SW Engineering CSC 648-848 Fall2023

SFStudentRent.com

Team03

Milestone 2: a) More Detailed Requirements, Specs, Architecture, UI mock-ups and b) vertical SW prototype

Version 2.0

Team Member	Roles
Jeremy Tran	Team Lead / Github Master /jtran43@mail.sfsu.edu
Geovanni Valadez	Front End Lead gvaladez@mail.sfsu.edu
Mozhgan Ahsant	Scrum Master mahsant1@mail.sfsu.edu
Anthony Silva	Database Lead asilva32@mail.sfsu.edu
Aman Khera	Back End Lead akhera@mail.sfsu.edu
Ivan Ayala-Brito	Documentation Master iayalabrito@mail.sfsu.edu
Daniel Enriquez	Front End Engineer denriquez@mail.sfsu.edu
Alex Huang	Back End Engineer xhuang20@mail.sfsu.edu

History Table

Milestone Version	Date Submitted
M1V1	9/21/23
M1V2	10/12/23
M2V1	10/12/23
M2V2	11/2/23

Table Of Contents

1. Data Definitions

1.User

- Unregistered: Limited to view-only access.
- Key Attitributes: N/A

2. Background Check

- Assesses registered user's financial and rental history.
- Key Attributes: Check_ID, Status

3. Registered User

- Full site access with functionalities like posting, messaging and joining groups.
- Key Attributes: User_ID, Password, Check_ID

4. Location of Rental Listings

- Precise location for rental listings.
- Key Attributes: Location_ID, Listing_ID

5. Region

- General location of the rental listing such as neighborhood.
- Key Attributes:Region_ID, Location_ID

6. Rental Listings

- A post that contains media, images or videos along with information regarding the property.
- Key Attributes: Listing_ID, User_ID

7. Posts

- Generic posts with or without media.
- Key Attributes: Post_ID, User_ID, Comment_ID

8. Messages

- Private communication between registered users.
- Key Attributes: Message_ID, User_ID(to), User_ID(from)

9. Notification

- Updates for users about thei account activities.
- Key attributes: Notification_ID, Notification_Type

10. Events

- User-created events either open or by invitation.
- Key attributes: Event_ID, User_ID(Host)

11. Friend Request

- Registered users can add other registered users to their friend list.
- Key Attributes: Request_ID, User_ID(to), User_ID(from)

12. Saved Properties List

- Registered users can save preferred rental listings.
- Key Attributes: List_ID, Saved_ID

13. Housing Cost Calculator

- Provides rental/buying suggestions based on input.
- Key Attributes: N/A, info not saved

14. Groups

- For group communications and activities.
- Key Attributes: Group_ID, User_ID

15. Reports

- Allows for users to report rental listings.
- Key Attributes: Report_Id, User_ID

2. Prioritized Functional Requirements

Priority 1:

1. **User:**

- (1.1) Users shall be able to register an account using a unique email and set up a password.

- (1.2) Users shall be able to browse the site without an account but will not be able to interact with posts or other users.

- (1.3) A user can reset their password if forgotten.

- (1.4) The system should verify user identities securely through email or phone number verification.

2. Background Check Integration

- (2.1) All registered users that wish to rent or rent out their place shall be subject to a background check.

- (2.3) The system shall integrate with predefined screening criteria and business rules, allowing property managers to set specific eligibility criteria based on factors such as credit score, criminal history, rental history, and income.

- (2.2) Registered users can open the status of the background check.

3. Registered User

- (3.1) Registered users shall be given the option to make posts on the site.

- (3.2) A Registered User shall be able to link their social media accounts such as Instagram and Twitter.

4. Location of Rental Listing

- (5.1) Locations shall include campus area, neighborhood, and proximity to public transport.

- (5.2) Users should be able to filter and view listings by clicking on map markers or drawing custom search areas.

- (5.5) Rental listings shall be separated into regions, where users can view by clicking interactable map region.

5. **Rental Listing:**

- (6.1) Landlord accounts shall be able to create, update, or delete their rental listings, specifying details such as the number of bedrooms, bathrooms, etc.

- (6.2) Landlord accounts shall be given the permission to add rental listing to the site.

6. **Housing Cost Calculator:**

- (13.1) Registered Users can generate approximate costs for chosen intervals and forecasts for the property.
-

7. **Posts:**

- (7.2) A post shall be able to contain text, multimedia content (photos, links), and other metadata.

- (7.3) Posts shall allow comments, each with a unique commentID and creation timestamp.

- (7.1) Posts shall be only edited by the author.

Priority 2:

1. **Registered User:**

- (3.2) Registered users shall save rental listings to their saved properties list.

- (3.7) A Registered User shall be able to form groups of other registered users they wish to house with.

2. **Rental Listing:**

- (6.3) Rental Listing shall have a minimum amount of information and media needed for posting.

3. **Messages:**

- (8.1) Registered users shall be able to initiate private messages with one other registered user.

4. **Saved Properties List:**

- (12.1) Users can create lists of saved properties.

- (12.2) Each saved list shall have a unique ListID.

- (12.3) Properties List shall be under a registered user and not viewable to other non-registered or registered users.

- (12.4) Properties list shall group properties by similarities or by a key word as desired by a registered user.

5. **Groups:**

- (14.2) A group shall share a saved properties list and all receive notifications for a property.

6. **Reports:**

- (15.1) Registered users can generate reports on rental listings, user activity, and feedback.

7. **Housing Cost Calculator:**

- (13.2) The Calculator shall generate a curated list of houses best fit for a registered user based on filters and location.

8. **Reports:**

- (15.2) The system shall provide functionality to generate various types of rental reports, including rental transaction summaries, rental history for specific customers, and rental equipment utilization reports.

- (15.3) Users shall be able to customize the content and format of reports, including selecting specific date ranges, sorting options, and the inclusion/exclusion of specific rental data field, such as rental duration, customer information, and equipment details.

Priority 3:

1. **Registered User:**

- (3.3) Registered users shall send and receive friend requests.

2. **Location of Rental Listing:**

- (5.3) Users may opt to receive notifications when new rental listings by distance from a specified location (e.g., within 5 miles of a particular address or point of interest).

- (5.4) The system shall provide autocomplete suggestions to assist users in specifying the location.

3. **Posts:**

- (7.5) A post shall be able to get pinned by the creator with a maximum of three for his or her account.

- (7.6) A post shall be able to get promoted by the creator at least twice with the boost lasting two days.

4. **Messages:**

- (8.4) The system should allow users to organize their messages, including archiving or deleting old messages.

- (8.5) Users should receive real-time notifications (e.g., push notifications or email notifications) when they receive a new message or when there are updates to existing messages.

5. **Notification:**

- (9.2) Registered users shall receive a notification upon receiving a new message.

- (9.3) A notification shall be able to be turned off or on by a register user or group.

6. **Events:**

- (10.1) Registered users shall be able to set up events (open house, etc) for other users to see.

- (10.2) An event shall have its own location, date, eventID, and a list of confirmed attendees.

- (10.3) An event shall only be made by verified registered users and have an expected list of attendees.

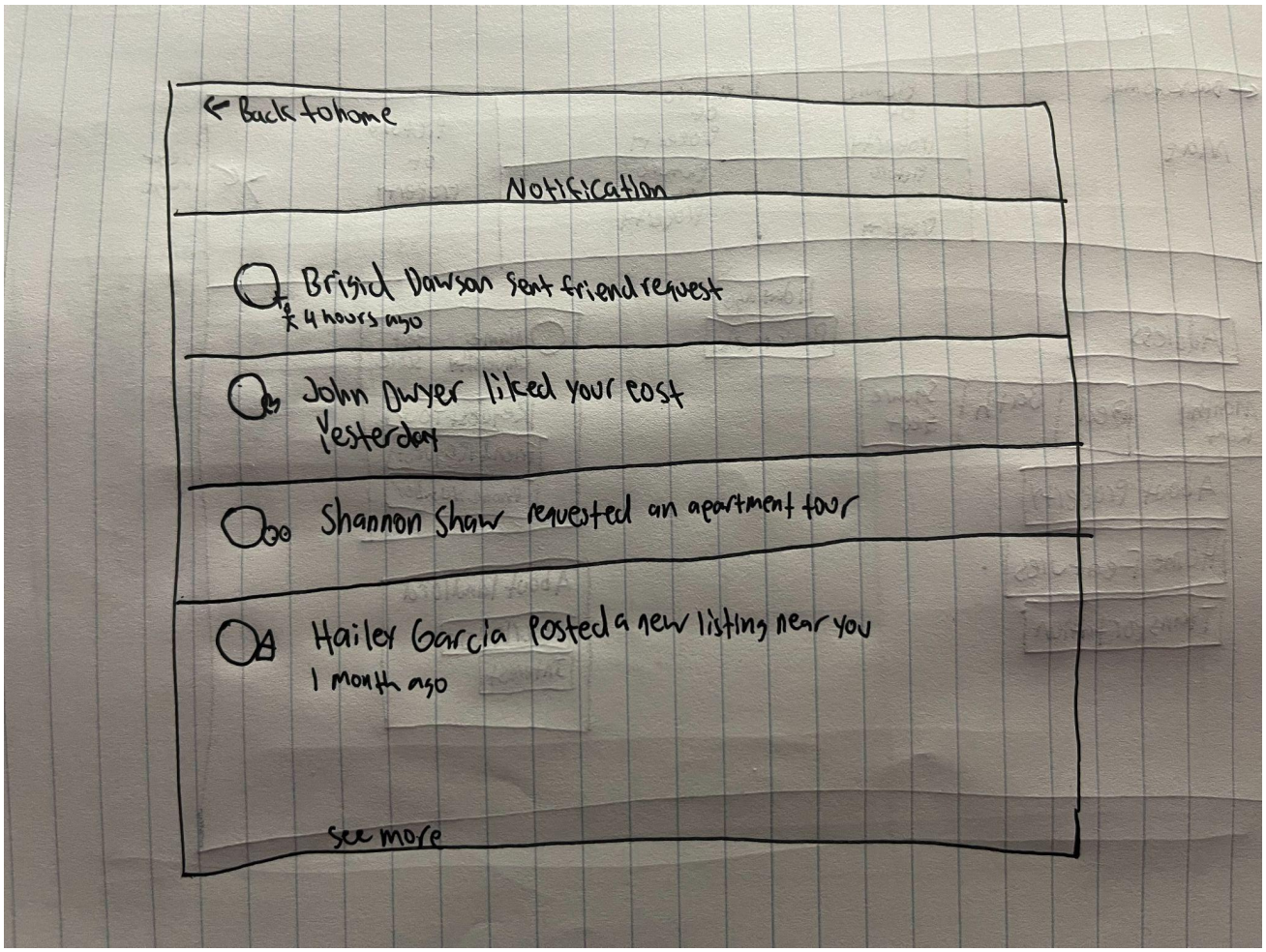
7. **Friend Request:**

- (11.3) Register users sending the friend request shall allow permission changes for the requesting friend to view content

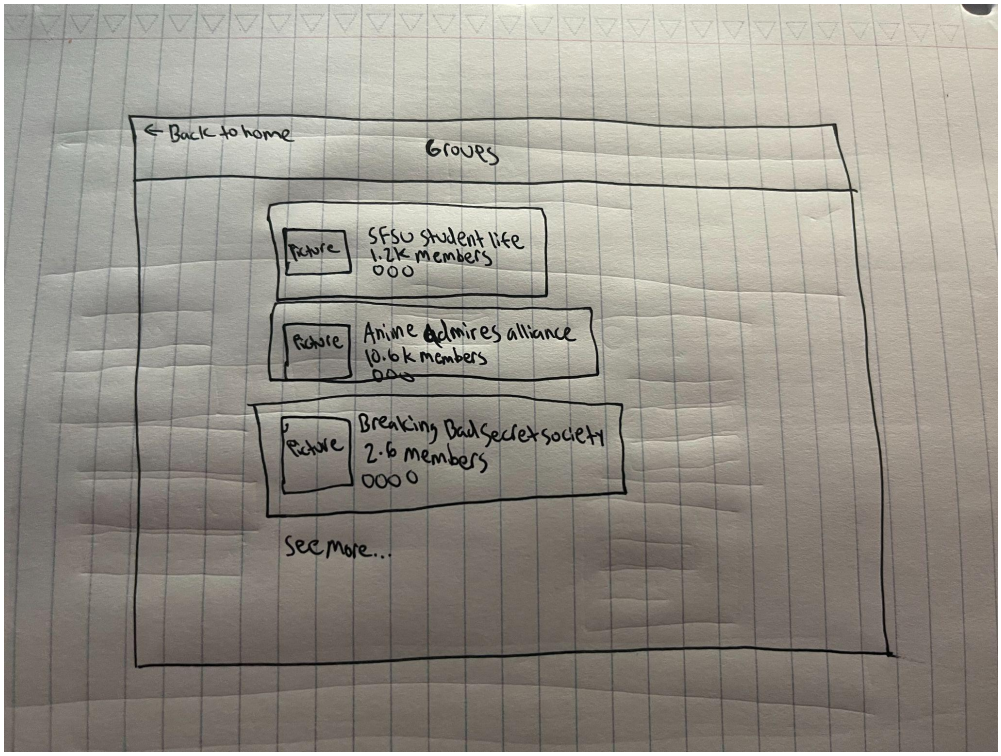
3. UI Mockups and Storyboards (high level only)

