

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 1.0

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

History Table

Milestone Version	Date Submitted
M2V1	10/12/23

Table Of Contents

1. Data Definitions.....	4
2. Prioritized Functional Requirements.....	8
3. UI Mockups and Storyboards.....	13
4. High level database architecture and organization.....	22
5. High Level API and Main Algorithm.....	29
6. High Level UML Diagrams.....	32
7. High Level Application Network and Deployment Diagrams.....	33
8. Identify actual key risks for your.....	35
9. Project Management.....	36
10. Detailed list of contributions.....	37

1. Data Definitions

1.User: Unregistered User will only have view privilege

An unregistered user will not have information kept in the database. They are free to use the site for viewing purposes only. All functionalities such as posting and messaging will not be available to these users.

2. Background Check: Check registered user's credit history, rental history and income

CheckID:

- This ID will be tied to a user so that we know who the background check belongs to.
- Status (Composite, Alphabetic): This is the status of the background check and let the user know whether its processing or finished.
- Credit_Score: Numeric
- Income: Numeric
- Rental_History: Alphanumeric
- Last 3 attributes are the information we are checking for.

3. Registered User: Will have full access to all functionality on site so they can post, message, join groups, etc.

- UID (User ID)
 - Tied to the user's account to identify who it is
- Email
- Name
- Username
- Filters
- Renter or Landlord
 - These last 5 attributes are information of the user we would be storing on their account

5. Location of Rental Listings: Shows exactly location of rental listing

- Maps: Alphanumeric
- Filter: Numerical
- Search: Alphanumeric

5. Region: General location of rental property, different neighborhoods

- RegionId: key
- LocationId: key

6. Rental Listings: Post containing texts and media about advertised rental property

- ListingID
- Bedrooms:Numerical
- Bathrooms:Numerical
- Price:Numerical
- Location:Alphanumeric
- Filters:Alphanumeric
- Time and Date:Timestamp
- Home Type: Alphanumeric
- Option to add more

7. Posts: General post that could contain texts or media

- Post ID
- Time and Date

8. Messages (STRONG Entity): Text sent privately from one registered user to another.f

- MessageID: Key, Numeric
 - A unique identifier assigned to each private message
- Message date and time: Timestamp
 - Date and time when the message was sent.
- CommentID: Numeric
 - A unique identifier assigned to each comment within a message.

9. Notification (STRONG Entity): Will let the registered user know if there is an update on their account.

- NotificationID: Key, Numeric
A unique identifier for each notification.
- Notification date and time: Timestamp
 - Date and time indicates when the notification was generated.
- Notification Type: Alphanumeric
 - Notification Type Indicates the type of notification. For example it indicates that the notification is a message or a friend request.

10. Events (STRONG Entity): Registered users can create an event page open to the public or select invite.

- EventID: Key, Numeric
 - A unique identifier assigned to each event.
- Event date and time: Timestamp
 - Date and time indicates the date and time of the event.
- Event Type: Alphanumeric
 - Event Type indicates the type of event. For example, an open house or private event.
- Confirmed Attendees: Numeric
 - A list of user IDs representing confirmed attendees for the event.

11. Friend Request: Registered users can save each other to a friend list

- Date and time of the request: Time Stamp
- Status: can either be accepted, denied, or delivered
- RequestID: unique id for every friend request that is sent
- SuggestFriendList: gives a recommended list of friends based on mutuals, locations, interests, and profile preferences.

12. Saved Properties List: Registered user can save rental listing(s) to the list

- Number of houses: Numeric
 - Max of 50 houses per registered account
- Similarities in houses: Alphanumeric

- Ability to compare houses by bedrooms, bathrooms, pricing, location, size or transportation options from list
- Pricing: Numeric
 - How much property will cost per month
- ListID: Key
 - Unique Id for every list that is created

13. Housing Cost Calculator: Allows users to input different information and will be given suggestion(s) based of the info

- Number of residents: Numeric:
 - How many are expected to live in property
- Duration of lease: Numeric:
 - Months lease is expected to last
- Expected growth of the area: Numeric:
 - Expected inline or decline of property area and repercussions on rent
- Interest: Numeric:
 - Only applicable to home buyers not renters

14. Groups: Registered user can form groups with one another for group messaging and group page

- GroupID: Key;
 - Unique ID for every group created
- Name: Alphanumeric;
 - Nameplace for the name of a group tied to GroupID

15. Reports: Generate a report of a rental listing

- ReportID: Key; Unique ID for every report created by a user
- Time and Date:Timestamp; time and date details for when the Report was created.

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.
- (1.4) 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.

3. Registered User

- (3.1) Registered users shall be given the option to make posts on the site.
- (3.4) A registered user shall have one password.
- (3.5) A registered user shall have one username.
- (3.6) A register user shall be able to create one free renter Account.
- (3.8) A Registered User shall be able to link their social media accounts such as Instagram and Twitter.

4. Location of Rental Listing

- (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. 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.
- (7.4) One post can have many comments.

Priority 2:

1. User:

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

2. Background Check Integration:

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

3. 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.

4. Location of Rental Listing:

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

5. Rental Listing:

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

6. Posts:

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

7. Messages:

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

8. 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.

9. Housing Cost Calculator:

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

10. Groups:

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

11. Reports:

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

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:

- (9.1) Each notification shall have a unique NotificationID.
- (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.

