SW Engineering CSC648-848-05 Spring 2023

BucketLyst

Team 1

Marie Shimizu[1], Arielle Riray[2], Francis Quang[3], Tommy Truong[4],
Kenneth Gee, Rabin Karki, Aaron Kuo

[1] Team Lead mshimizu4@sfsu.edu
[2] Front End Lead
[3] Back End Lead
[4] Github Master

Milestone 3

Milestone	Date
M3V1	4/27/2023
M2V2	4/24/2023
M2V1	4/3/2023
M1V2	4/2/2023
M1V1	3/2/2023

Table Of Contents

1. Data Definitions

1. Users

Users are people that use our platform. After registering and successfully logging on, they can use our customer facing features. This includes creating, sharing, deleting lists, or viewing provided data analytics of your business. There are two types of users and each user type has access to the specific functionalities.

a. PersonalUser

A PersonalUser intends to use this app to make and view lists for themselves. They will only be able to create, read, update, and delete lists given appropriate permissions. They can also make changes to their account details. Fields they contain include the followings:

- ID: unique integer that refers to the user. Primarily used for internal user search.
- Username: unique user-decided String, searchable by other users.
- Email: unique String, for communication between BucketLyst admin and user.
- Password: String.
- isPublic: privacy setting for the user.
- FollowerList: a list of PersonalUsers that have this specific user has, following them.
- FollowingList: a list of PersonalUsers that this user follows.
- BlockedList: a list of PersonalUsers that this user does not want in their following list. This should always be private.
- LocationLists: Lists of LocationList that this user can edit or view. Lists may be viewable based on the visibility setting.
- Profile: PersonalUser has a Profile to store their name, description and

Location

b. BusinessUsers

BusinessUsers are users that own a business that uses our application to promote their company. They promote their company by running ads on BucketLyst. Their functions include creating and removing ads, viewing active ads. When they register, they will select a business type account with their company details.

- ID: unique integer that refers to the user. Primarily used for internal user searching.
- Username: unique user-decided String, searchable by other users.
- Email: unique String, for communication between BucketLyst admin and user.
- Password: String.
- PhoneNumber: String, a phone number of this business account
- CompanyName: String, the BusinessUser's company
- PaymentInformation: JSON of their payment information
- runAds: boolean, option for BusinessUsers to choose whether the advertisements are active or inactive.

2. Profile

Profile is a class associated with PersonalUsers. It is a page that contains the user's name, description, and where they are located. The profile's fields will have the following:

- FirstName: String, first name of the user
- LastName: String, last name of the user
- Description: String, description of themselves

- Location: Optional for users to set where they are located

3. **Comment**

Comment allows PersonalUsers and BusinessUsers to express their opinions, provide feedback, or ask questions about a place. The comment's field are the following:

- ID: unique id used to identify and track comments. To ensure comments are organized and analyzable.
- Author: refers to the PersonalUser
- addedDate: The date when comment was added
- isEdited: boolean, shows that comment was edited or not
- Content: String for description

4. **LocationList**

LocationList is a list of Locations. It is used by PersonalUsers to edit, view and share their list of locations that they like. It's editable by multiple users, has visibility options, and a description. It can only be deleted by the original owner. Thus, the following fields:

- ID: unique integer that refers to this LocationList object. Primarily used for internal searching.
- Name: String, a searchable name for PersonalUsers
- Tags: Collection of Strings to describe the contents of this LocationList
- Description: String, descriptor
- Location: The general location or area where all these Locations are located
- Owner: PersonalUser that owns this LocationList

- Collaborators: a list of PersonalUsers that can edit this LocationList.
- Locations: Collection of Locations that this LocationList contains
- isPublic: Indicates visibility of List

5. Location

Location is a container for a Location Data Object where a Personal user may edit fields they want to describe this Location.

- Creator: original user that created this specific Location
- Name: Name that the Personal user would want to call this specific Location
- Description: String descriptor
- Tags: Collection of Strings to describe this Location
- GoogleMapLink: link to their Google Maps page, grabbed from Location Data
- Object
- LocationDataObj: Location Data Object that contains internal data about this
- location

6. LocationDataObject

This is an internal reference to the specific location. Separating the location data details from the Personal user to access allows for individual customizability for Personal users.

- DateAddedToDatabase: date first created by a user
- GPSCoordinate: From a Google Maps link, parse the specific longitude and latitude for location manipulation
- GoogleMapsLink: Direct string of the location on Google Maps
- IncludedInList: number of people that include this in their list

2. Prioritized Functional Requirements

Priority 1:

New User:

1. User should create their username
2. User should enter their email
3. User should create a password
4. Same username and email cannot exist in database

Registered User:

5. User should be able to sign in with their registered details
6. Users should be able to reset their password in case they forget it
7. User can create a list
8. User can manage a list
9. User can own multiple lists
10. User can name a list
11. User can add tags
12. User can edit lists
13. User can edit location data fields
14. User can edit user profile
15. User can edit notes on the lists
16. User can edit their account details
17. User can edit the visibility of the list (public, only with friends, private)
18. User can add lists
19. User can add their followers
20. User can add additional information to profile
21. User can add notes on the lists
22. User can add a location to list
23. User can delete lists
24. User can delete notes on the lists
25. User can delete their account
26. User can delete location from their list
27. User can share their list with others with link
28. User can search for locations by tags

29. User can search for locations by categories
30. User can search other users on the platform
31. User can follow other users
32. Users can see other's profiles
33. User can sign out
34. User can explore lists through the recommendations
35. User has option to register using Google account
36. User has option to register using facebook account
37. User can delete followers
38. User can delete their following list
39. Users can own shared lists as a group so that everyone can edit them
40. User can search for locations by radius from the user's current location
41. User can search for locations by open now
42. User can search public lists with keyword
43. User can send a follower request to other users
44. User can accept follower request
45. User can deny follower request
46. Users can rate a place on a scale 1-5 (tier)
47. Users can view notifications
48. Users can turn off notifications
49. Users can turn on notifications
50. User can be notified via email
51. User gets notified for new activities in other people's list
52. Using the data to create popular lists like "most favorited places"
53. Users can view recommendations from other users
54. Users can view a map that displays the location of their saved places and recommendations.
55. Users can import their existing lists from Google Maps

Business User:

56. Business user should enter their company name
57. Business user should enter their email
58. Business user should create a password
59. Business user should be able to sign in with their registered details
60. Business user should be able to reset their password in case they forget it
61. Business user can sign out
62. Business user can market their establishment on platform

Location:

63. Location contains google map link that allow user to jump to the location on google map seamlessly
64. Location has “Navigation” that allow users to open google map navigation

Priority 2:

New User:

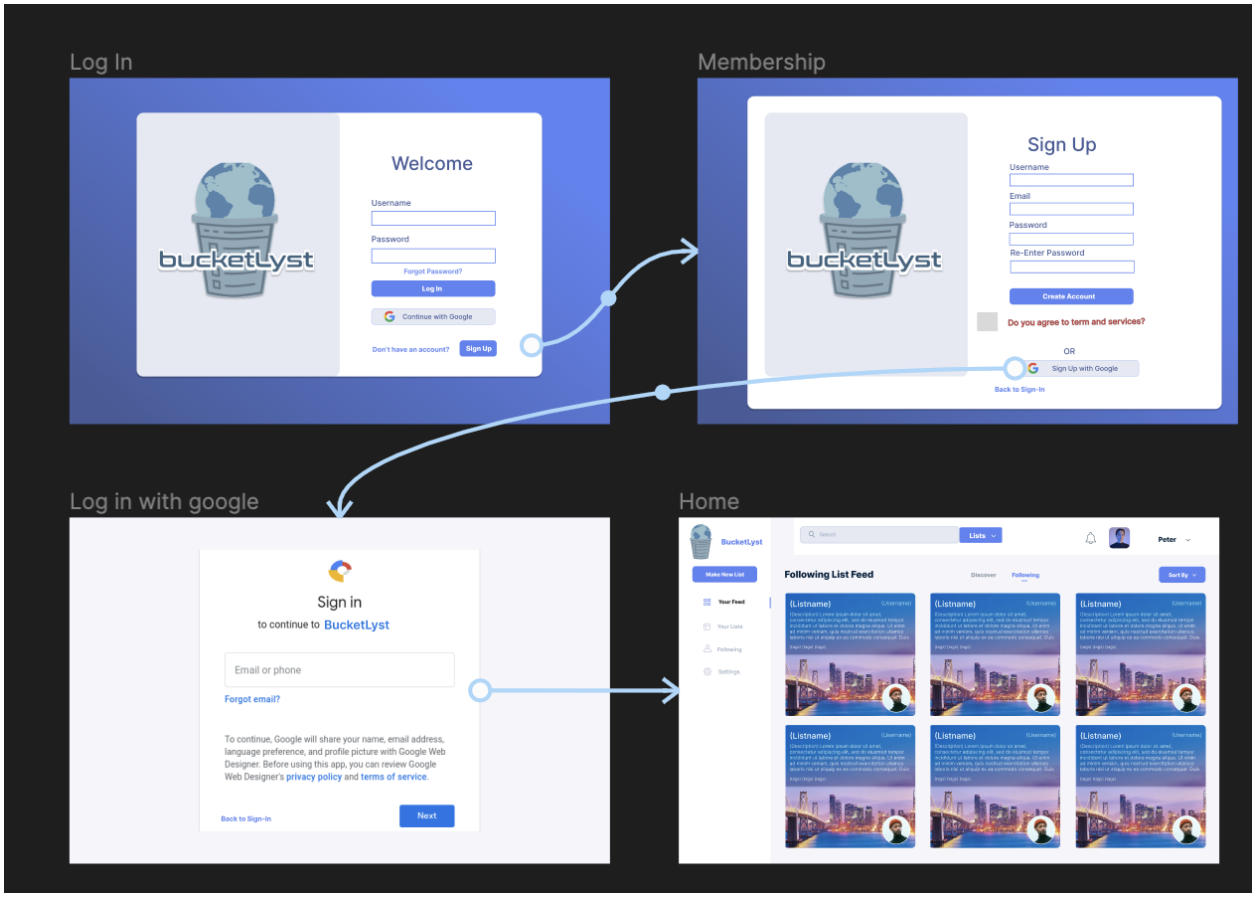
1. New user can see homepage
2. New user can see some of the public lists

Registered User:

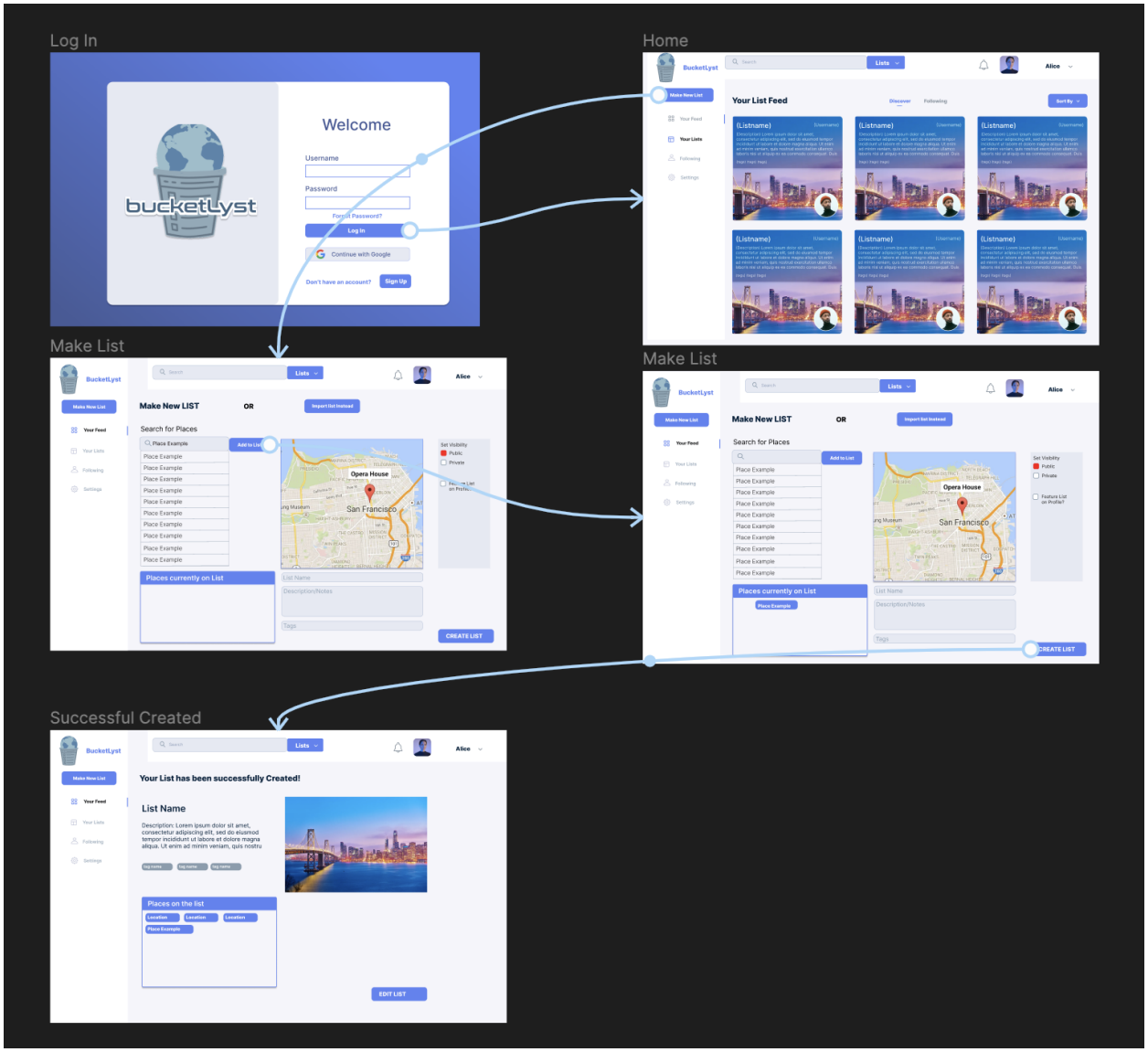
3. User can compare their list with selected list and see the difference
4. User can copy other’s list to make it their own list
5. User can temporarily disable their account
6. User can choose the default privacy setting for their list
7. User has ability to choose to get notified - changes on the list
8. User has ability to choose to get notified - changes friend's activity in general
9. Users can report abuse, such as inappropriate comments or content

