Design Analysis for *Storable*, an AirBnB for Storage

(Changes in Grey below)

1. Goals

Storable is an online marketplace for storage spaces. We bring people together to provide an easy self-storage solution. We make it easy for hosts to post listings about possible storage spaces they own, whether it's a closet, some space in their basement or garage. On the flip side, we make it easy for renters to quickly find storage space based on various parameters such as location, type of space, time needed, amount of space, etc. There are very few companies in this space; ShareMyStorage is a notable one that is run in the UK, but there is nothing targeted towards or optimized for U.S. residents. There are a lot of policies that will need to be perfected and those will be detailed in this design doc.

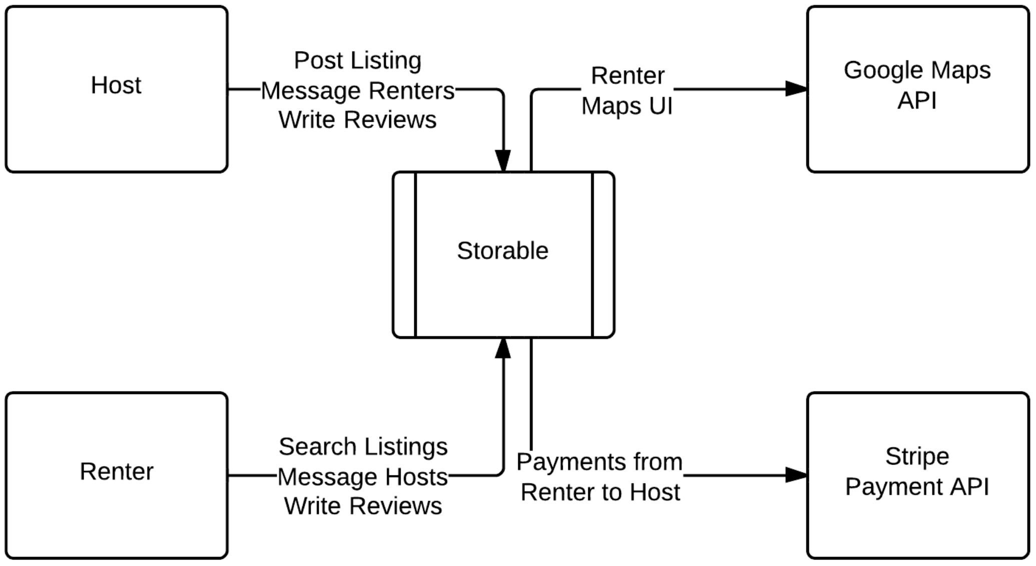


Figure 1: Context Diagram for Storable. A striped box represents an internal component, while the other boxes represent external components

2. Concepts

MVP Concepts:

A **user** is defined as anyone who uses the Storable site. A user is subclassified as either a **host** or a **renter**. A host is a user that has space available for storage, and a renter is a user that wants to rent storage space. Renters rent from hosts.

A host can create **listings**. A listing contains a **location**, price (per night), duration of time available, dimensions, and images of the space. A **location** consists of an approximate address (for privacy purposes).

When a renter expresses interest in a listing of a host, he creates a **message** from himself to the host. In doing so, he inherently creates a **conversation** between himself and the host containing the one message. Dialog between the host and renter will continue to add messages to this conversation.

Extension Concepts:

Once a host and renter have come to an agreement, they can proceed to create a **transaction**. A transaction contains information very similar to a listing, such as price and duration of time, but also contains identifiers for the host and renter.

All users can create **reviews** of other users / locations. A review will contain comments and ratings related to another user and/or location.