8. Housing Cost Calculator:

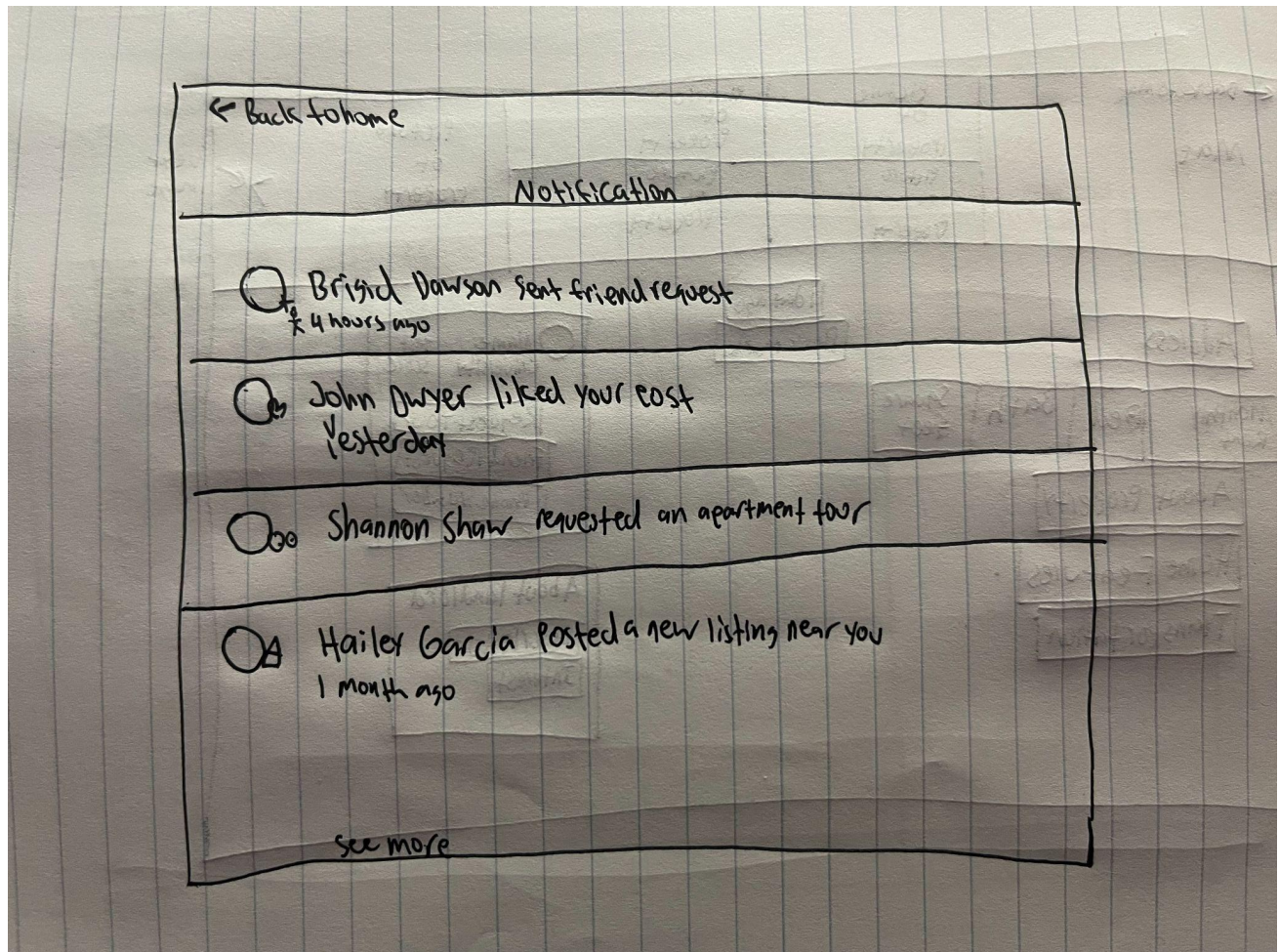
- (13.2) The Calculator shall generate a curated list of houses best fit for a registered user based on filters and location.

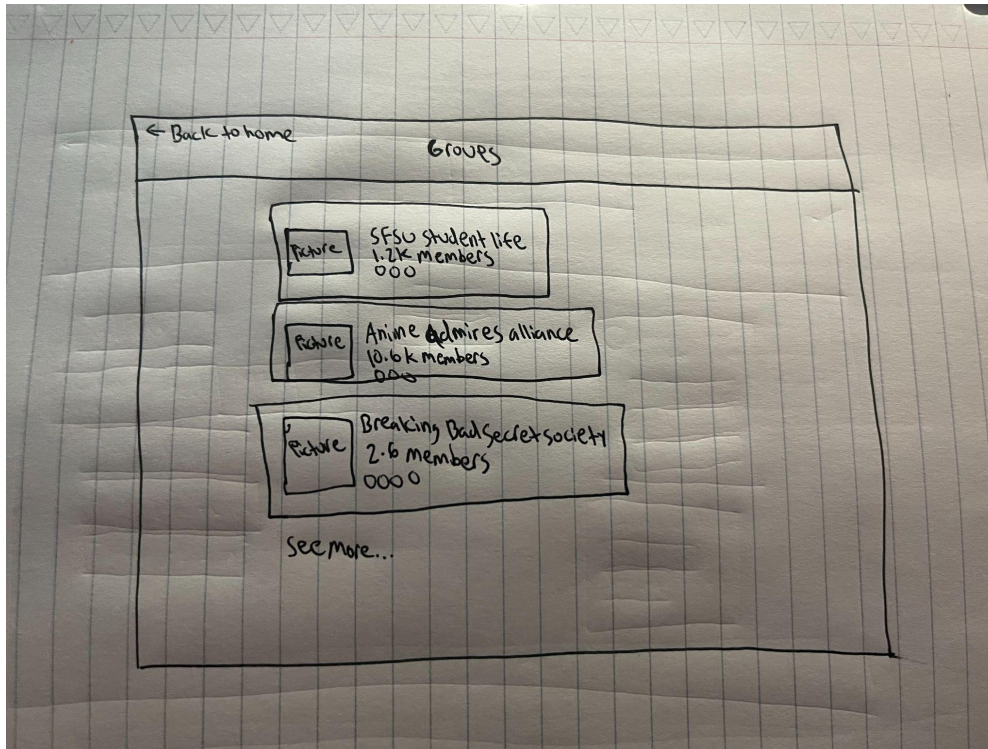
9. Reports:

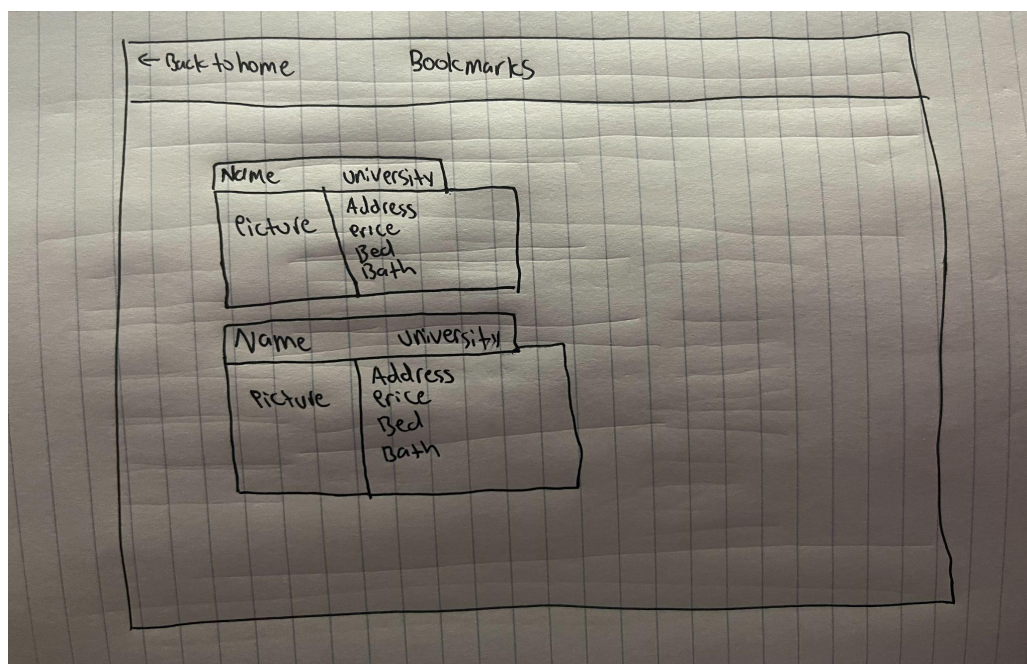
- (15.2) The system shall provide functionality to generate various types of rental reports, including rental transaction summaries, rental history for specific customers, and rental equipment utilization reports.
- (15.3) Users shall be able to customize the content and format of reports, including selecting specific date ranges, sorting options, and the inclusion/exclusion of specific rental data field, such as rental duration, customer information, and equipment details.