3. Wireframes

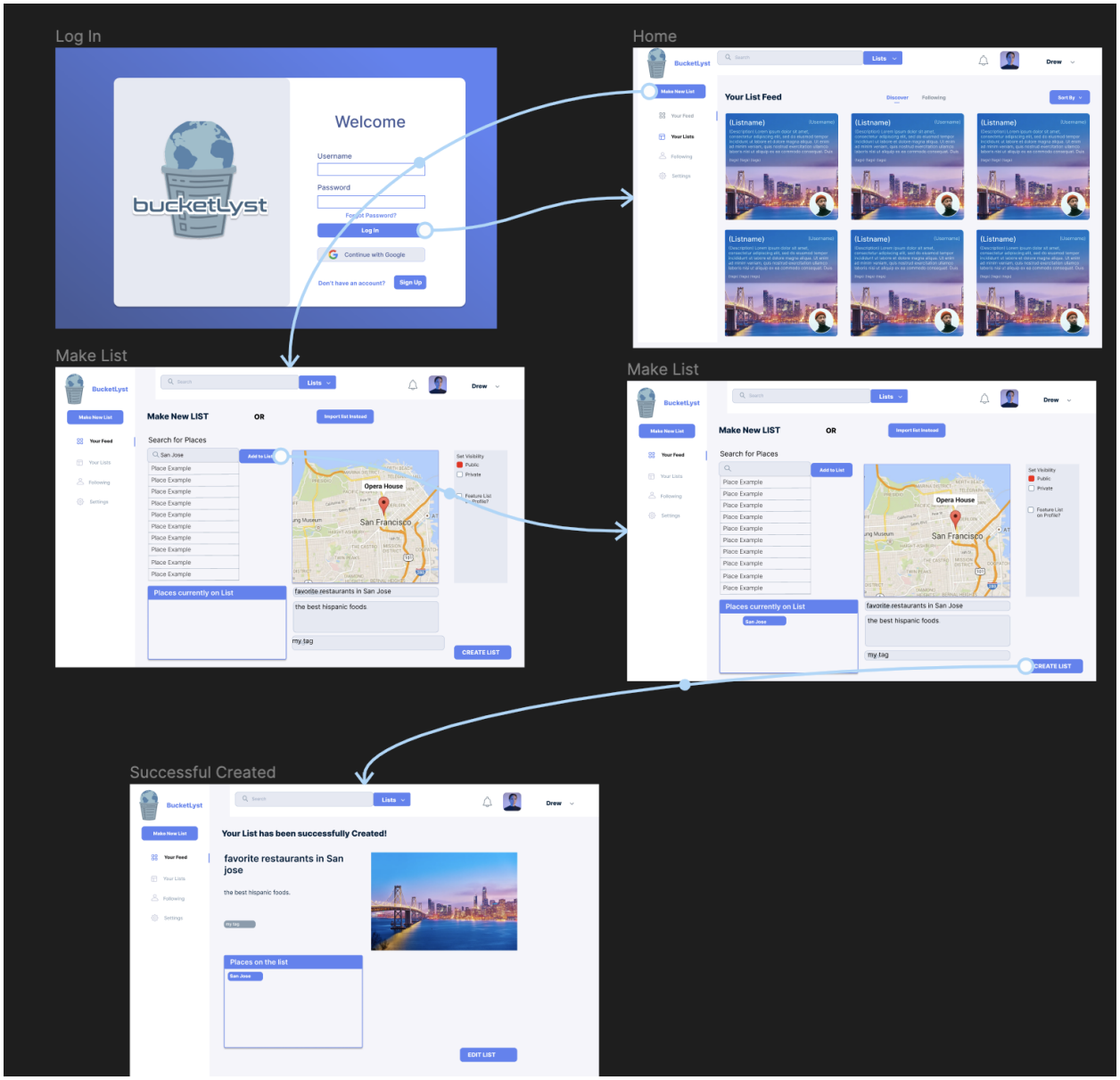
Use case 1: Signing up



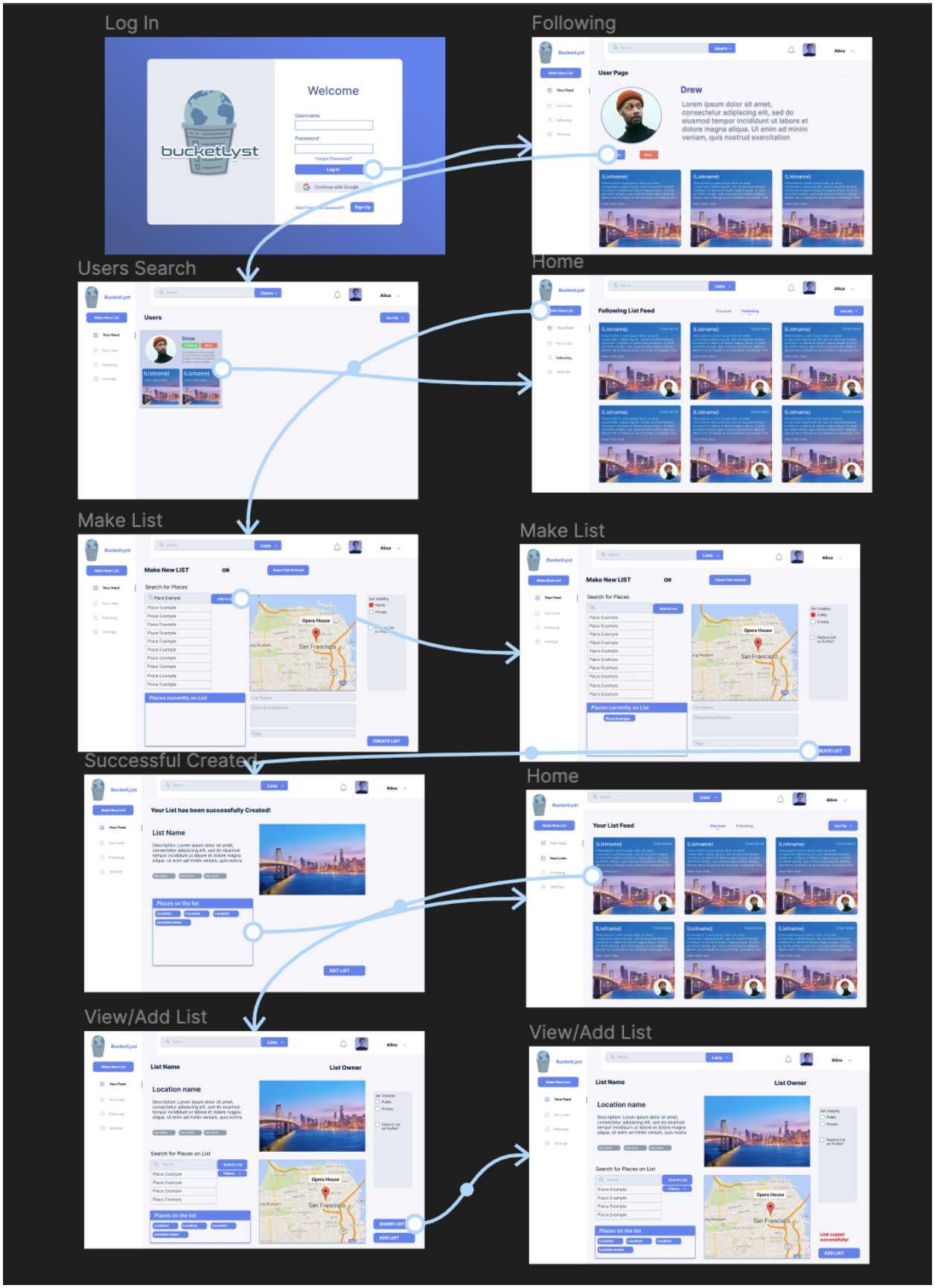
Use case 2: Create a list



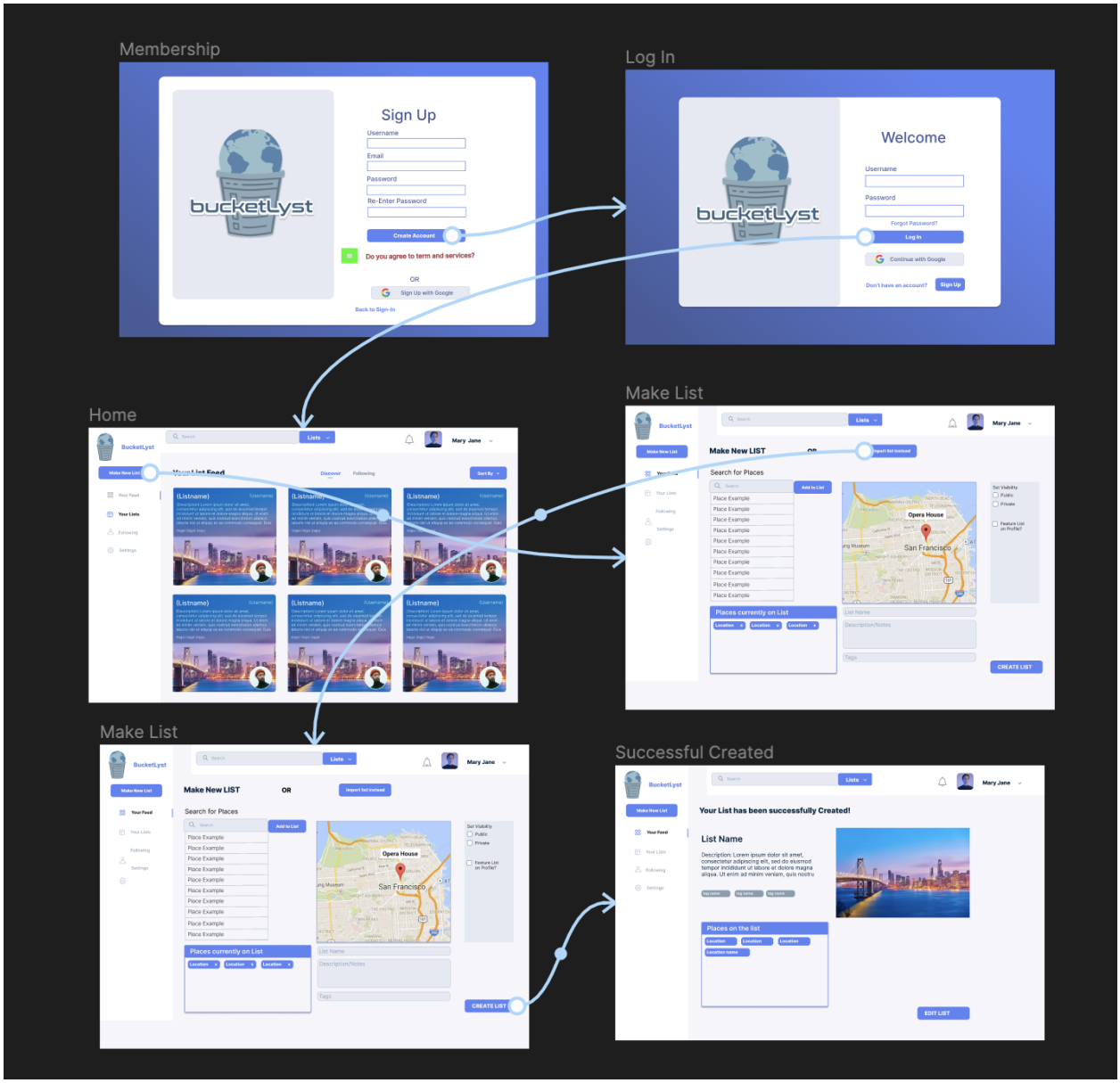
Use Case 3: Add Details on One Location



Use Case 4: Share a list

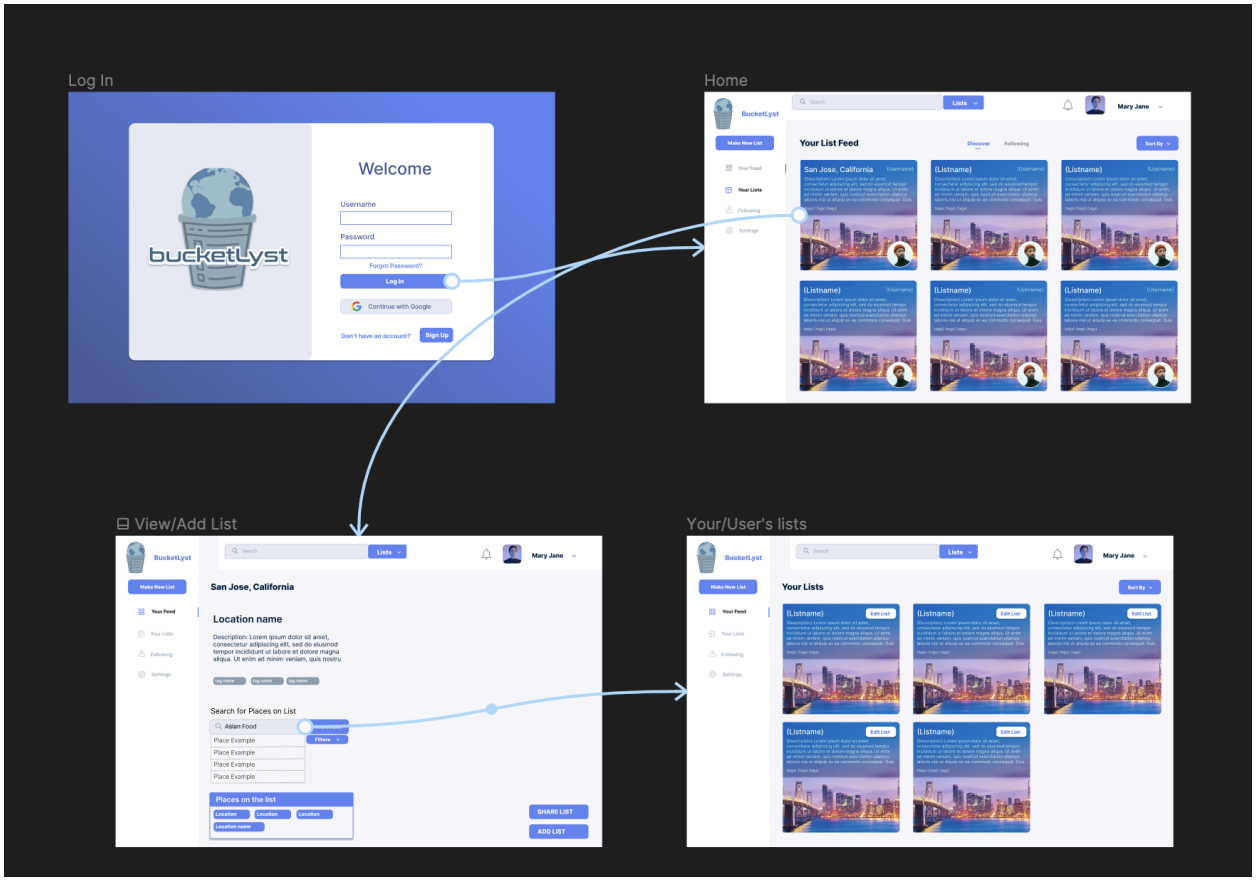

Log In
Welcome
Following
User Page
Drew
Users Search
Users
Home
Following List Feed
Make List
Make New LIST
OR
Search for Places
Opera House
San Francisco
Places currently on List