MOCKUPS:

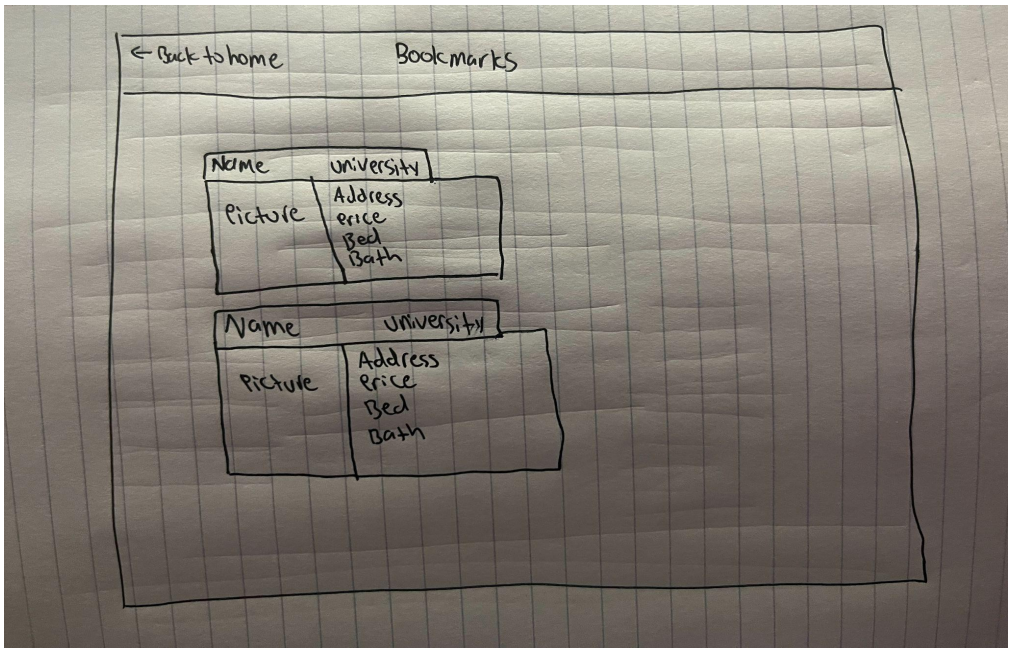
Notification List:



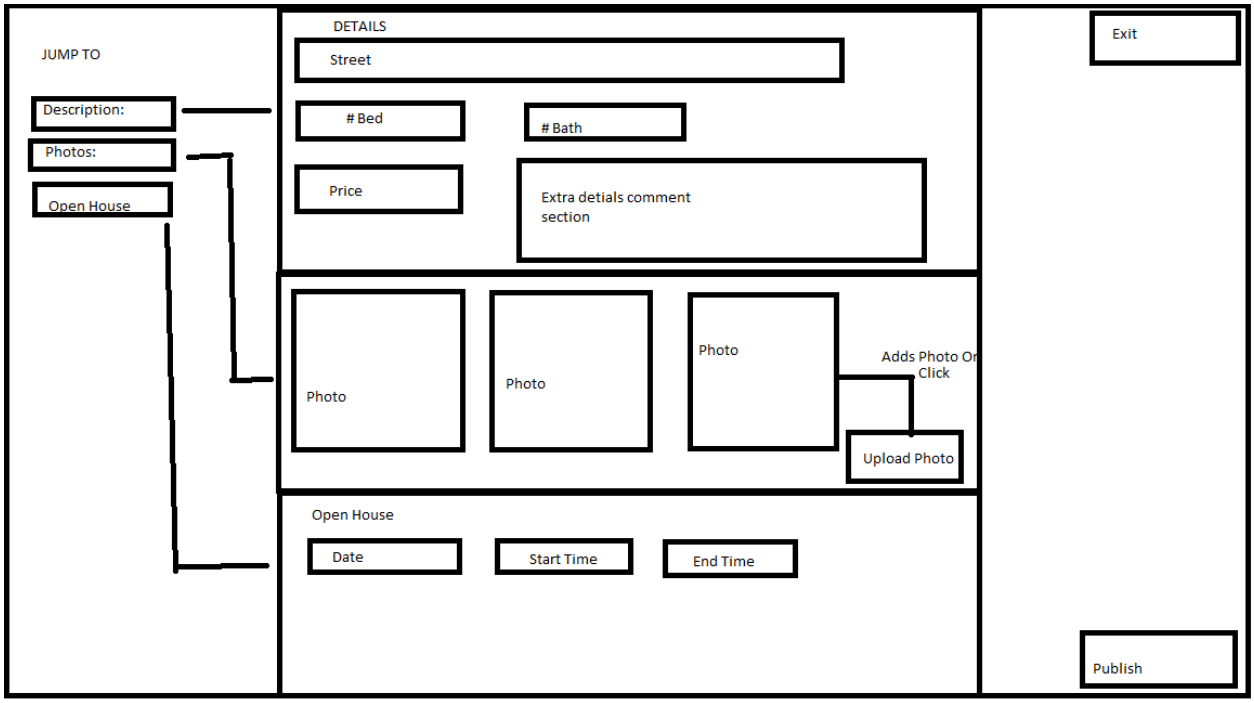
Groups:



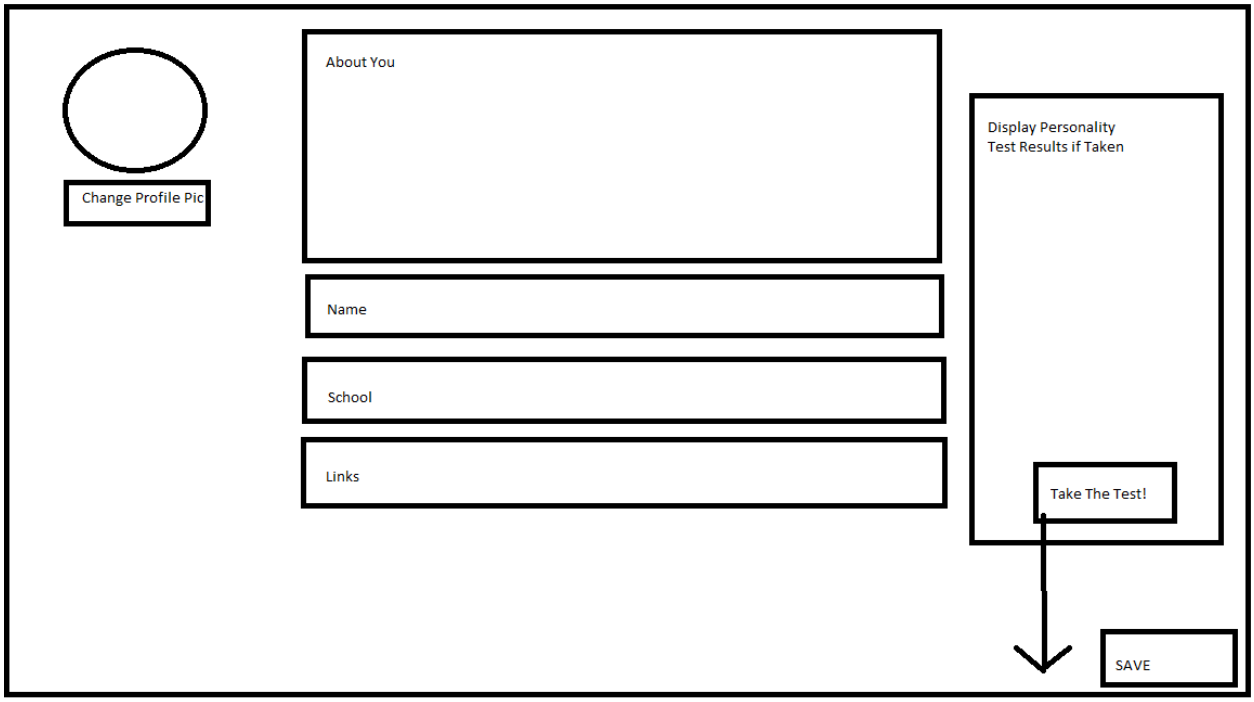
Bookmark:



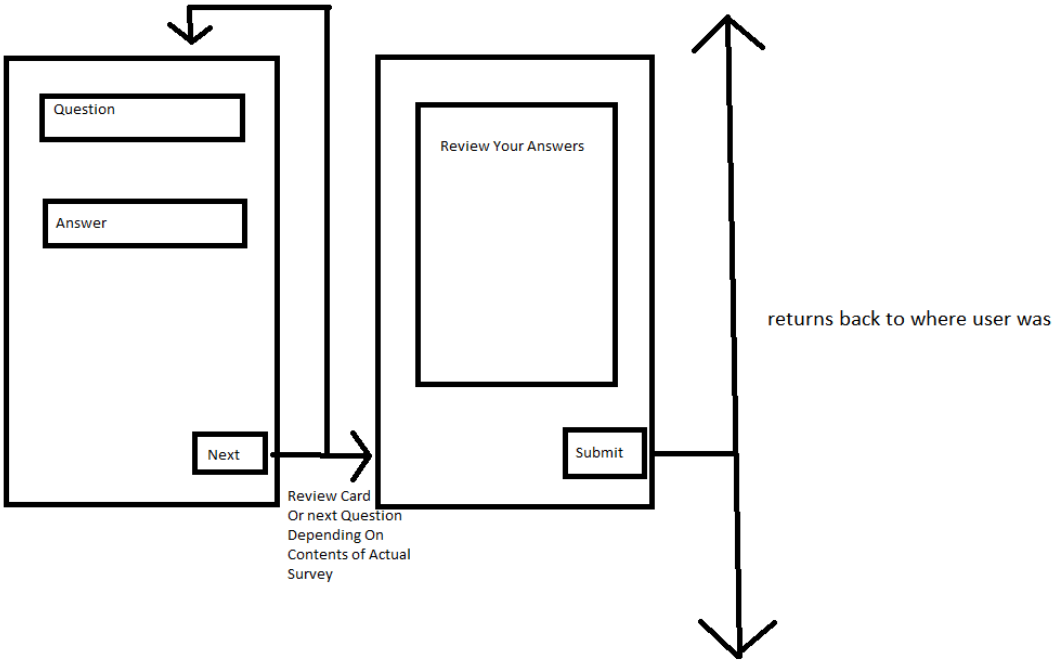
Edit Listing:



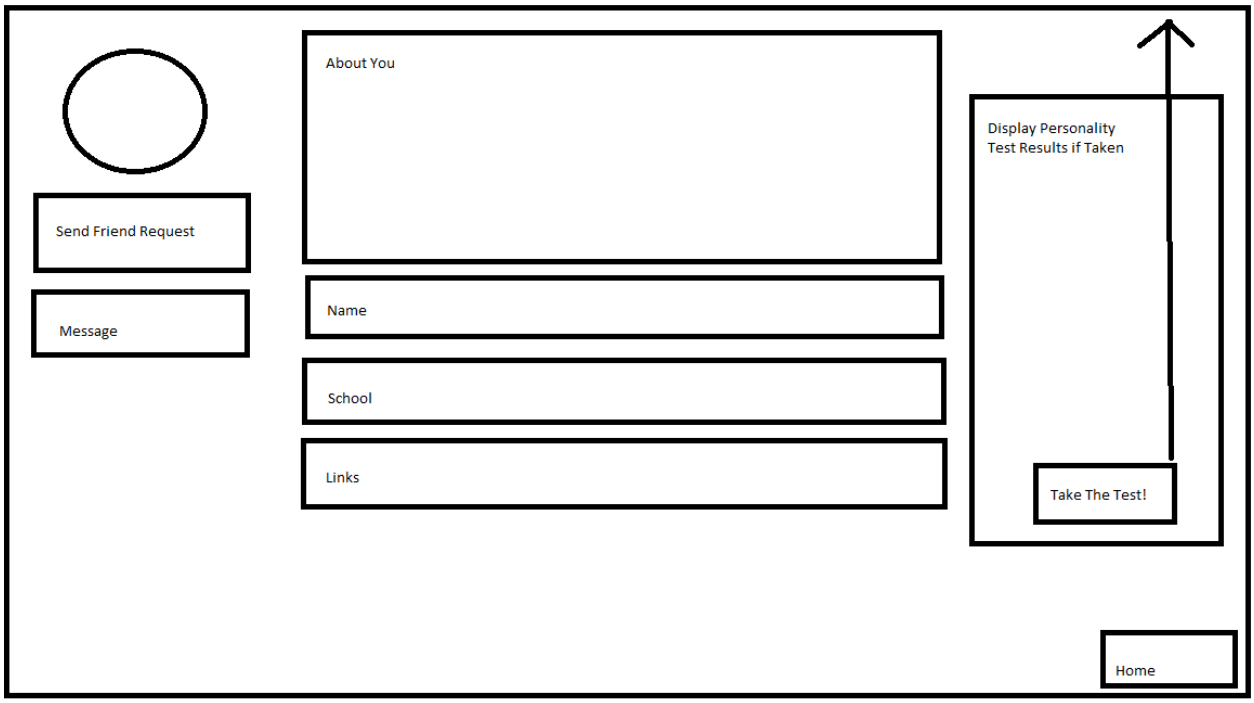
Edit Profile:



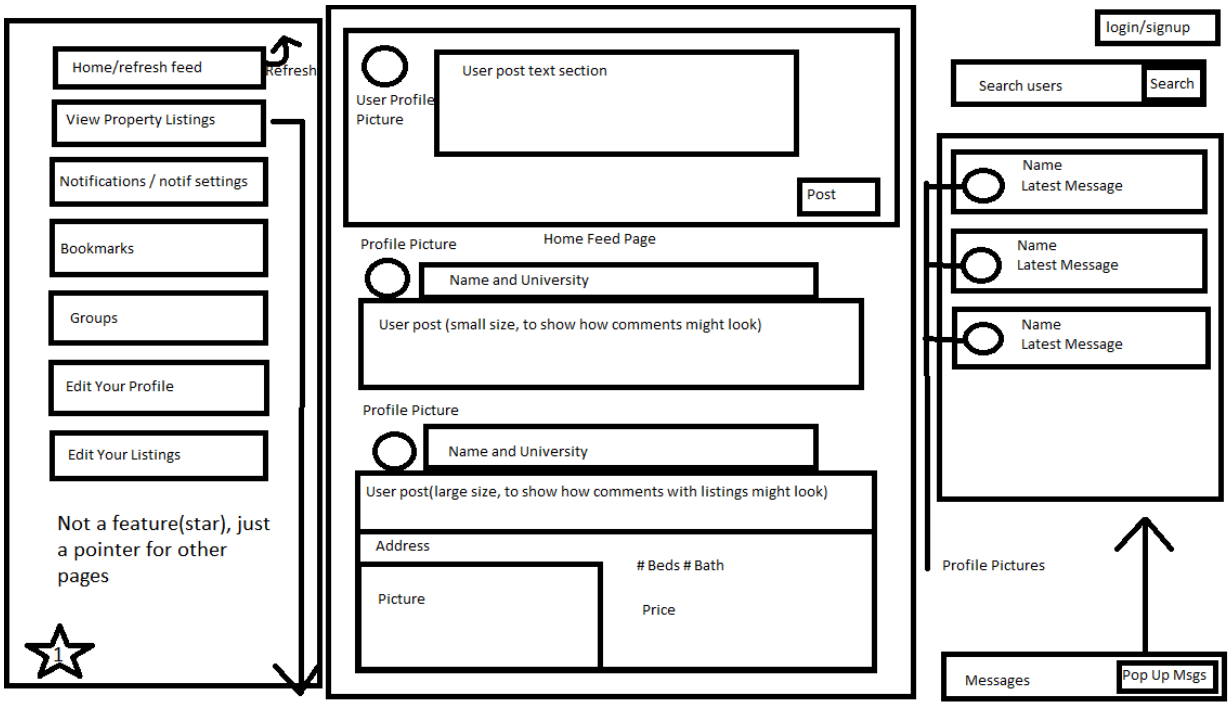
Personality Test:



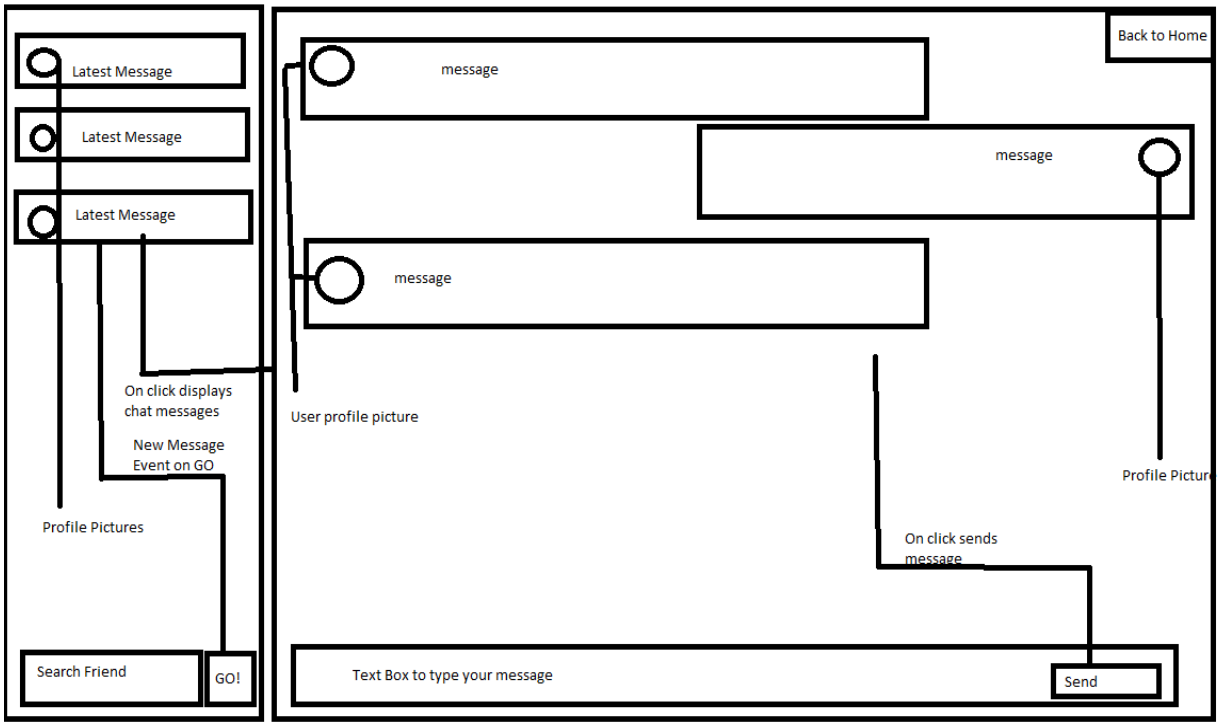
View Profile:



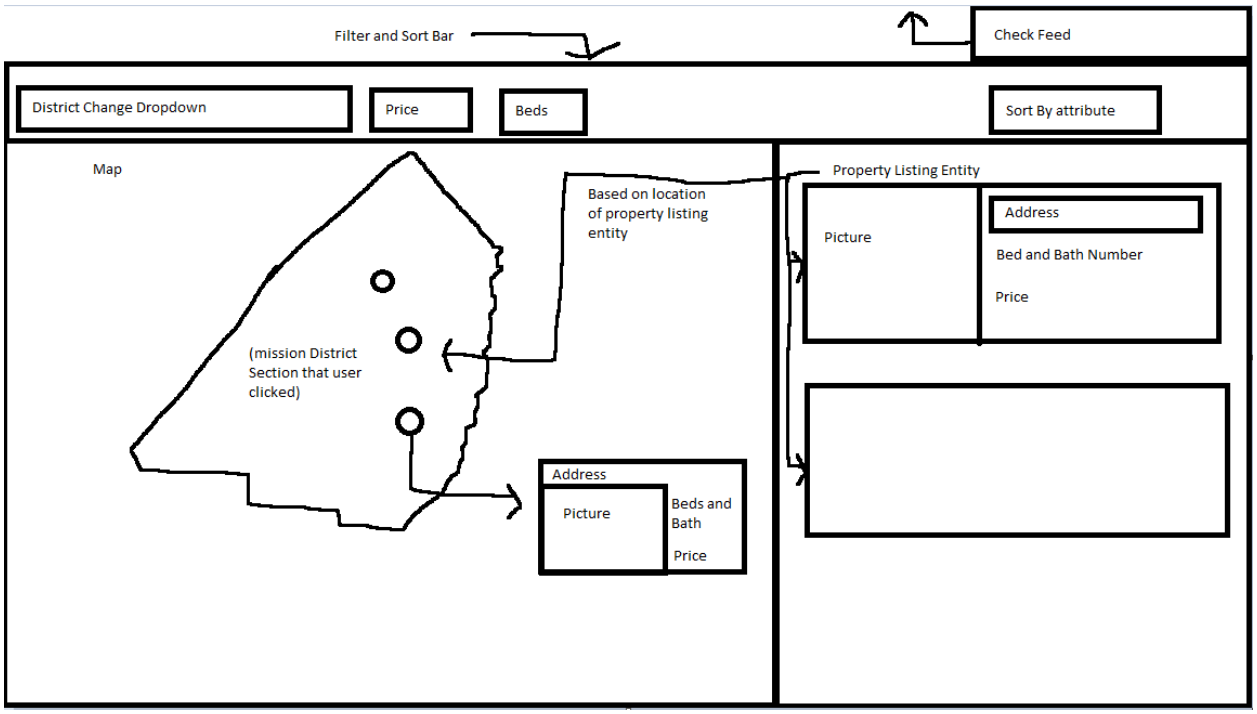
Social Page:



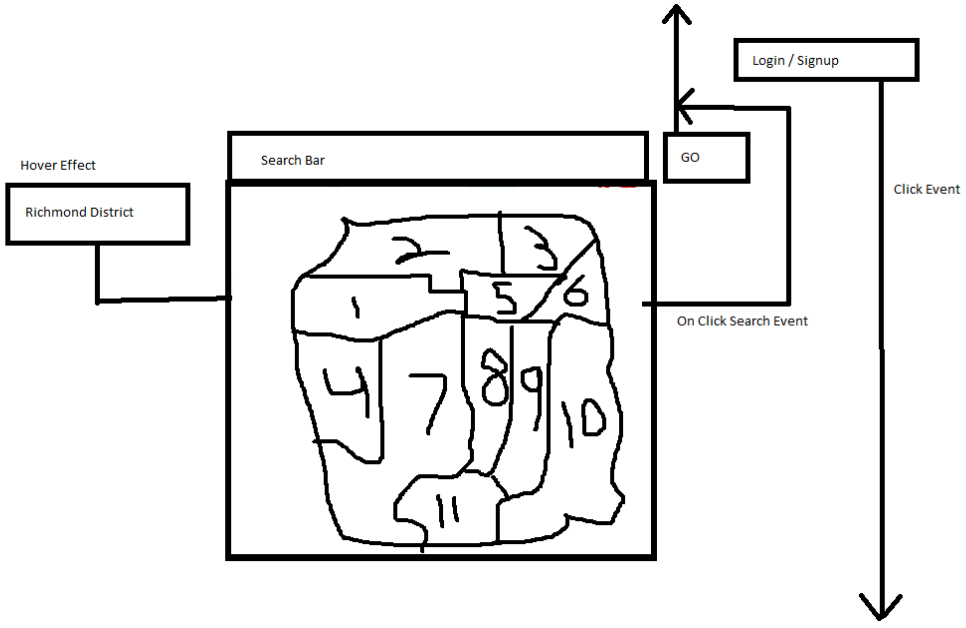
Messages:



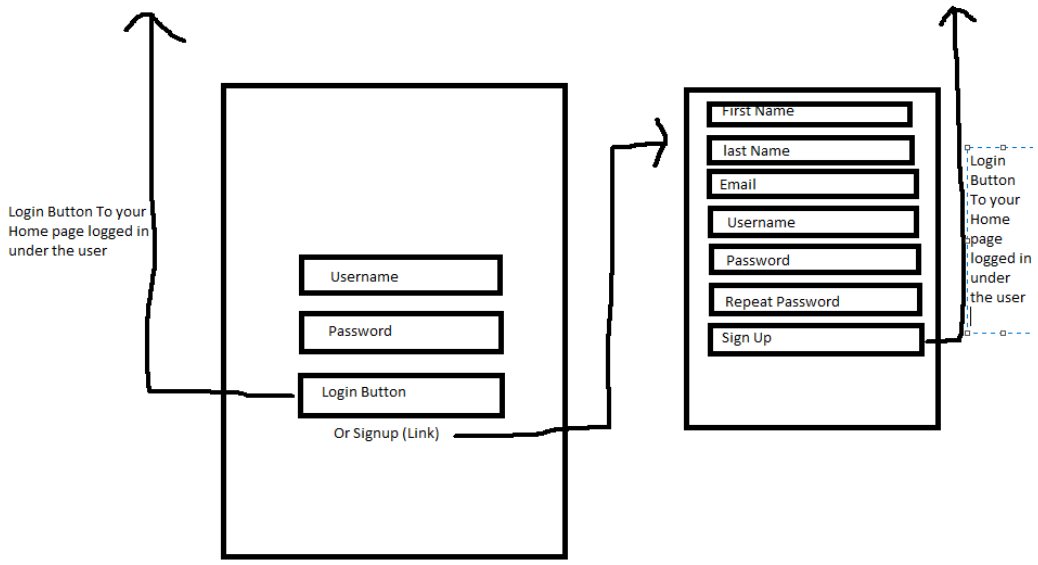
Property Listing:



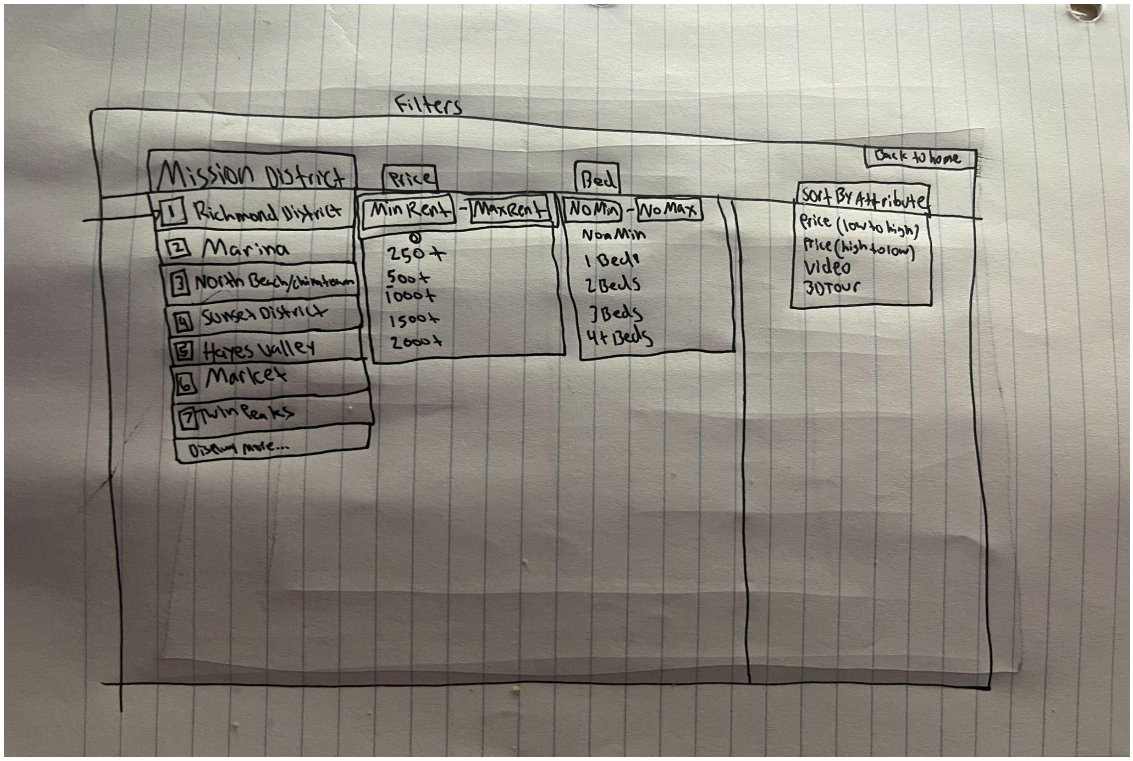
Main Search:



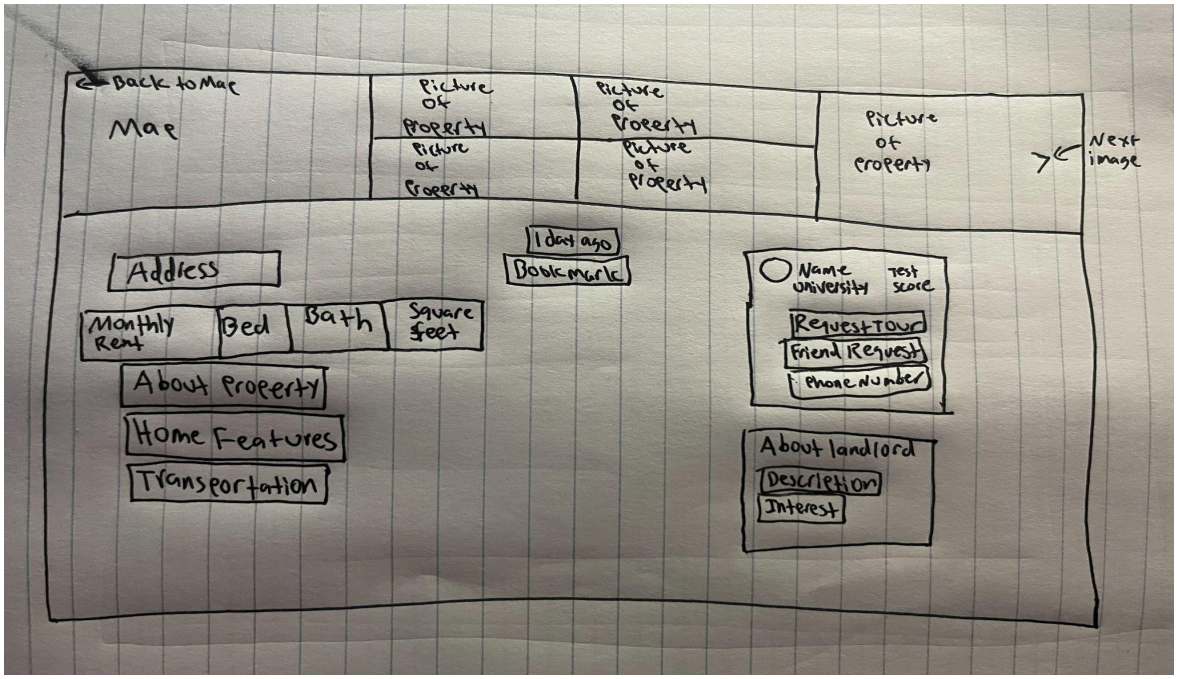
Login/Sign up Page:



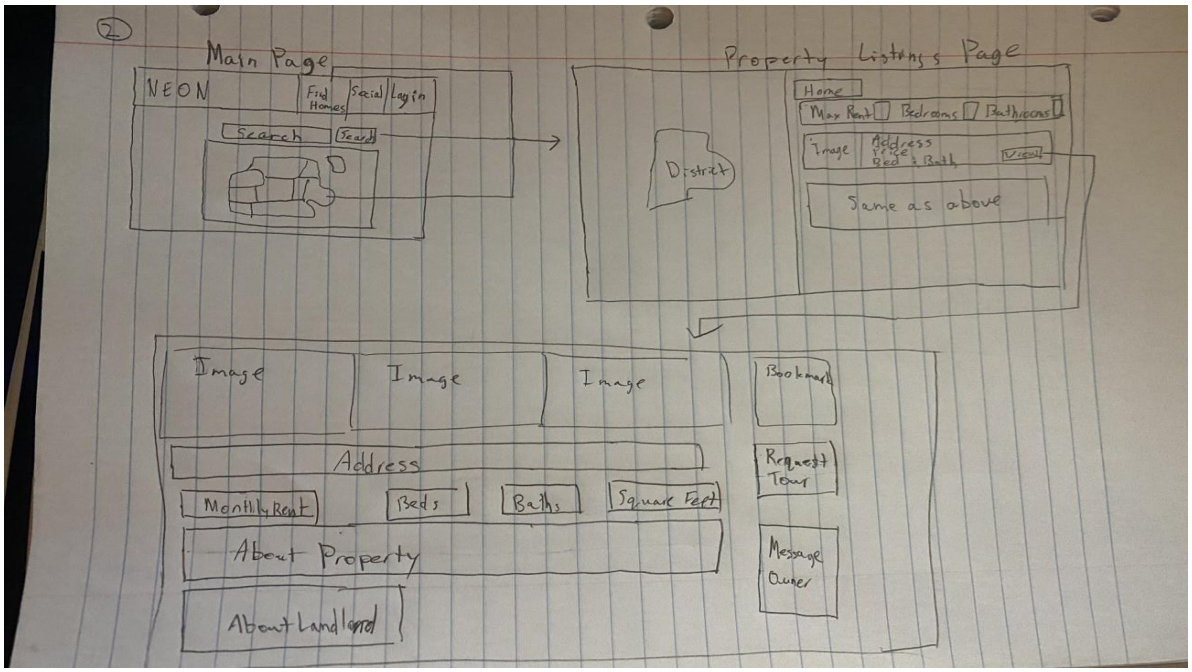
Filters:

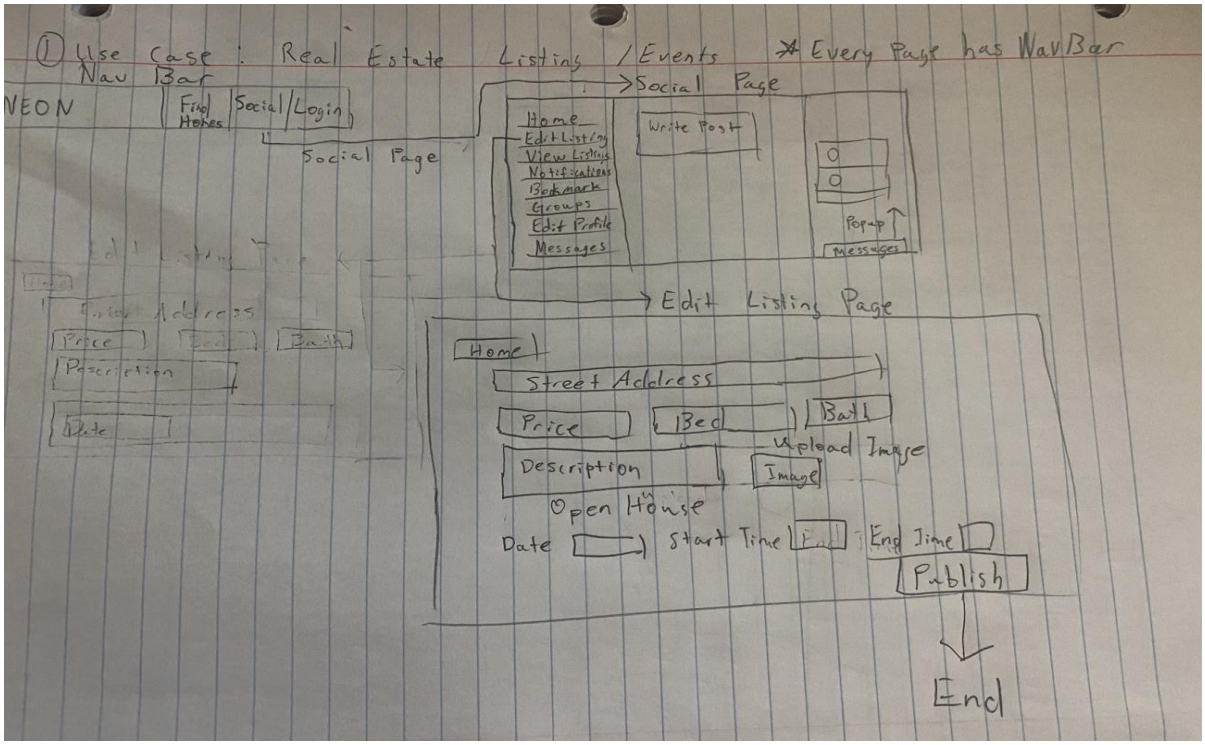


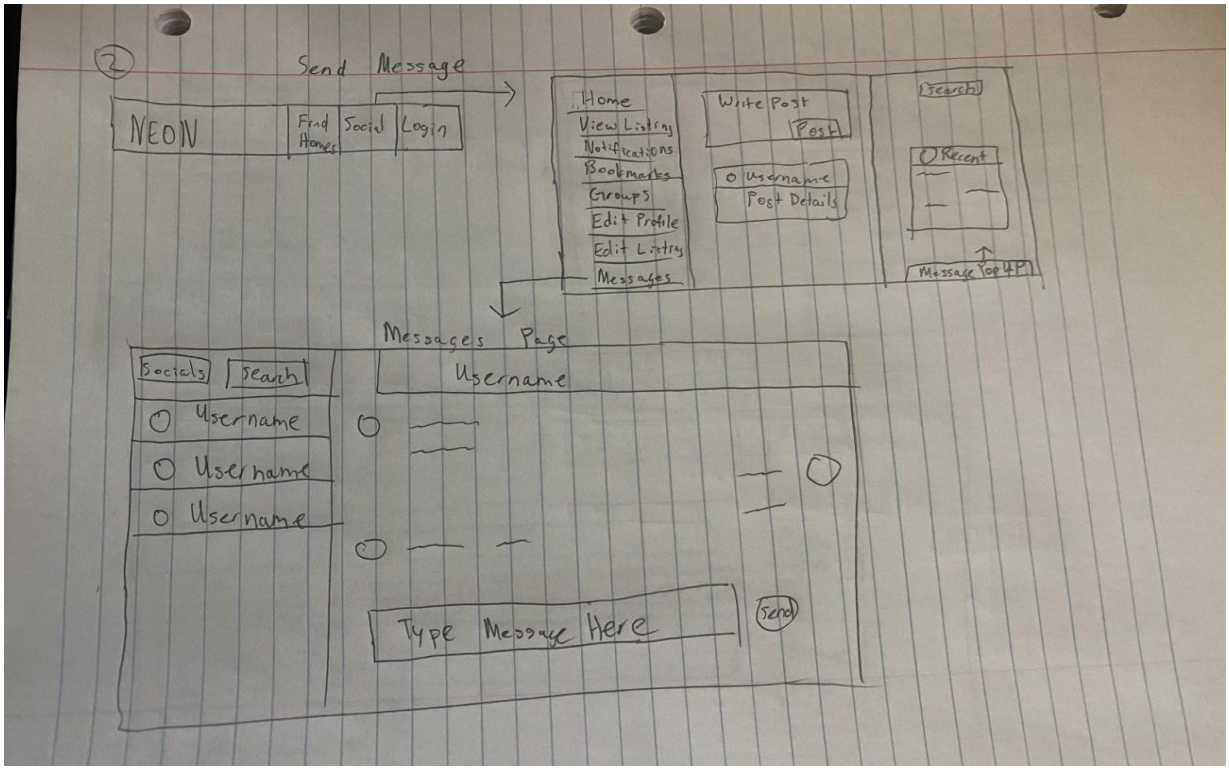
Displaying Property Listing:

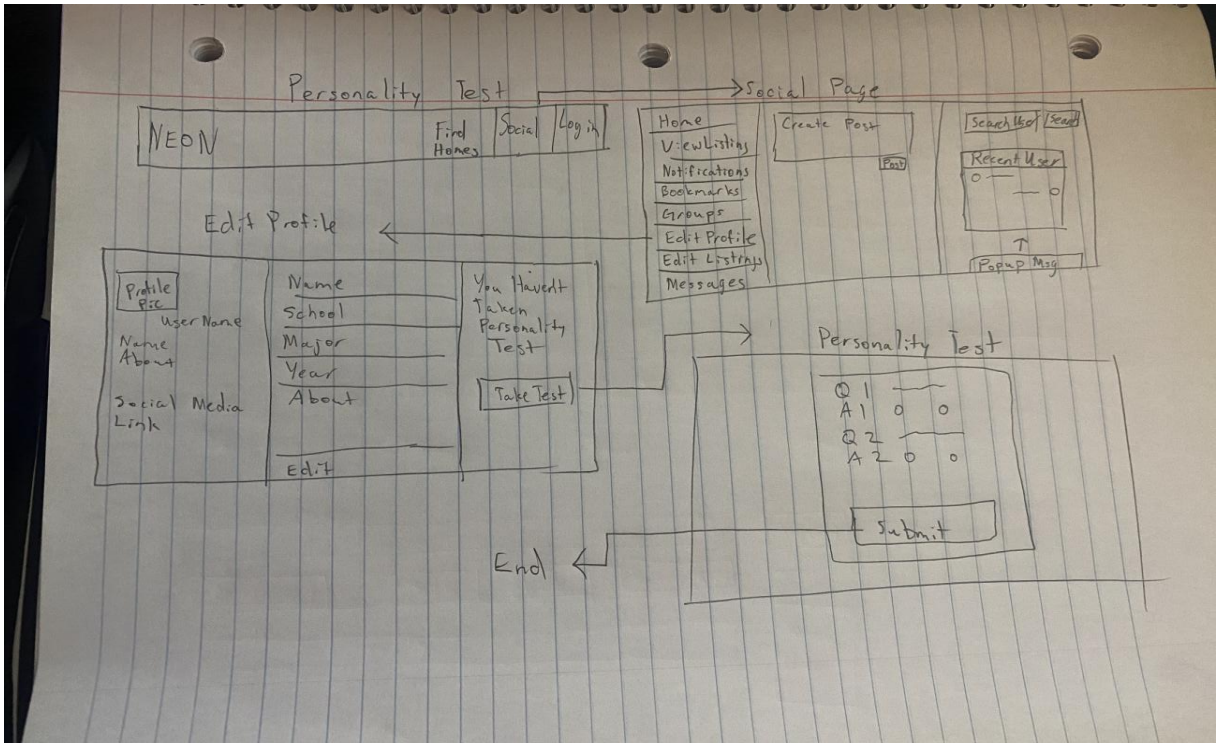


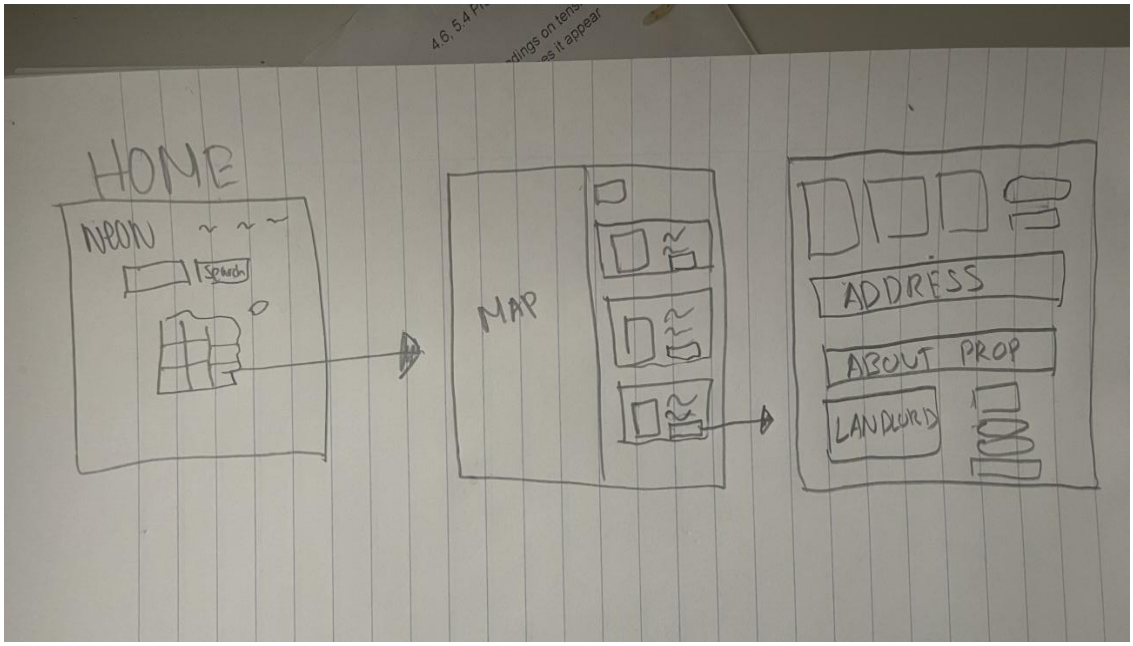
Storyboards:

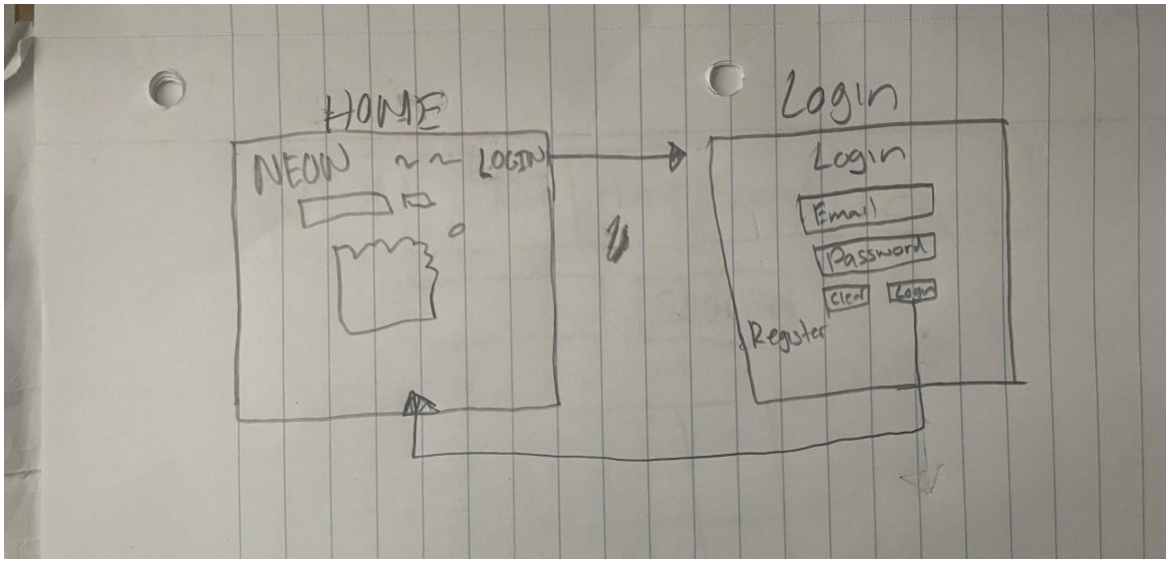

2
Main Page
NEON
Find Homes
Social
Login
Search
Search
Property Listings Page
Home
Max Rent
Bedrooms
Bathrooms
Image
Address
Bed & Bath
View
District
Same as above
Image
Image
Image
Bookmark
Address
Monthly Rent
Beds
Baths
Square Feet
Request Tour
About Property
About Landlord
Message Owner

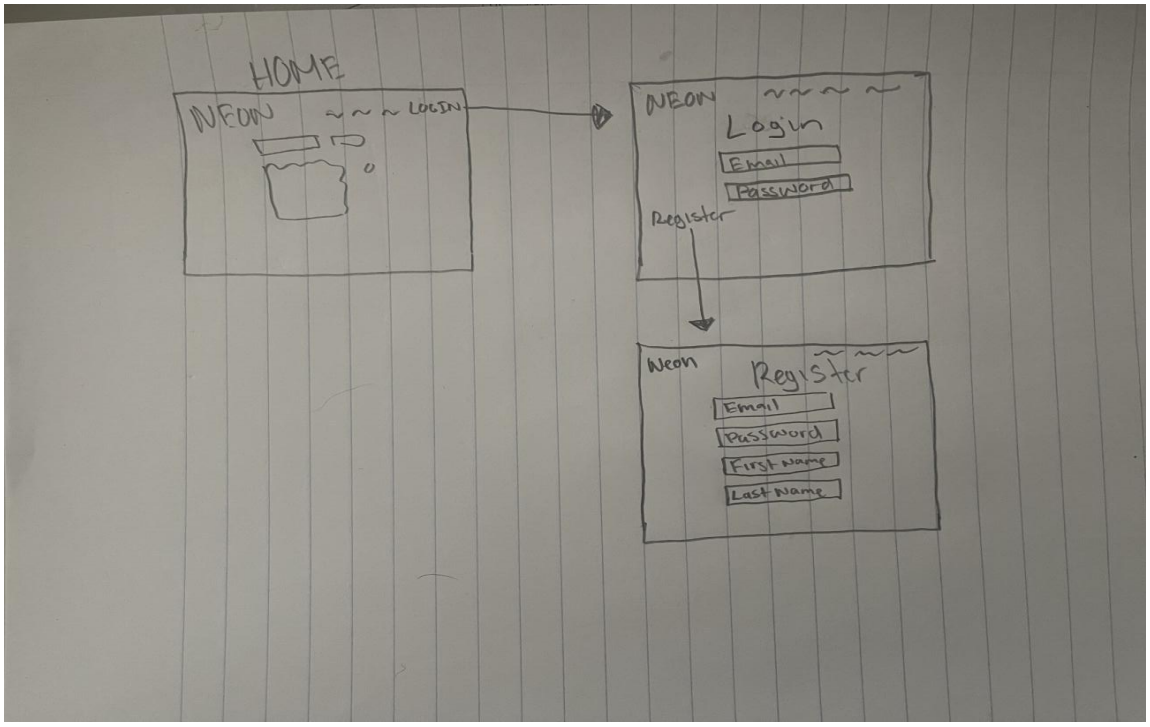

1 Use Case: Real Estate Listing / Events
Every Page has NavBar
Nav Bar
NEON
Find Homes
Social
Login
Social Page
Social Page
Home
Edit Listing
View Listings
Notifications
Bookmark
Groups
Edit Profile
Messages
Write Post
Pop-up
Messages
Edit Listing Page
Home
Street Address
Price
Bed
Bath
Description
Upload Image
Image
Open House
Date
Start Time
End Time
Publish
End