3. UI Mockups and Storyboards (high level only)

Notification List:



Groups:**Bookmark:**



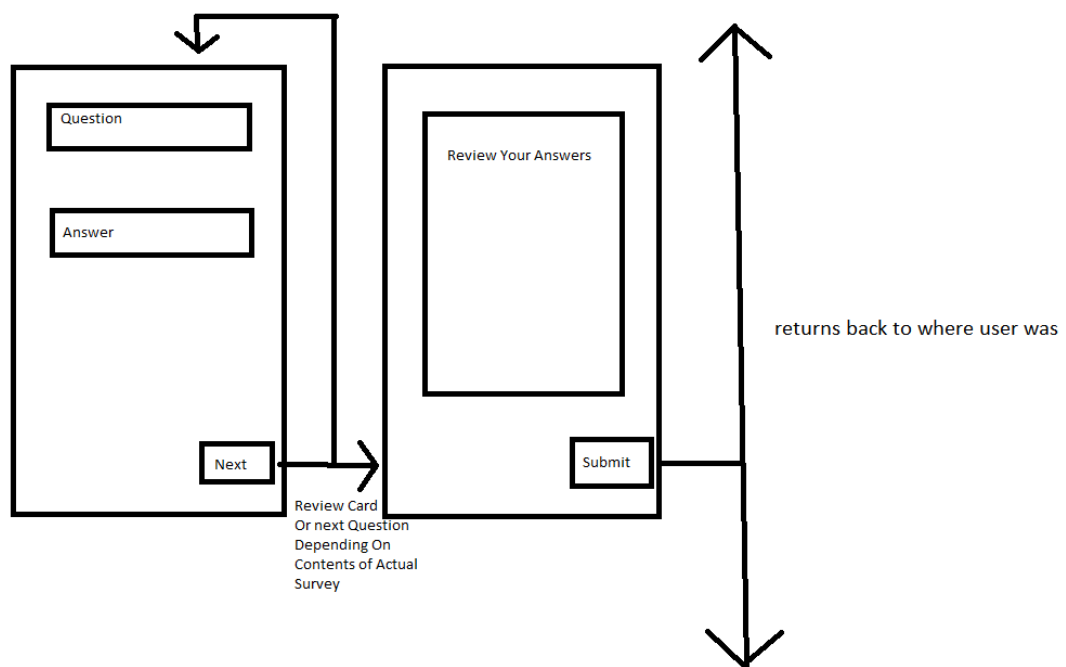
Edit Listing:

<p>JUMP TO</p> <p>Description:</p> <p>Photos:</p> <p>Open House</p>	<p>DETAILS</p> <p>Street</p>		<p>Exit</p>
	<p># Bed</p>	<p># Bath</p>	
	<p>Price</p>	<p>Extra details comment section</p>	
	<p>Photo</p> <p>Photo</p> <p>Photo</p> <p>Upload Photo</p> <p>Adds Photo Or Click</p>		
	<p>Open House</p> <p>Date</p> <p>Start Time</p> <p>End Time</p>		<p>Publish</p>

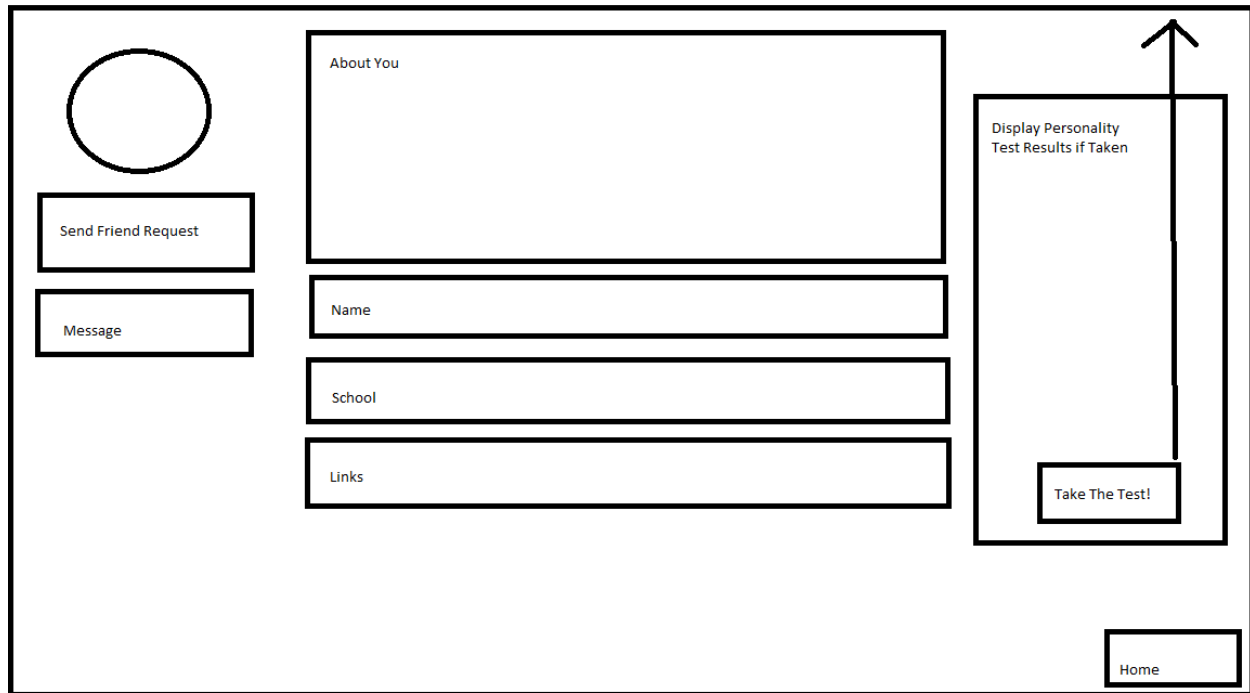
Edit Profile:

A wireframe of a user profile page. On the left, there is a circular profile picture placeholder with a 'Change Profile Pic' button below it. To the right of the profile picture is a large 'About You' text area. Below this are three input fields labeled 'Name', 'School', and 'Links'. On the far right, there is a vertical container with the text 'Display Personality Test Results if Taken' at the top and a 'Take The Test!' button at the bottom. An arrow points from the 'Take The Test!' button to a 'SAVE' button located at the bottom right of the page.

