All I Want [for Christmas]

Cristhian Ulloa, Dirk Stahlecker, Kristin Asmus, Tricia Divita

[Updated] Design Doc

Changes

Unsubscribe Functionality

Our app was planned to send users email notifications for various actions - confirming that you've claimed a gift, alerting you when the owner of a wishlist modifies or deletes the list itself or a specific gift that you've claimed, and notifying you when a gift you've split with others has reached 100% of the cost covered. We realized that some users might wish to opt out of these emails, and since we don't support deleting accounts, there should be an option to unsubscribe. This is manifested as an Unsubscribed User in the data design, a subtype of User who will not receive email notifications. It will also create a small change to the Account Page's UI by including a checkbox to subscribe or unsubscribe to emails.

Public Wishlist Discovery Page

This is the biggest change to our design. We wanted to include public wishlists that can be seen by anybody; however, our UI only offered the ability to navigate to wishlists that you created or that have been shared with you. To remedy this, we've added a Public Wishlist Discovery Page to our wireframes and page flow diagram, where the user will be able to search for relevant wishlists by their titles and descriptions, and if desired share these wishlists with themselves to save them to their sidebar.

Motivation

All I Want [for Christmas] is an online app with the goal of simplifying and optimizing gift giving. Its creation is motivated by the fact that gift-giving is often a disorderly task, and many people do not have a system to handle it well.

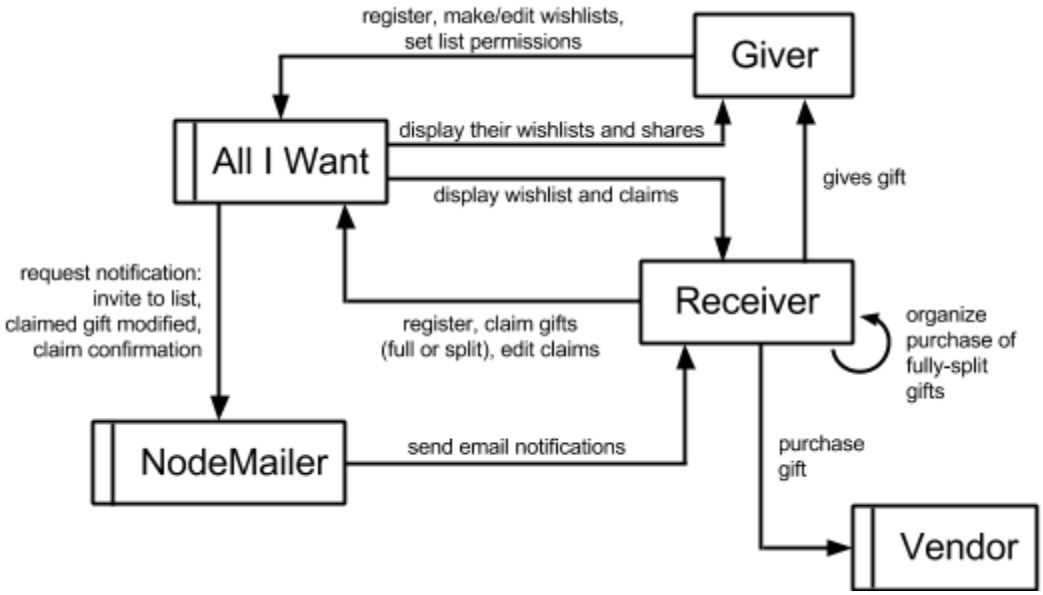
The purposes are as follows:

1. **Get the gifts you want.** Users create wishlists of gifts they want, and share them with their friends or make them public, so that everyone knows what they would like to receive.
2. **Give only gifts you know they'll like.** Other users can view wishlists and choose to give one or more of the desired gifts.

3. **Avoid duplicate gifts.** Users can make claims on gifts in a wishlist, so gift givers can view what gifts have already been claimed, and avoid getting the same gift for a user.

There are a few aspects that existing solutions do not allow for.
CheckedTwice is one existing site that has much of our functionality, with a few significant exceptions. in CheckedTwice, each user only has a single wishlist, and gifts have the option of being shared differently. But there is no option for sharing with the public, or even with specific users. All gifts are either private or shared with friends and family.
Other more widely known options, like *Amazon Wishlist* or *store registries*, only allow for gifting of specific items from their associated vendors. Our solution will enable users to link to specific items on the web or include descriptive info for a product from a specific store, but will also offer them the freedom of describing what they want without having a specific vendor's product in mind.
Another major point is that there do not seem to be any solutions that allow for the splitting of gifts between multiple users, which will make our app unique in its functionality.