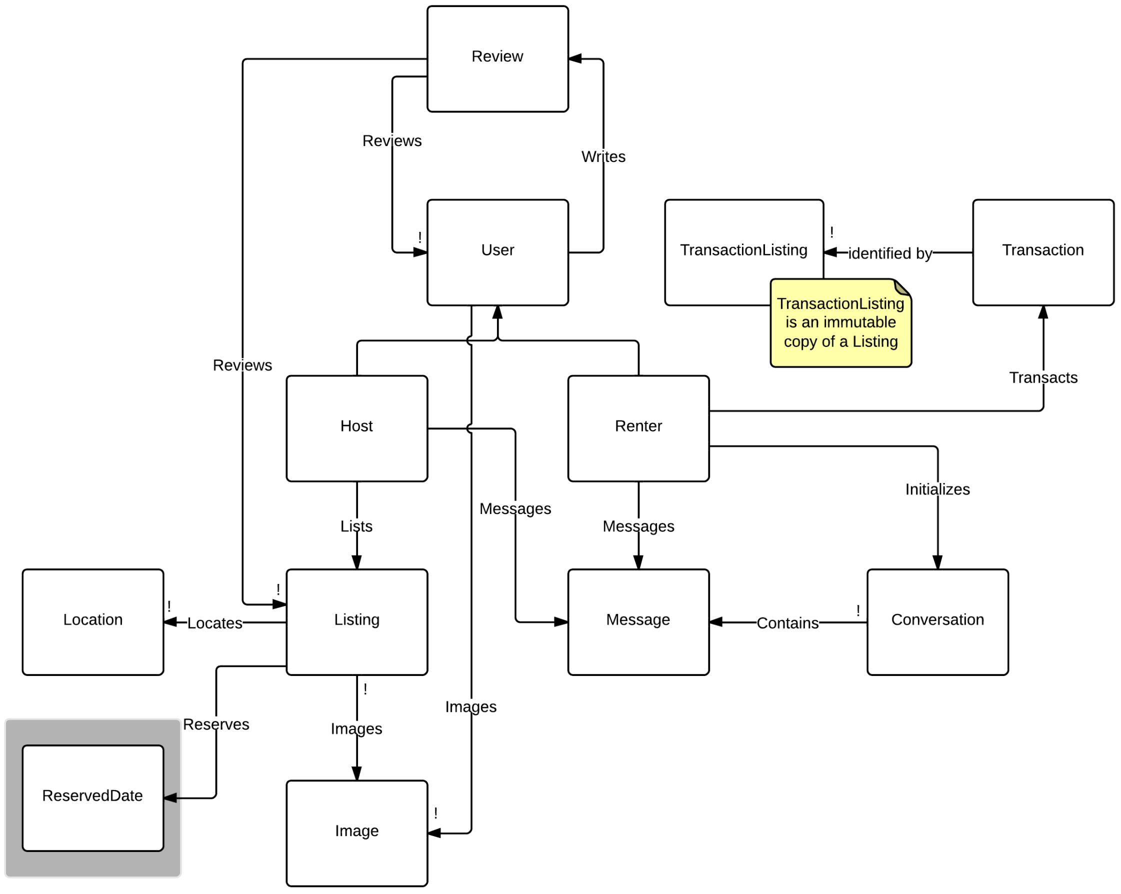


Figure 2: Object Model for Storable (Grey background denotes changes)

3. Behavior

3. 1. Features

Storable provides the following features:

MVP

- Host
    - Create a posting for his/her storage space
    - Respond to messages from potential renters
- Renter
    - See a list of all basic listing and details
    - Contact a host and inquire about details
- Privacy
    - No phone numbers or email addresses are displayed --> must use the internal messaging system, unless the host agrees to share contact information.

Extension

- Host
    - Receive payments straight through the site
- Renter
    - Search for storage spaces using various parameters such as location, type of space, time needed, amount of space
    - View reviews and ratings of hosts and be able to rate a hosts
    - Pay right through the site with integrated payments system (Stripe)
- Privacy
    - Hosts have their storage location anonymized so that only their general location (w/in a 1 mile radius) is revealed in search results

3.2. Security Concerns

Storable addresses the following security requirement: There must be robust user management in place. Only a host must be able to edit his/her listing and payments must be handled very securely.