CREATE LIST
Make List
Successful Created
Your List has been successfully Created!
List Name
Places on the list
EDIT LIST
Home
Your List Feed
View/Add List
List Name
List Owner
Location name
Search for Places on List
SHARE LIST
ADD LIST
View/Add List
List Name
List Owner
Location name
Search for Places on List
Places on the list
ADD LIST

Use Case 5: Import a List from Google Maps



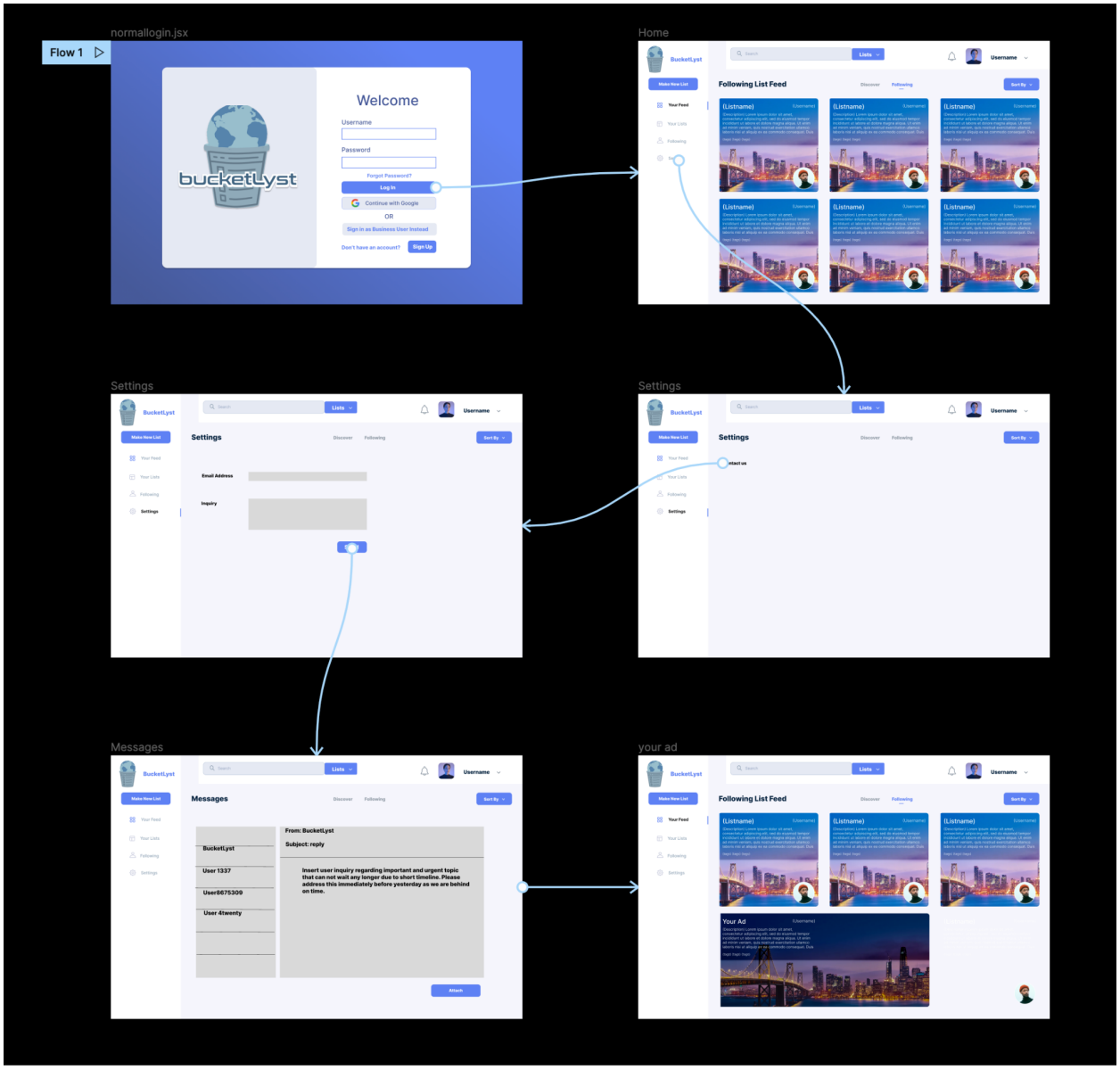
Use Case 6: Exploring New Places from Friends' List

