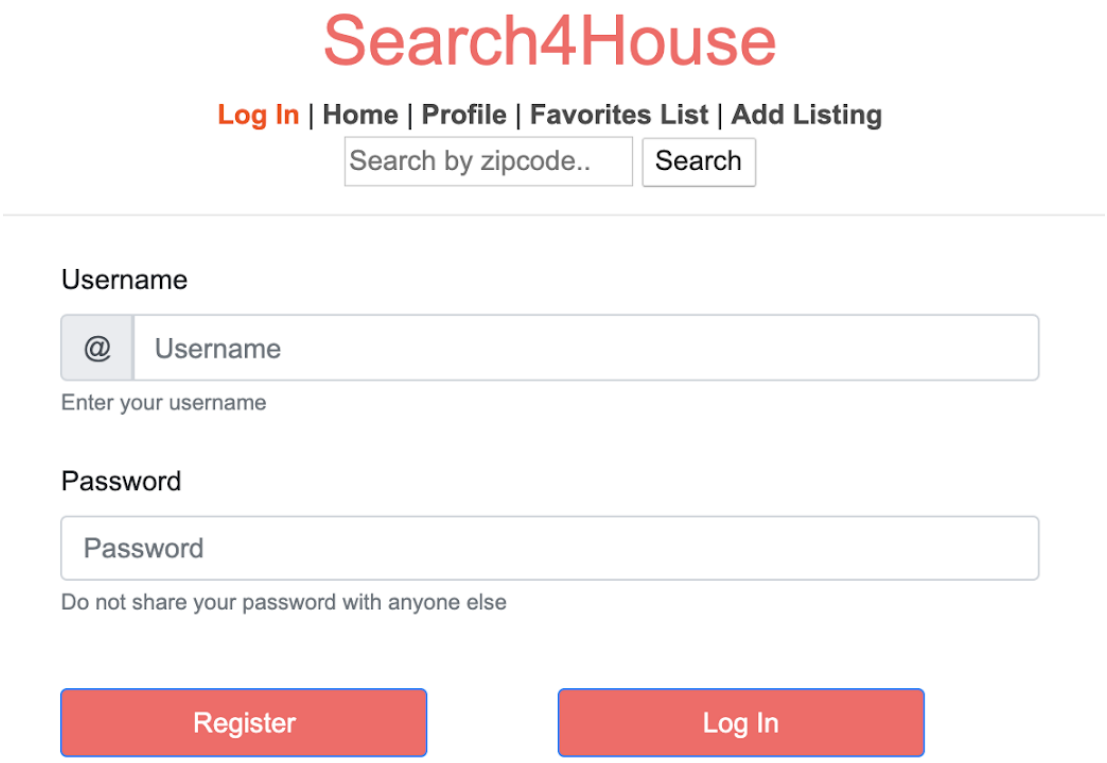
Result Grid Filter Rows: Search Edit: Export/Im

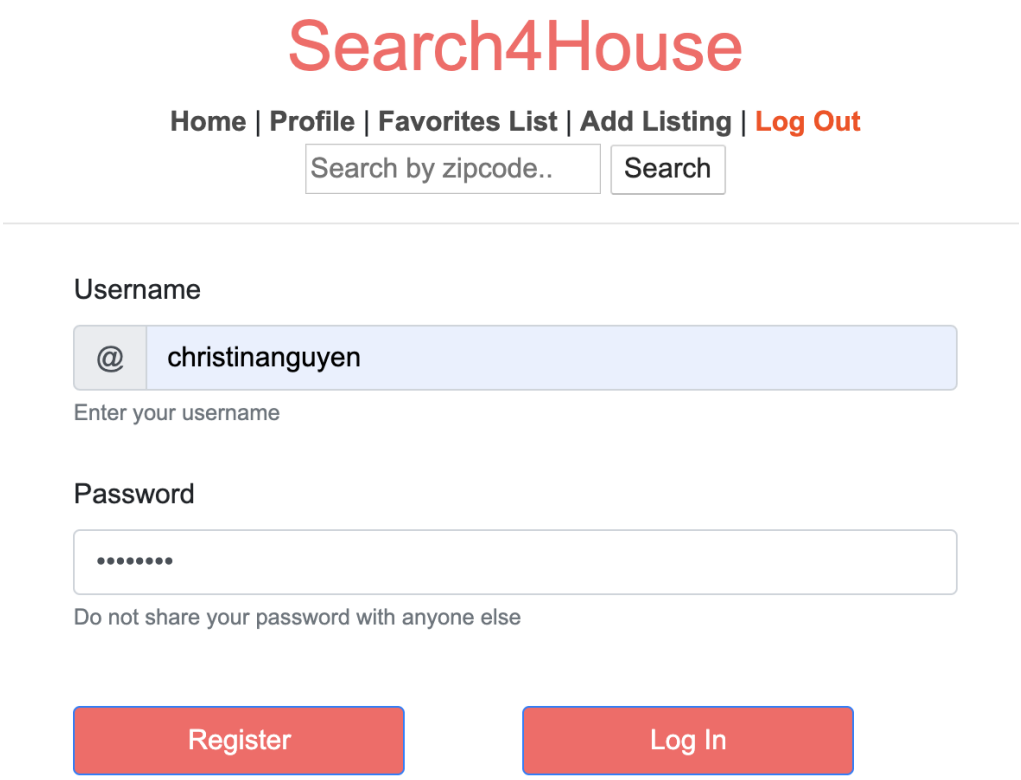
| user_id | list_id |
|---|---|
| 1 | 1 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | 10 |
| 11 | 11 |
| 12 | 12 |
| 13 | 13 |
| 14 | 14 |
| 15 | 15 |
| 16 | 16 |
| 17 | 17 |
| 18 | 18 |
| 19 | 19 |
| 20 | 20 |
| 21 | 21 |
| 22 | 22 |
| 23 | 23 |
| 24 | 24 |
| NULL | NULL |

### 6.2.1.1: Accessing Restricted Page (not logged in)




### 6.2.2: Log In:

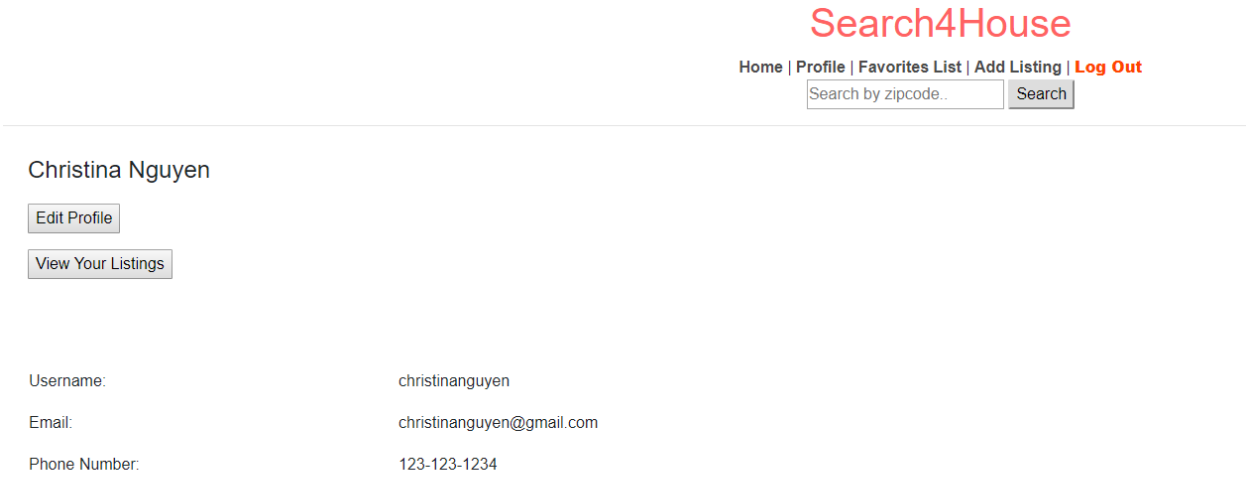





This page uses the following SQL query:
ResultSet rs = stmt.executeQuery("SELECT * FROM search4houses.Accounts WHERE username='"+aUsername+"'
AND password='"+aPassword+"';");

It checks from accounts table a username and password and logs the user in.