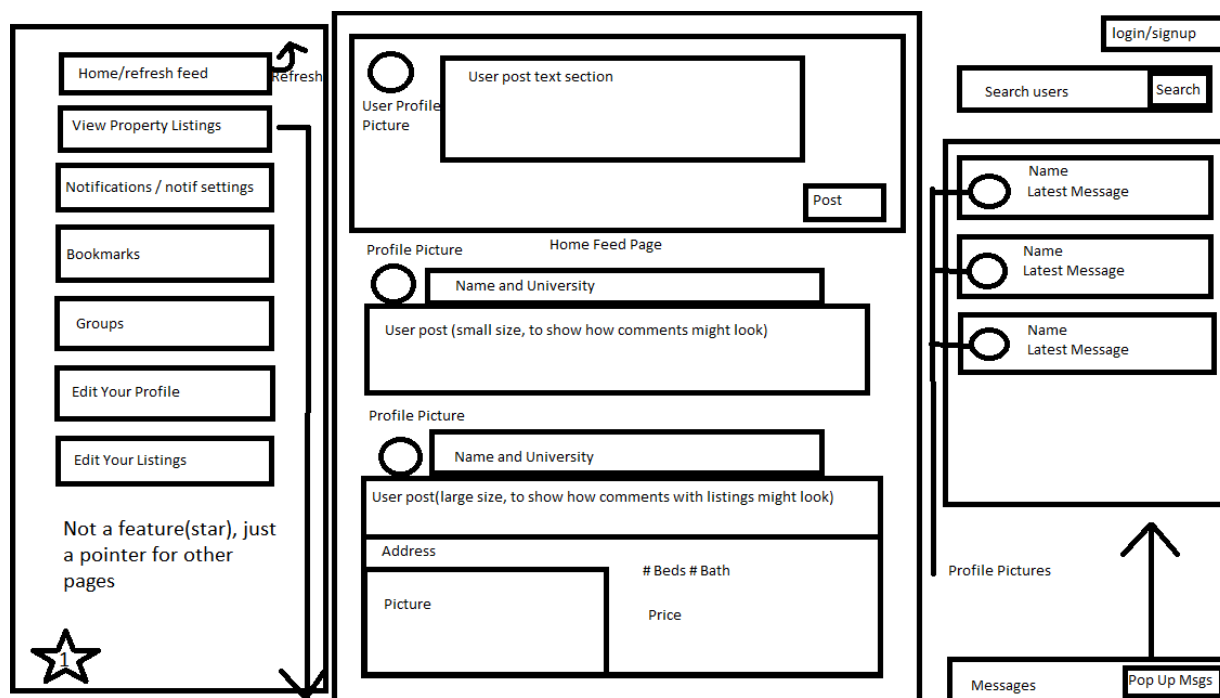
Personality Test:



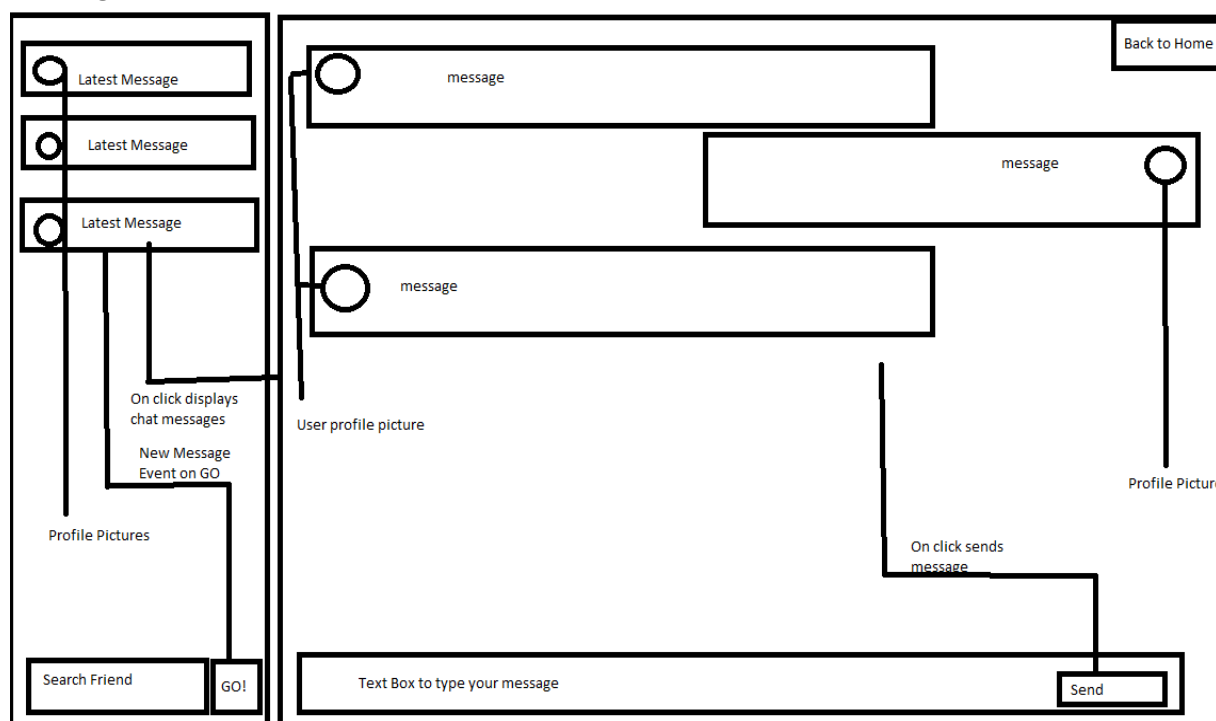
View Profile:



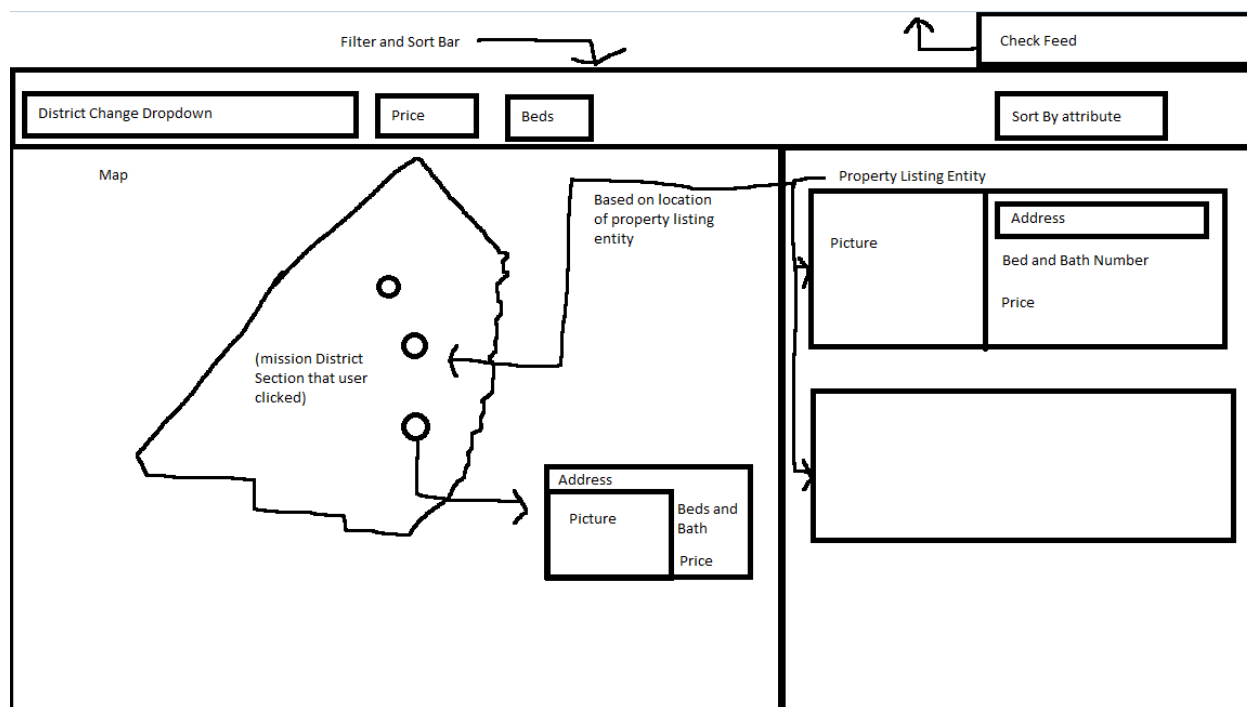
Social Page:



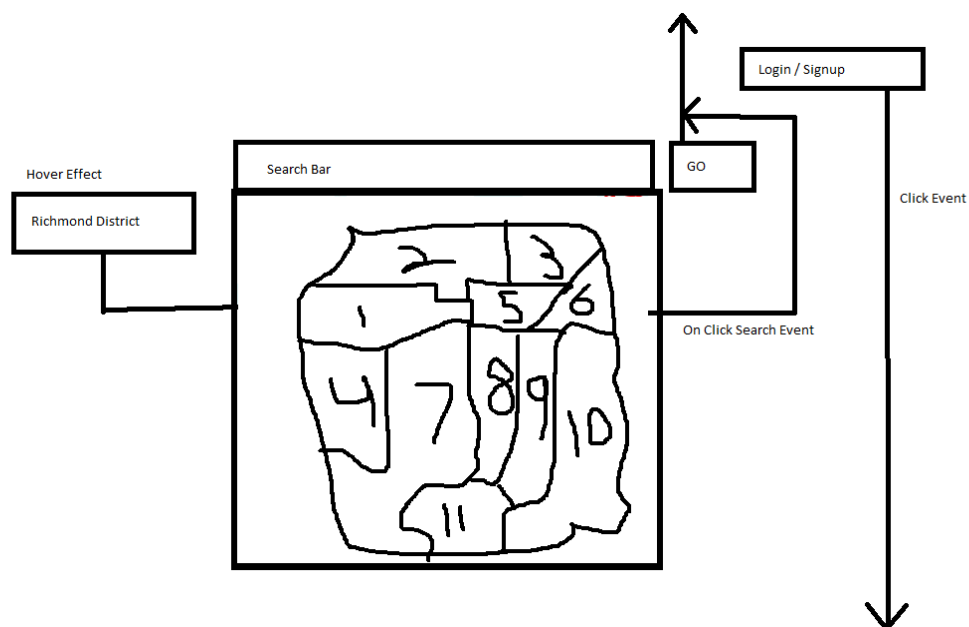
Messages:



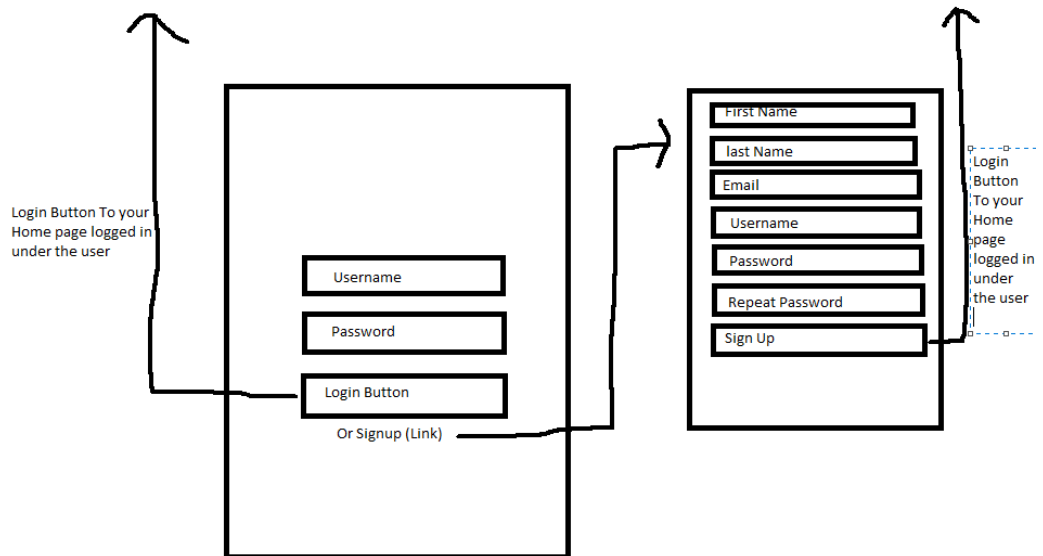
Property Listing:



Main Search:



Login/Sign up Page:



Filters:

Filters

Mission District	Price	Bed	Sort By Attribute
1 Richmond District	Min Rent - Max Rent	No Min - No Max	Price (low to high)
2 Marina	0 250+	No Min	Price (high to low)
3 North Beach/Point	500+	1 Bed	Video
4 Sunset District	1000+	2 Beds	3D Tour
5 Hayes Valley	1500+	3 Beds	
6 Market	2000+	4+ Beds	
7 Twin Peaks			
Display more...			