Use Case 7: Search within a List with Criteria

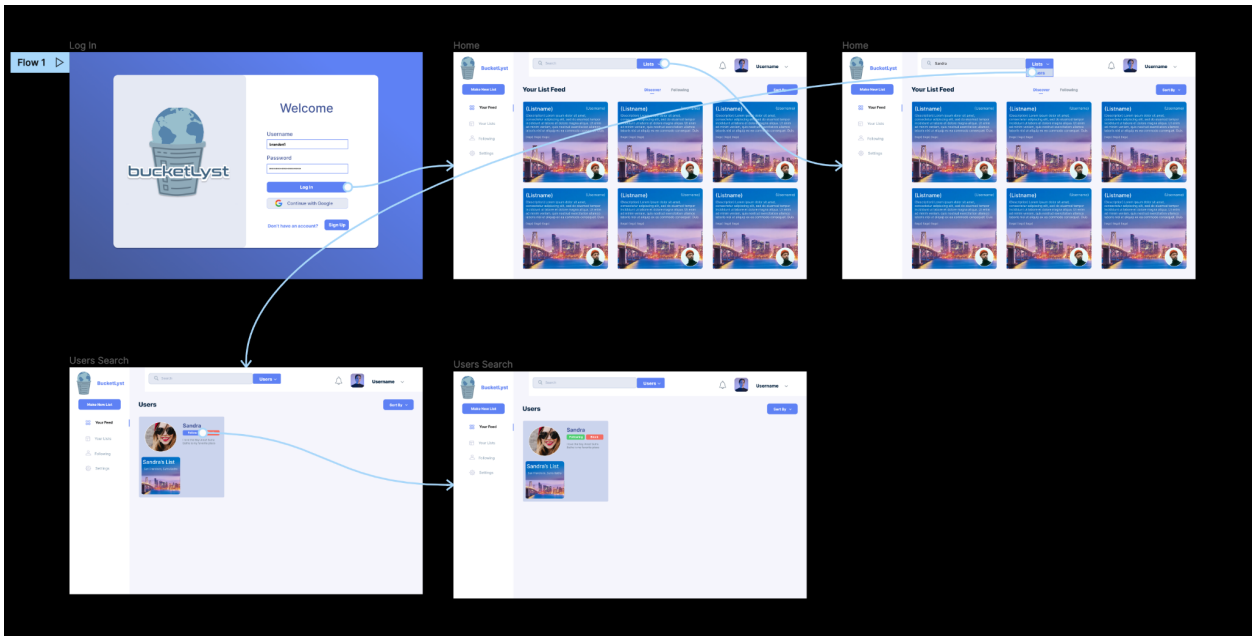


Use Case 8: Edit an Existing List

Use Case 9: Boost a Business Presence:



Use Case 10: Gaining a follow on BucketLyst



4. Database Architecture and Organization

A. The Database Requirements (Business Rules)

Based on the competitive analysis we conducted in Milestone 1, we identified the ability for users to customize their lists as the primary unique feature of our application. This includes the ability to create, edit, add, delete, search within, and import from Google Maps. All functional requirements associated with these features are considered of equal importance (Priority 1). The requirements below:

1. User
    i. A User must have at least one username, password and email.

2. PersonalUser
    i. A PersonalUser can have at most one UserProfile
    ii. A PersonalUser can have at least zero LocationLists.
    iii. A PersonalUser can make at least zero locations.
    iv. A PersonalUser can follow at least zero other users.
    v. A PersonalUser can be followed by at least zero other users.

3. BusinessUser:

    i. A BusinessUser must have at least one username and email.
    ii. A BusinessUser can have at most one phone number, company name , and payment information.
    iii. A BusinessUser can run at least zero ads on the platform

4. UserProfile
    i. A user can make at most one UserProfile.
    ii. A UserProfile must have at least one first name and last name.

5. LocationList
    i. A locationList must have a single owner user.
    ii. A LocationList can be public or private.
    iii. A LocationList can contain at least zero locations.

6. Location
    i. A Location must have GPS coordinates or a Google Maps link.
    ii. A Location can have a rating system.
    iii. A Location can have at least zero tags.

7. LocationDataObject
    i. A LocationDataObject must have GPS coordinates or a Google Maps link.
    ii. A LocationDataObject can be part of at least zero Locations.

8. Follower
    i. A Follower must have at least one follower and following PersonalUser.
    ii. A follower can have at least one request status.

9. Comment:.
    i. A comment must have at least one author, added date, and content.
    ii. A comment can be edited at most once.
    iii. A comment can be added to at least zero locations.

10. Notification:
    i. A notification must have at least one user and content.
    ii. A notification must have at most one type.
    iii. A notification can be marked as read at most once.

B. The Entities, Their Attributes, and Domains at the High Level

1. **User** (Strong): This table stores basic user information such as username, email, and password.

    - user_id: primary key, numeric
    - username: strong key, alphanumeric
    - email: strong key , alphanumeric
    - password: weak key, alphanumeric
    - usertype: weak key, alphanumeric

2. **PersonalUser**(Strong): This table stores the information about PersonalUsers including a flag indicating whether the user's profile is public or not.
    - personal_uid: primary key, numeric
    - user_fk: foreign key referencing to User entity, numeric.
    - is_public: weak key, boolean
    - profile_photo: weak key, alphanumeric

3. **BusinessUsers**(Strong): This table stores the information about the business users so that they can promote their business on our platform.
    - business_id: primary key, numeric

- phone_number: weak key, numeric
- Company_name: weak key, alphanumeric
- payment_info: weak key, alphanumeric
- runs_ads: weak key, boolean
- user_fk: foreign key referencing to User entity, numeric

4. **UserProfile** (Strong): This table stores the information about the profile information of users.

    - profile_id: primary key, numeric
    - first_name: weak key, alphabet
    - last_name: weak key, alphabet
    - description: weak key, alphanumeric
    - date_of_birth: composite key, numeric
    - user_fk: foreign key, numeric

5. **Followers** (Weak): This table stores the information about the user's followers and followings.