### 6.2.3: Profile Page:

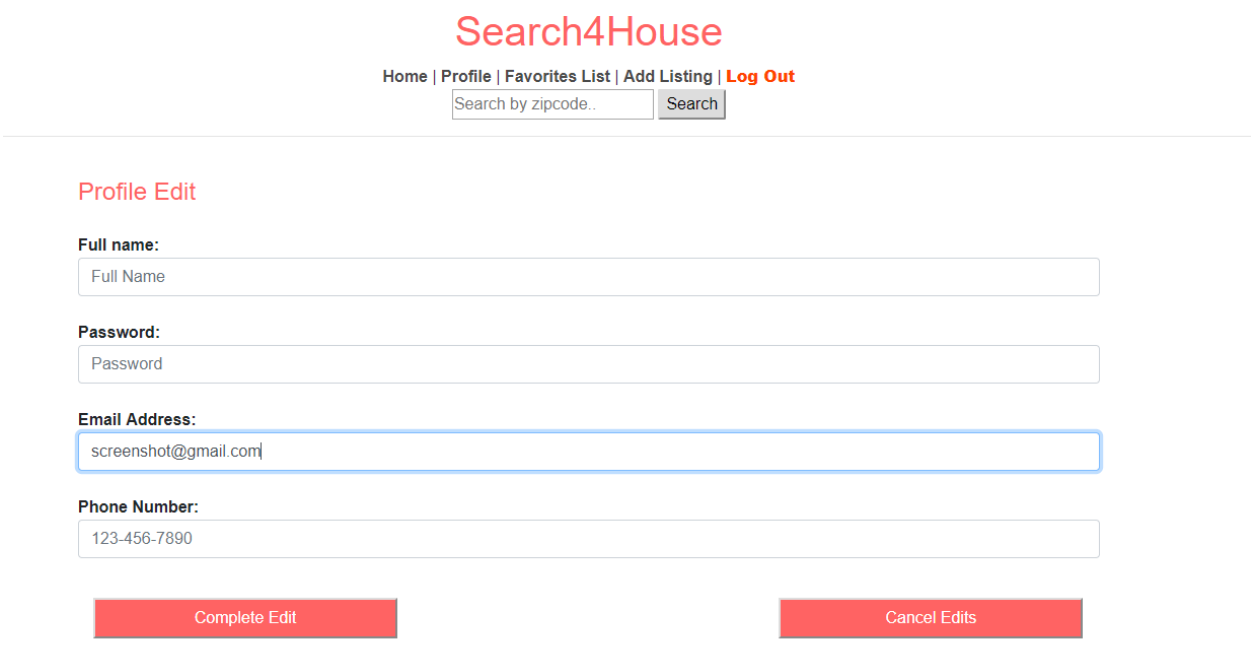


This uses the following SQL query:

ResultSet rs = stmt.executeQuery("SELECT * FROM search4houses.accounts WHERE search4houses.accounts.accID = " + userID);

And gets everything in the row with that specific userID that we get from logging in and prints out the values onto the profile page.

### 6.2.4: Edit Profile Page:




This use the same SQL query from the registration process, but cuts it up into different pieces depending on which text box the user has filled and then updates profile.

stmt2.executeUpdate("UPDATE search4houses.accounts
SET search4houses.accounts.password = '" + password + "'
WHERE search4houses.accounts.accID = " + sessionUserID);

Where password is replaced with email, name or phoneNum depending on the filled in text box.

| | | | | | | |
|---|---|---|---|---|---|---|
| 7 | kylek | kylek2000 | kyle kennedy | www.kyle.com/profilepic | kyle@gmail.com | 408-000-0001 |
| 8 | lennyl | lennyl2001 | lenny leonard | www.lenny.com/profilepic | lenny@gmail.com | 408-000-0002 |
| 9 | mom | mom2002 | mo matsbe | www.mo.com/profilepic | mo@gmail.com | 408-000-0003 |
| 10 | nickn | nickn2003 | nick noran | www.nick.com/profilepic | nick@gmail.com | 408-000-0004 |
| 11 | oscaro | oscaro2004 | oscar obrien | www.oscar.com/profilepic | oscar@gmail.com | 408-000-0005 |
| 12 | alexa | alexa1990 | alex armedano | www.alex.com/profilepic | alex@gmail.com | 408-012-3456 |
| 13 | bobb | bobb1991 | bob barnes | www.bob.com/profilepic | bob@gmail.com | 408-123-4567 |
| 14 | candacec | candace1... | candace conney | www.candace.com/profil... | candace@gmail.com | 408-234-5678 |
| 15 | dannyd | dannyd1993 | danny davis | www.danny.com/profilepic | danny@gmail.com | 408-345-6789 |
| 16 | elliee | ellie1994 | ellie eyamore | www.ellie.com/profilepic | ellie@gmail.com | 408-456-7890 |
| 17 | frankf | frankf1995 | frank feeney | www.frank.com/profilepic | frank@gmail.com | 408-567-8901 |
| 18 | garyg | garyg1996 | gary gomez | www.gary.com/profilepic | gary@gmail.com | 408-678-9012 |
| 19 | harryh | harryh1997 | harry holmes | www.harry.com/profilepic | harry@gmail.com | 408-789-0123 |
| 20 | ivani | ivani1998 | ivan isador | www.ivan.com/profilepic | ivan@gmail.com | 408-890-1234 |
| 21 | johnj | johnj1999 | john jacobson | www.john.com/profilepic | john@gmail.com | 408-901-2345 |
| 22 | stina | stina | stina | NULL | stina@gmail.com | 123-456-7890 |
| 23 | nick | password | nickel back | NULL | nick@nick.com | 111-111-1111 |
| 24 | christina... | password | Christina Nguyen | NULL | screenshot@gmail.com | 123-123-1234 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

ıccounts 2 ×

Search4House

Home | Profile | Favorites List | Add Listing | Log Out

Search by zipcode.. Search

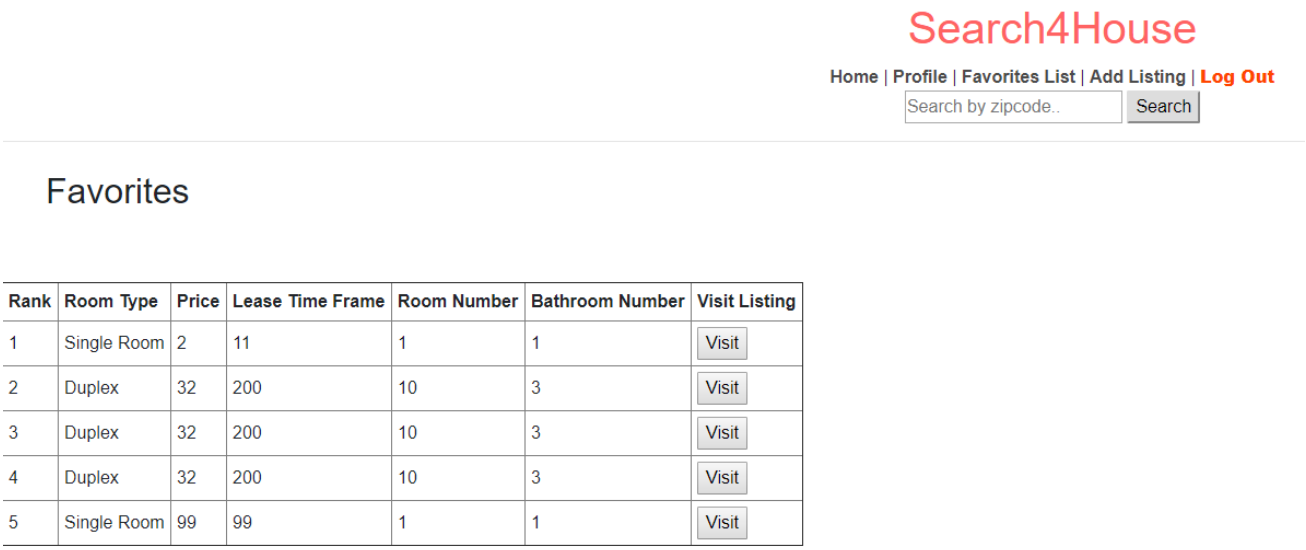
## Christina Nguyen

Edit Profile

View Your Listings

| | |
|---|---|
| Username: | christinanguyen |
| Email: | screenshot@gmail.com |
| Phone Number: | 123-123-1234 |

### 6.2.5: View Favorites Page:

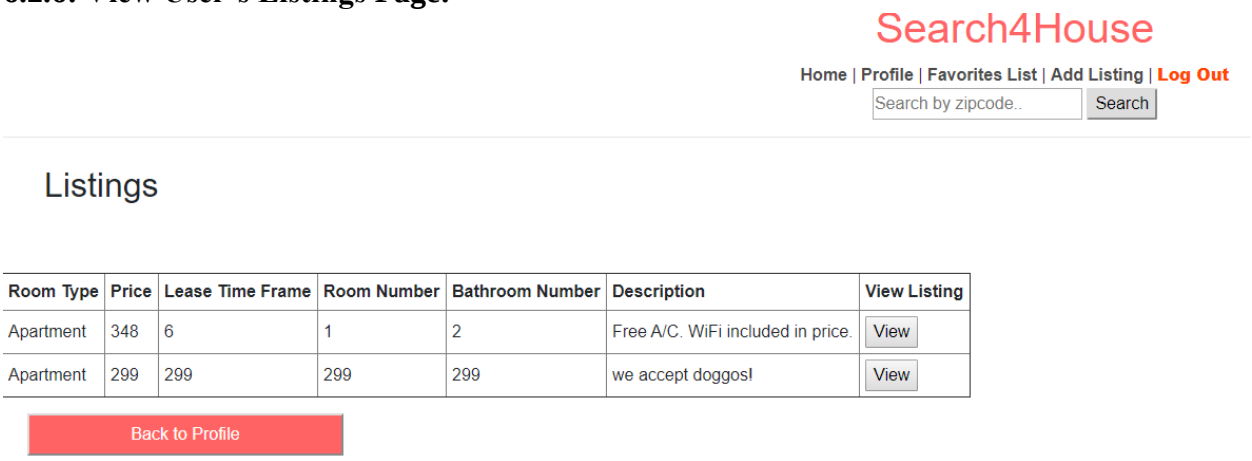



This uses the following SQL query:

```
("Select * FROM listings, favorites WHERE
        (SELECT listing" + rankCounter + "ID
        FROM search4houses.favorites, search4houses.user_favorites
        WHERE search4houses.user_favorites.user_id = "+ userID + " AND
        search4houses.favorites.listID = search4houses.user_favorites.list_id)
= listings.listingID");
```