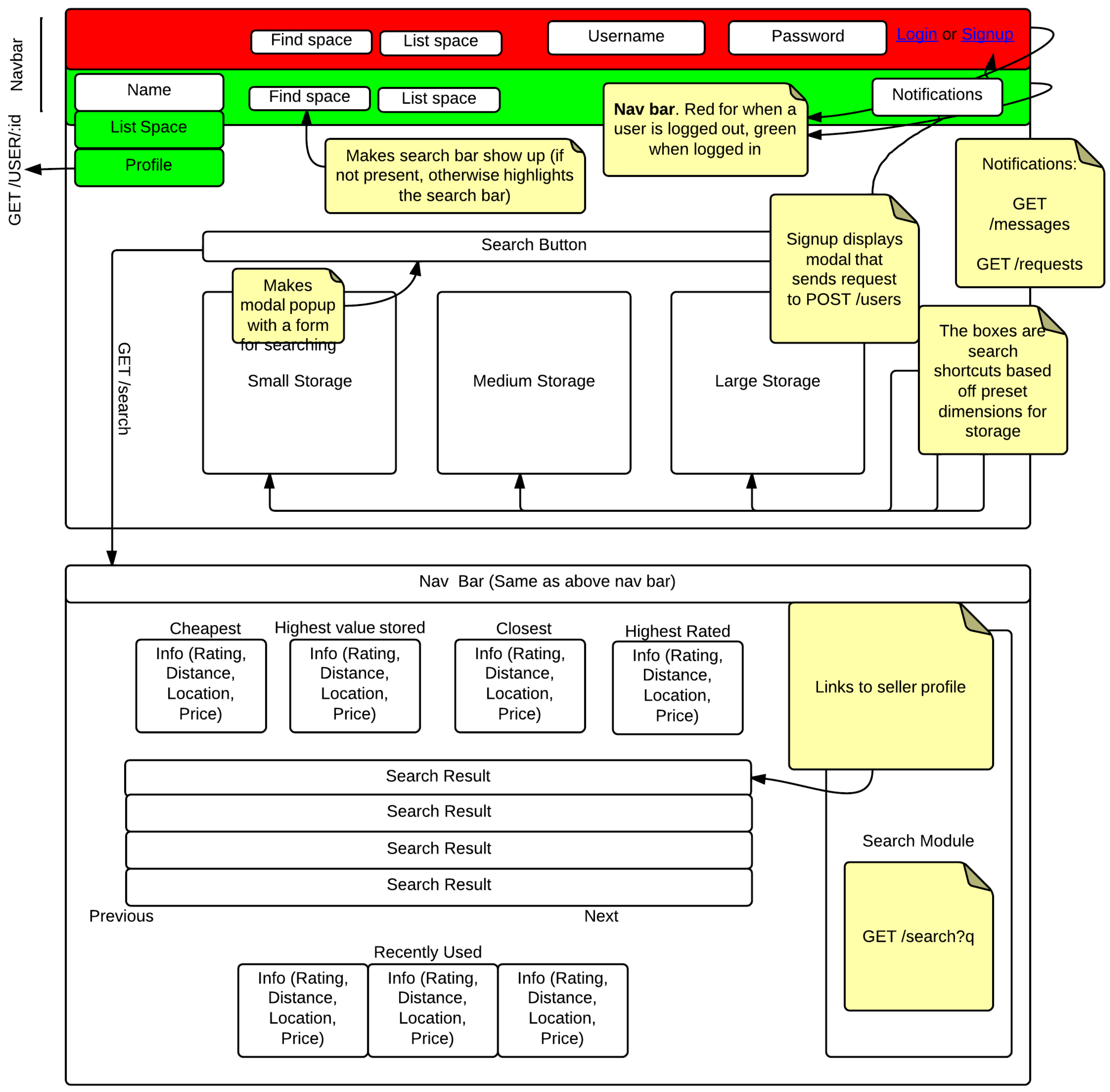
The possible security threats are:

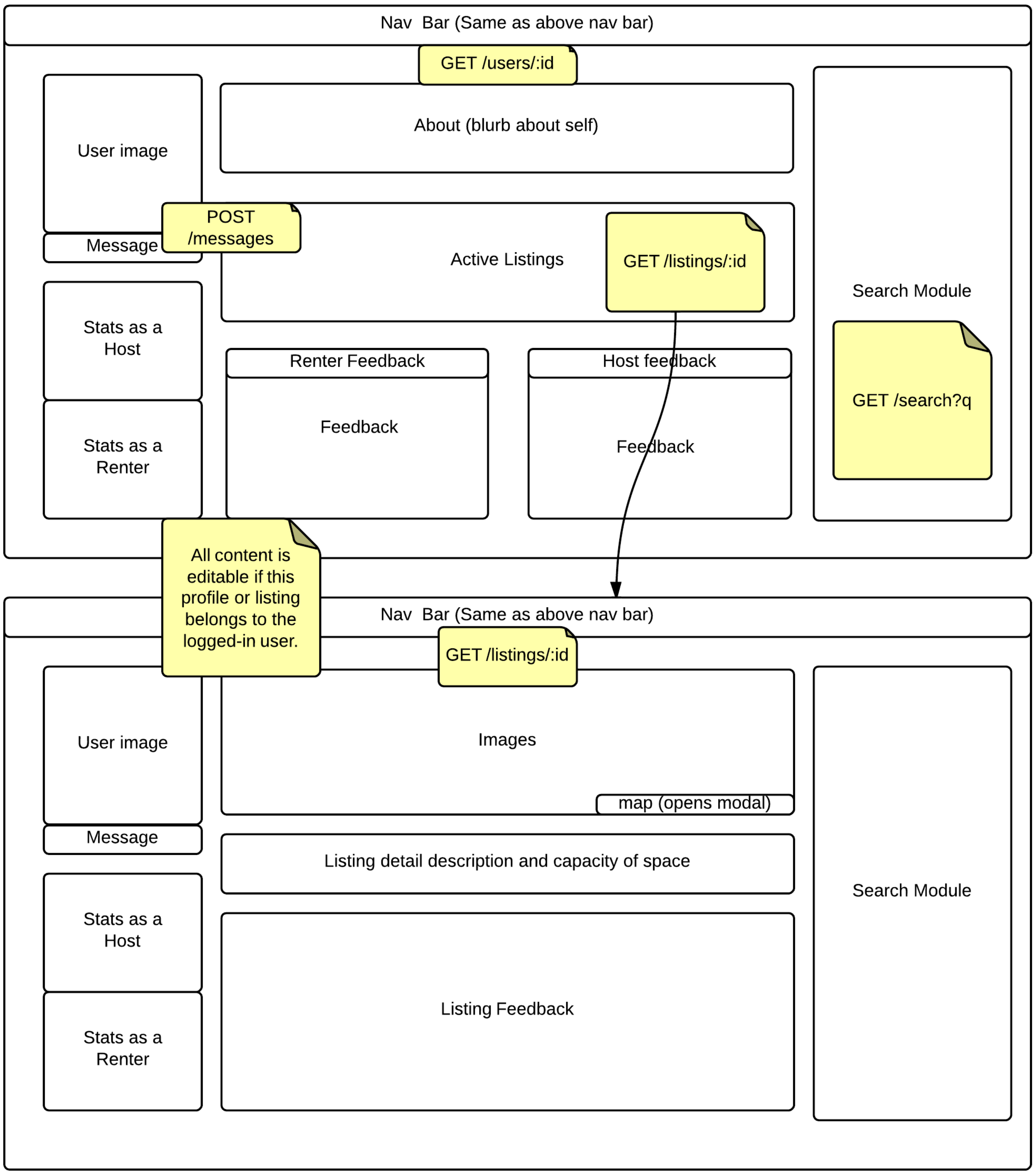
- Payment information must be PCI compliant otherwise if we store credit card information on our servers then there is a huge security risk of being hacked, etc. We will use Stripe so that we don’t store any sensitive information on our servers.
- Hosts could run away with the material that a renter has deposited. We will require that until the host has stored an amount of items worth a certain amount, the hosts put up a security deposit on the site until they have been vetted with enough good reviews/ratings.
- Renters could store illegal material such as narcotics, weapons, etc, so we will maintain that the host has the right to examine the renter’s property before accepting it.