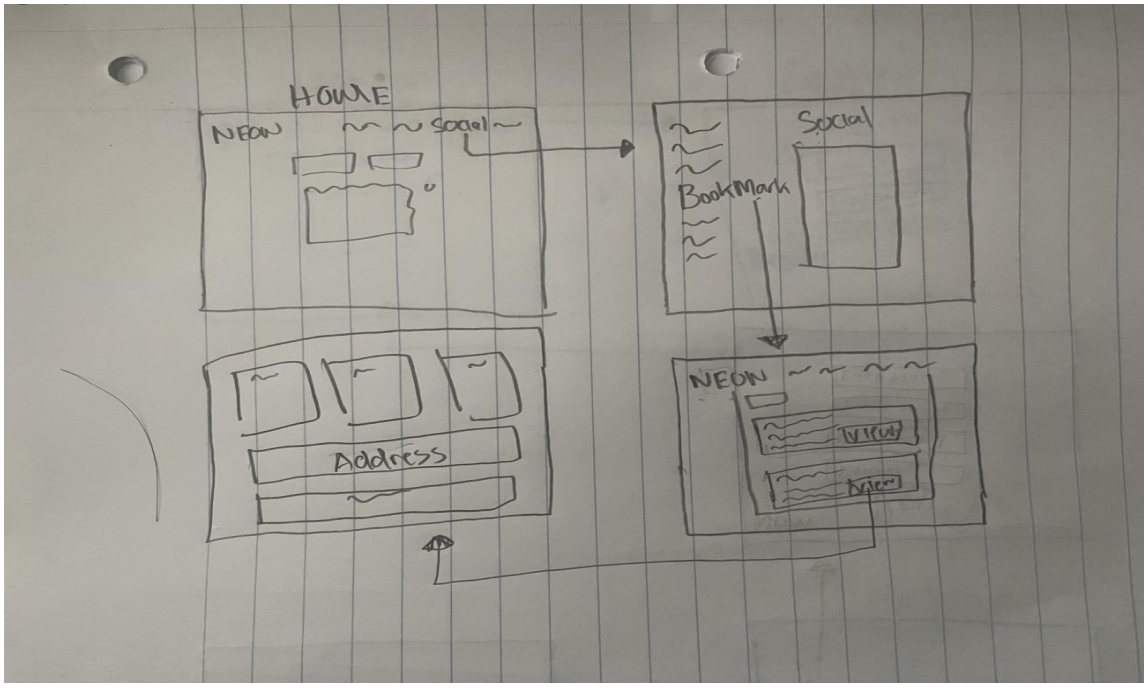

2
Send Message
NEON
Find Homes
Social
Login
Home
View Listing
Notifications
Bookmarks
Groups
Edit Profile
Edit Listing
Messages
Write Post
Post
Username
Post Details
Search
Recent
Message Pop UP
Messages Page
Socials
Search
Username
Username
Username
Username
Type Message Here
Send


Personality Test
Social Page
NEON
Find Homes
Social
Log in
Home
View Listing
Notifications
Bookmarks
Groups
Edit Profile
Edit Listings
Messages
Create Post
Post
Search Bar
Search
Recent User
Popup Msg
Edit Profile
Profile Pic
User Name
Name
About
Social Media Link
Name
School
Major
Year
About
Edit
You Haven't Taken Personality Test
Take Test
Personality Test
Q 1
A 1
Q 2
A 2
Submit
End


HOME
NEON
Search
MAP
ADDRESS
ABOUT PROP
LANDLORD


HOME
NEON
LOGIN
Login
Login
Email
Password
Clear
Login
Register


HOME
NEON
LOGIN
NEON
Login
Email
Password
Register
Neon
Register
Email
Password
First Name
Last Name


HOME
NEON
Social
Social
BookMark
Address
NEON

4. High level database architecture and organization

1. Background Check:

- A background check shall have a unique ID.
- A background check shall be tied to a userID.
- A background check shall contain Employment Record of userID.
- A background check shall contain Education Record of userID.
- A background check shall contain Financial Record of userID.

2. Registered User:

- A registered user shall have one password
- A registered user shall have one username
- A registered user shall have one or more email on the account.
- A registered user shall have either be a renter or a landlord.

3. Location of Rental Listings:

- Each location shall have 1 and only 1 locationID.
- A location shall 1 and only 1 address
- A location shall 1 and only 1 GPS Coordinates
- A location shall 1 and only 1 Region ID

4. Region

- Each Region shall have 1 and only 1 RegionID
- Each Region shall have many LocationID’s.
- Each Region shall have 1 and only 1 District

5. Rental Listings:

- One rental listing can have many posts.
- A rental listing shall have 0 or many beds
- A rental listing shall have 0 or many bathrooms
- A rental listing shall have one and only one address
- A rental listing shall have one and only one price
- A rental listing shall have one and only one property type
- A rental listing shall have many details

6. Posts:

- One post can have many comments.
- Each post shall have one and only one postID.
- Each post shall be tied one and only one userID.

7. Messages:

- Each message shall have one and only one FROM userID.
- Each message shall have one and only one TO userID.
- Each message shall have one and only one timestamp.
- Each message shall contain text(s).

8. Notification:

- Each notification shall be tied to one or more registered user.
- A notification shall be tied to a message or post.
- A notification shall contrarian one and only one time stamp.

9. Events:

- Each event shall have one and only one eventID.
- Each event shall have at least 1 registered user.
- Each event shall have one or more date and time.

10. Friend Request:

- Each friend request shall have one requestID.
- Each friend request shall have only one FROM userID.
- Each friend request shall have only one TO userID.
- Each friend request shall contain a time and date.

11. Saved Properties List:

- Each saved property shall have zero or many PropertyID.
- Each saved property shall have one and only one timestamp.
- Each saved property shall have one and only one userID.
- Each saved property shall have zero or many listingID

12. Groups:

- A group shall have many registered users.
- A group shall have zero or more posts.

- A group shall have one and only one groupID.

13. Reports:

- A report shall have one and only one reportID.
- A report shall be tied to a listingID.
- A report shall contain a time and date.

14. Comments:

- Comments shall have 1 and only one UserID.
- Comments shall have 1 and only one timestamp.
- Comments shall contain texts.

Entities, Attributes, Relationship, and Domains

1. **Registered User**
   - User_ID: Key, Alphanumeric
   - Password: Composite, Alphanumeric
   - BC_Report_ID: Key, Numeric
   - First_Name: Alphabetic
   - Last_Name: Alphabetic
   - Email: Composite, Alphanumeric

2. **Background Check Report(Unsure)**
   - Report_ID:Key, Numeric
   - User_ID:Key, Alphanumeric
   - Status: Composite, Alphabetic
   - Credit_Score: Numeric
   - Income: Numeric
   - Rental_History: Alphanumeric

3. **Location of Rental Listing**
   - Location_ID: Key, Numeric
   - Address: Composite, Alphanumeric

4. **Region**
   - Region_ID: Key, Alphanumeric
   - Region_name: Alphanumeric
   - Location_ID: Key, Numeric

5. **Rental Listing**
   - User_ID: Key, Alphanumeric
   - Listing_ID:Key, Numeric
   - Location_ID: Key, Numeric
   - Rooms: Numeric
   - Bathrooms: Numeric
   - Price: Numeric
   - Property Type: Composite, Alphabetic

6. **Posts**
   -Posts_ID:Key,Numeric
   - User_ID:key, Alphanumeric

- Comment_ID:Key,Numeric
- Post_Content: Alphanumeric

7. **Comments**
   - Comment_ID:Key,Numeric
   - User_ID:key, Alphanumeric
   - Text:composite, Alphanumeric
   - Date/timestamp: Multivalue, Timestamp

8. **Messages**
   - Message_ID:Key,Numeric
   - To_User_ID:Key, Alphanumeric
   - From_User_ID:Key, Alphanumeric
   - Text:composite, Alphanumeric
   - Date/timestamp: Multivalue, Timestamp

10. **Events**
    - Event_ID:Key,Numeric
    - Host_User_ID:Key, Alphanumeric
    - Event_Name: Alphanumeric

11. **Event Guest**
    - Event_Guest_ID:Key, Alphanumeric
    - User_ID:Key, Alphanumeric
    - Event_ID:Key,Numeric

12. **Friend Request**
    - Request_ID: Key,Numeric
    - To_User_ID: Key, Alphanumeric
    - From_User_ID: Key, Alphanumeric

13. **Friend Request List**
    -Request_List_ID: Key,Numeric
    -To_User_ID:key, Alphanumeric
    -From_User_ID: Key, Alphanumeric

14. **Friend List**
    -Friend_List_ID: Key,Numeric
    -Friend_ID: Key, Alphanumeric
    -User_ID:Key, Alphanumeric

15. Saved Properties

-Saved_Properties_ID: Key,Numeric
- User_ID: Key, Alphanumeric
- Listing_ID: Key, Alphanumeric

16. Groups

-Group_ID: Key,Numeric
-Host_User_ID: Key, Alphanumeric
-Group_Name: Alphanumeric

17. **Group Guests**

- Group_Guest_ID: Key,Numeric
- Group_ID: Key,Numeric
- Guest_User_ID: Key, Alphanumeric

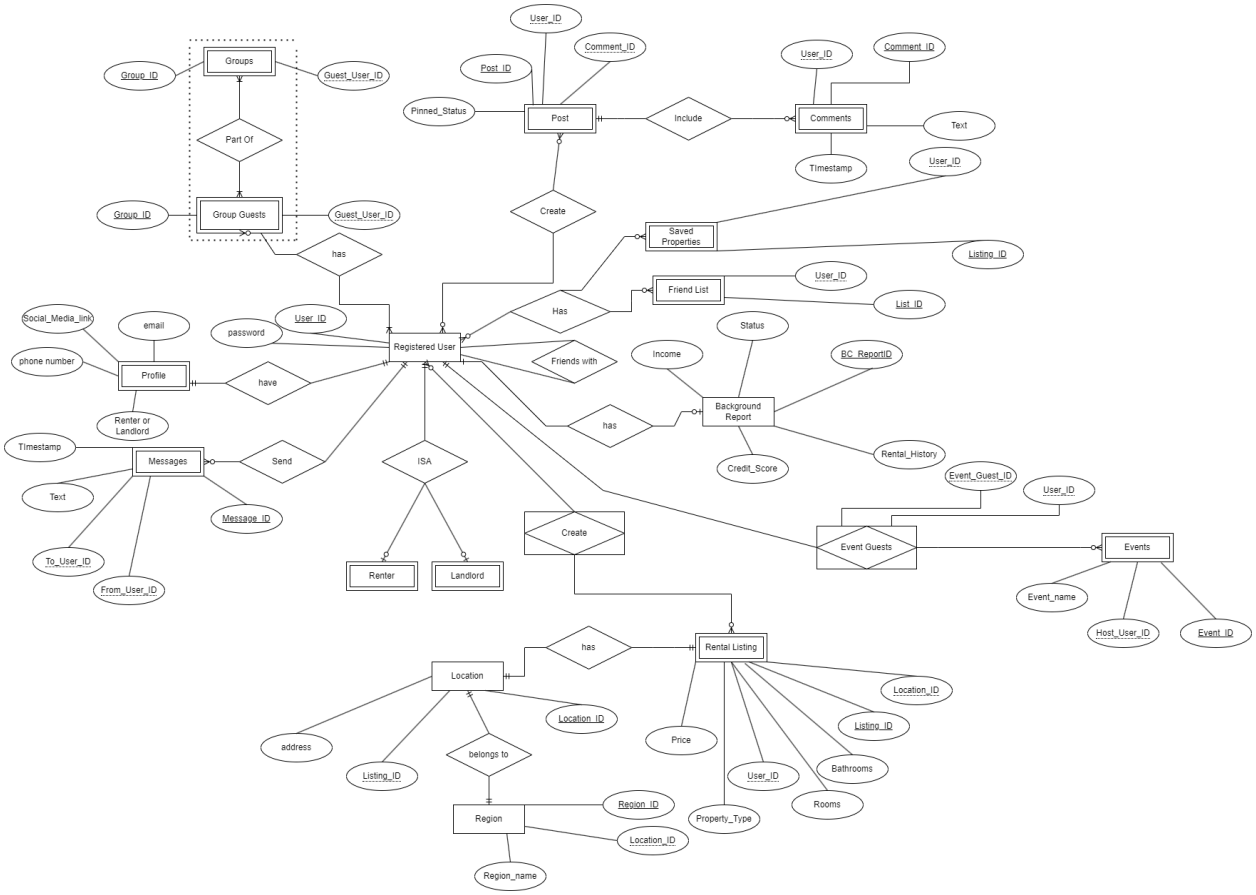
18. Reports

-Report_ID: Key,Numeric
-User_ID: Key, Alphanumeric
-Report_INFO: Alphanumeric

19. Profile

- Profile_ID: Key,Numeric
- User_ID: Key, Alphanumeric
- Phone number: Numeric
- Social_Media_Link: Alphanumeric
- Account_Type: Alpha

Entity Relationship Diagram (ERD)



- We chose MySql because it is a relational database management system that is widely known and adopted in many different fields.

