SW Engineering CSC648/848 Spring 2019
GatorState
Team 6

Team Members:

Team Lead: Rowvin Dizon (email: rdizon1@mail.sfsu.edu)
Front End Lead: Jonathan Gurdal
Front End: Daisy Sun
Back End Lead: Marlon Johnson
Full Stack: Minho Cha
Full Stack: Rene Elias
GitHub Master: Kayla Musleh

Milestone 2
March 22nd, 2019

History Table

Date Submitted:	**3/22/19**
Date Revised:	**N/A**

1. **Data Definitions V2**

Classes:

- **Home Page:** Default page that provides access to login, navigation of site, and search
- **User Account:** Able to see basic info of listings, favorite/bookmark listings, can create listings
    - Database Elements:
        - Name: A string of User's first name, can be displayed on User profile
        - Username: A string for User's ID credentials that allows user to be found via search
        - Email: One email required for login credentials
        - Password: A string of characters to authorize user at login
        - Age: Integer for
        - Photo/Image: One .jpg or .jpeg file for User Profile Picture
        - Listings: A list that contains user's own posted listings
        - Favorites: A list that contains user's bookmarked listings
- **Listing:** Holds the basic and necessary information of a home. Must be created by Registered User
    - Database Elements:
        - Address: String that holds the physical address/location of listing
        - Images: Multiple .jpg or .jpeg files of Listing
        - Home Type: Falls under either apartment, condo, house category.
        - Number of Rooms
        - Square Feet: Size of listing
        - Price: Cost of listing per month, or total overall cost.
        - User ID: A String of the original poster's User ID.
- **Search:** Retrieves specific information from the database depending on User's input
    - Database Elements:
        - Listing Address: Elements and data from Listing class are retrieved
        - Thumbnail: Small image to represent a larger image
        - Price: Pricing for listings can be pulled up
        - Rating: A percentage that shows the total average of all ratings.

2. **Functional Requirements V2**

Priority 1 *(Must have)***:**

1. **Buy**
    a. Be able to see multiple listings.
        i. Filters shall be applied to narrow down posting selection.
    b. Be able to search different locations to tailor postings to user's location.
2. **Sell**
    a. Be able to show other listings in the area.
    b. Create a button saying "Post a listing" to add a new posting/listing to the site.
        i. Once clicked, the button links to a form containing posting details such as name, address, etc.
            1. At the end, if the user isn't logged in, they shall get a registration pop up.
                a. The registration pop up requires user email and password.
3. **Account**
    a. Name
    b. Photo
    c. Personal Info/Bio
    d. View your listings
    e. Edit Profile
    f. Save Listing (Priority 2)
4. **Search**
    a. Partial Address
        i. Zip Code
        ii. City
        iii. Street
        iv. Neighborhood
    b. Full Address
        i. Everything in partial
    c. Filter (Priority 2)
5. **Login/Registration**
    a. Email
    b. Password

Priority 2 *(Desired)***:**

1. **Filter**
    a. Distance
    b. Bed
    c. Bath
    d. Price
    e. Type of Building