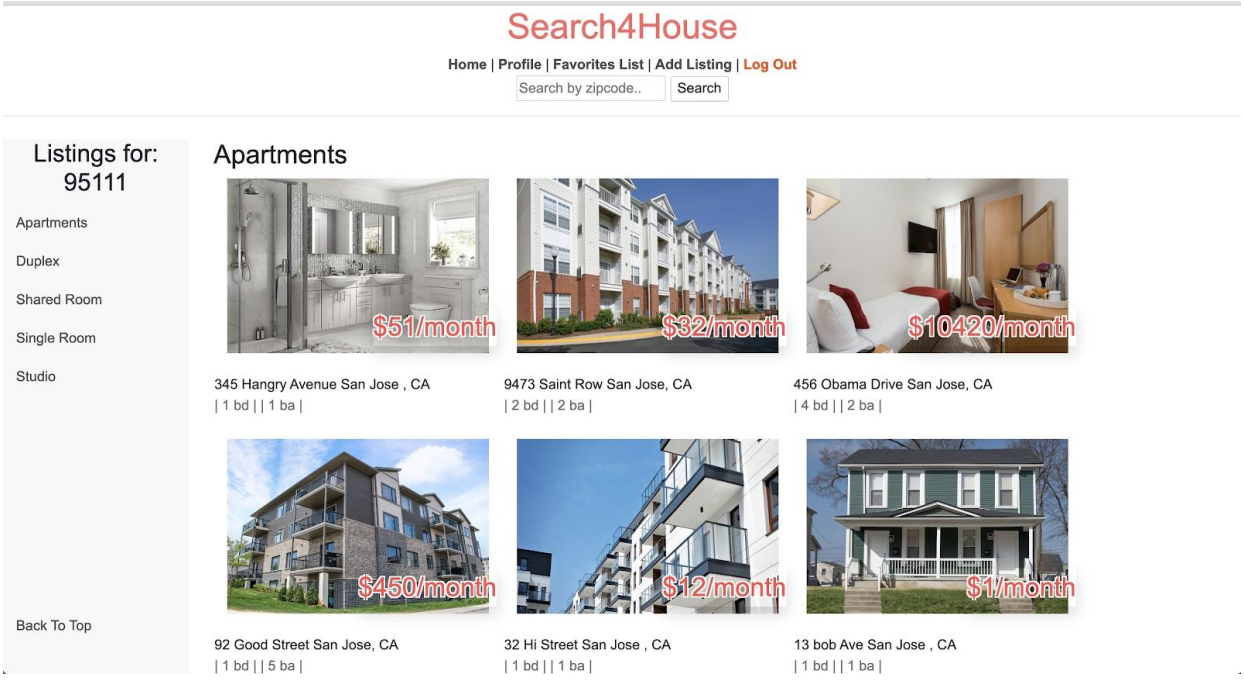
This gets the favorite listing ID from each favorites list, and uses that to find the listing id from each listing and returns the values of each listing into the HTML table.
It uses the listingId as well to hyperlink to that listing page.

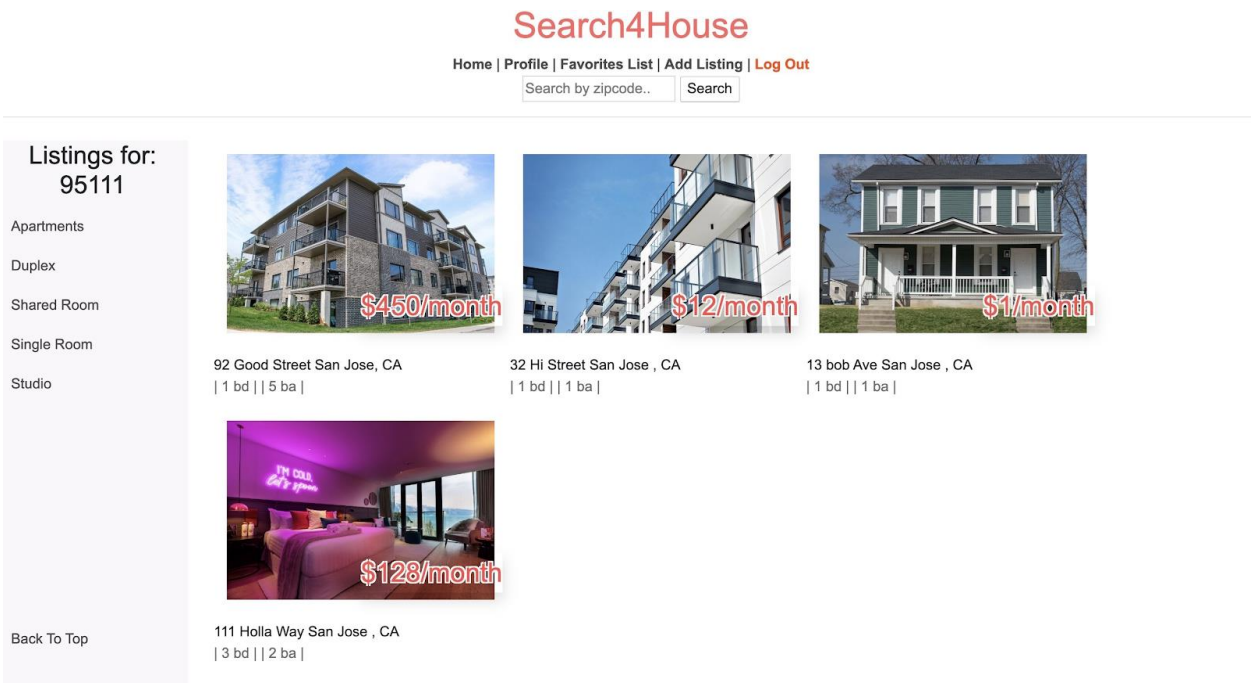
### 6.2.6: View User's Listings Page:




This page uses the same exact stuff from Favorites list, however the listings are the listings the user owns. The hyperlink redirects the user to the listing page, and on that listing page allows for the user to edit that listing.

### 6.2.8: Listings Page:






This page uses many of the same queries to get all the information and display it onto the web page.

ResultSet rs = stmt.executeQuery("SELECT Listings_Photos.listing_id,photo_id, addr_id FROM search4houses.Listings_Photos, search4houses.Listings,search4houses.Photos,search4houses.Addresses,search4houses.Listing_Address

WHERE roomType

LIKE 'Apartment%'

AND Listings_Photos.listing_id=Listings.listingID

AND Listing_Address.listing_id=Listings.listingID

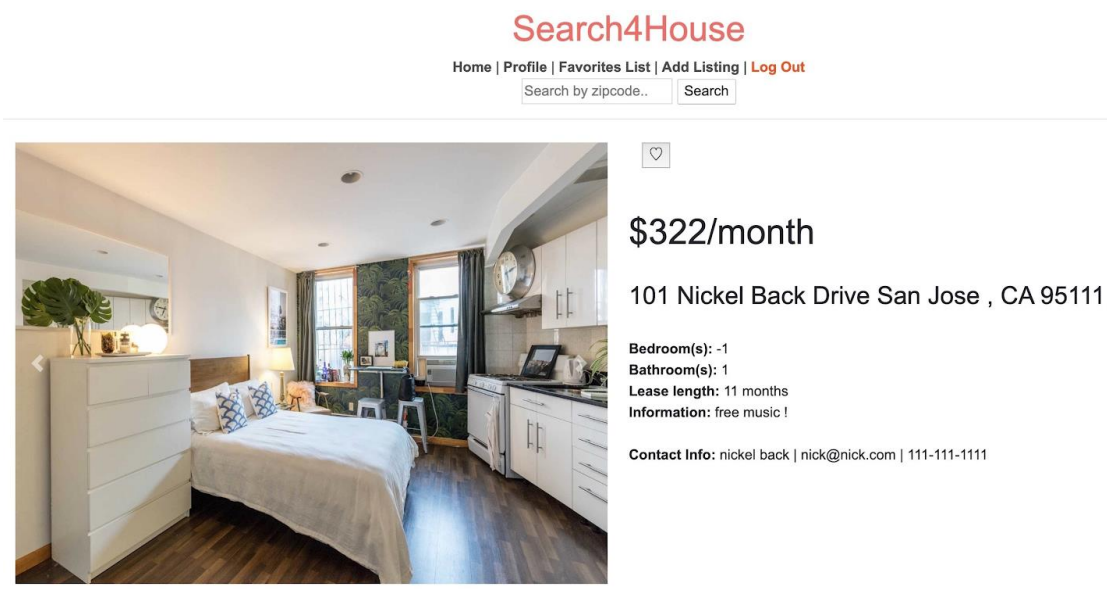
AND photo_id=photoID AND addr_id=addrID

AND zipCode LIKE "'+theZip+'"

ORDER BY Listing_Address.listing_id ASC, Listings_Photos.photo_id DESC;");

This is used multiple times to get the different rows in the web page.