    - id: primary key, numeric
    - request_status: weak key, alphanumeric
    - follower_fk: foreign key referencing to PersonalUser entity, numeric
    - following_fk: foreign key referencing to PersonalUser entity, numeric

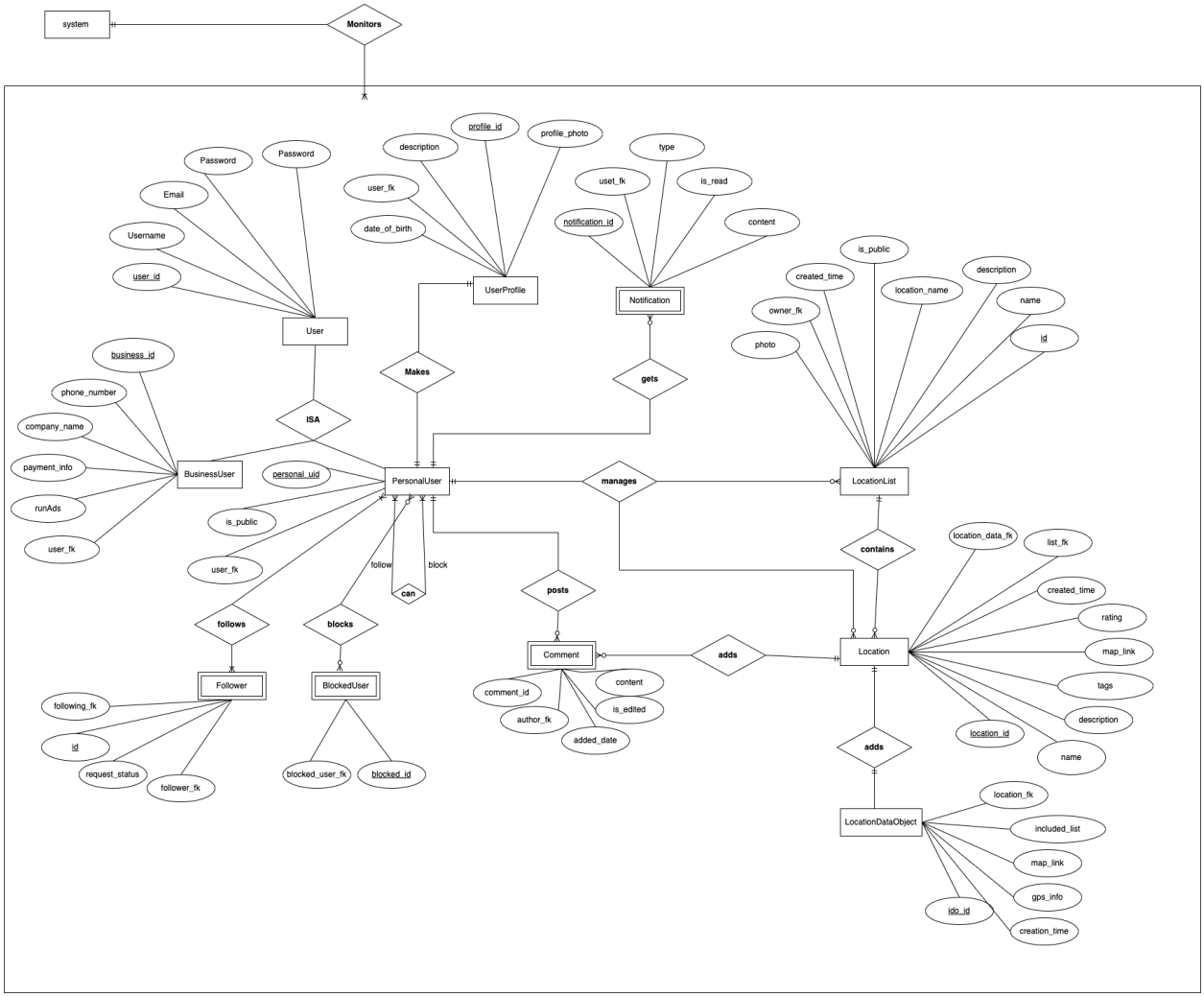
6. **BlockedUser**(Weak): This table stores the information about the blocked Personal users.
    - blocked_id: primary key, numeric
    - Blocked_user_fk: foreign key referencing to PersonalUser entity.

7. **LocationList** (Strong): This table stores the information about the LocationLists such as the name, description, and creator of the list, as well as whether it is public or not.

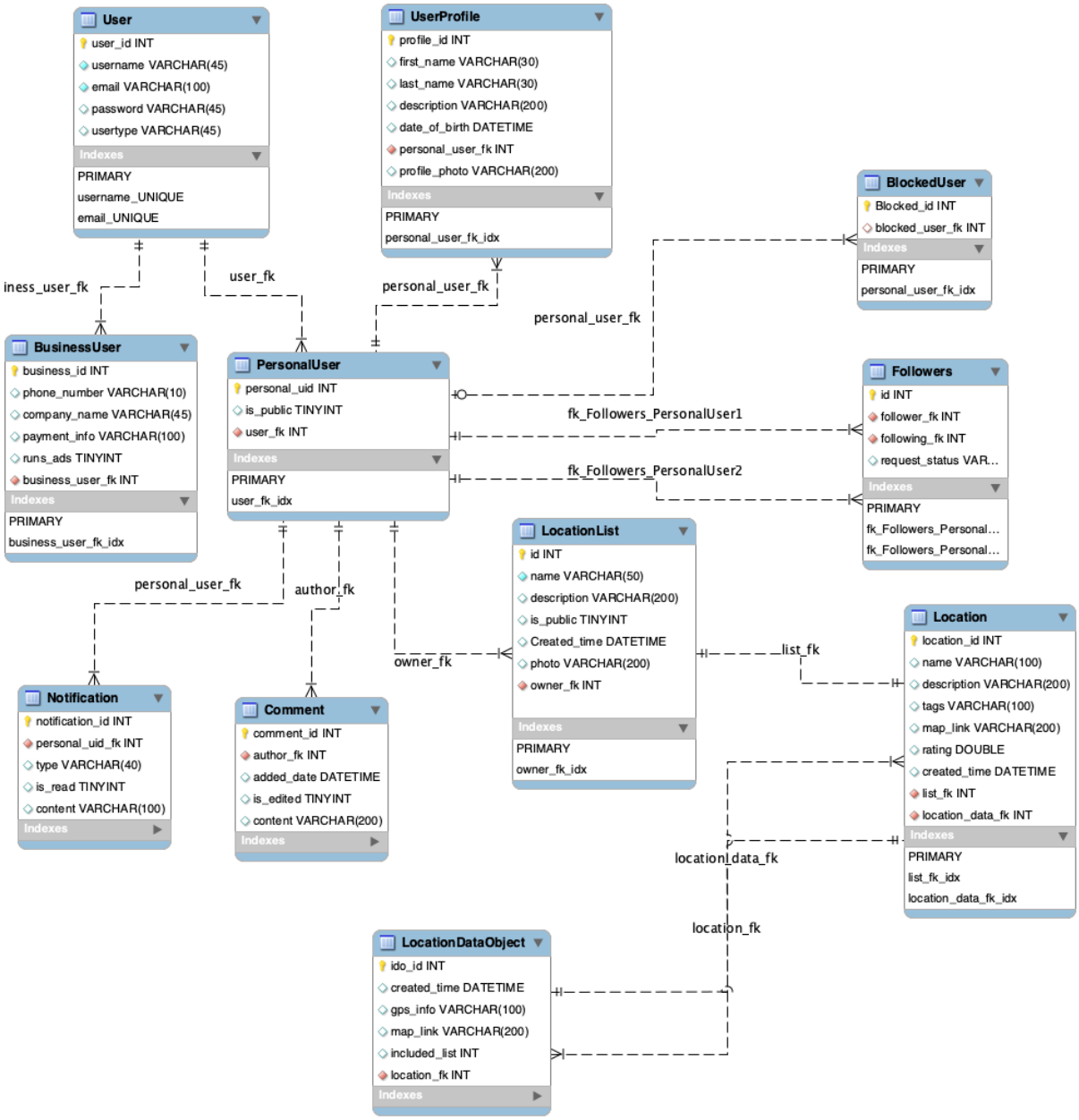
    - id: primary key, numeric
    - name: weak key, alphanumeric
    - description: weak key, alphanumeric
    - location_name: weak key, alphanumeric
    - is_public: weak key, boolean
    - created_time: weak key, alphanumeric
    - owner_fk: foreign key referencing PersonalUser entity, numeric
    - photo: unique key, alphanumeric