Media storage

The application requires two media storage types: images for each listing and GPS locations to use with Google Maps API. To handle images, they will be stored in a file system since blobs within the MySQL database would significantly impact performance. To handle images, an initial directory will be created with subdirectories for each district. Then, each time the user makes a new listing with images, the API will generate a new folder within the district directory in which the listing is located. The folder name will correlate to the address with a set number of characters to prevent exceeding the limit of the directory path attribute stored within the database. After the directory is created, the images will be stored in the folder, and the folder path will be stored in the images attribute of the listing entity. Once complete, the API will be able to use the folder path from the images attribute to retrieve the images for that listing. To handle the GPS location for each listing, the MySQL POINT type will be used. When the user creates a new listing, the Google Maps API will verify the address, and the longitude and latitude values will be stored in the location POINT for the listing entity. Then, when the user requests a listing, the POINT values can be used to view the location within Google Maps.

Search/filter architecture and implementation

To handle searching and filtering for listings, a MySQL query builder function will be created based on the user's requirements for district, price range, and number of rooms. The function will filter listings from the selected Region with a price range between the chosen lower and upper bounds and the required number of rooms. The sort attribute will then sort the resulting listings and return a maximum of 10 listings.

5. High Level APIs and Main Algorithms

User authentication API (Priority 1):

1. Creation of accounts:

This API allows users to create accounts and register on our platform. Includes steps:
The Unregistered user inputs their account information like username, email, password, repeat password, and etc.
Client side validation ensures that the data is correctly formatted.
When the user clicks the register button, a POST request is sent to the Node API server.
Server-side validation checks if the provided data is valid.
If everything is valid, the user's information is added to the database.
Upon successful registration, the user is redirected to the login screen.

2. Logging into accounts:

This API is for user login:
Users input their username and password.
A POST request is sent to the server to check if the user exists in the database.
If the username is validated, the API creates a session for the user, allowing them to remain authenticated while using the platform. Then the user is redirected to their Home page.
The API also handles errors effectively, providing meaningful feedback to users to resolve any login issues. If the user enters an incorrect username or password, the API will display an error message to the user.

3. Background Checking (Priority 1):

This API will retrieve data about any potential tenant or landlord using existing. APIs to search for any misdemeanors, court records, driver history,rental history,

income, and credit history. Once the background checking process has concluded a checkmark symbol will appear next to an registered user's account denoting that they are background checked.
When a landlord triggers a background check, the API should initiate a series of verification processes.
This might include sending requests to external credit agencies and databases to cross-reference the provided information.
The API should handle the responses from these external services and maintain logs of these interactions.

4. Group Notification and Saved Properties API (Priority 2):
This API is responsible for managing group notifications and saved properties within our platform.
Users can receive and view group notifications related to properties they are interested in.
Additionally, users have the option to save properties for future reference.
The API handles the storage and retrieval of these notifications and saved property data
Implement authorization and authentication checks to ensure that only authenticated users can access their notifications.
Query the database for notifications associated with the user's account and return the data in a structured format, typically JSON.

5. Quality of Client API (Priority 3):
Implement code to continuously monitor key performance metrics, such as page load times and responsiveness, in real-time.
Set up automated systems that detect performance issues in real-time.
Use alerting mechanisms, such as email, or webhook notifications, to immediately inform administrators when issues are detected.
Store detailed issue information in logs or a database for further analysis.

Data retrieval API (Priority 1):

1. **Listing and retrieving rental listings Information:**

This API handles the creation of new rental listings, ensuring that all required fields are filled in. It also retrieves rental listings information, including title, description, image, price, number of bed and bath, and address.
Create Listings: This API will handle validation to ensure that the provided listing information is complete and valid, including details like property type, location, price, number of bed and bath, image, title, property description.
Store the listing information in a database, associating it with the user who created the listing.
Retrieve Listings: This API will handle query parameters that allow users to filter listings based on criteria such as location, price range, and property type.
Query the database to fetch relevant listings based on the specified filters and return the results in a structured format, such as JSON.
Update Listings: This API will handle validation to ensure that only the listing owner can modify their own listings.
Update the listing information in the database based on the provided changes.
Delete Listings: This API will handle authorization checks to confirm that only the listing owner can delete their listings.
Remove the listing from the database, ensuring it's no longer accessible.

Search and recommendation API (Priority 1):

2. **Search and recommendation API:**

   Property Search: This API will handle search queries and generate personalized recommendations based on user preferences and behavior. The purpose is to enhance user engagement and content discovery within the platform

   This API will handle query parameters for search criteria such as location, property type, price range, and other relevant filters.
   Query the database to fetch properties that match the specified criteria and return the results in a structured format, such as JSON.

   Recommendations: This API will handle recommendation algorithms or models based on user behavior, preferences, and historical data.
   This API uses these algorithms to suggest properties that are likely to match the user's interests and return the recommendations in a structured format.

Notification and Messaging API (Priority 2):

3. Notification and Messaging:

This API will manage notifications, messages, and alerts sent to users. It maintains an array of messages, with each message containing the ID of the message poster, the message itself, and the timestamp. The purpose is to enable real-time communication and keep users informed about relevant activities.
This API will handle validation to ensure that the message content is provided, the recipient is valid, and the sender has the necessary permissions.
Also, it stores the messages in the database, associating them with sender and recipient information.

4. Filtering of Listing by Location API (Priority 1):

This API will manage our listing based on the user's location filtering preferences. It will show all the listings within a specific geographic area. The purpose is to ensure the user has an option to filter out her listing giving them the results that aligns with their specific location criteria.
This API will handle parameters for location-based filtering, such as city, neighborhood.

5. Detecting Inappropriate Posts or Accounts (Priority 1):

This API will detect and handle reports and the inappropriate content that is being posted on the app. The purpose is to ensure the user can browse through safe and suitable content that is free from scam and other fraudulent activities.

User Reporting: This API will handle validation to ensure that the reported content or account is properly identified and reported for legitimate reasons.
Image Analysis: This API will recognize and analyze reported images to identify inappropriate content.
Blocking or Flagging: This API will develop a mechanism to flag or block inappropriate content or accounts. It allows administrators to review flagged content and take appropriate action, such as removing content or suspending accounts.

6. API to Link Social Media Accounts such as Twitter or Instagram (Priority1):

This API will allow users to link their social media accounts to their SFStudentRent account. The purpose of linking social media accounts is to enhance the app's security by preventing fraudulent accounts and enabling the users to find potential roommates.
This API will handle authentication protocols to allow users to authorize access to their social media accounts securely. Once authorized, associate the linked social media account with the user's account on your platform. It stores the association in the database, mapping the user's account to their social media account identifiers.

Non-trivial algorithms or processes:

1. Housing Price Forecast Calculation (Priority 2):

This process will forecast possible changes in rent or housing prices for the upcoming years using existing APIs and public data available based on the zipcode of a property. It will take a variety of factors and retrieve data for other properties in the same area, look at an area jobs' outlook and schooling to come up with an accurate prediction of how a property forecasts to increase in price or rent.

2. Deep Learning/AI Housing Forecast Calculation (Priority 3):

This process will attempt to predict the same rent or housing prices but for a longer period such as the upcoming ~5 to 20 years and will do so by using machine learning algorithms and will use existing APIs in the beginning to create its own API to model the regression or progression that it expects a property to have for a given time interval. The model will get smarter and better with more data as it gets fine-tuned and the results of its predictions come in. This will be especially useful for home buyers on deciding what the best area to buy a property will be.
This API will define input parameters, including property details such as location, type, historical pricing data, and forecast period. This API will Integrate linear regression, time series analysis, or machine learning models for price forecasting.

Posts and Messages Integration API:

1. Real-time Chat Synchronization Process (Priority 2):

This API will make store all the messages that take place as an array of objects, with each object having its own message ID, ID of the message poster, and timestamp. This process ensures that messages sent between users in real-time chats are synchronized across devices.The purpose is to provide a seamless messaging experience for users.

2. Promotion or Pinned Post API (Priority 2):

This API will first identify the content that is desired to be promoted or pinned to an user account and it will strategically push the listings to applicable users.

Promoted listing will have a changed priority at the same level as a brand new post and will have a greater change at showing up in a user's recommendations.

The pinned posts are subject to the user's preferences with a limit of three per account and promoted listing are intialized by landlords who wish to get a larger reach for their property or sell their property quicker with a limit of two free promotions a week.

3. Editing or commenting Post API (Priority 1):

This API will store each post in an array of objects having its own post ID, ID of the user who created Post, and timestamp of post. Posts can either come in the form of landlord posting rental listings, or renters posting looking for roomates.

If any changes are made by a user to update his or her post, the API will retrieve and change the current data of the post. In addition it also allows users the functionality to edit their own post, comment, replies storing each of the comments in its own array with comment ID, ID of commenter, and a timestamp. The poster has capability of allowing comments for all users or only registered users or only friends.

All comments will go through the the Spam Messages API and are subject to be deleted for highlighted for an account to be banned if the API detects any unusual or uncanny activity.

4. Detecting Inappropriate Messages or Spam Messages API (Priority 2): This API will first make a call to see if an account has an IP address, email, or domain that is on a blacklist and if it detects that it does it will block the account which will be very useful in preventing against spam or bot regristrations.

Any future accounts that get banned will be added to the blacklist and any users with the blocked IP will be unable to use the service.

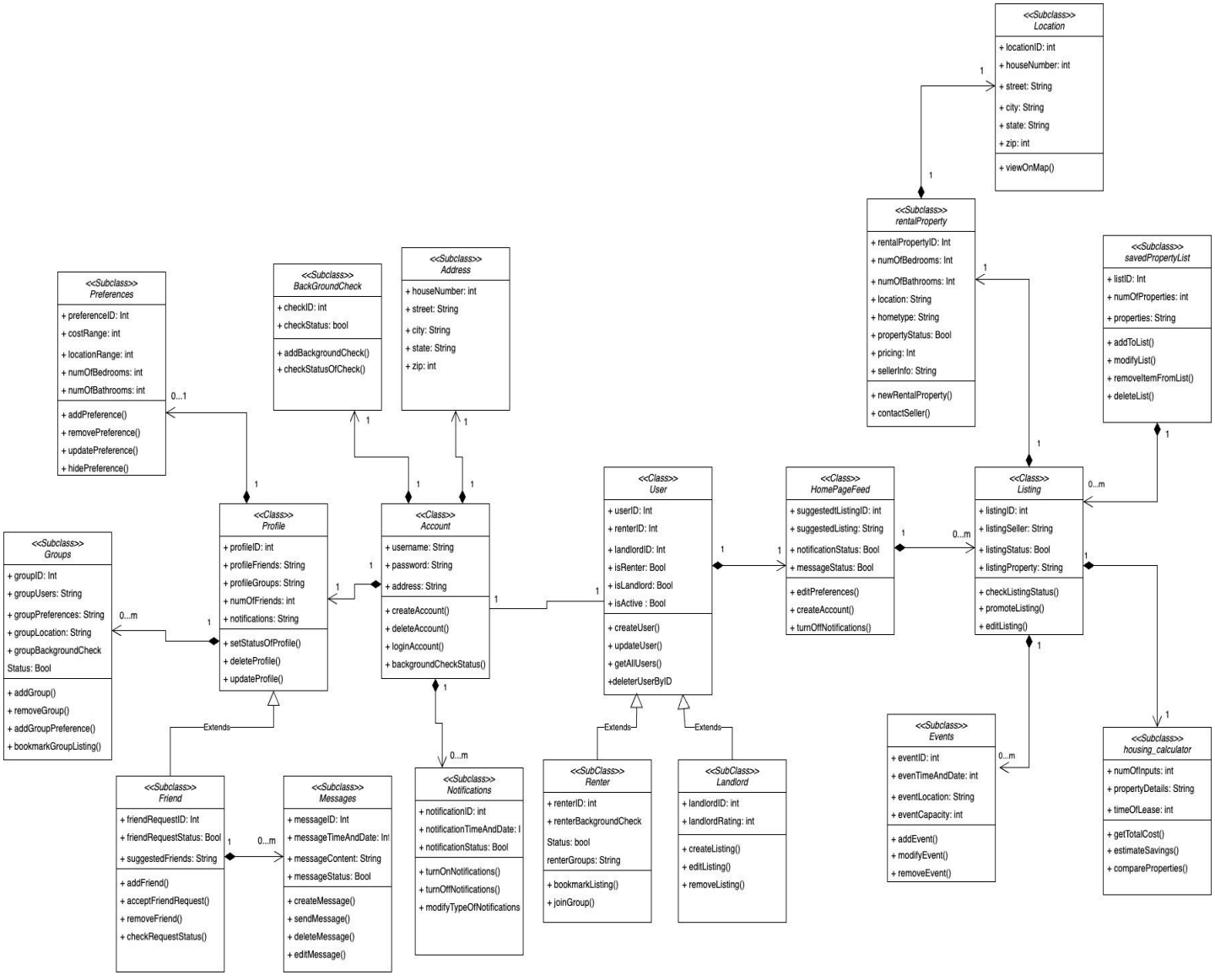
There will also be an API call to scan for inappropriate words or phrases and immediately be highlighted and subject to review by administrators who will decide what to do with the in question account.