Back to home

Displaying Property Listing:

A hand-drawn wireframe for a property listing page on a grid background. The layout is as follows:

- Top Section:** A horizontal bar containing a "Back to Map" link with a left arrow, a grid of four "Picture of Property" placeholders, and a "Next image" link with a right arrow.
- Left Column:** A vertical stack of boxes for "Address", "Monthly Rent", "Bed", "Bath", "Square feet", "About Property", "Home Features", and "Transportation".
- Right Column:** Contains a "Bookmark" button, a "1 day ago" timestamp, and a "Request to View" button.
- Bottom Right Section:** A box for "About landlord" containing "Description" and "Interest" fields.
- Form Elements:** Includes a "Name" field with a radio button, a "University" field, a "Test score" field, a "Friend Request" button, and a "Phone Number" field.

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_ReportID: Key, Numeric

2. Background Check Report(Unsure)

- Report_id:Key, Numeric
- Status: Composite, Alphabetic
- Credit_Score: Numeric
- Income: Numeric
- Rental_History: Alphanumeric

3. Location of Rental Listing

- Location_id: Key, Numeric
- Listing_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, Alphanumeric
- Rooms: Numeric
- Bathrooms: Numeric
- Location_id: Key, Alphanumeric
- Price: Numeric
- Property Type: Composite, Alphabetic

6. Posts

- User_ID:key, Alphanumeric
- Posts_ID:key, Alphanumeric
- Comment_ID:Key, Alphanumeric
- Pinned_Status: Alphanumeric

7. Comments

- Comment_ID:Key, Alphanumeric
- User_id:key, Alphanumeric
- Text:composite, Alphanumeric
- Date/timestamp: Multivalue, Timestamp

8. Messages

- Message_ID:Key, Alphanumeric
- To_User_ID:Key, Alphanumeric
- From_User_ID:Key, Alphanumeric
- Text:composite, Alphanumeric
- Date/timestamp: Multivalue, Timestamp

10. Events

- Event_ID:Key, Alphanumeric
- Host_User_ID:Key, Alphanumeric
- Event_Name: Alphanumeric

11. Event Guest

- Event_Guest_ID:Key, Alphanumeric
- User_ID:Key, Alphanumeric

12. Friend Request

- From_USER_ID: Key, Alphanumeric
- To_User_ID: Key, Alphanumeric
- RequestID: Key, Alphanumeric

13. Friend Request List

- User_ID:key, Alphanumeric
- From_USER_ID: Key, Alphanumeric

14. Friend List

- User_ID:Key, Alphanumeric
- List_Id: Key, Alphanumeric

15. Saved Properties

- User_ID: Key, Alphanumeric
- Listing-ID: Key, Alphanumeric

16. Groups

- Group_ID: Key, Alphanumeric
- Host_User_ID: Key, Alphanumeric

17. Group Guests

- Group_ID: Key, Alphanumeric
- Guest_User_ID: Key, Alphanumeric

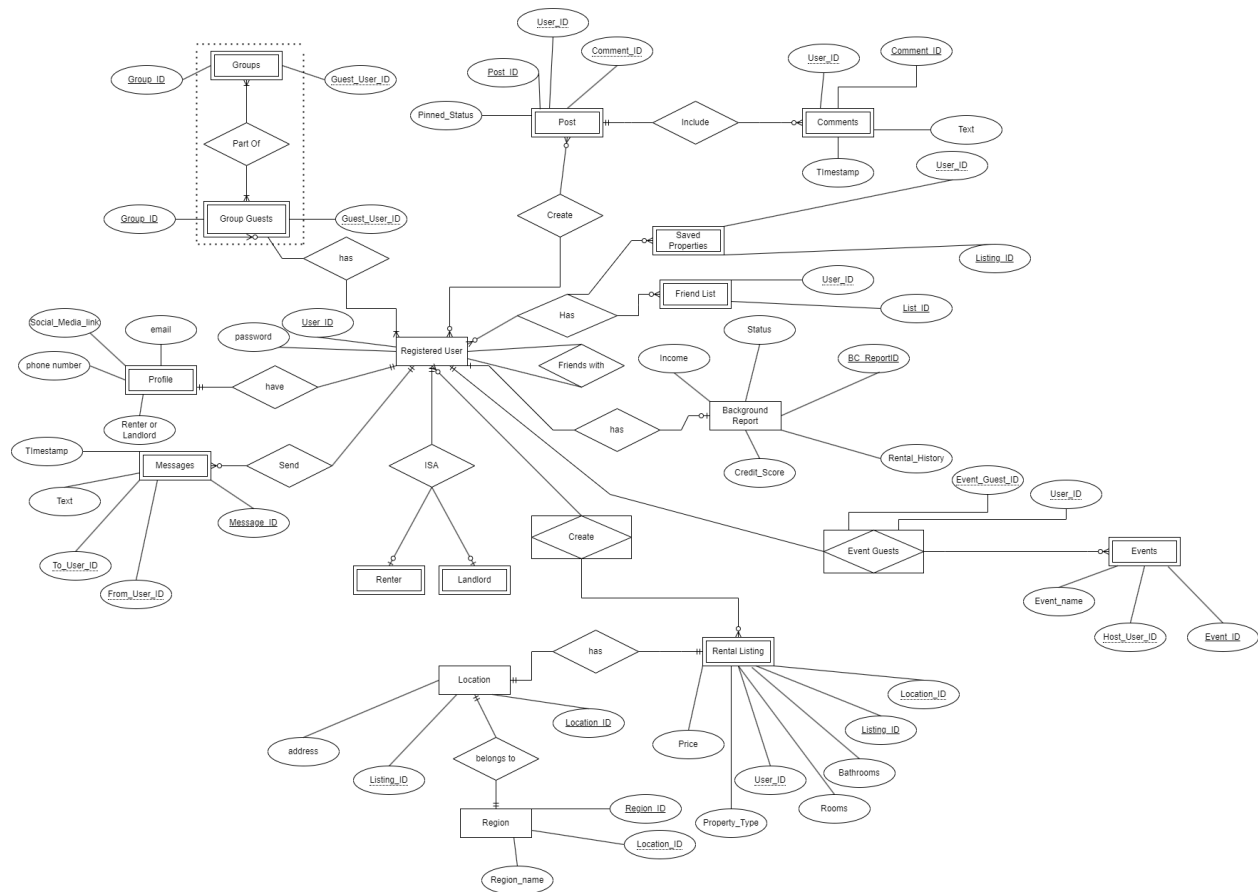
18. Reports

- Report_ID: Key, Alphanumeric
- User_ID: Key, Alphanumeric

19. Profile

- Profile_ID: Key, Alphanumeric
- User_ID: Key, Alphanumeric
- Email: Composite, 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.