8. **Location** (Strong): This table stores the information about the location of the lists.
    - location_id: primary key, numeric
    - name: weak key alphanumeric
    - description: weak key alphanumeric
    - tags: weak key alphanumeric
    - map_link: weak key alphanumeric
    - rating: weak key, numeric
    - created_time: weak key, alphanumeric
    - list_fk: foreign key referencing LocationList entity, numeric
    - location_data_fk: foreign key referencing to LocationDataObject entity, numeric
9. **LocationDataObject** (Strong): This table stores the actual location data, such as GPS coordinates and Google Maps links, and is referenced by the Location table.
    - ldo_id: primary key, numeric
    - creation_date: weak key, alphanumeric
    - gps_info: weak key, alphanumeric
    - map_link: strong key, alphanumeric
    - included_list: weak key, numeric
    - location_fk: foreign key referencing Location entity, numeric
10. **Comment**(weak): This table stores the information about comments made by PersonalUsers.
    - comment_id: primary key , numeric
    - author_fk: foreign key referencing PersonalUser entity, numeric
    - added_date: weak key, alphanumeric
    - is_edited: weak key, boolean
    - content: weak key, alphanumeric
11. **Notification**(weak): This table stores the information about the notifications sent to users.
    - notification_id: primary key, numeric
    - user_fk: foreign key referencing a PersonalUser entity.
    - type: weak key, alphanumeric
    - is_read: weak key, boolean
    - content: weak key, alphanumeric

C. ER-Diagram Based on the above requirements



D. EER


User
user_id INT
username VARCHAR(45)
email VARCHAR(100)
password VARCHAR(45)
usertype VARCHAR(45)
Indexes
PRIMARY
username_UNIQUE
email_UNIQUE
UserProfile
profile_id INT
first_name VARCHAR(30)
last_name VARCHAR(30)
description VARCHAR(200)
date_of_birth DATETIME
personal_user_fk INT
profile_photo VARCHAR(200)
Indexes
PRIMARY
personal_user_fk_idx
BlockedUser
Blocked_id INT
blocked_user_fk INT
Indexes
PRIMARY
personal_user_fk_idx
iness_user_fk
user_fk
personal_user_fk
personal_user_fk
BusinessUser
business_id INT
phone_number VARCHAR(10)
company_name VARCHAR(45)
payment_info VARCHAR(100)
runs_ads TINYINT
business_user_fk INT
Indexes
PRIMARY
business_user_fk_idx
PersonalUser
personal_uid INT
is_public TINYINT
user_fk INT
Indexes
PRIMARY
user_fk_idx
fk_Followers_PersonalUser1
fk_Followers_PersonalUser2
Followers
id INT
follower_fk INT
following_fk INT
request_status VAR...
Indexes
PRIMARY
fk_Followers_Personal...
fk_Followers_Personal...
LocationList
id INT
name VARCHAR(50)
description VARCHAR(200)
is_public TINYINT
Created_time DATETIME
photo VARCHAR(200)
owner_fk INT
Indexes
PRIMARY
owner_fk_idx
personal_user_fk
author_fk
owner_fk
list_fk
Location
location_id INT
name VARCHAR(100)
description VARCHAR(200)
tags VARCHAR(100)
map_link VARCHAR(200)
rating DOUBLE
created_time DATETIME
list_fk INT
location_data_fk INT
Indexes
PRIMARY
list_fk_idx
location_data_fk_idx
Notification
notification_id INT
personal_uid_fk INT
type VARCHAR(40)
is_read TINYINT
content VARCHAR(100)
Indexes
Comment
comment_id INT
author_fk INT
added_date DATETIME
is_edited TINYINT
content VARCHAR(200)
Indexes
location_data_fk
location_fk
LocationDataObject
ido_id INT
created_time DATETIME
gps_info VARCHAR(100)
map_link VARCHAR(200)
included_list INT
location_fk INT
Indexes

E. Defining DBMS

We opted for MySQL as the DBMS to build our database because Amazon RDS offers a free tier for MySQL instances, which helps us save costs. Additionally, MySQL is a well-known and widely used open-source DBMS that supports relational databases.

F. Media Storage

The images and thumbnails in our application will be stored to AWS S3 cloud service and it will produce a URL for the image and the URL will be stored in the database as VARCHAR(200).

G. Search & Filter Algorithm in

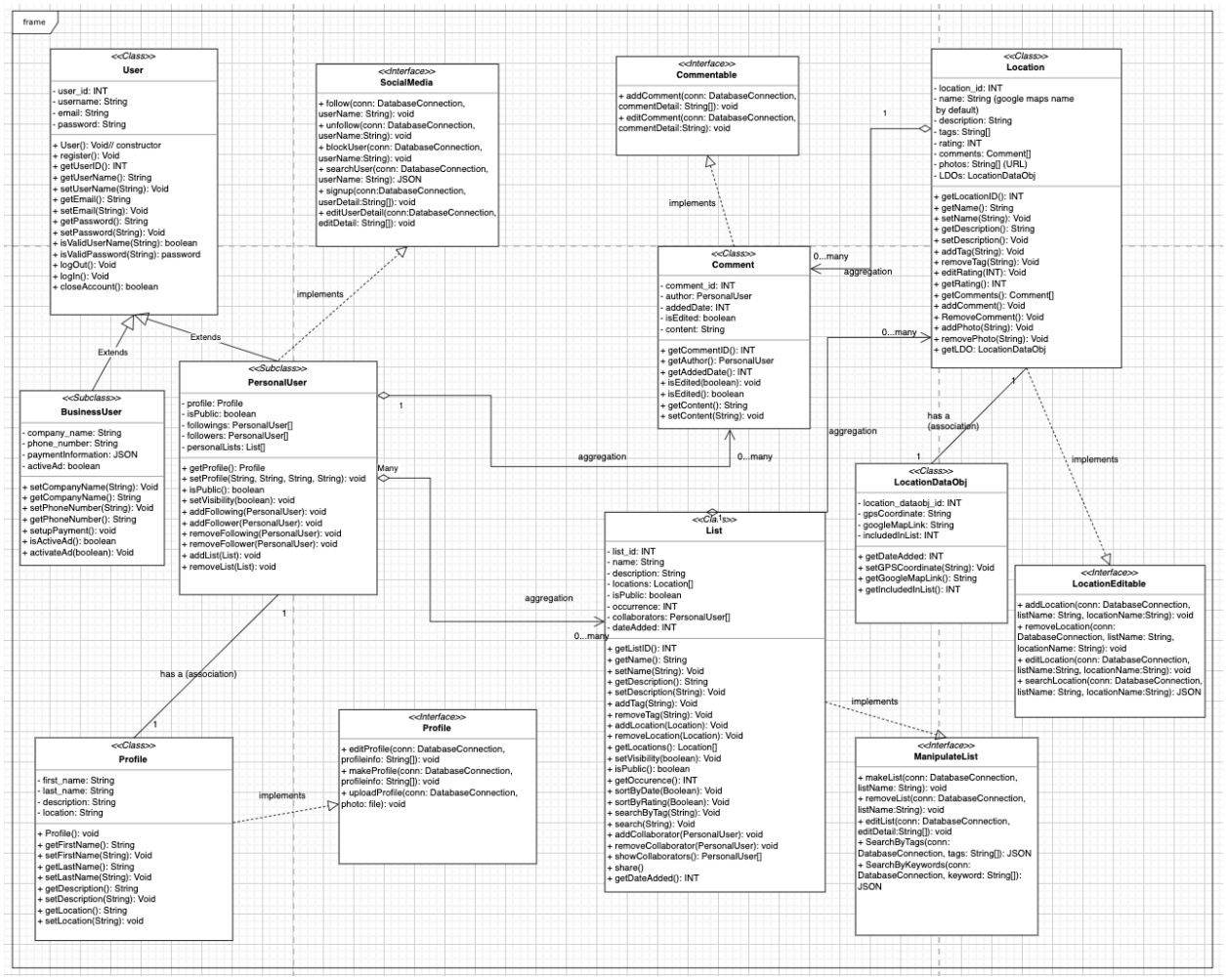
Our search/filter architecture and implementation prioritize searching for users and location lists based on specific criteria. For the GET /users and POST /searchUser endpoints, the search algorithm is a straightforward SELECT query with a WHERE clause that filters the results based on the user's name. As for the GET /LocationList endpoint, the search algorithm involves a SELECT query that filters the location lists based on both the LocationListName and LocationName attributes.

To facilitate the search, we created a LocationList entity in the database with the LocationName and LocationListName attributes. We use the %like operator in the SELECT query to match any characters before or after the search Location. The query selects all the lists from the LocationList table that have at least one list item with a LocationName that matches the search term LocationName.