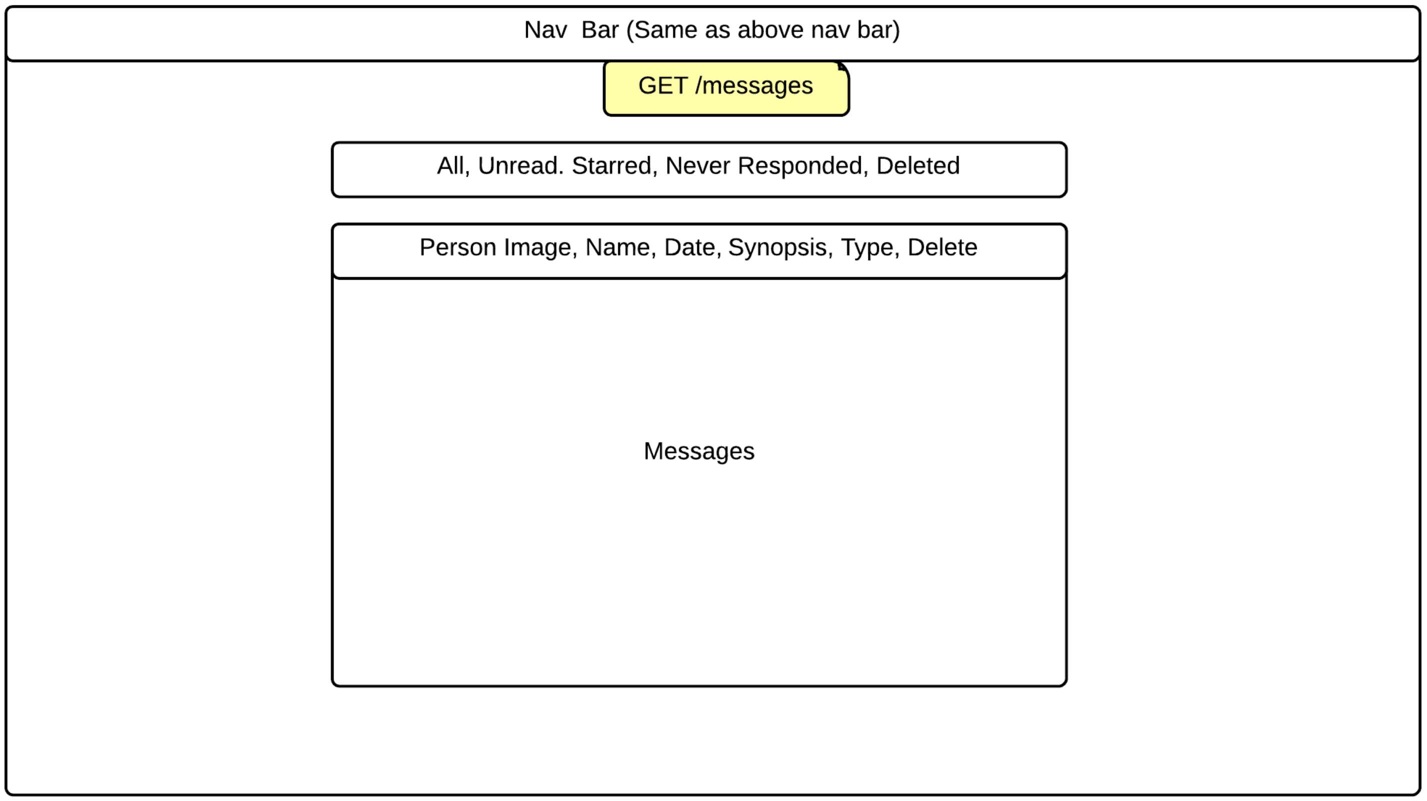
- An attacker could try to attack the database and get access to user's passwords. To address this, we will store only hashed versions of passwords with an appropriate salt that is different for every user.
- An attacker could sign up as a host and then attempt a cross-site scripting (XSS) attack by embedding malicious code in things like the storage space description. When this description is displayed to a renter, this malicious script could be executed. We will sanitize any external input into the system