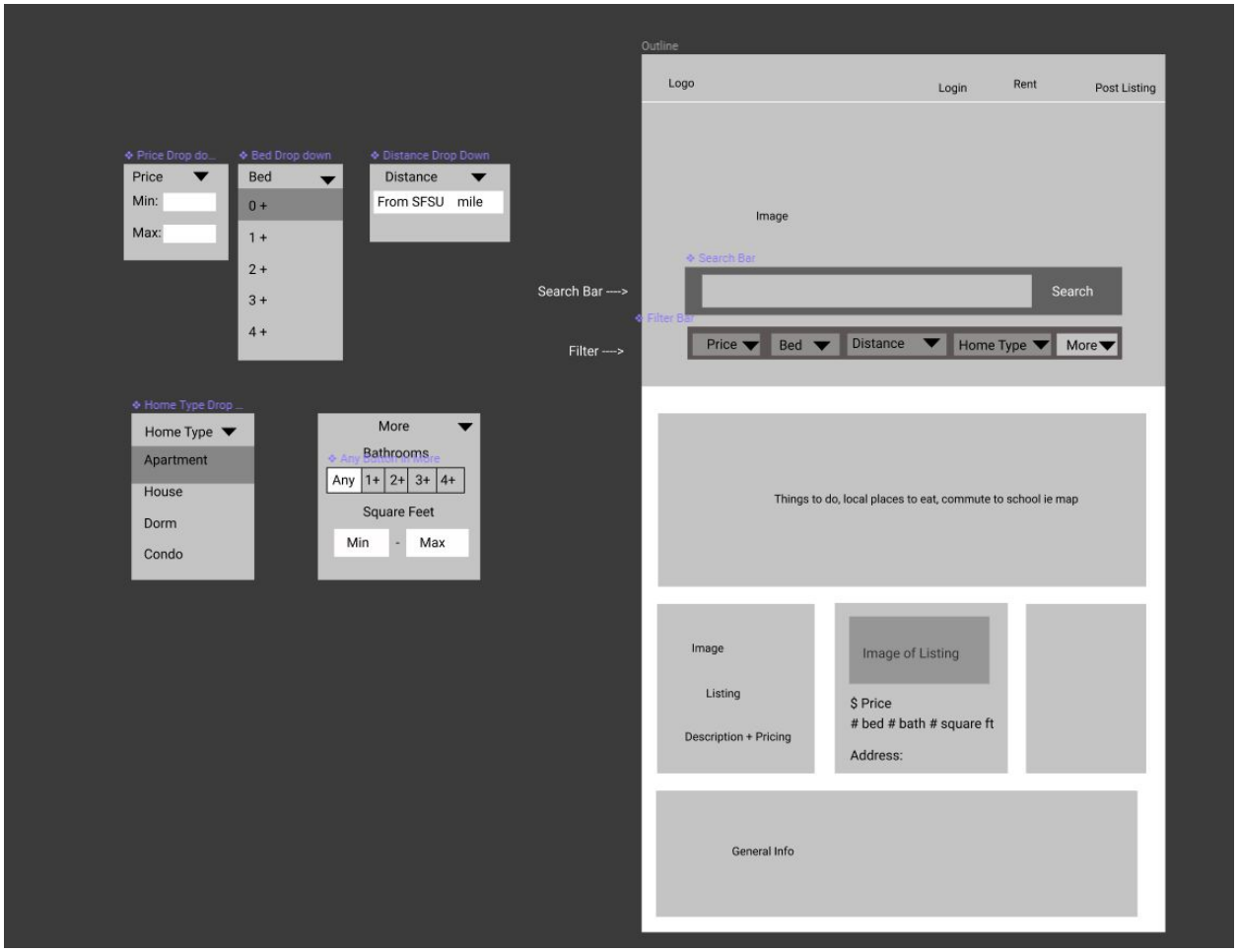
2. **View Listing**
    a. Square footage
    b. Price
    c. Address
    d. General Description
    e. Contact