Context Diagram



Concepts

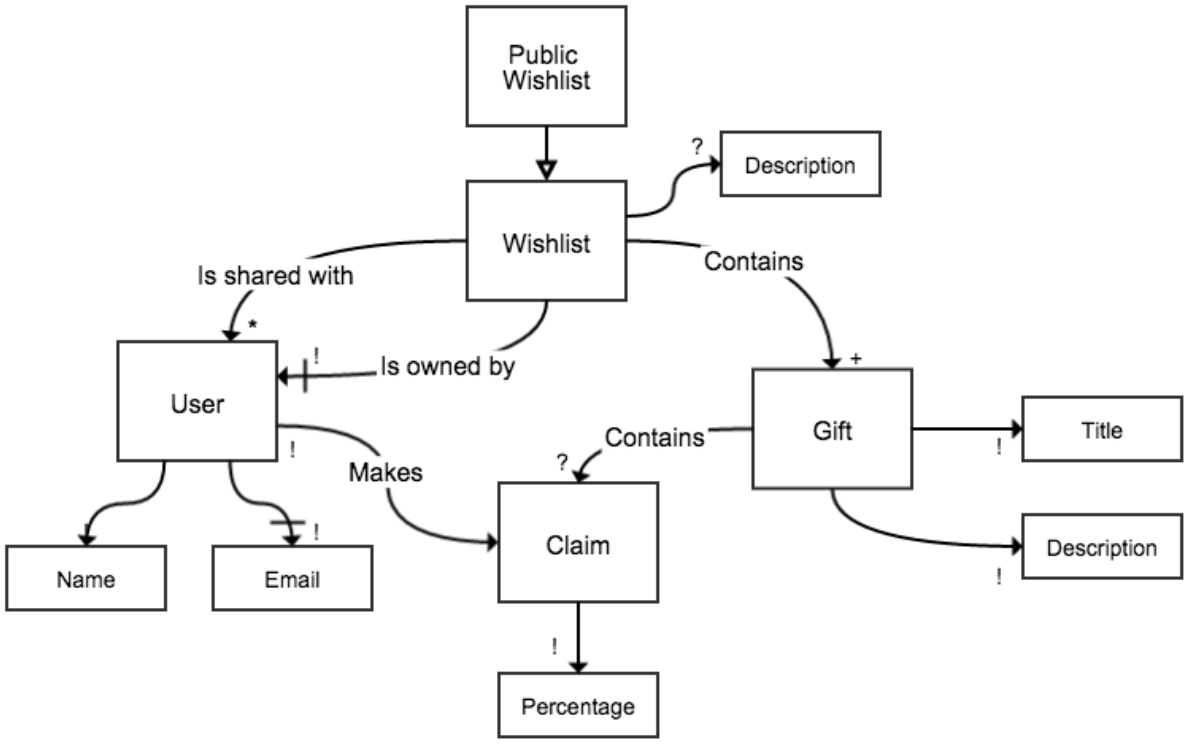
Wishlist: A list of gifts a user is requesting. A wishlist can be shared with other users by its creator, so that they can view it to help them decide what to give the wishlist owner. (Motivated by purposes 1 and 2.)

Gift: An item a user wishes to receive. Each gift belongs to a wishlist of a specific user. (Motivated by purpose 1.)

Claim: A promise that one user will give a wishlist gift to its requester. Users can only claim gifts from wishlists that have been shared with them. (Motivated by purpose 3.)

Split Claim: Like a claim, but the claimant specifies the percentage of the gift that they are willing to cover.
(Motivated by purpose 3.)

Data Model



Gift can only be claimed by users who can view the wishlist containing that gift

Private wishlists can only be viewed by users that the list is shared with

Public wishlists can be viewed by anyone, but can also be shared with specific users

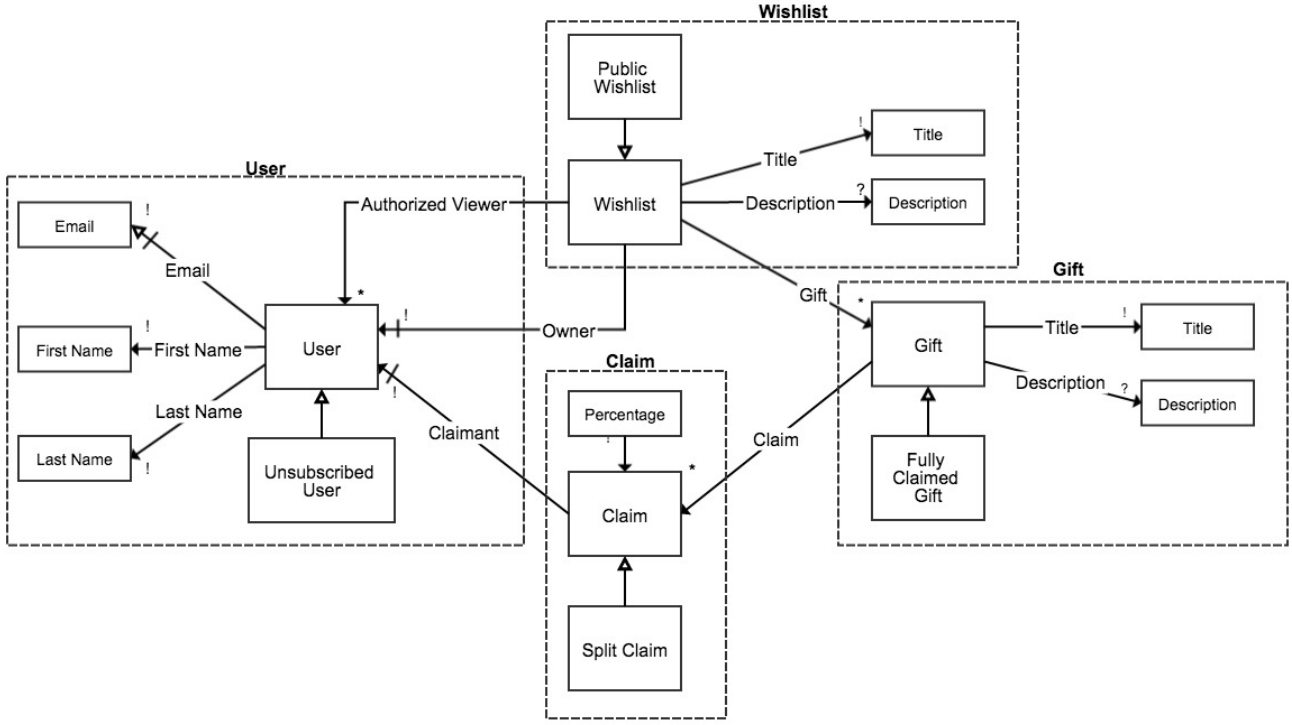
Users can't claim gifts on a wishlist they own

Data Design

All I Want has four main models: Users, Wishlists, Gifts, and Claims. The reason that they are separated in this manner is due to the references that each make to one another. Both wishlists and claims have a user associated with them, so neither of these three can be combined. Likewise, gifts are associated with both wishlists and claims.

The reason we chose not to combine claims and gifts is that separating them helps display them more easily in different contexts, such as showing claims in a user's claimed items list, or displaying gifts without their claims on a user's own wishlists. The separation also allows for much easier filtering for individual claims or gifts.