### 6.2.9: View Listing Page:

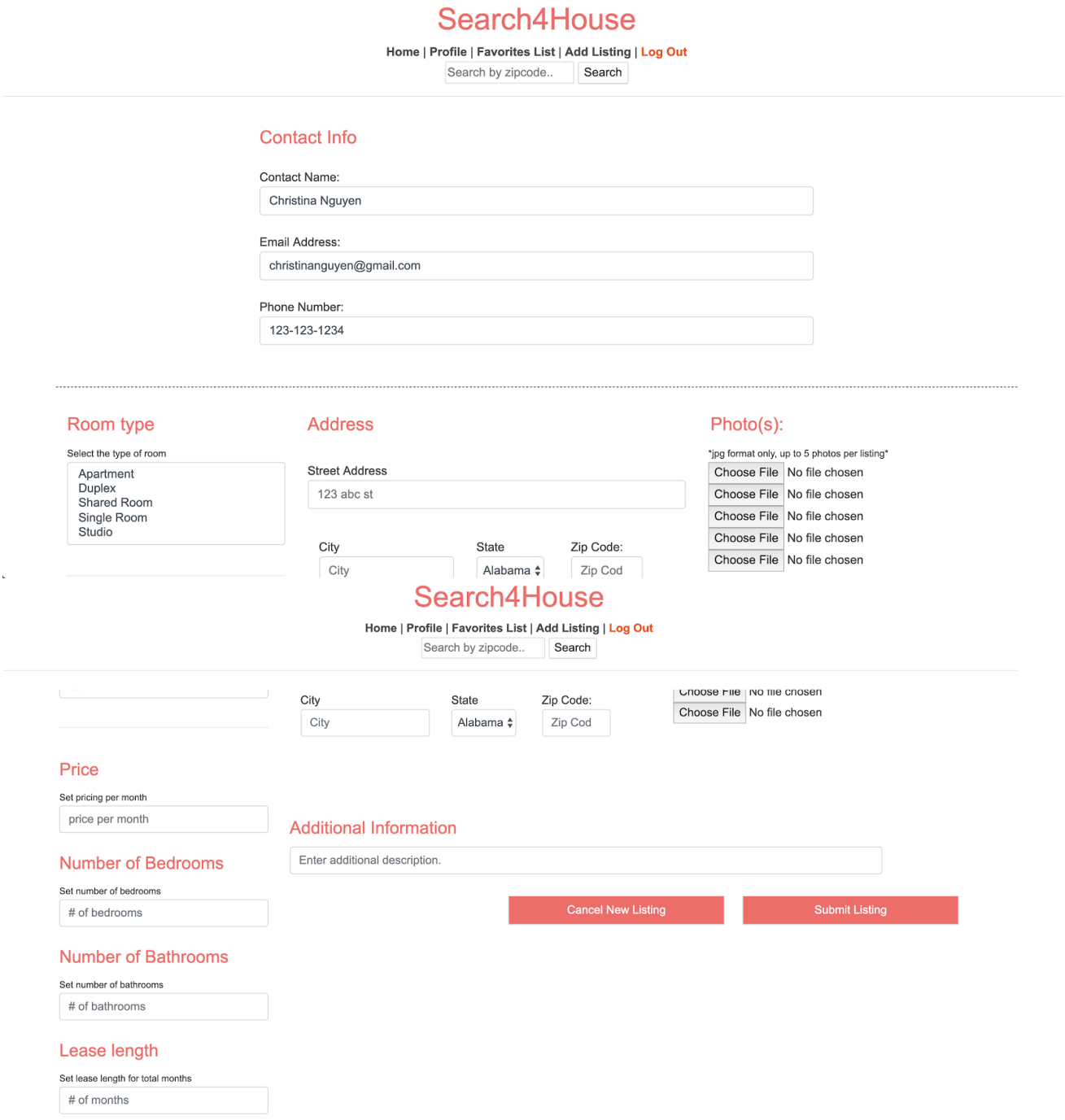


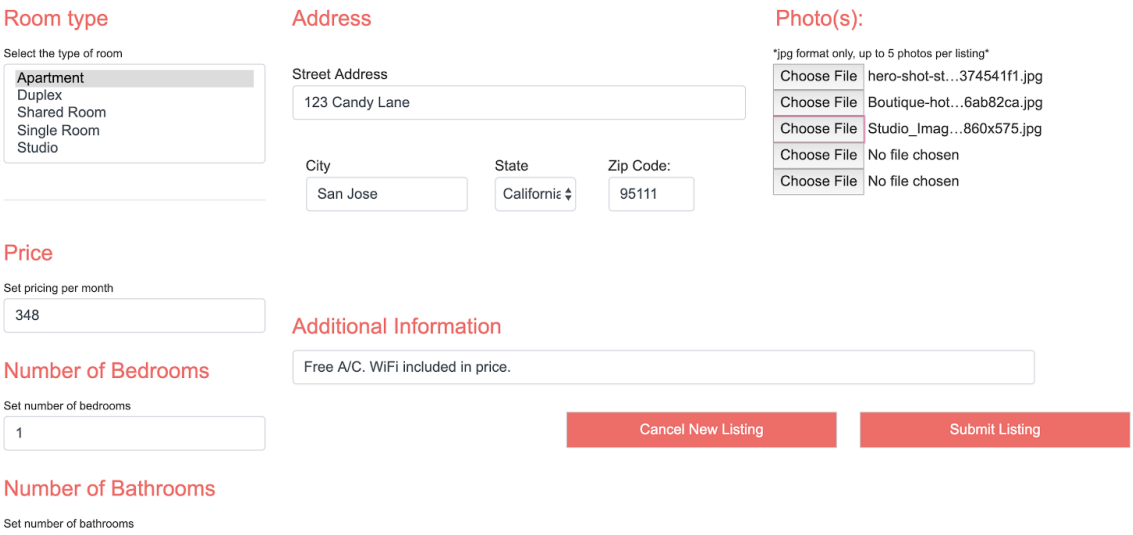

This page uses two different queries to get the listing information and the additional contact information:

```
ResultSet rs2 = stmt3.executeQuery("SELECT * FROM
search4houses.Addresses, search4houses.Listing_Address
WHERE addrID=addr_id
AND listing_id="+listing_id);

And

ResultSet rs3 = stmt4.executeQuery("SELECT * FROM
search4houses.AdditionalContacts, search4houses.Listings_AdditionalContact
WHERE contactAccID=addContact_id
AND listing_id="+listing_id);
```

The information is that was gathered is displayed on the page.

## 6.2.10: Add Listing Page:



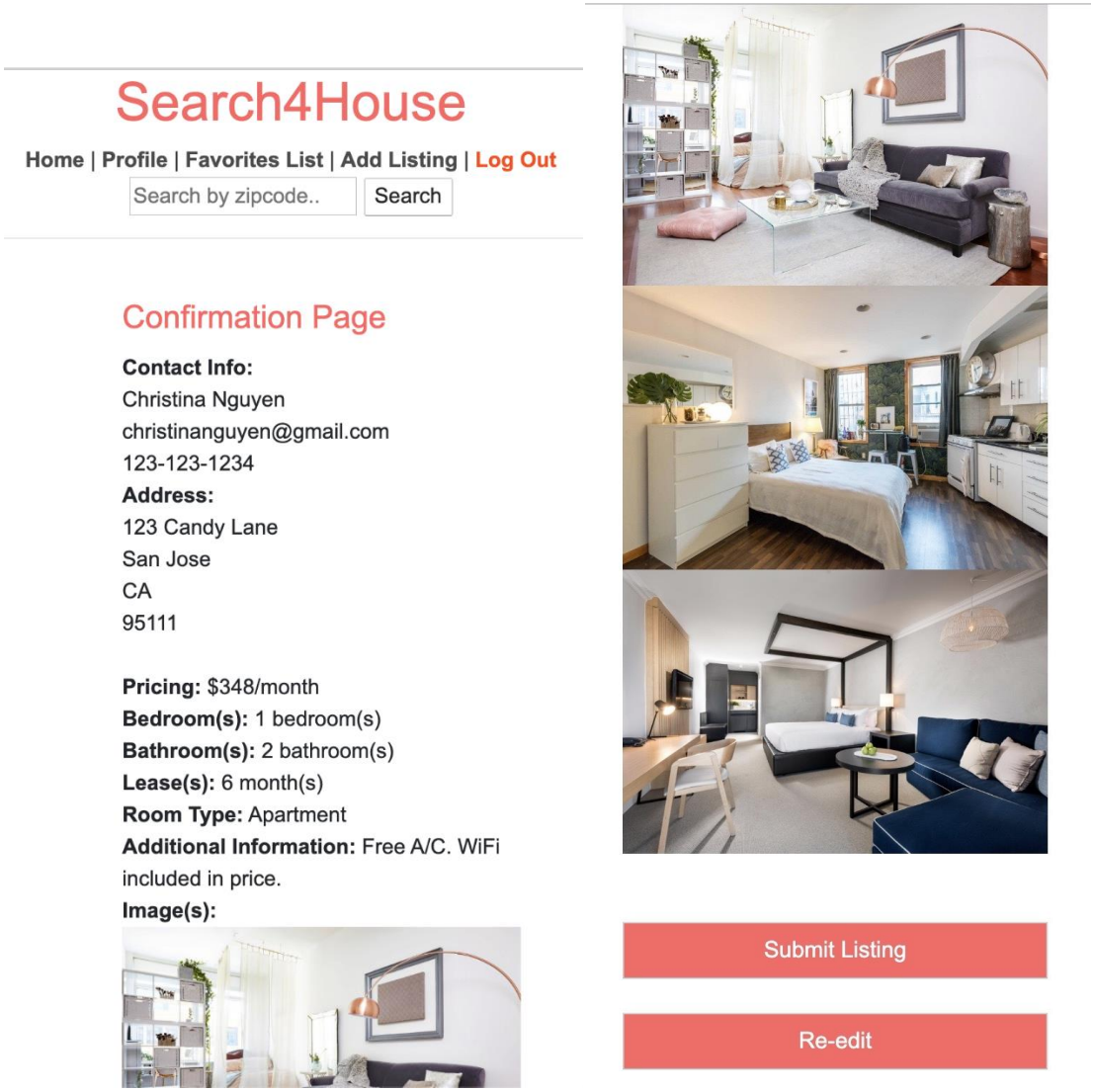



The add listing page gets the information from the user and puts that into the the listing table.