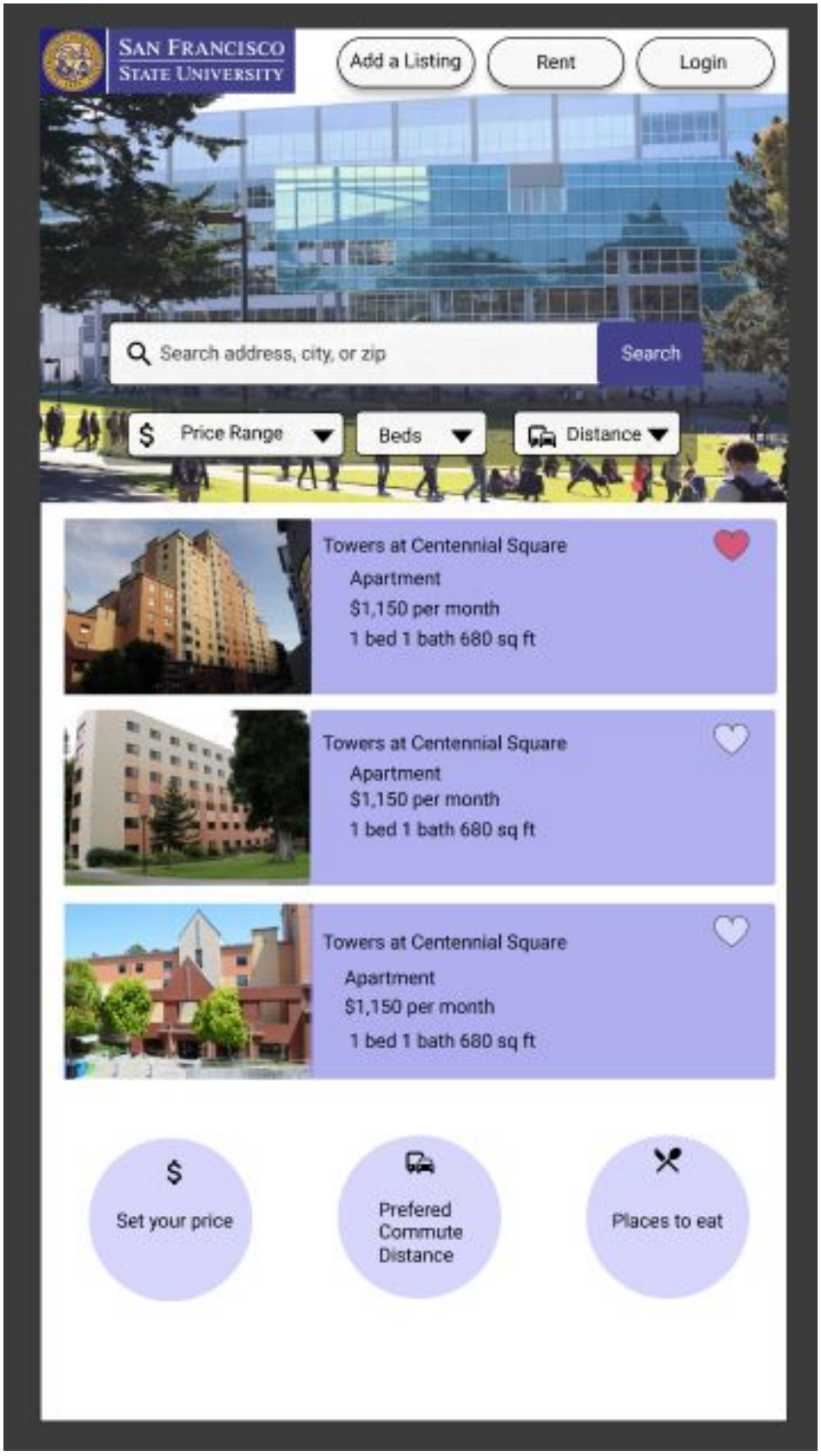
3. **Save Listing**
    a. Save listings to your profile page to allow it to be viewed again later.
4. **Messaging**
    a. Ability to message the seller or buyer directly with direct messages.

Priority 3 *(Opportunistic)***:**

1. **Compare**
    a. Ability to select two listings and compare them side by side to easily compare the different attributes of the listings.
2. **Things to do**
    a. General recommendations of things to do in the area based on your location.
3. **Directions**
    a. Different methods of transportation to get from your listing to SFSU.
4. **Reviews**
    a. Depending on the type of listing, you are able to leave reviews on the listing for everyone to read about the listing.
5. **Tutorial**
    a. Step by step instructions on the posting page to help less technical users post the listings with greater ease.
6. **Feedback**
    a. Feedback button to allow users to message the developers with feedback.