Subclasses were used to represent different states. For example, an "unsubscribed user" subclasses a user because an unsubscribed user has the property of not receiving emails, while a user does receive emails. This representation was used to avoid using a boolean to represent the different states.



Gift can only be claimed by users who can view the wishlist containing that gift.

Private wishlists can only be viewed by users that the list is shared with.

Public wishlists can be viewed by anyone, but can also be shared with specific users.

Users can't claim gifts on a wishlist they own.

A splitclaim has a percentage less than 100% while all other claims (full claims) must have a percentage of 100%.

Unsubscribed users are the same as a regular user, except they don't receive email notifications

Security Concerns

Key Security Requirements:
Because our app involves only claiming gifts and not directly purchasing them, our security concerns are considerably less critical. We want to be able to authenticate users mainly for the purpose of being able to share wishlists with other users with the assurance that only the intended people can actually see it. Users will sign up with an email, name, and password, and may be required to verify their email in order to authenticate. This account information will be hidden from others and encrypted as necessary, with the exception of their name.

Mitigating Standard Attacks:

- **Hoax public wishlists posing as charities.** Because we allow anyone to make an account and create a public wishlist that anyone can contribute to, it is definitely possible that a user could create a public wishlist posing as a charity for their own personal benefit. However, because our site only enables users to claim gifts, they must deliver gifts on their own. There are two likely scenarios for this - a user could hand-deliver the gift, or mail a package. If they hand-deliver it, they can likely judge for themselves the trustworthiness of the receiving organization. If they mail it, the owner of the public wishlist must include an address or contact info in the description, which conscientious users could verify independently. To encourage this practice, our site will include notes on public wishlists encouraging creators to include information to prove their validity, including a verifiable contact number, address, and/or website. Likewise, we will include a note warning users that All I Want does not verify the owners of public wishlists.
- **Grinches (especially on public pages).** One possible abuse of the claiming system is that a malicious user (a “grinch”) could claim items from a wishlist with the intent of never giving them, and have their claims discourage other users from giving those gifts as well. This is mitigated on private wishlists by the fact that the owner can choose who is able to view and therefore claim items from them. However, the worst case scenario would be a grinch on a public wishlist who arrives uninvited but unimpeded, and claims all of the gifts so that other users who might have contributed give nothing. We’re still considering how best to combat this, but one potential solution would be to allow public wishlists to disable claims completely (especially for wishlists where there’s no need to limit the number of each gift received). Another solution could be to limit the number of claims a user can make on each public wishlist.
- **Hackers attempting to extract user information (emails and passwords).** We will encrypt emails and passwords in our internal database in order to mitigate this threat.
- **Malicious data entry.** We will clean all user inputs to attempt to prevent injection attacks.

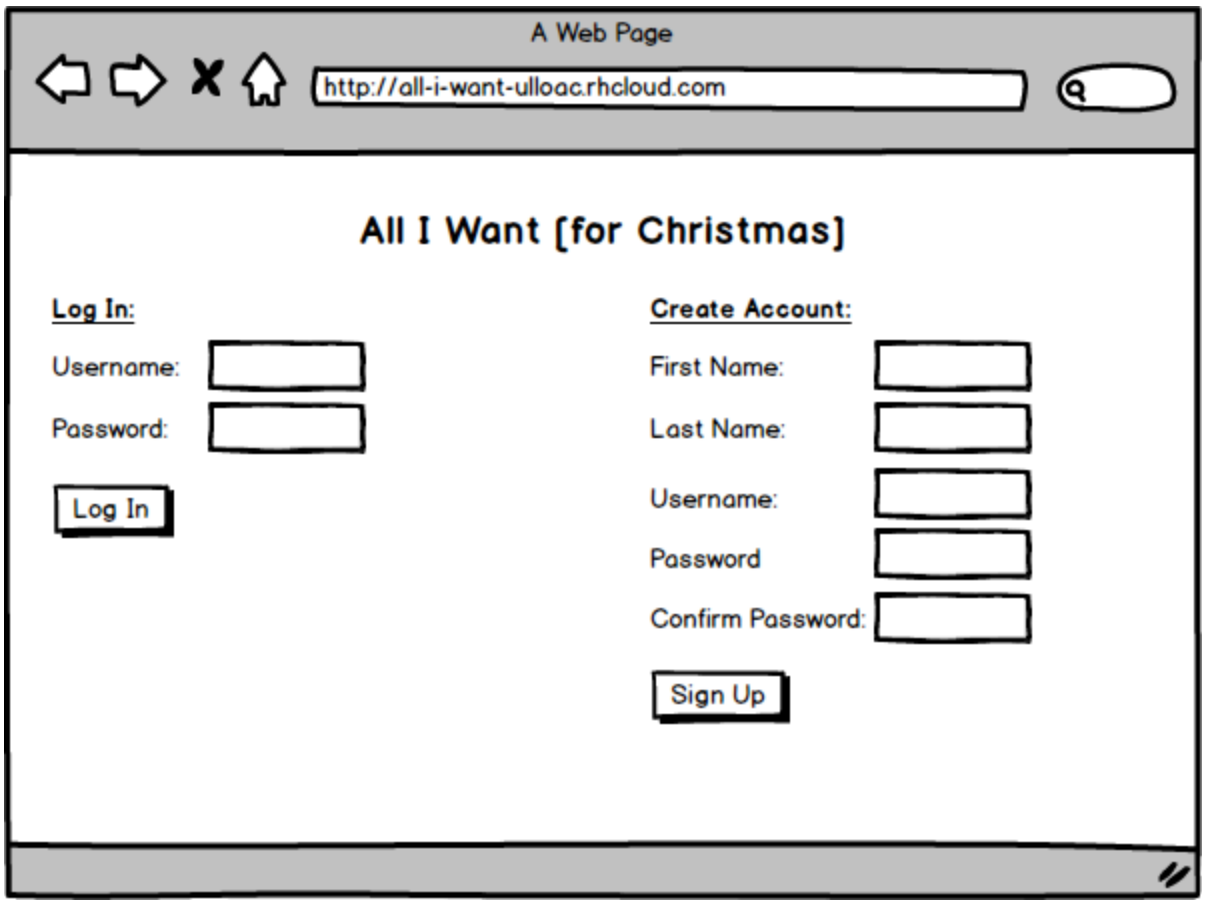
Threat Model

- Can assume no major criminal attacks due to the small scale of the project and the minimal information stored (email).
- Hoaxes are less likely given that our project doesn't involve actually purchasing and sending items. However, this is still a threat that should be given some consideration.
- There isn't much motivation for grinches, but the internet is full of trolls so we should do our best to mitigate the negative effects they could cause for other users.