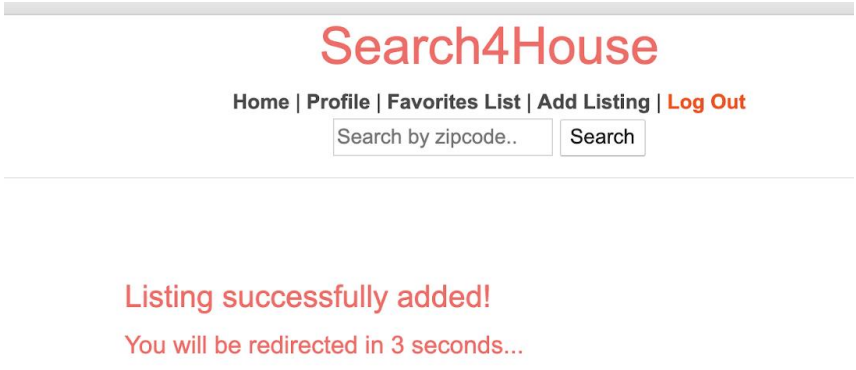
stmt.executeUpdate("INSERT INTO search4houses.Listings (listingID, roomType, price, leaseTimeFrame, roomNum, bathroomNum, description)

VALUES ("+newGeneratedID+", '"+roomType+"', "+price+", "+lease+", "+bedrooms+", "+bathrooms+", '"+info+"');");

### 6.2.10.1: Confirmation Page




### 6.2.10.2: Adding Listing Success Page:



# Search4House

Home | Profile | Favorites List | Add Listing | Log Out

Search by zipcode.. Search

♡

## $348/month

### 123 Candy Lane San Jose , CA 95111

**Bedroom(s):** 1
**Bathroom(s):** 2
**Lease length:** 6 months
**Information:** Free A/C. WiFi included in price.

**Contact Info:** Christina Nguyen | christinanguyen@gmail.com | 123-123-1234

```
1  SELECT * FROM search4houses.Accounts_Listings;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| user_id | listing_id |
|---|---|
| 1 | 37 |
| 1 | 38 |
| 23 | 39 |
| 23 | 40 |
| 23 | 41 |
| 23 | 42 |
| 23 | 43 |
| 24 | 44 |
| NULL | NULL |

Accounts_Listings 1

```
1  SELECT * FROM search4houses.Addresses;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| addrID | streetAddress | city | state | zipCode |
|---|---|---|---|---|
| 37 | 13 bob Ave | San Jose | CA | 95111 |
| 38 | 111 Holla Way | San Jose | CA | 95111 |
| 39 | 101 Nickel Back Drive | San Jose | CA | 95111 |
| 40 | 133 Beens Ave | San Jose | CA | 95111 |
| 41 | 65 Good Babbies Street | San Jose | CA | 95111 |
| 42 | 90 Rad street | San Jose | CA | 95111 |
| 43 | 33 Cute Drive | San Jose | CA | 95111 |
| 44 | 123 Candy Lane | San Jose | CA | 95111 |
| NULL | NULL | NULL | NULL | NULL |

Addresses 1

```
1  SELECT * FROM search4houses.Listing_Address;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| listing_id | addr_id |
|---|---|
| 36 | 36 |
| 37 | 37 |
| 38 | 38 |
| 39 | 39 |
| 41 | 41 |
| 42 | 42 |
| 43 | 43 |
| 44 | 44 |
| NULL | NULL |

Listing_Address 1

```
1  SELECT * FROM search4houses.AdditionalContacts;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| contactAccID | phoneNum | email | name |
|---|---|---|---|
| 30 | 408-000-0020 | matt@gmail.com | Matt |
| 31 | 785-424-2121 | mrbob@gmail.com | Mr Bob |
| 33 | 123-456-7890 | stina@gmail.com | stina |
| 34 | 104-248-2482 | henry@gmail.com | Henrie |
| 35 | 104-248-2482 | henry@gmail.com | Henry |
| 36 | 408-537-2892 | hi@gmail.com | Bob |
| 37 | 111-111-1111 | nick@nick.com | nickel… |
| 38 | 222-222-2222 | Chill@gmail.com | Coke C… |
| 39 | 333-333-3333 | jennie@yahoo.com | Jennie |
| 40 | 123-123-1234 | christinanguyen@g… | Christi… |
| NULL | NULL | NULL | NULL |

AdditionalContacts 1

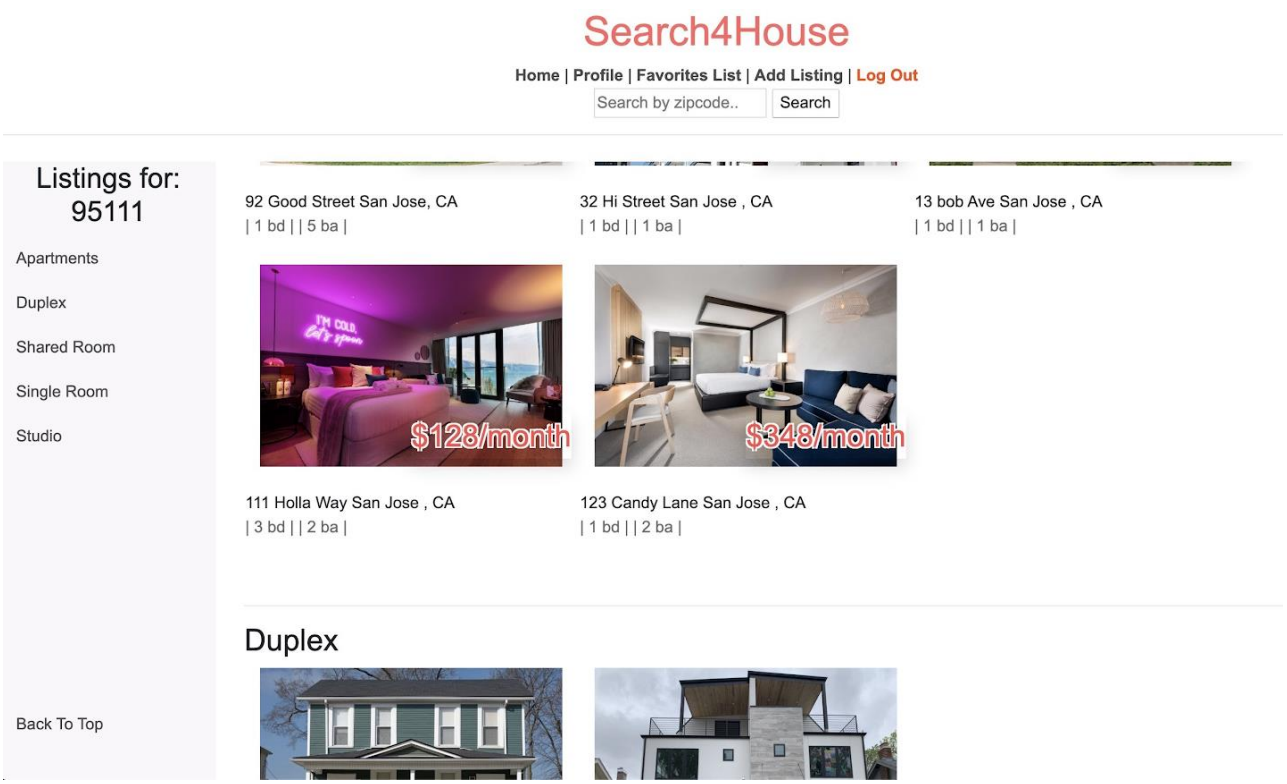
```
1  SELECT * FROM search4houses.Listings_AdditionalContact;
```

100% 1:1

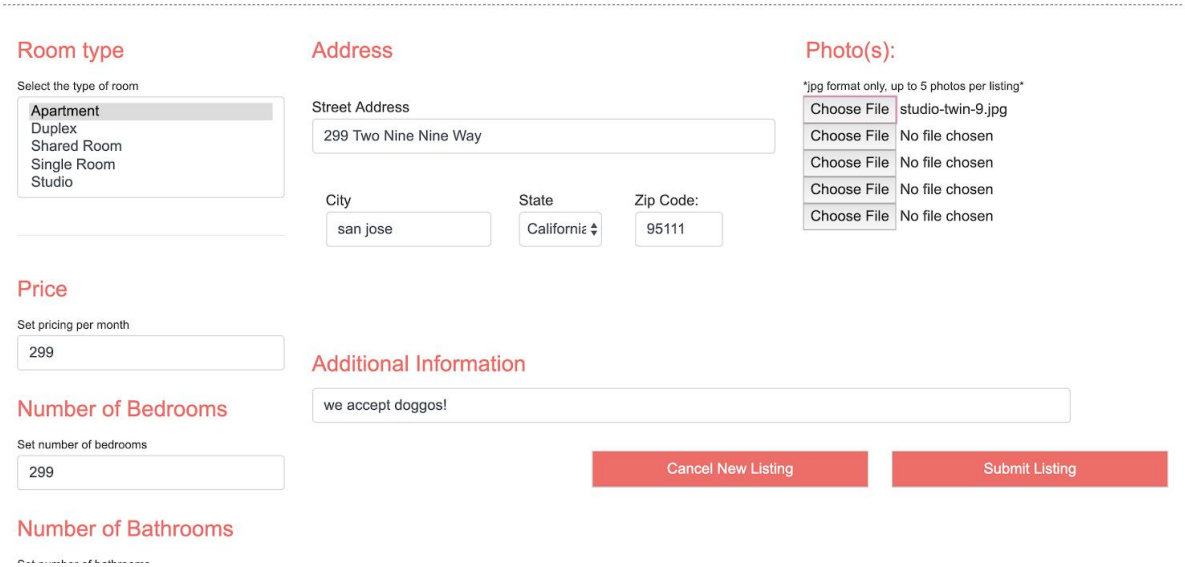
Result Grid Filter Rows: Search Edit: Export/Import:

| listing_id | addContact_id |
|---|---|
| 33 | 36 |
| 34 | 37 |
| 36 | 36 |
| 37 | 36 |
| 38 | 36 |
| 39 | 38 |
| 41 | 38 |
| 42 | 38 |
| 43 | 38 |
| 44 | 40 |
| NULL | NULL |