3. **UI Mockups and Storyboards (high level only) :** The Diagram in Figma




San Francisco State University
Add a Listing
Rent
Login
Search address, city, or zip
Search
Price Range
Beds
Distance
Towers at Centennial Square
Apartment
$1,150 per month
1 bed 1 bath 680 sq ft
Towers at Centennial Square
Apartment
$1,150 per month
1 bed 1 bath 680 sq ft
Towers at Centennial Square
Apartment
$1,150 per month
1 bed 1 bath 680 sq ft
Set your price
Prefered Commute Distance
Places to eat

4. **High level Architecture, Database Organization**

DB Organization:

These are the each of the database tables along with the attributes contained within them:

- User
  - UserID
  - Name
  - Username
  - Email
  - Password
  - Age
  - Photo
  - UserListingsListID
  - UserFavoritesListID
- Users
  - UserID(1)...UserID(n)
- Listing
  - ListingID
  - UserID
  - StreetAddress
  - City
  - ZipCode
  - State
  - NumberOfRooms
  - NumberOfBathrooms
  - Images
  - HomeType
  - SquareFeet
  - Price
  - Rating
  - ThumbnailPhoto
- Listings
  - ListingID(1)...ListingID(n)
- UserListingsList
  - UserListingsListID
  - ListingID(1)...ListingID(n)

- UserFavoritesList
  - UserFavoritesListID
  - ListingID(1)...ListingID(n)

Media Storage:

We have decided to use a file system to store our data, images as well as gps data.

Search/filter architecture and implementation:

We shall organize the searches we display first and foremost around the specification of the user. We shall display listings that are within the range of the filters specified. Additionally we can show recommended listings that are close to the users specifications but are slightly* out of range. (We need to define what slightly means)

Algorithm Overview:

Entered in from user search:

1. Address
   a. If given, show the building of the exact address that was given
   b. Database items used:
      i. Listing
         1. Street Address
         2. City
         3. ZipCode
         4. State
2. Zip code
   a. If we are only given a zip code, show all listings in that given zip code
      i. Filters will prioritize which listings are shown first
   b. Database items used:
      i. Listing
         1. ZipCode
3. City
   a. Show Listings in that city across the city, group by neighborhood.
      i. Filters will prioritize what is shown first.
   b. Database items used:
      i. Listing
         1. City

Filter priority rating:

- High: Only find listing that are within what the user specified.
- Low: Show listings that are within what the user input, however if a listing has been found within the filtered input except this one, then dismiss this one.

Filters that can be added:

1. Price
    a. High priority
    b. If a user specifies price our algorithm will then only show listings within that range.
    c. Database items used:
        i. Listing
            1. Price
2. Distance
    a. High priority
    b. Stay within the users input.
    c. Database items used:
        i. Listing
            1. ZipCode
            2. City (optional)
3. Home type
    a. Low priority
    b. Show only the type of buildings the user specifies
        i. Dorm
        ii. Apartment
        iii. House
        iv. Condo
    c. Database items used:
        i. Listing
            1. HomeType
4. Bed
    a. Low priority
    b. Lower priority
    c. Database items used:
        i. Listing
            1. NumberOfRooms

5. Bath
    a. Low priority
    b. Database items used:
        i. Listing
            1. NumberOfBathrooms
6. Square Feet
    a. Low priority
    b. Database items used:
        i. Listing
            1. SquareFeet

If the user changes a filtered search after the initial search, prioritize showing all the new listings that fall within the filter.