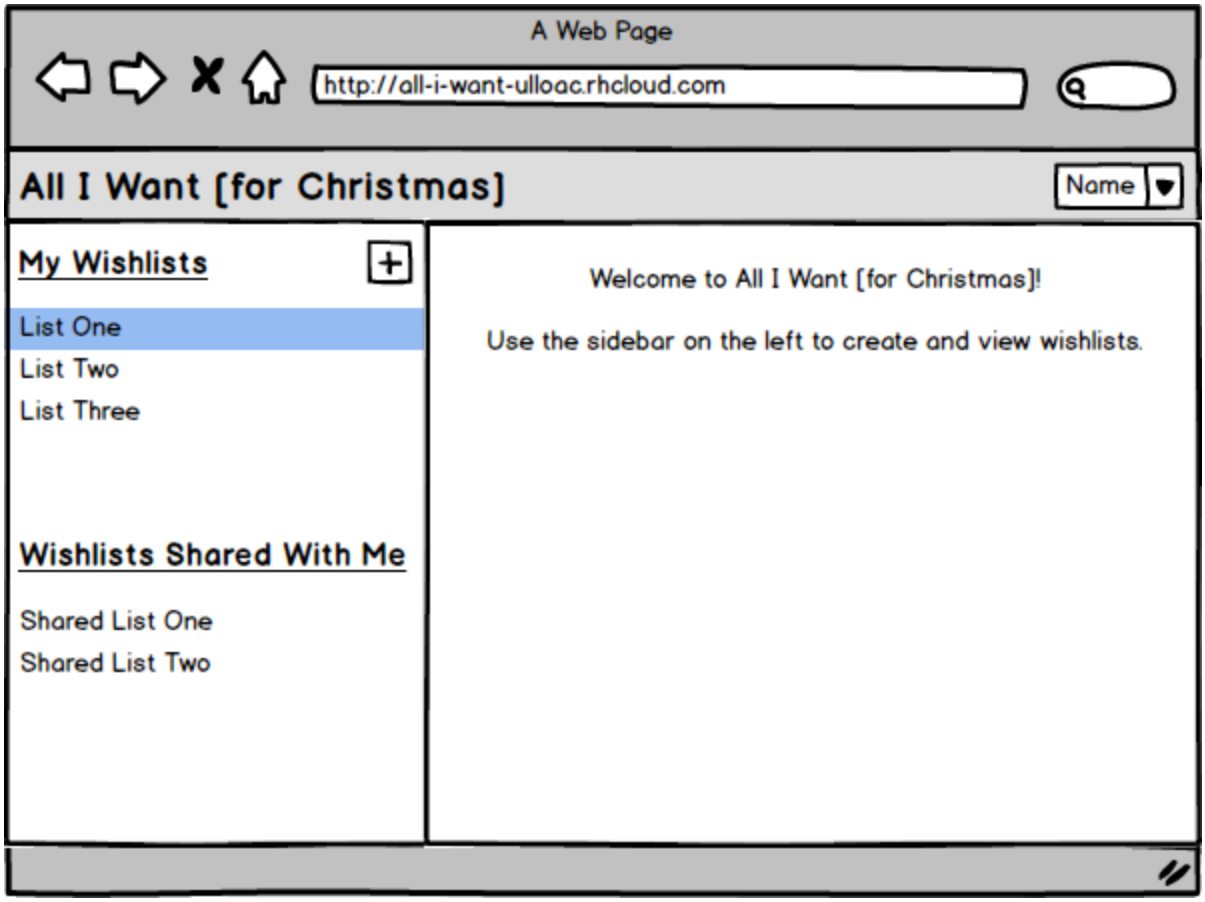
User Interface

Wireframes

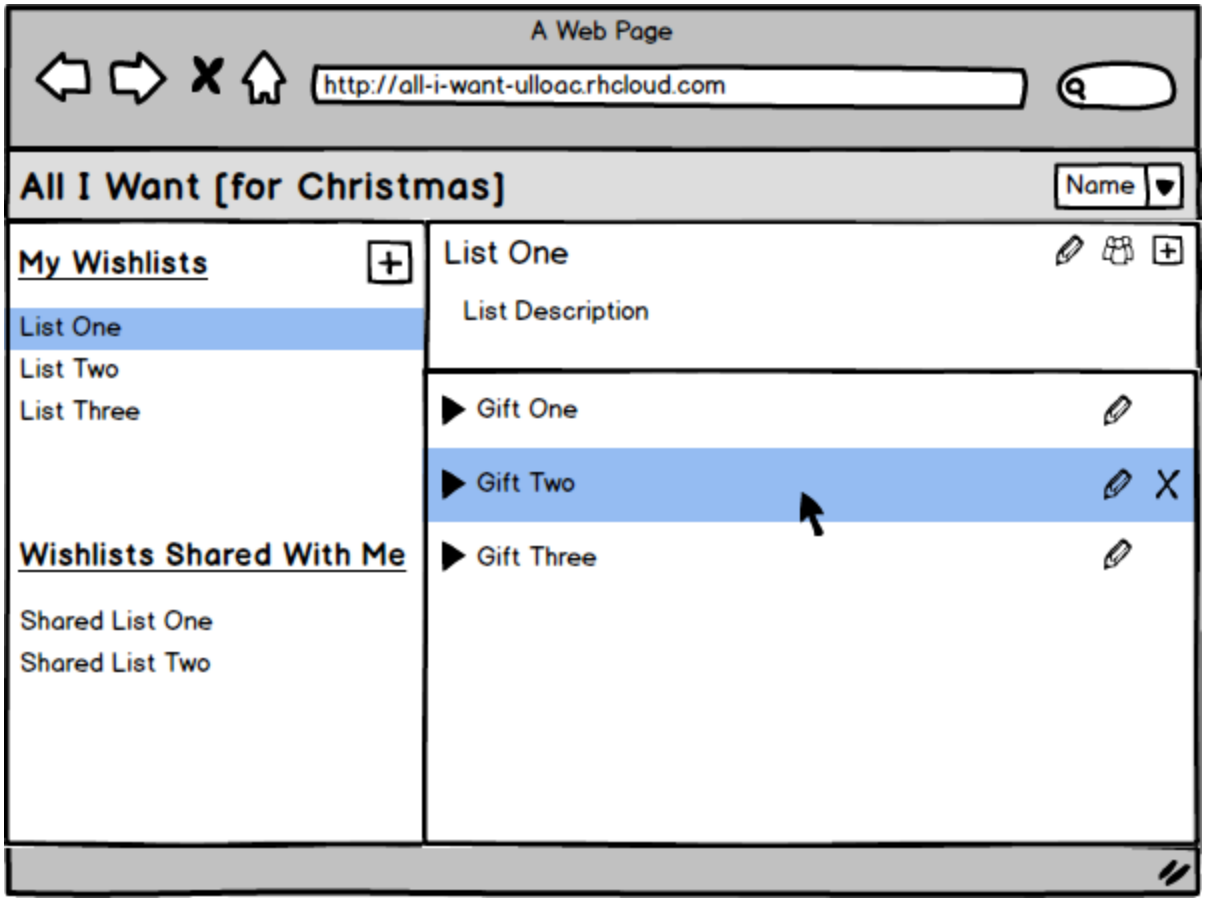