Finally there will be an interface for reporting all misdemeanors and inappropriate actions of another user with administrators being the final step in determining the status of whether or not to ban and blacklist an account.

6. High Level UML Diagrams

• High-level UML class diagrams for implementation classes of core functionality, i.e. functionality with provided interfaces. Focus on a main high-level classes only (one or at most two levels deep). Your UML diagram must be implemented using Object Oriented or Protocol Oriented approaches as seen in Class.

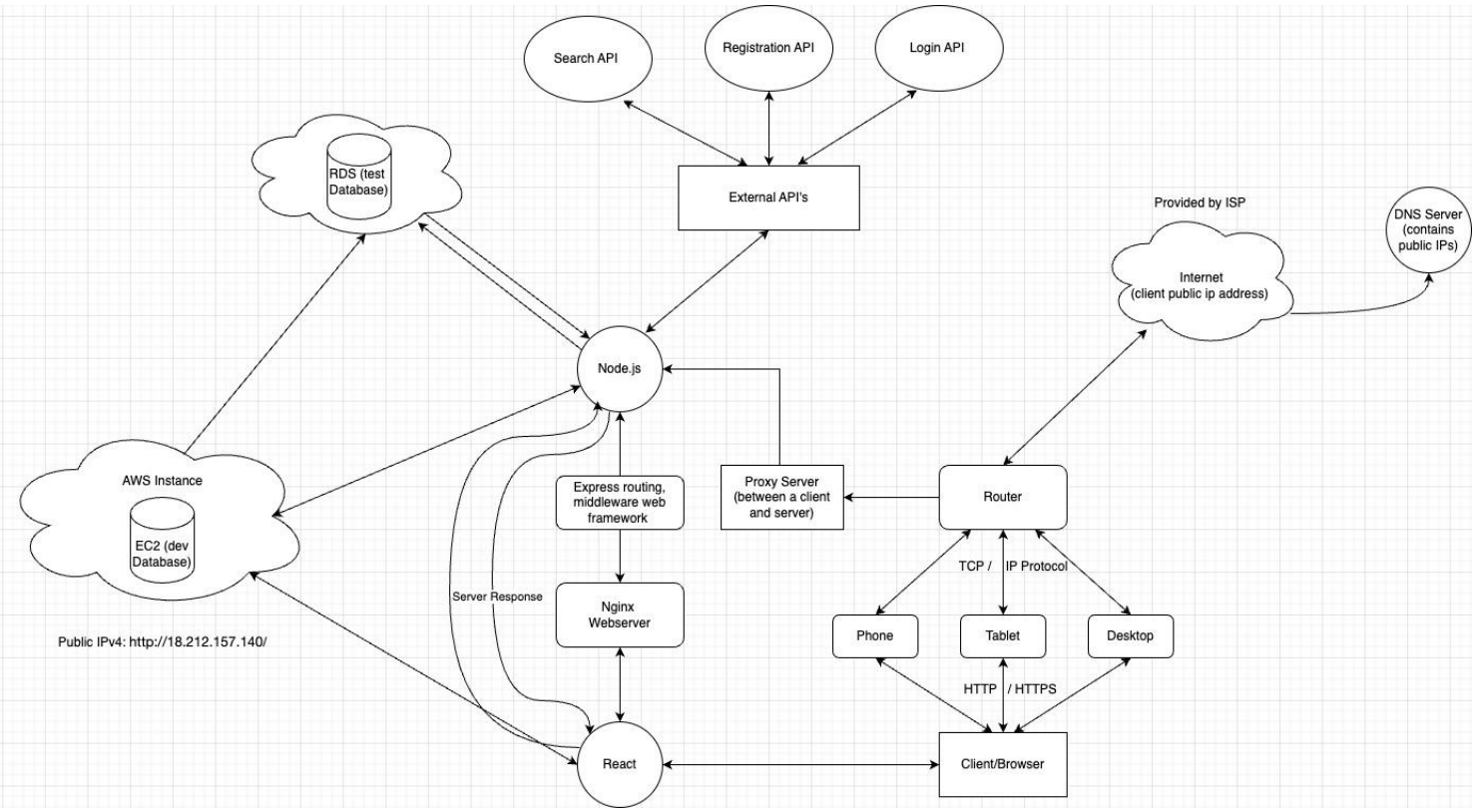
• Use data terms and names consistently with Data Definition Section 1 above.
https://drive.google.com/file/d/1OCo-y7JNa5F4_hOWmugzJgZMKJdR4jFo/view?usp=sharing



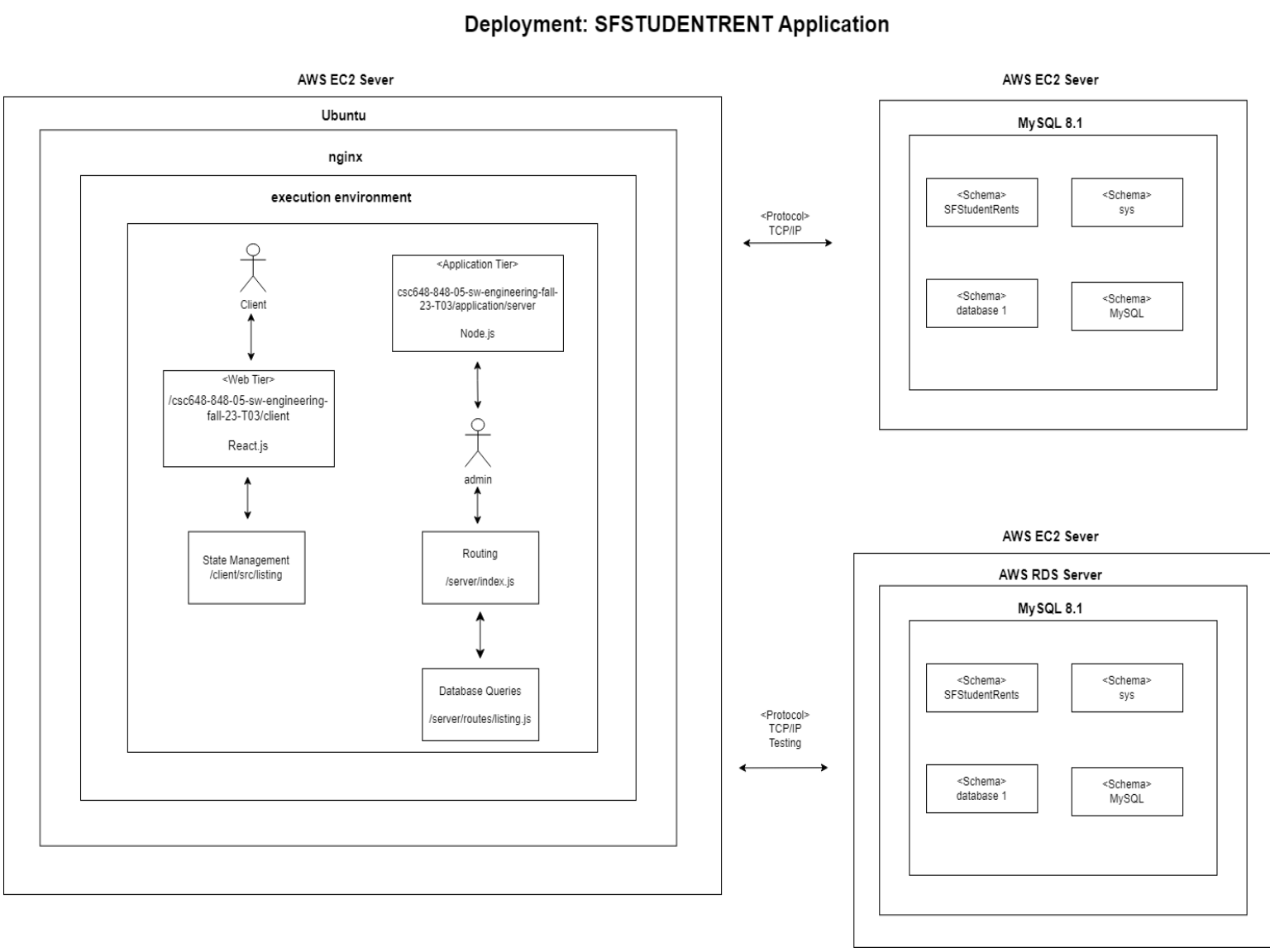
7. High Level Application Network and Deployment Diagrams

Application Network Diagram:

https://drive.google.com/file/d/1UWdu-Du5dFywLpYWD4owqKvRh9_iZX-9/view?usp=sharing



Deployment Diagram:
https://drive.google.com/file/d/1uRXj8tXfLRSPx-sQbFEYQnTkgl7Q9_LZ/view?usp=sharing



8. Key Risks

• **Schedule Risks:**

- A problem we noticed was that the top down project cannot begin until the database has something to work with, for example we cannot create our data hook api to connect to the front end. A way to resolve that is to ensure we have things done on time.

- We have a 7th member now and we can not find a time where we can all meet up. Since he is placed at the frontend time, we have decided to leave it to the frontend lead to fill him in on information he missed and the work he needs done.

• **Legal Risks:**

- Potential problem with certain discrimination housing laws in CA, discrimination on criminal history. Felons are ok to, but petty misdemeanors for example may or may not necessarily warrant certain limitations. Way around this is maybe a minimum standard contract, something like that, where homeowners can put a minimum standard for their tennant, but they can probably more easily relate that to a different protected class which makes it a very grey area. A way to resolve this is the work around stated or focus on this part later to get the proper research on the legalities.

•**Technical Risks:**

- Our team of eight members and four of us want to go into the frontend team. Our backend and database team only has two people there. We plan on resolving this by shifting people around to rebalance the teams and the team lead plans to go to the team that needs the most support.

- The team is working with Amazon AWS using the EC2 instance to host our server and RDS to host our website. With the way Amazon has their cloud services set up, the team runs into error accessing certain part of the application. This could be resolved by starting the work early and finding the roadblocks before the due date gets too near.

9. Project management:

• So far we have been using discord to chat and manage our tasks, with individual calls and split each other into teams. We have been coordinating our work through Google Docs by leaving suggestions and comments for the next person to see. Currently we are now using Trello to manage each task in a more organized fashion that offers a unified dashboard view of all tasks and statuses and as well as who is assigned to each task. We also are using the notes and comments section in Trello cards to add further details to each task to allow us to manage our project properly. We also have color coordination on our Trello so the team knows what is being worked on, not started and what is finished. Currently this is doing a decent job to manage each task, thus we will probably continue to use this in the future as our main managerial tool. We also now have the ability to pin chat messages in discord, and that has been useful for our important links and feedback in that channel.

10. Detailed list of contributions (this section must be done by the team lead)

Members	Contributions	Score
Jeremy Tran	● Added data definitions for M2 ● Prioritized functional requirement ● Worked on Database Architecture ○ Worked on Database requirement and data definition, schema ○ Revised ERD ● Learned and left comments for revision on each section of M2 ● Revised M1 document based on feedback for V2 ● Created and invited team to Trello ● Checked and edited M1 and M2 document for consistency	
Geovanni Valadez	● Revised M1 document based on feedback for V2 ● Did data definition for M2 ● Prioritized functional requirements ● Created UI mockups and storyboard ● Lead frontend development of prototype ○ Split work among Ivan and Daniel ● Attended every meeting and contributed in discussions ● Has been on top of internal due dates	10
Mozhgan Ahsant	● Added data definitions for M2 ● Researched and planned out APIs needed for project ● Wrote Description of section 5 ● Has attended every meeting ● Finished work by due date	8
Anthony Silva	● Added data definition for M2 ● Attended every meeting and has contributed to discussions ● Worked on Database Architecture ○ Worked on database requirements,	10

	data definitions ○ Created ERD and schema	
Aman Khera	● Added data definition for M2 ● Assisted in finding API and algorithms ● Created rough UML diagram and transferred to draw.io for production diagram ● Created rough Netwok and Deployment Diagram ● Has attended every meeting and contributed to group discussion ● Lead backend development of prototype	6
Ivan Ayala-Brito	● Added data definition for M2 ● Organized M1 and M2 document ● Created UI Mockups and Storyboards ● Worked on frontend development of prototype ● Has attended every meeting	7
Daniel Enriqurez	● Worked on frontend development ○ Login Page ○ Register Page	10
Alex Huang	● Was proactive on getting caught up with the team ● Organized the document	10