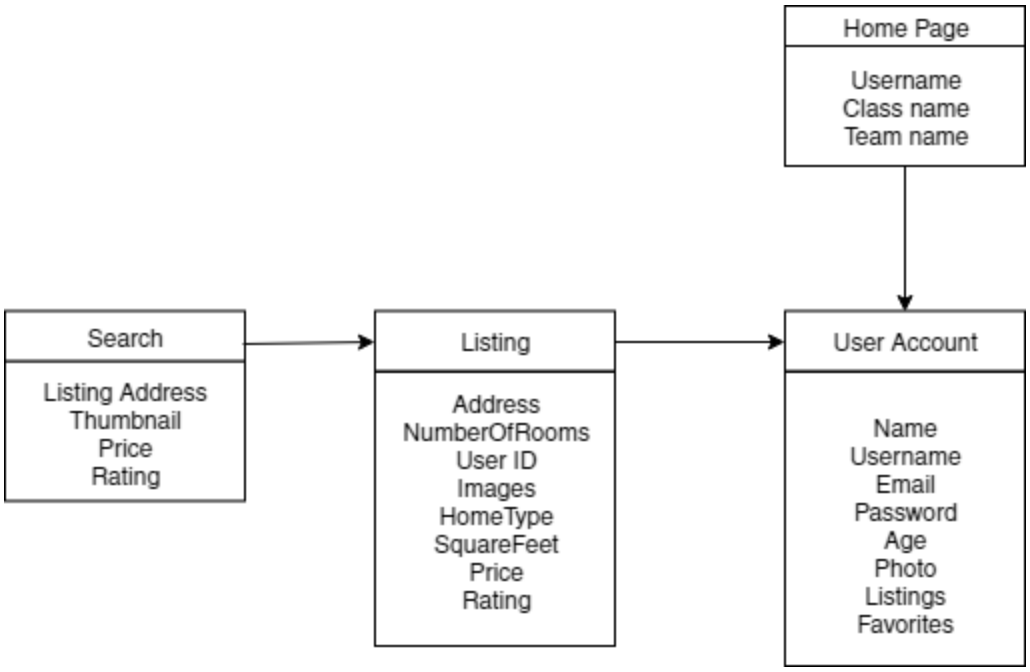
How information will be retrieved from the database:

Depending on which filters are specified, only the necessary attributes needed to determine which listings are desired to populate in the search results and in which order they should be displayed will be initially queried. The requesting app will then parse through the information provided by the database, and send a second request which will be have the desired order, will request more information on the listings to display, and will omit any listings from the initial query that are undesired. Here are couple of examples of how this will generally work:

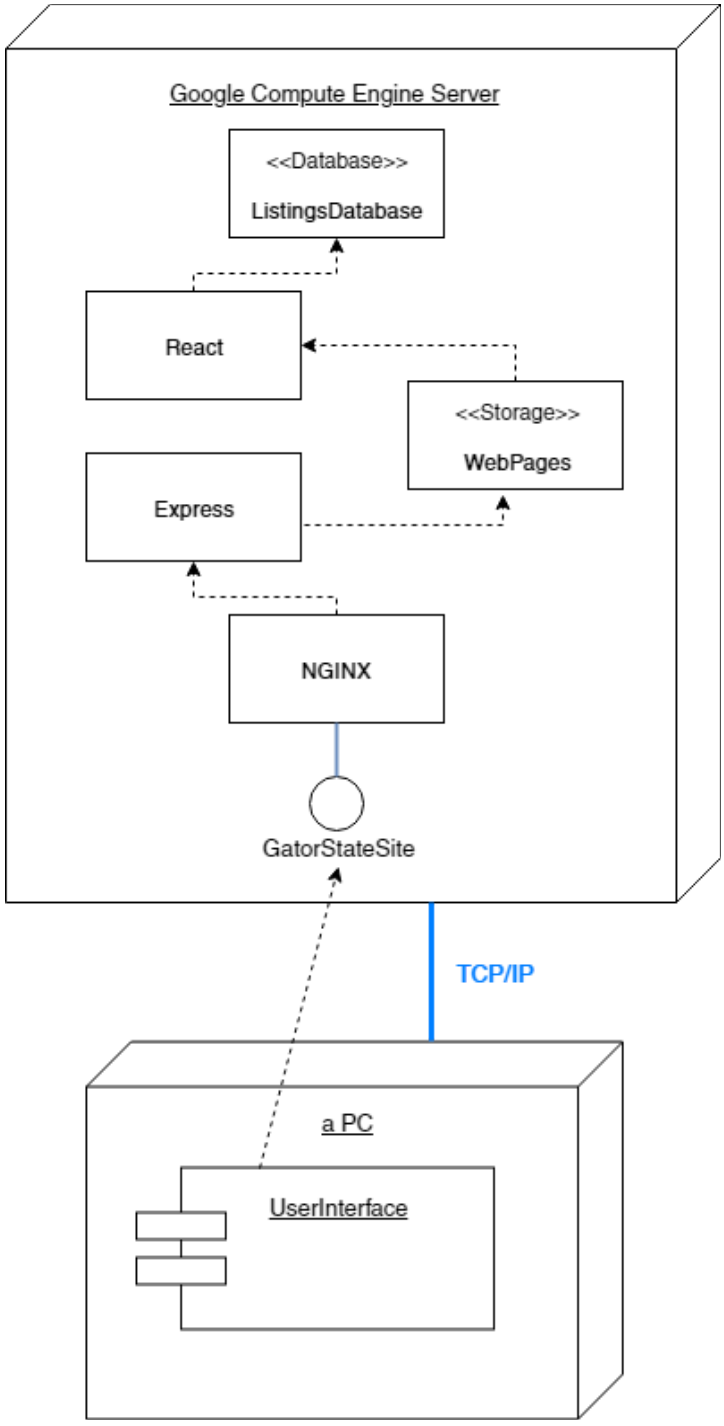
1. No filters specified:
    a. ListingID and ZipCode/City of all listings will be requested from the database. Requesting app will then parse through results and re-query for all the listings in distance order and omit any results that are undesired in the new query.
2. Price range specified:
    a. ListingID and ZipCode/City of all listings will be requested from the database using a query that only returns results within the specified price range using the Price attribute in Listing. Requesting app will then parse through results and re-query for the listings in a new order that uses distance and pricing as weights and omit any results that are undesired in the new query.