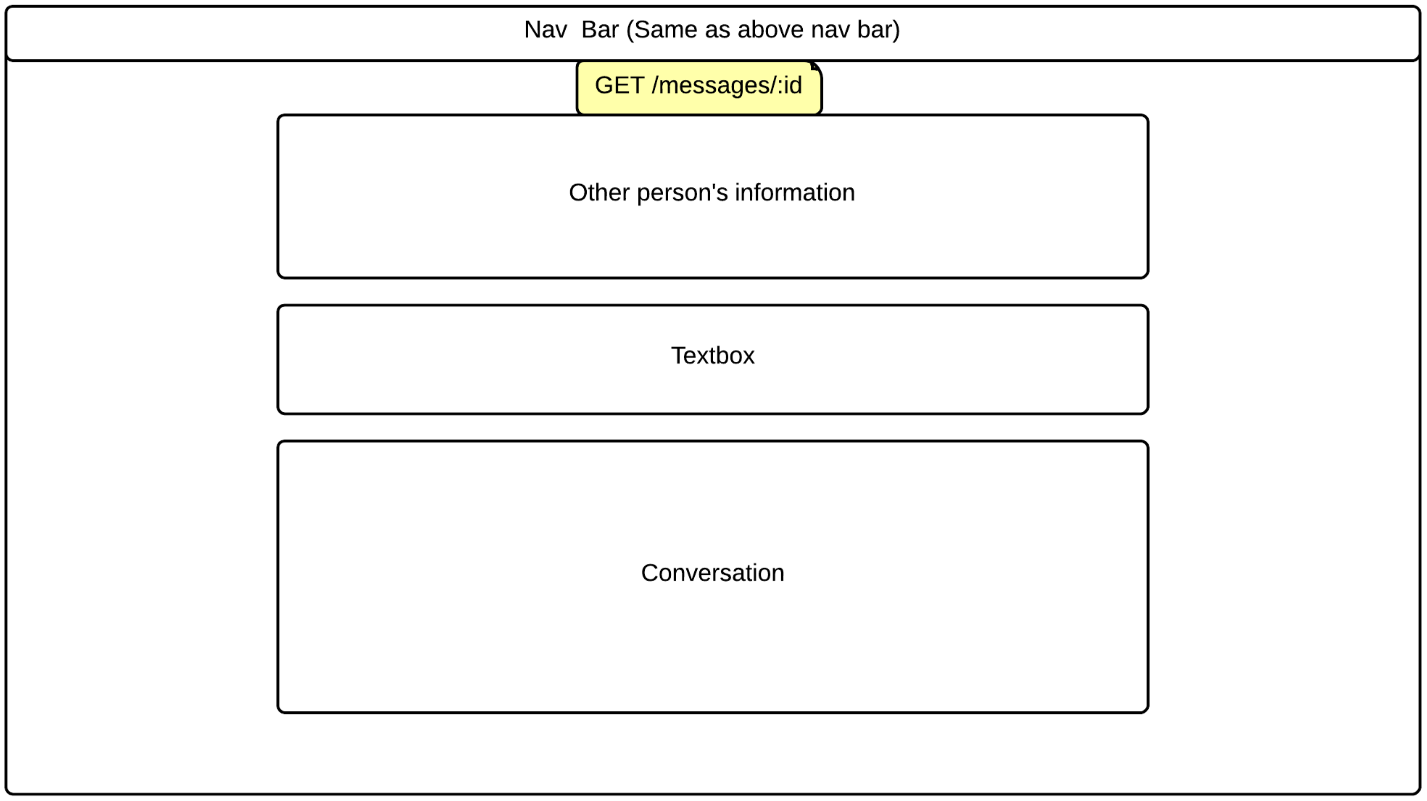
3.3. User Interface

The user interface is outlined below in Figure 3.