Listings_AdditionalContact 1



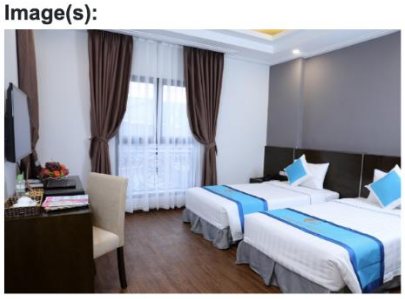

### 6.2.10.3: Adding Another Listing:



## Confirmation Page

**Contact Info:**
Christina Nguyen
christinanguyen@gmail.com
123-123-1234
**Address:**
299 Two Nine Nine Way
san jose
CA
95111

**Pricing:** $299/month
**Bedroom(s):** 299 bedroom(s)
**Bathroom(s):** 299 bathroom(s)
**Lease(s):** 299 month(s)
**Room Type:** Apartment
**Additional Information:** we accept doggos!
**Image(s):**

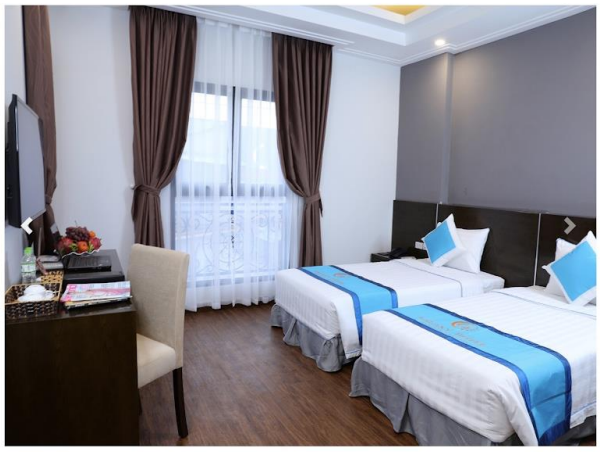


Submit Listing

## Search4House

Home | Profile | Favorites List | Add Listing | Log Out

Search by zipcode.. Search



♡

## $299/month

299 Two Nine Nine Way san jose , CA 95111

**Bedroom(s):** 299
**Bathroom(s):** 299
**Lease length:** 299 months
**Information:** we accept doggos!

**Contact Info:** Christina Nguyen | christinanguyen@gmail.com | 123-123-1234

# Search4House

Home | Profile | Favorites List | Add Listing | Log Out

Search by zipcode.. Search

Listings for: 95111

- Apartments
- Duplex
- Shared Room
- Single Room
- Studio


92 Good Street San Jose, CA
| 1 bd | | 5 ba |


32 Hi Street San Jose , CA
| 1 bd | | 1 ba |


13 bob Ave San Jose , CA
| 1 bd | | 1 ba |


111 Holla Way San Jose , CA
| 3 bd | | 2 ba |


123 Candy Lane San Jose , CA
| 1 bd | | 2 ba |


299 Two Nine Nine Way san jose , CA
| 299 bd | | 299 ba |

Back To Top

## Duplex

```
1  SELECT * FROM search4houses.Addresses;
```

100% 1:1

Result Grid Filter Rows: Search Edit:

| addrID | streetAddress | city | state | zipCode |
|---|---|---|---|---|
| 35 | 13 bob Ave | San Jose | CA | 95111 |
| 36 | 32 Hi Street | San Jose | CA | 95111 |
| 37 | 13 bob Ave | San Jose | CA | 95111 |
| 38 | 111 Holla Way | San Jose | CA | 95111 |
| 39 | 101 Nickel Back Drive | San Jose | CA | 95111 |
| 40 | 133 Beens Ave | San Jose | CA | 95111 |
| 41 | 65 Good Babbies Street | San Jose | CA | 95111 |
| 42 | 90 Rad street | San Jose | CA | 95111 |
| 43 | 33 Cute Drive | San Jose | CA | 95111 |
| 44 | 123 Candy Lane | San Jose | CA | 95111 |
| 45 | 299 Two Nine Nine Way | san jose | CA | 95111 |
| NULL | NULL | NULL | NULL | NULL |

Addresses 1

```
1  SELECT * FROM search4houses.Listing_Address;
```

100% 1:1

Result Grid Filter Rows: Search Edit:

| listing_id | addr_id |
|---|---|
| 34 | 34 |
| 35 | 35 |
| 36 | 36 |
| 37 | 37 |
| 38 | 38 |
| 39 | 39 |
| 41 | 41 |
| 42 | 42 |
| 43 | 43 |
| 44 | 44 |
| 45 | 45 |
| NULL | NULL |

Listing_Address 1

```
1  SELECT * FROM search4houses.Accounts_Listings;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| user_id | listing_id |
|---|---|
| 1 | 38 |
| 23 | 39 |
| 23 | 40 |
| 23 | 41 |
| 23 | 42 |
| 23 | 43 |
| 24 | 44 |
| 24 | 45 |
| NULL | NULL |

```
1 • SELECT * FROM search4houses.Listings;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| listingID | roomType | price | leaseTimeframe | roomNum | bathroomNum | description |
|---|---|---|---|---|---|---|
| 35 | Single Room | 1 | 1 | 1 | 1 | nice to mee tyou |
| 36 | Single Room | 1 | 1 | 1 | 1 | |
| 37 | Apartment | 1 | 1 | 1 | 1 | |
| 38 | Apartment | 128 | 9 | 3 | 2 | holla holla |
| 39 | Studio | 322 | 11 | -1 | 1 | free music ! |
| 40 | Studio | 201 | 9 | 1 | 1 | very cool |
| 41 | Studio | 900 | 13 | 9 | 3 | very good here |
| 42 | Studio | 600 | 13 | 9 | 3 | very good here |
| 43 | Shared Room | 400 | 10 | 3 | 2 | cuuute~ |
| 44 | Apartment | 348 | 6 | 1 | 2 | Free A/C. WiFi included in price. |
| 45 | Apartment | 299 | 299 | 299 | 299 | we accept doggos! |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Listings 1

```
1 • SELECT * FROM search4houses.Listings_AdditionalContact;
```

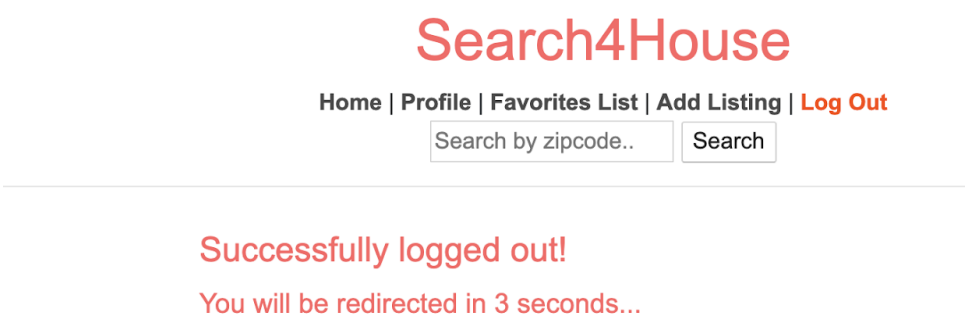
100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