5. **High Level UML Diagrams**

High-Level UML:



Component/Deployment Diagram:



6 .**Identify *actual* key risks for your project at this time**

Skills (Risks)

- Back End
    - Not properly structure the database entities
        - Having clear entity name that relate to their function and using proper naming conventions.
    - Race conditions
        - Mutex, semaphore locks.
    - Setting proper access depending on user priority.
        - Write the right rules for each priority level of the users.
    -
- Front End
    - Having improper states and calling improper functions
        - Setting obvious and clear names for our states and functions.
    - Not properly linking sources to our back end.
        - Actively debug and do code reviews. Create test cases for functions in our front end.
- Scheduled risks
    - Pushing bad code to the main branch
        - Push to the development branch and make sure it runs properly before pushing it to the main branch
    - Recently pushed code breaks the site
        - Restore to the previous version of the site.
- Technical Risk
    - Map API
        - Look into other programs that integrate google map api and use it as a reference.
    - Linking front end and back end while using react.
        - Study react in depth and find how others use react when having the front end communicate with the back end.
    - SQL injections
        - Proper checks on any data that is being inputted by the user.
    - Properly protecting sensitive information that the users may have.
        - Identify what the information is and set proper checks and restrictions on that data.
    - Prevent bots from flooding the sites with postings to crash it.
        - Implement a captcha when posting

- Teamwork Risk
    - People are doing something they aren't comfortable coding
        - Clear communication among teammates and discussing what people need help with or changing the tasks people are doing.
    - Teammates not showing up to class and or scheduled meetings.
        - Discuss to see whether it is an issue of the time the group is meeting or talk to the team member who is missing the meetings and see if there is a reason why and if there is a way to resolve it. If all else fails and the group member refuses to meet up or work on the project, contact the professor to see what the next steps should be.
- Legal
    - Make sure users are posting listings that are actually theirs and not just someone else’s home.
        - Have a team to verify the listing before listings are posted.
        - Have a report button that users can use.
    - Properly licensing any material that is not ours, i.e. SFSU logo
        - Don’t use them and create are own, or contact the owner of the material to see if we can get the rights to use the material.

7. **Project management**

We managed all of Milestone 2 requirements by organizing all the tasks that were assigned into Trello. From there we meet up as a group to discuss all the requirements and agreed on how to separate the tasks that we felt should be handled by front end and back end. We organized the priorities of the tasks that needed to be completed first in order to make the rest of the tasks easier. From there we assigned tasks to team members that felt the most confident with completing them.