Navbar
GET /USER/:id
Find space
List space
Username
Password
Login or Signup
Name
Find space
List space
Notifications
List Space
Profile
Nav bar. Red for when a user is logged out, green when logged in
Makes search bar show up (if not present, otherwise highlights the search bar)
Notifications: GET /messages GET /requests
Signup displays modal that sends request to POST /users
Search Button
Makes modal popup with a form for searching
GET /search
Small Storage
Medium Storage
Large Storage
The boxes are search shortcuts based off preset dimensions for storage
Nav Bar (Same as above nav bar)
Cheapest
Highest value stored
Closest
Highest Rated
Info (Rating, Distance, Location, Price)
Links to seller profile
Search Result
Search Result
Search Result
Search Result
Previous
Next
Search Module
GET /search?q
Recently Used
Info (Rating, Distance, Location, Price)


Nav Bar (Same as above nav bar)
GET /users/:id
User image
About (blurb about self)
POST /messages
Message
Active Listings
GET /listings/:id
Search Module
Stats as a Host
Stats as a Renter
Renter Feedback
Feedback
Host feedback
Feedback
GET /search?q
All content is editable if this profile or listing belongs to the logged-in user.
Nav Bar (Same as above nav bar)
GET /listings/:id
User image
Images
map (opens modal)
Message
Listing detail description and capacity of space
Search Module
Stats as a Host
Stats as a Renter
Listing Feedback


Nav Bar (Same as above nav bar)
GET /messages
All, Unread. Starred, Never Responded, Deleted
Person Image, Name, Date, Synopsis, Type, Delete
Messages


Nav Bar (Same as above nav bar)
GET /messages/:id
Other person's information
Textbox
Conversation