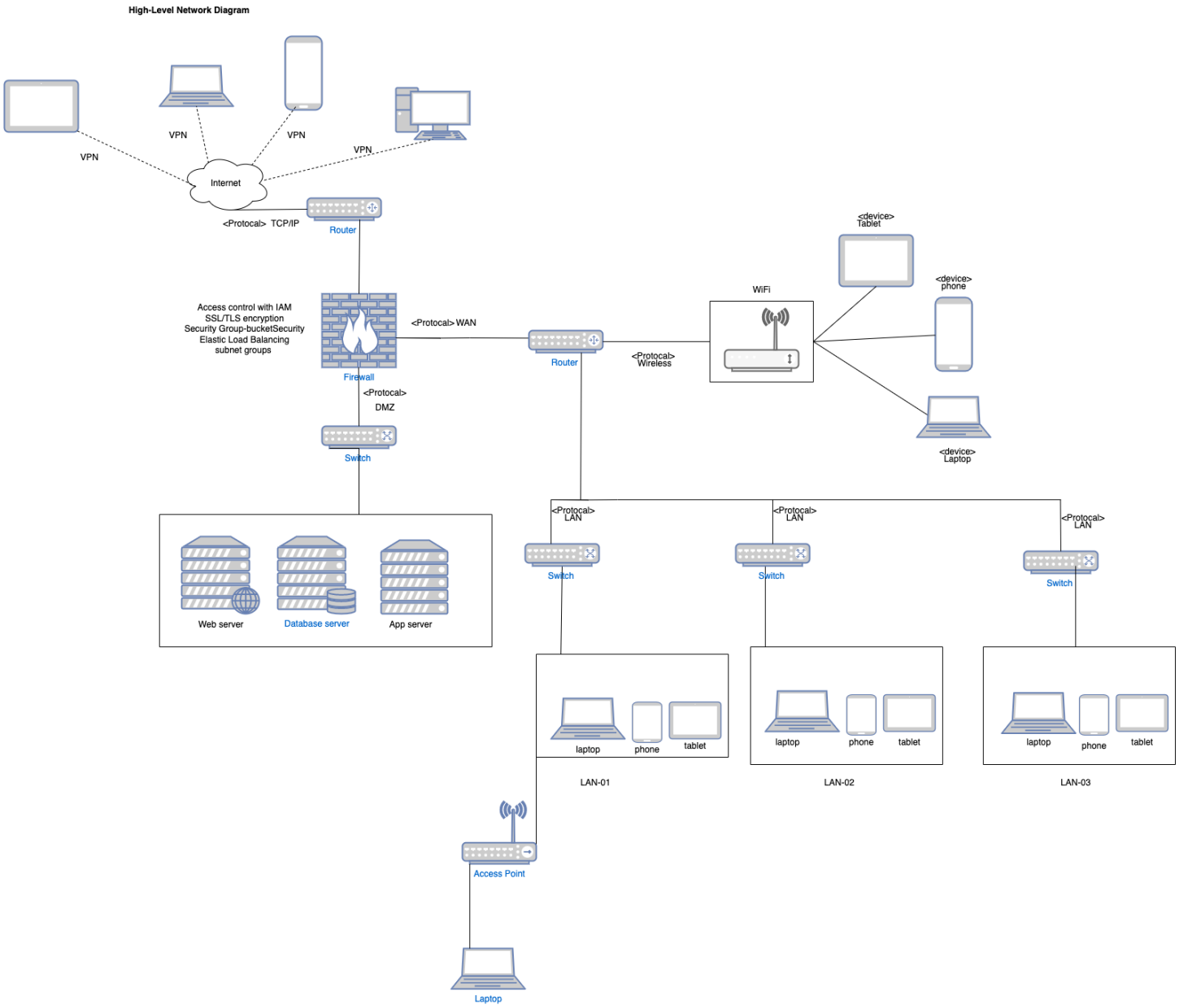
Overall, the search/filter implementation in the code is based on query parameters and SQL queries with WHERE and LIKE clauses to filter the results based on username and LocationList information.

5. High Level Diagram

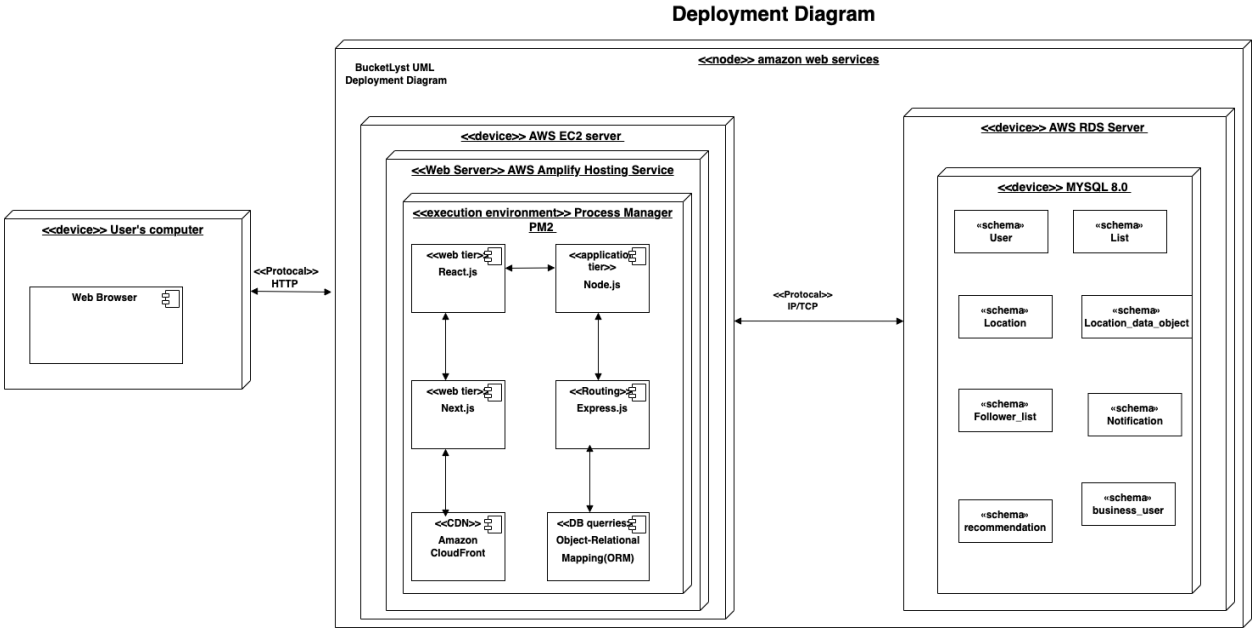
1. UML



2. High-level Network Diagram



3. Deployment Diagram



6. Team Contribution

Name	Role	Contribution
Arielle Riray 10/10	Frontend Lead	- Made and Designed the Figma frames/components for rest of frontend team to make use case - Coded majority of UI Horizontal prototype from scratch (Couldn't export it from Figma) - -Implemented most of the P1 functionalities into the Horizontal UI - Established codes/stylesheets to make it more easier for other frontend members to contribute to the components, and delegated those tasks
Francis Quang 9/10	Backend Engineer	- Updated data definitions - Worked on makeList API - Added APIs: /displaylist/{list id}, /viewprofile/{user id}, /followuser/{user id}, /blockuser/{user id}
Tommy Truong 10/10	Document Editor	- Edited M3 - Finished up Your Feed, Your Lists, Following front end web pages - Finished up Sort by Filter - Finished up wireframes for 9 and 10
Kenneth Gee 10/10	Frontend Engineer	- Helped to edit ERD diagram - Touched up Functional Requirements - Finished up Login and Membership web pages
Rabin Karki 10/10	Database Master	- Worked on ERD and EER diagrams. - Re-designed db. - Worked on High-level Network Diagram - Worked on wireframes for use case 1 to 8.
Aaron Kuo 10/10	Github Master/ Backend Lead	- Reviewed high level diagrams - Reviewed UML diagram from m2 - Worked on s3 bucket upload api and link retrieval - Directed back end meetings to provide direction - Met with front end lead to discuss application specifics - Designed general outlines for APIs - Created API for image upload