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 roommates.

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 registrations.

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.

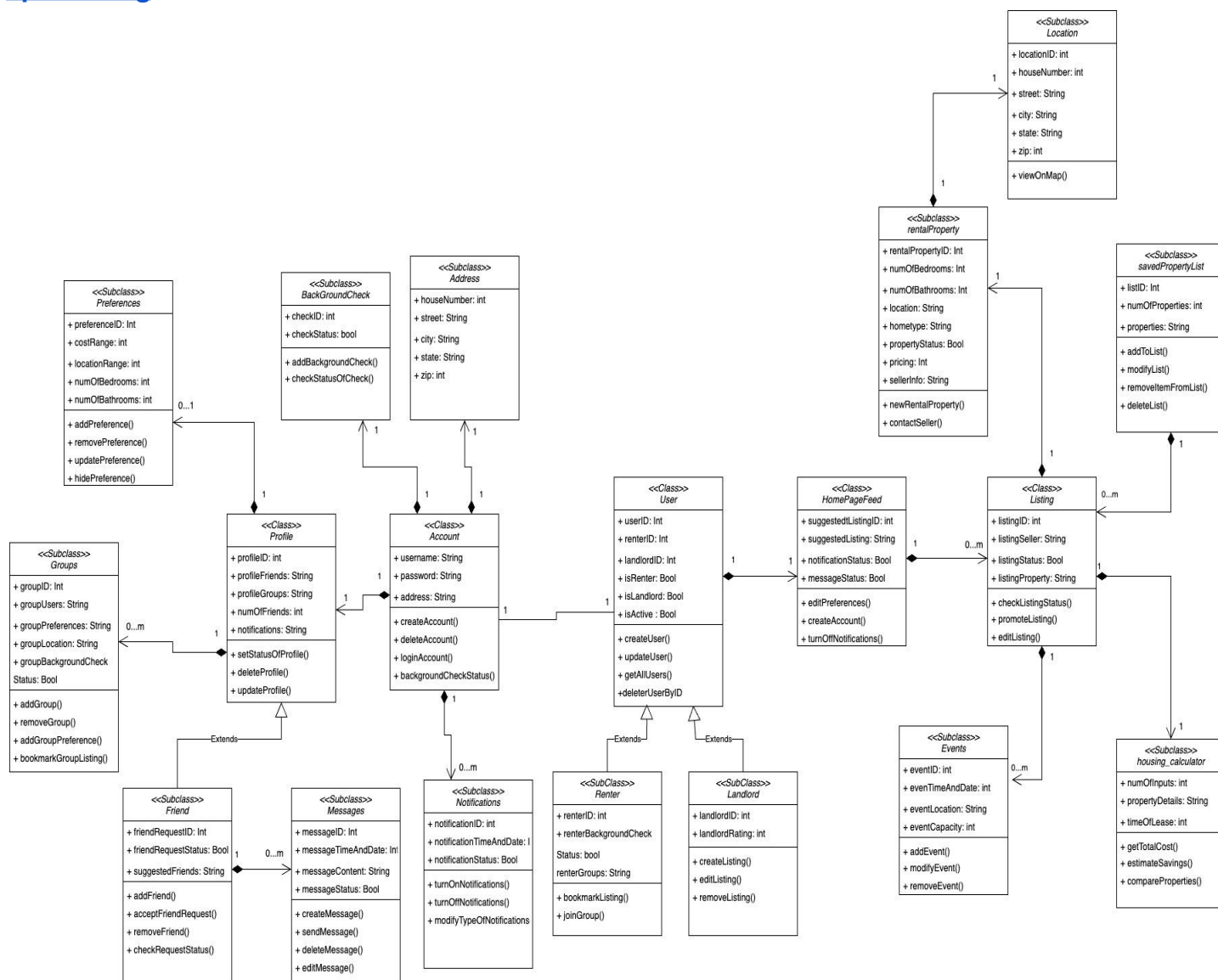
Finally there will be an interface for reporting all misdemeanors and inappropriate actions of another user with administrators being the final step in determining the status of whether or not to ban and blacklist an account.

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.

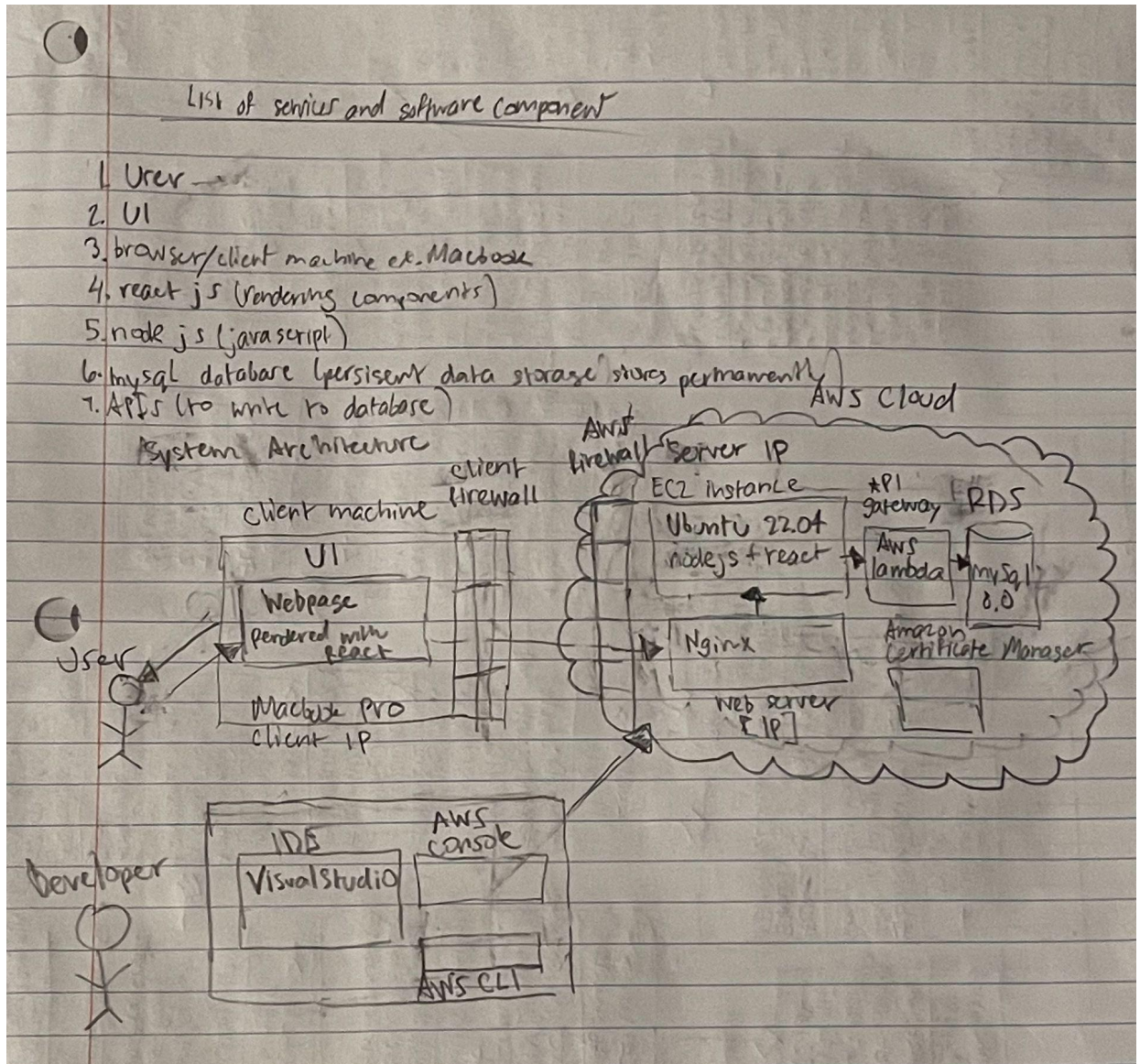
- Use data terms and names consistently with Data Definition Section 1 above.