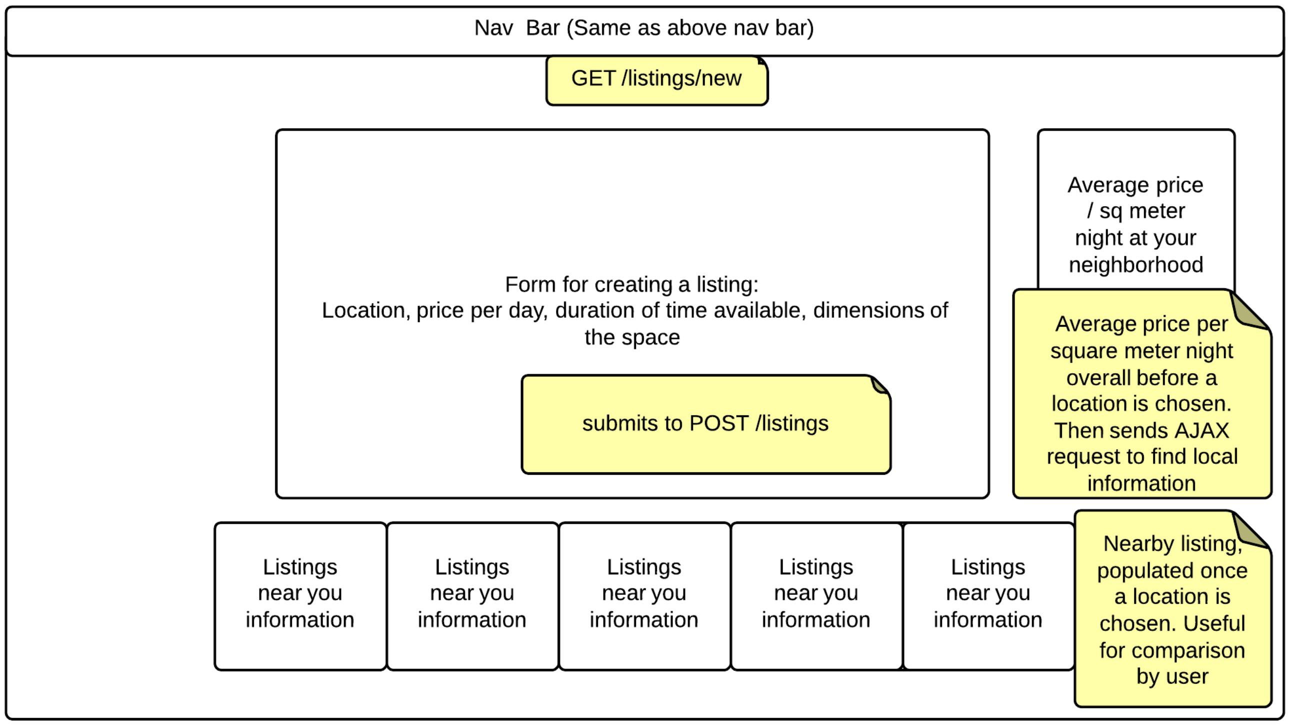

Nav Bar (Same as above nav bar)
GET /listings/new
Form for creating a listing: Location, price per day, duration of time available, dimensions of the space
submits to POST /listings
Average price / sq meter night at your neighborhood
Average price per square meter night overall before a location is chosen. Then sends AJAX request to find local information
Listings near you information
Listings near you information
Listings near you information
Listings near you information
Listings near you information
Nearby listing, populated once a location is chosen. Useful for comparison by user

Figure 3: User Interface

4. Challenges

4.1. Policy Challenges

Validity of Host Listings: Renters want to make sure that the host is reliable so that they can be sure their stuff is kept safe. They don't want the host to run away with their stuff.
1. Storable will screen hosts: There will be a screening process before hosts can become vetted hosts on the site
2. Use reviews and ratings: Renters will have to pick hosts at their own risk. We will provide them as much information as we can such as past ratings and reviews.
3. Design policies to ease host concerns: We will require that a host only store items worth less than a certain amount, until he/she has enough unique reviews / ratings to be essentially "vetted."

We will use a combination of options 2 and 3. We want renters to feel safe with their choice of host so we will try to provide them as much information as possible. Besides ratings and reviews of past users, we will try to design policies like the one mentioned in option 3 to assuage renter concerns. Since this is a marketplace, those hosts with higher ratings will naturally rise to the top in search results, so late comers will have to have stellar service and lower prices to get customers. Market forces will help ensure quality control as well. In the unlikely case that the host runs away with their stuff, we will use their stored payment information to charge them whatever the amount of the stolen goods. Note that since a new host can only store items worth a small amount of money, even if they do run away, it should be much easier for us to charge that penalty. A veteran host is unlikely to do illegal things like steal items.

Safety/Legality of Renter Storage Items: Hosts want to be sure that renters aren't storing illegal material in their storage spaces
1. Heavily fine renters who store illegal / inappropriate items: Since we have access to the credit card information of renters we can charge a substantial fine
2. Allow hosts to check the items of renters first: Give hosts the ability to go through all of the renter's stuff before accepting it into their storage space.
3. Have the renters sign an e-document that ensures they don't store illegal materials: this document can be used in any legal proceedings as a result of illegal behavior

We will use all 3 options to protect hosts from renter misbehavior. A mixture of policy, financial deterrents, and legal documents should be sufficient to protect hosts.

Payment Policies: Hosts want to get paid for their services, but they should only be paid provided that they complete their service for the duration of their listing.

1. Only pay hosts once the entire duration has passed: Keep the money paid by the renter in an escrow account, and only pay the hosts when the entire period of storage has passed.
2. Pay hosts on a weekly/monthly basis: Pay the hosts a portion of their fees every successful week/month that they carry out their storage services

We will use a mixture of options 1 and 2. As a host starts getting more good ratings we can fast-forward their payment schedules because we have confidence in them. When they first start out though, we will pay them at end of their service.

Host etiquette: Renters want to be sure that hosts will be able to faithfully hold their stuff for whatever the allotted time is. They don't want the host to be in a situation where he/she has to move away and their stuff is left stranded
1. Make it very clear as part of the site policy that the host cannot move away: The host commits to being around for the entire time period of storage. If they can't, then they should make it very clear in the listing.
2. Provide policy that lets a host move away if absolutely necessary: If a host moves away, then he/she is responsible for finding a temporary vetted storage location and paying for that location for whatever time he/she is unable to provide the service

We will use option 2 since we don't want to be draconian towards hosts if some extenuating circumstances arise. If a host has to move away, then he simply needs to find a professional storage location for the remainder of the storage period. Yet we will encourage hosts to not put themselves in this situation, i.e. only post listings when you're sure you can honor the time period required.

Cancellation Policy: Both renters and buyers want their bookings to go through without last minute cancellations.
1. Heavy fine for cancellation: If either party cancels, then charge high fines. These fines increase as it gets closer to the start date of the storage contract.
2. Force hosts to find another storage space for renter: Host will have to find a professional storage facility and pay for it if he/she cancels

We will chose option 1, in accordance with the policies that most other similar sites like AirBnb adopt. We will refund the renter the money he paid if the host cancels before the start date. If the renter cancels, then we give the host a portion of the cancellation fee.