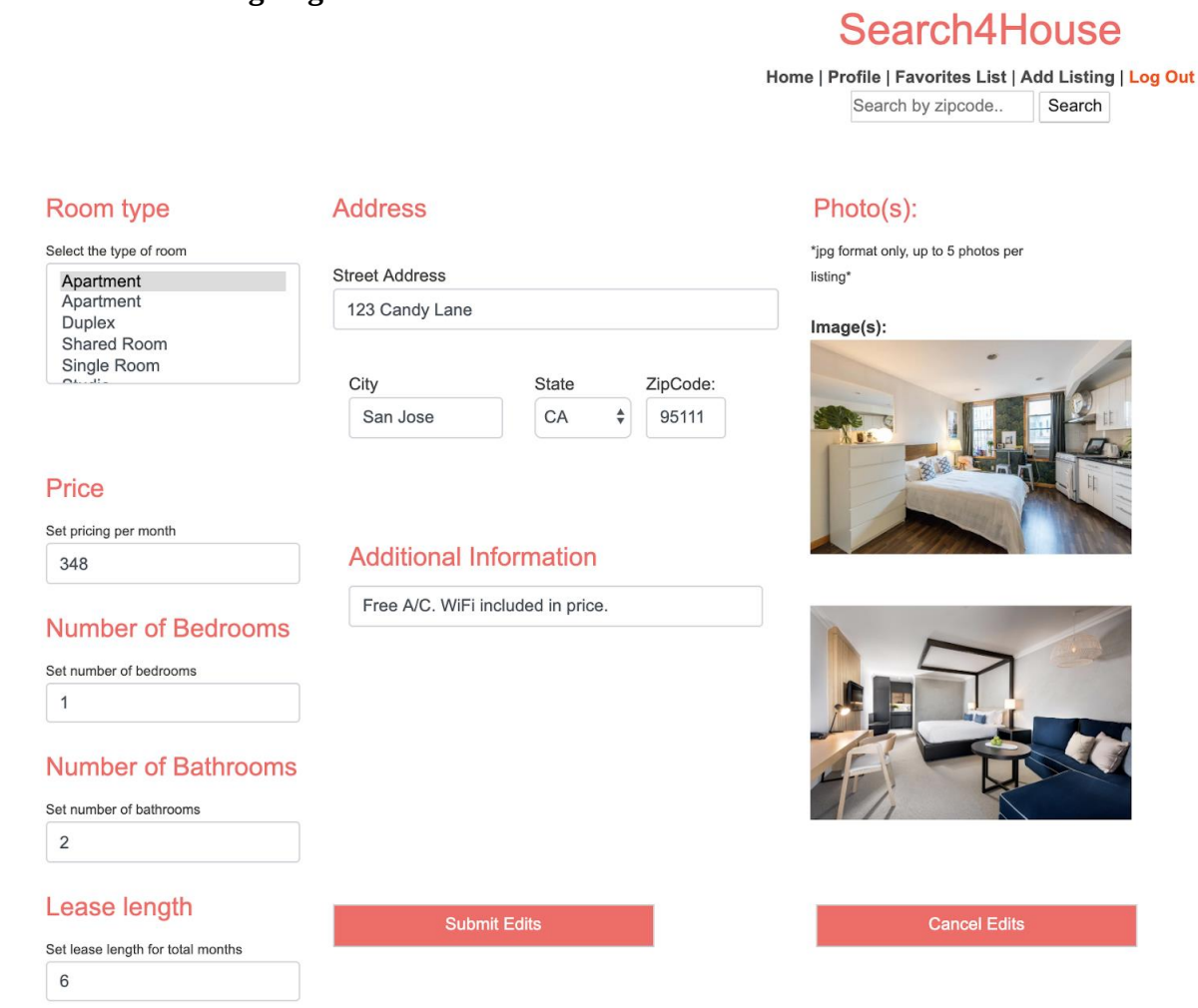
| listing_id | addContact_id |
|---|---|
| 33 | 36 |
| 34 | 37 |
| 36 | 36 |
| 37 | 36 |
| 38 | 36 |
| 39 | 38 |
| 41 | 38 |
| 42 | 38 |
| 43 | 38 |
| 44 | 40 |
| 45 | 40 |
| NULL | NULL |

Listings_AdditionalContact 1

### 6.2.11: Log Out Page:




### 6.2.12: Edit Listing Page:




This page uses the same query to the listing page, but will use different queries where it will get the user's listings and display it into the text boxes already provided. It will then

take these values that the user may or may not have changed and use them to replace the values already in the table.

ResultSet rs = stmt.executeQuery("SELECT * FROM search4houses.Listings

WHERE listingID="+listing_id+";");

ResultSet rs1 = stmt1.executeQuery("SELECT * FROM search4houses.Photos, search4houses.Listings_Photos
WHERE photoID=photo_id
AND listing_id="+listing_id+";");(edited)

ResultSet rs2 = stmt2.executeQuery("SELECT * FROM search4houses.Addresses, search4houses.Listing_Address

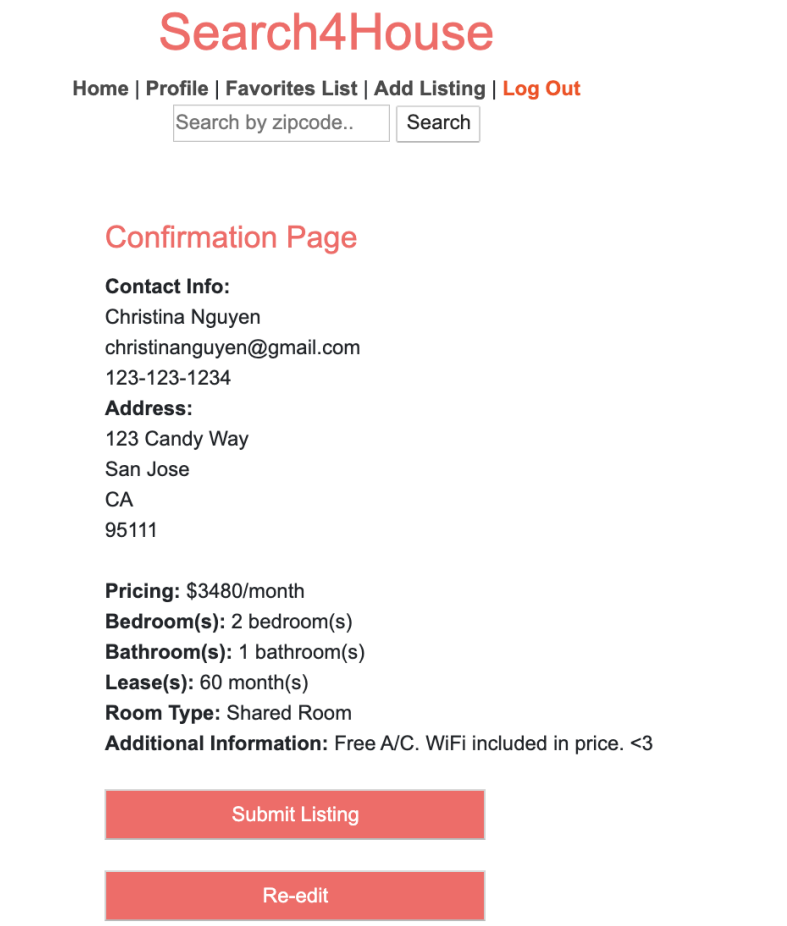
WHERE addrID=addr_ID

AND listing_id="+listing_id+";");

ResultSet rs3 = stmt3.executeQuery("SELECT * FROM search4houses.AdditionalContacts, search4houses.Listings_AdditionalContact WHERE contactAccID=addContact_ID

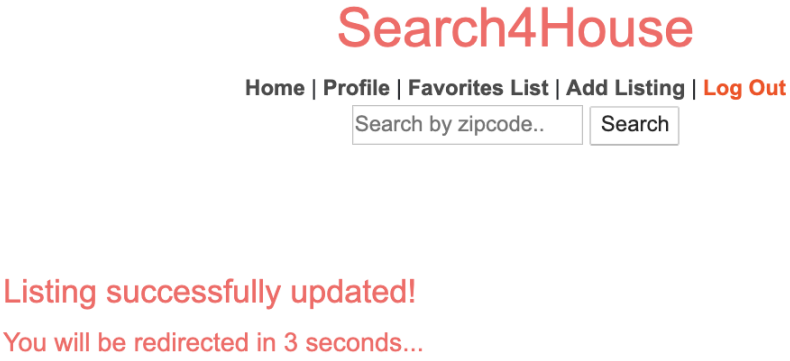
AND listing_id="+listing_id+";");

### 6.2.12.1: Confirmation P




**age**

### 6.2.12.2: Editing Listing Success Page:



Search4House

Home | Profile | Favorites List | Add Listing | Log Out

Search by zipcode.. Search

♡

## $3480/month

123 Candy Way San Jose , CA 95111

**Bedroom(s):** 2
**Bathroom(s):** 1
**Lease length:** 60 months
**Information:** Free A/C. WiFi included in price. <3

**Contact Info:** Christina Nguyen | christinanguyen@gmail.com | 123-123-1234

Search4House

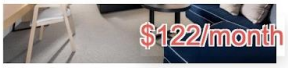
Home | Profile | Favorites List | Add Listing | Log Out