https://drive.google.com/file/d/1OC0-y7JNa5F4_hOWmugzJgZMKJdR4jFo/view?usp=sharing



7. High Level Application Network and Deployment Diagrams

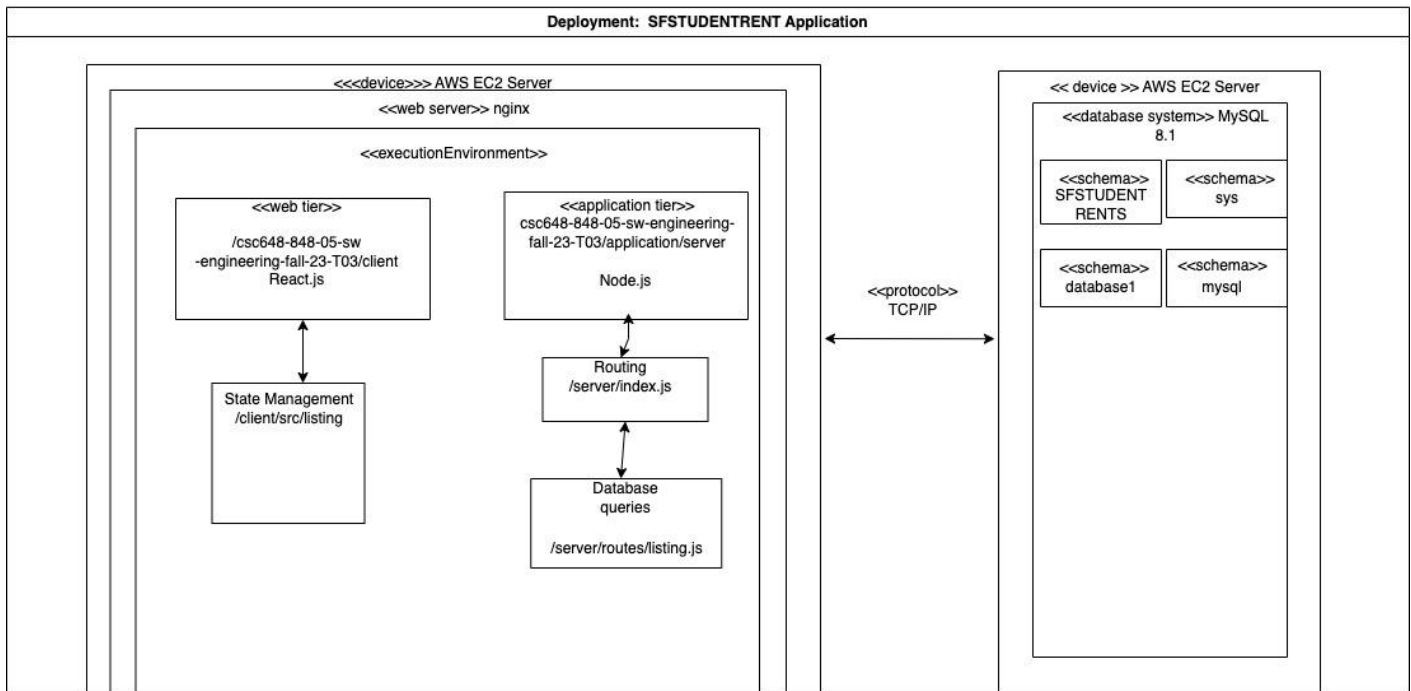
Application Network Diagram:



Deployment Diagram

<https://drive.google.com/file/d/1TMTLDTJMQIWBo8HwxzhOf1tqQy0nFjIq/view?usp=sharing>

Deployment Diagram



8. Identify actual key risks for your project at this time

• 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 tenant, 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	<ul style="list-style-type: none"> • Added data definitions for M2 • Prioritized functional requirement • Worked on Database Architecture <ul style="list-style-type: none"> ◦ 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 	

	<ul style="list-style-type: none"> • Created and invited team to Trello • Checked and edited M1 and M2 document for consistency 	
Geovanni Valadez	<ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> ◦ Split work among Ivan and Daniel • Attended every meeting and contributed in discussions • Has been on top of internal due dates 	10
Mozhgan Ahsant	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Added data definition for M2 • Attended every meeting and has contributed to discussions • Worked on Database Architecture <ul style="list-style-type: none"> ◦ Worked on database requirements, data definitions ◦ Created ERD and schema 	10
Aman Khera	<ul style="list-style-type: none"> • 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 Network and Deployment Diagram • Has attended every meeting and contributed to group discussion • Lead backend development of prototype 	6
Ivan Ayala-Brito	<ul style="list-style-type: none"> • 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 Enriquez	<ul style="list-style-type: none">• Worked on frontend development<ul style="list-style-type: none">◦ Login Page◦ Register Page	10
Alex Huang	<ul style="list-style-type: none">• Was proactive on getting caught up with the team• Organized the document	10