4.2. Implementation Challenges

Access control: In order to meet security / usability requirements, we considered the following options.
1. Accounts: Both renters/hosts would be required to create an account with password protection
2. Only require account creation/login for certain actions: Renters would only need to create accounts when they want to book a listing or send a message to a host. Sellers would need to create accounts to create a new listing or view their messages, etc.
3. Don't require accounts/ just use cookies. A buyer would enter a key that would uniquely identify his/her store. Users could then just add items to their cart and order from it but there would be no persistence, i.e. when the cookie expired, so would the cart items.

We choose option 2 as it provides the most flexibility for both the host and renter. A host can create his/her listings and be sure that only they have the permission to edit it. Similarly a renter can feel comfortable paying for storage space on the site and know that their information is handled securely. To ensure this level of safety, we require that both sellers and buyers create accounts before doing any of the important, sensitive actions on the site. However, renters can still search for listings without logging in.

Design of Payments Infrastructure: Renters need to be able to pay hosts and hosts need to be able to get paid.
1. **Do payments processing ourselves:** Store encrypted versions of all the credit card / bank information for the users
2. **Use a third party API (i.e. Stripe):** Use another service for the processing so that we don't have to worry about PCI compliance or other security issues.

We chose option 2 since we wanted to focus on building the main product and not worry about all the problems that have already been solved like payments. We won't store any sensitive information such as credit card numbers on our servers.

Anonymization of location / profile information: Users might not want to reveal their email addresses, phone numbers, or exact location of their storage space.
1. **Allow the users to reveal their information using the messaging infrastructure:** Don't reveal the exact location or personal information in publicly viewable profiles but allow for the sharing of that information with a renter during a private messaging conversation.
2. **Make everything public:** Let everything be public and warn users to post at their own risk.
3. **Allow users to choose what to make public/private:** Put the power in the hands of the user. They should be able to choose what to make public. For example if they have an email account specifically dealing with their storage spaces then they might want to make that public.

We chose option 1 and 3 since we wanted to balance functionality with user privacy. By choosing these options, users can reveal information publicly that is not too sensitive and at the same time can share more sensitive information through private messaging provided by our site.

Ratings and Reviews: Renters want to be able to review both hosts for their service and listings for its various features. Hosts want to be able to review renters.

1. **Allow users to review others / listings at any time:** Let the user review anything, even if he/she hasn't specifically stored something in a storage space or interacted with a buyer.
2. **Only allow users to review once a transaction has gone through:** Users can only review each other/listings if a transaction has occurred. This makes sure that a renter has actually used a storage space and interacted with the host. A person can review right after the transaction goes through and doesn't need to wait until the end of the storage lease.
3. **Only allow users to review once a storage lease is complete:** A user can only do a review of a listing / user once the storage lease is complete

We chose option 2 because we wanted people to have the freedom to review in the middle of their lease without waiting till the absolute end. On sites like AirBnB most leases are several days or a week long so it makes sense to force the user to wait, but on our site, reviews could be several months long, so waiting that long might not make sense. This policy will need to be tested though based on user feedback.

Notes on Code Design: Most of our coding was pretty smooth. Early on we decided that the frontend would be written in Backbone so there was a very clear separation of concerns. All the backend was written to respond with json objects that could be assimilated into the Backbone models.

Some specific interesting issues did arise with the backend though. In designing an efficient search algorithm, we ran into several problems. Since a user could search using a mix of several filters, it was complicated to come up with an efficient query to accommodate all of them, so we ended up using a hybrid approach. For example if the user wanted to search by all listings that were available between a certain start and end date. We'd first do a query to filter all listings whose listed availability was between the dates given. Then we'd check if any of these listings already had reserved dates that conflicted with the given dates. These conflicts were stored in an exclusion set and subtracted from our candidate set of listings. Then for other search filters (like amount of storage space needed), we could just do the query on the candidate set by appending a where clause; this would lead to much more efficient queries since our search space is not all listings but just the candidate ones.

Moreover, we were using a ruby gem to do geolocation search and it only operated on models that had a latitude and longitude defined. Our listing model contained a reference to the location object, so we would have to do the query on the location model and then find the set of all the parents and return those from the search method.

Also for payments, we realized that after a renter pays for a listing, we need to wait for the host to accept it before we can charge the renter's card. To do this securely, when a renter tries to book a listing, we create a transaction but don't actually charge the renter. We store a Stripe token instead that references this transaction. We then send a message to the host, asking if the payment can be approved. If the host agrees, then the transaction identified by the token is carried out and the transaction is complete.