Search by zipcode.. Search

Listings for: 95111

- Apartments
- Duplex
- Shared Room
- Single Room
- Studio

Back To Top




333 Good Drive San Jose , CA
| 2 bd | | 2 ba |

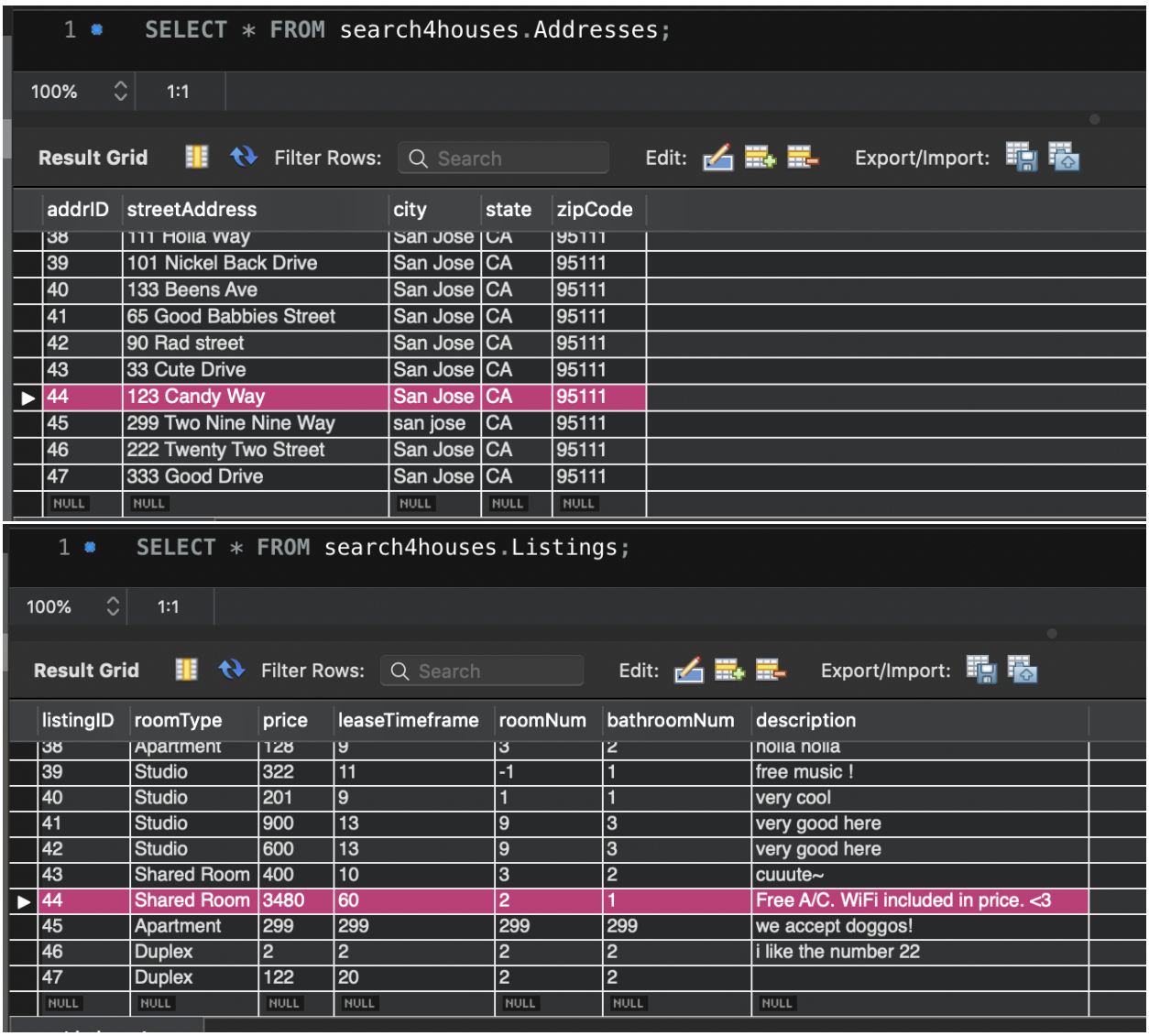
### Shared Room


232 Pepe Way San Jose, CA
| 1 bd | | 1 ba |


33 Cute Drive San Jose , CA
| 3 bd | | 2 ba |


123 Candy Way San Jose , CA
| 2 bd | | 1 ba |

### Single Room



```
1 •  SELECT * FROM search4houses.Addresses;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| addrID | streetAddress | city | state | zipCode |
|---|---|---|---|---|
| 38 | 111 Holla Way | San Jose | CA | 95111 |
| 39 | 101 Nickel Back Drive | San Jose | CA | 95111 |
| 40 | 133 Beens Ave | San Jose | CA | 95111 |
| 41 | 65 Good Babbies Street | San Jose | CA | 95111 |
| 42 | 90 Rad street | San Jose | CA | 95111 |
| 43 | 33 Cute Drive | San Jose | CA | 95111 |
| ▶ 44 | 123 Candy Way | San Jose | CA | 95111 |
| 45 | 299 Two Nine Nine Way | san jose | CA | 95111 |
| 46 | 222 Twenty Two Street | San Jose | CA | 95111 |
| 47 | 333 Good Drive | San Jose | CA | 95111 |
| NULL | NULL | NULL | NULL | NULL |

```
1 •  SELECT * FROM search4houses.Listings;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

| listingID | roomType | price | leaseTimeframe | roomNum | bathroomNum | description |
|---|---|---|---|---|---|---|
| 38 | Apartment | 128 | 9 | 3 | 2 | holla holla |
| 39 | Studio | 322 | 11 | -1 | 1 | free music ! |
| 40 | Studio | 201 | 9 | 1 | 1 | very cool |
| 41 | Studio | 900 | 13 | 9 | 3 | very good here |
| 42 | Studio | 600 | 13 | 9 | 3 | very good here |
| 43 | Shared Room | 400 | 10 | 3 | 2 | cuuute~ |
| ▶ 44 | Shared Room | 3480 | 60 | 2 | 1 | Free A/C. WiFi included in price. <3 |
| 45 | Apartment | 299 | 299 | 299 | 299 | we accept doggos! |
| 46 | Duplex | 2 | 2 | 2 | 2 | i like the number 22 |
| 47 | Duplex | 122 | 20 | 2 | 2 | |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

### 6.3 Step-by-Step

To set up and run our system, these are the steps for how to follow our procedure:

1. Install MySQL and set the password to “newpassword”.
2. Install Workbench.
3. Create search4House schema.
4. Download the required files from the GitHub at [www.github.com/CS157A-Team35/CS157A-Team35](www.github.com/CS157A-Team35/CS157A-Team35).
5. Import all the .sql files to create the databases correctly.

6. Download and install Eclipse IDE for Enterprise Java Developers.
7. Create a new Dynamic Web Project.
8. Name it search4houses.
9. Download Tomcat Apache 7.0.
10. Download the MySQL Connector Java.
11. Set up Tomcat Apache 7.0 with Eclipse.
12. Copy the SQL Connector Java .jar file to search4House > WebContent > WEB-INF > lib.
13. Copy / Import all of the .jsp files for the pages, the style.css file, and the image sources into the search4House > WebContent folder.
14. Refresh Eclipse.
15. Run index.jsp on the Tomcat Apache 7.0 server.
16. You can now use http://localhost:8080/search4Houses to open the project on your browser of preference.

# 7 Project Conclusion

## 7.1 Lessons Learned

### 7.1.1: Yusuf Amer

I learned a lot from this project. Learning new languages and applying it to a project are both a lot of work, and individually can be straining, but when doing them together, a lot of support is needed. I struggled a lot in understanding and implementing the ideas we had into code. My groupmates helped me understand both MySQL and javascript better, and also different ways to implement HTML and CSS code around a web page. I really enjoyed seeing our databased become a part of a website, and how our project grew with every commit that we pushed to GitHib. There was a lot of work put into creating what I thought would be a simple project, and even more debugging than I had expected. Using MySQL became a lot easier as the class moved forward, and I learned a lot about back-end sides of websites through this project.

I believe that I could have and should have done more for this project, but I learned that balance is a very important thing, because spending too much time on homework, or other things, did not leave me with much time for the project, and that personal lives do also need some time, and that I need to set aside more time for the next project that I work on. My previous projects had a lot less front end work to it, but I did enjoy making things look nicer, and the better that it looked, the less that this project felt like school work, but rather a fun assignment to enjoy. Nevertheless, it could have been aligned better, and with resizing and placement issues, I think we could have improved it even more.

These things that I learned come from both educational material through the course, and behavioral skills that I must learn to prioritize like time management and asking for support when needed. I believe that as a Computer Scientist, both features will be

very important for me in the rest of my life, and I hope that they stick with me and I become a better programmer and a better person because of it.

### 7.1.2: Christina Nguyen

I learned a lot from this project. I had experience with NoSQL (Firebase) prior to this, but they are both very different from each other. It was interesting to learn how tables can be updated and the figuring out the code to do so was quite challenging at times, but nevertheless, it was a good hands-on experience with data manipulation. I also learned how much easier it was to pull and update data with relationships because I would only need to update one schema versus two and pulling data that were relative to one another was much easier once I updated my ERD and made my database to be BCNF. Overall, doing the SQL queries were challenging at first, but once I started implementing it more throughout my pages, it became easier and made more sense.

Some future improvements that I would like to implement would be the over UI of the application. I believe that UI is important in selling a product, and I would like to try to keep it as minimalistic as possible; nowadays, a lot of people prefer a more straight forward website versus one that is tech-savvy due to the low learning curve of more simple designs. I would like to try to minimize as much scrolling as needed because the less scrolling the user has to do, the more convenient it would make the application feel.
I would also like to implement Java servlets and more JQuery in the future as well. I am not too familiar with Java servlets so the application was not as dynamic as I would like it to be. I would implement more JQuery frameworks in the future as well such as Node.js or React.js because those libraries are more commonly used within web development.

### 7.1.3: Arman Sandher

I learned quite a bit from this project. All the way from html to JQueries. Prior to this project, I had no experience with this html, CSS, JavaScript or SQL other than what we had learned in the lectures to this class. With much support from Christina, I was able to get over many of these hurdles. Now I have some basic knowledge of html, CSS and JavaScript that I can take to the future. Thanks to the lectures and the class assignments, I have a good understanding of SQL and how manipulate it towards this project. The project itself was interesting to say the least and it was not an easy task to go through. At first, I had trouble implementing the SQL queries into the pages simply due to wrong formatting, and even took more than a few days trying to implement some of them correctly. Deletion of some values in the table were hard to do, and because of that, our program does not implement deletion of listings or favorites. This process did however allow me to learn many other things in the attempt to delete values. This project was a good way to instill hands on knowledge

to the students. after learning and using it myself, I do feel like this class is an important part of coding since databases are used pretty much everywhere on the internet.

There are a few things that I would change/implement in the future of this program. Deletion of listing and favorites would be the first thing to attempt. Even if we couldn't get the listing to delete, maybe having another attribute to listing to see if a listing was sold to be true or false, and having a .jpg to cover the listing photo that reads sold. Another thing I want to implement is a profile picture for each user that can be edited to the user's liking to give the user more uniqueness. There are a few other things that I want to do as well like making the pages more appealing to the eyes and fixing the home page so that it is sized correctly.