Login Page:



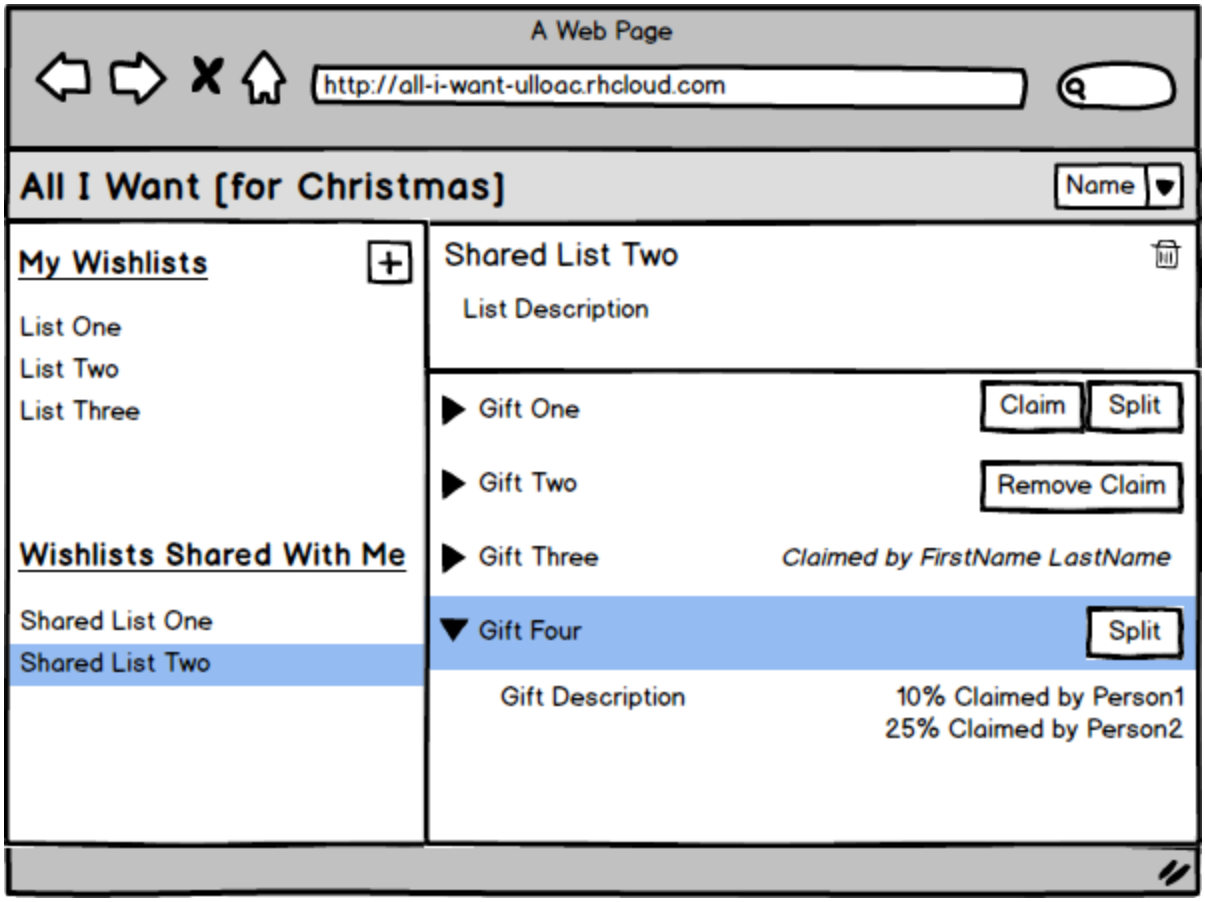
Landing Page:



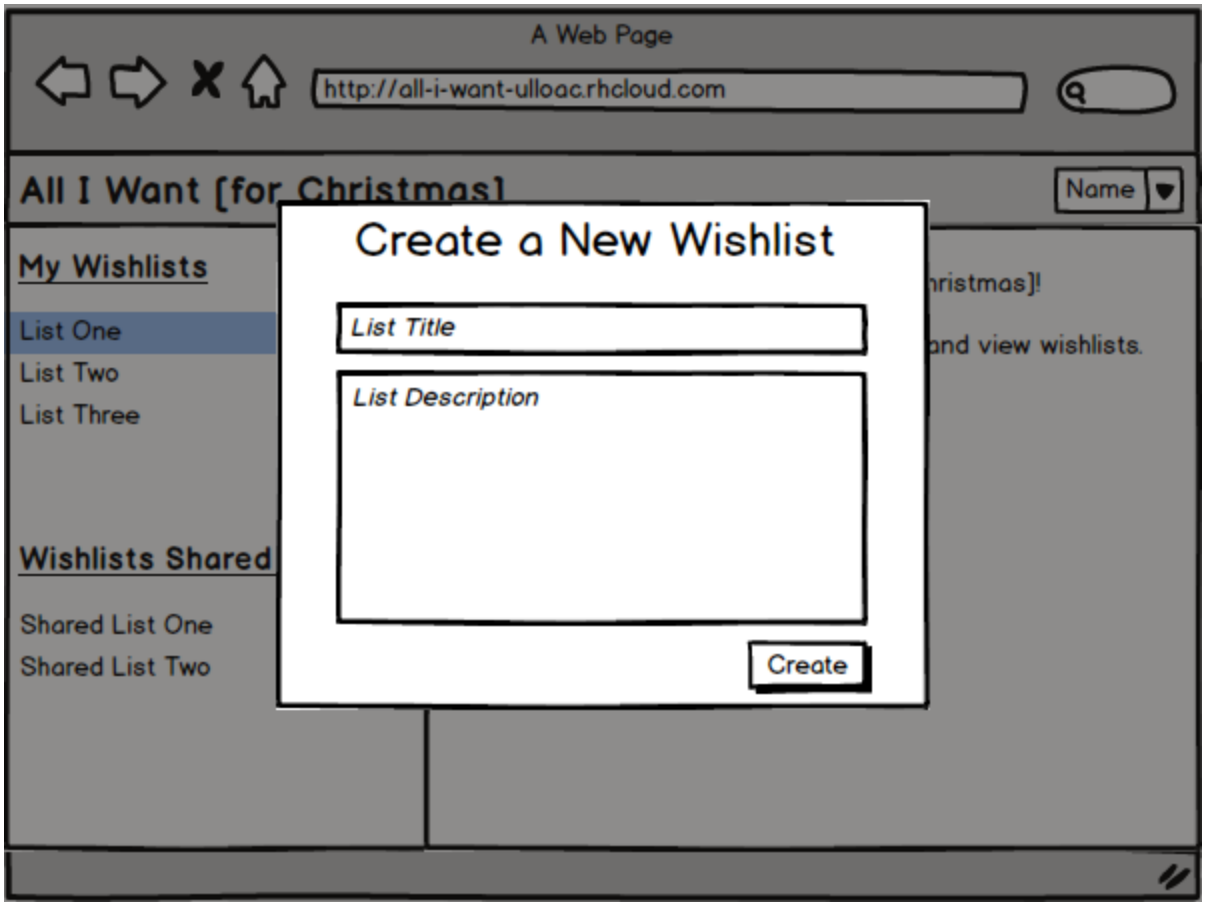
My Wishlist Page:



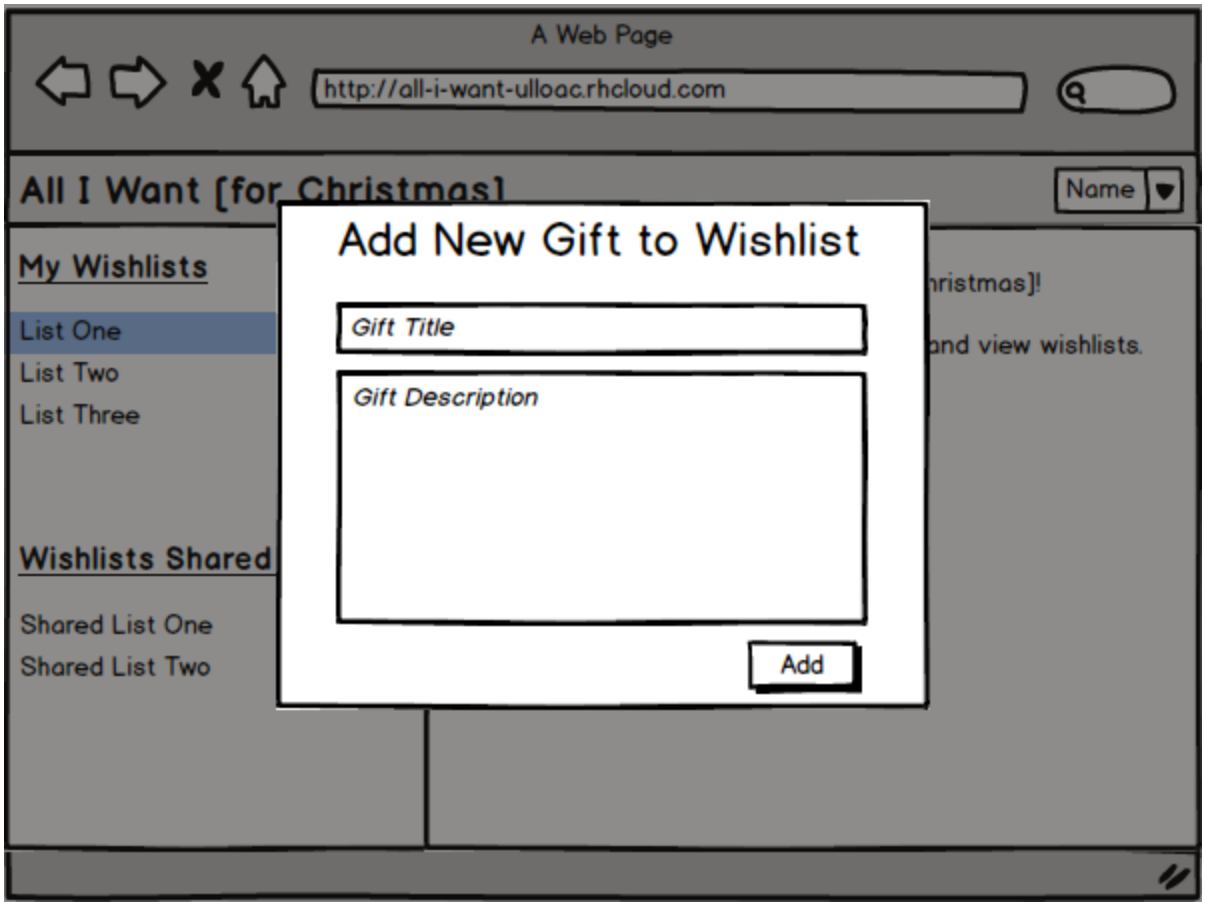
Wishlist Shared with Me Page:



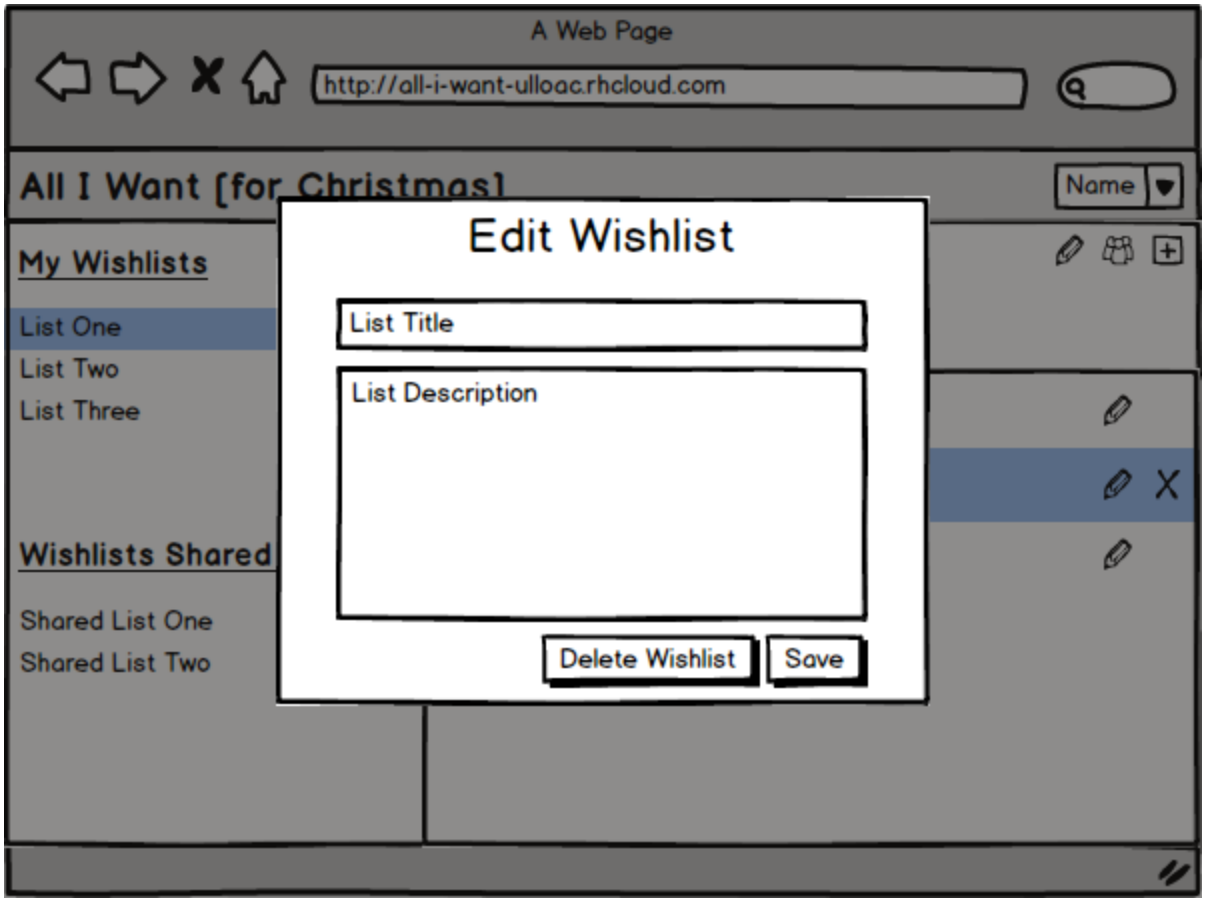
Create Wishlist Modal:



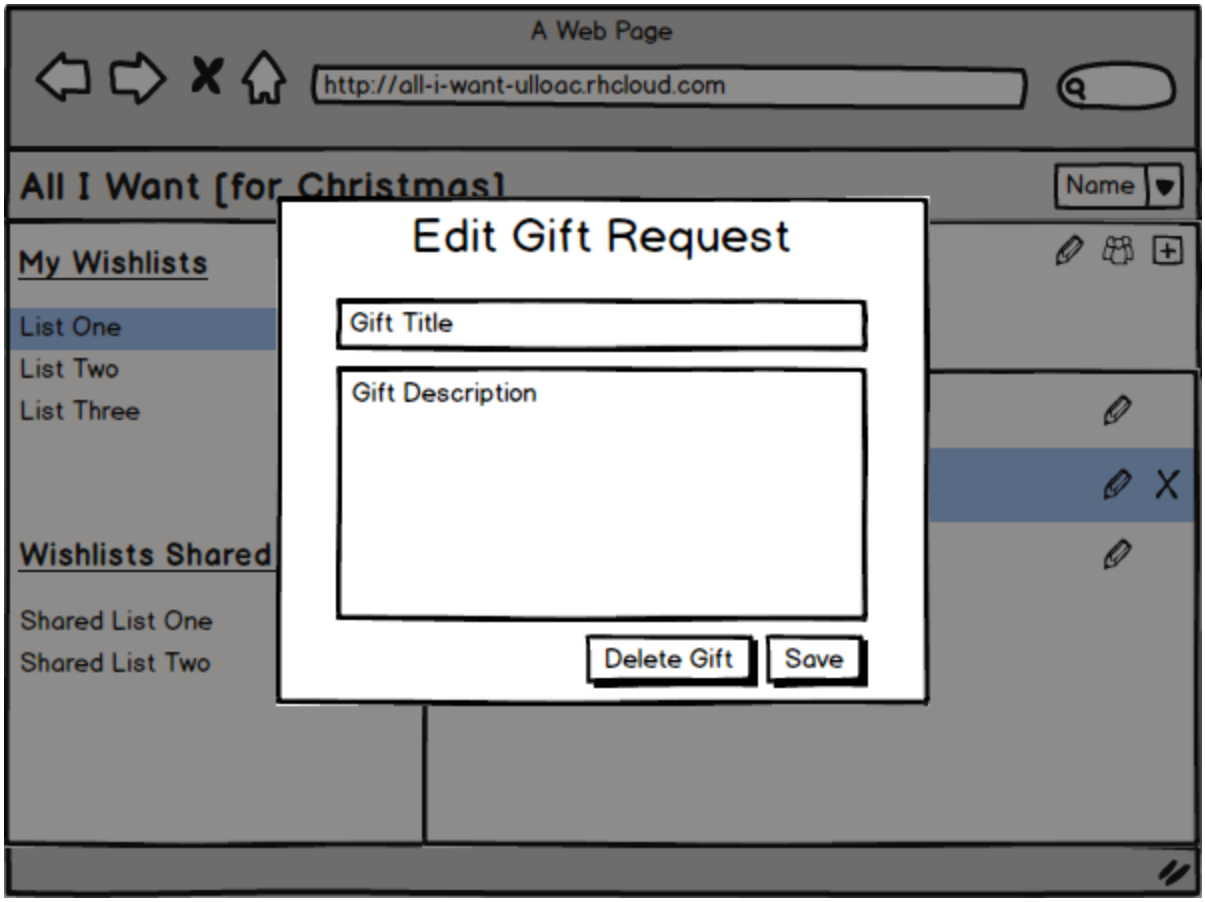
Add Gift to Wishlist Modal:



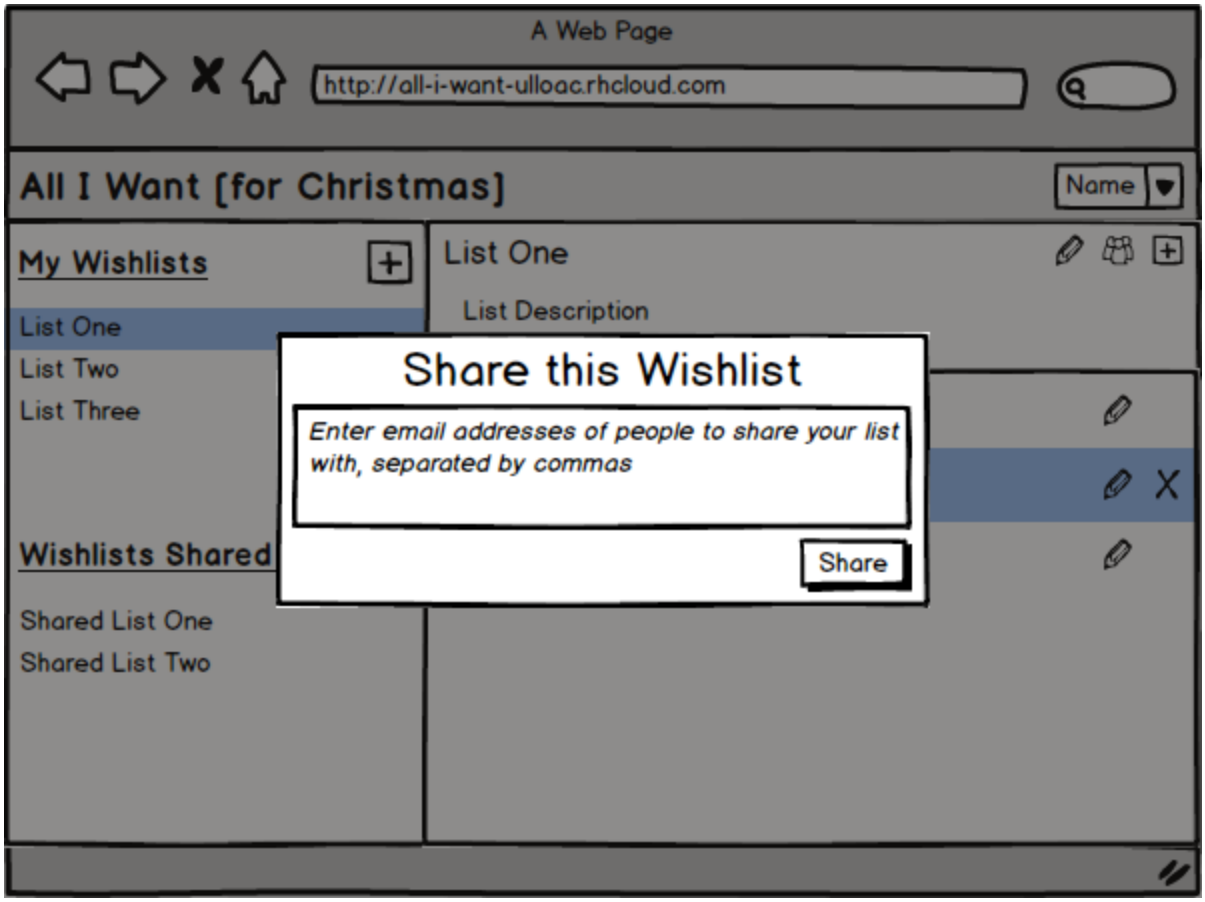
Edit Wishlist Modal:



Edit Gift Modal:

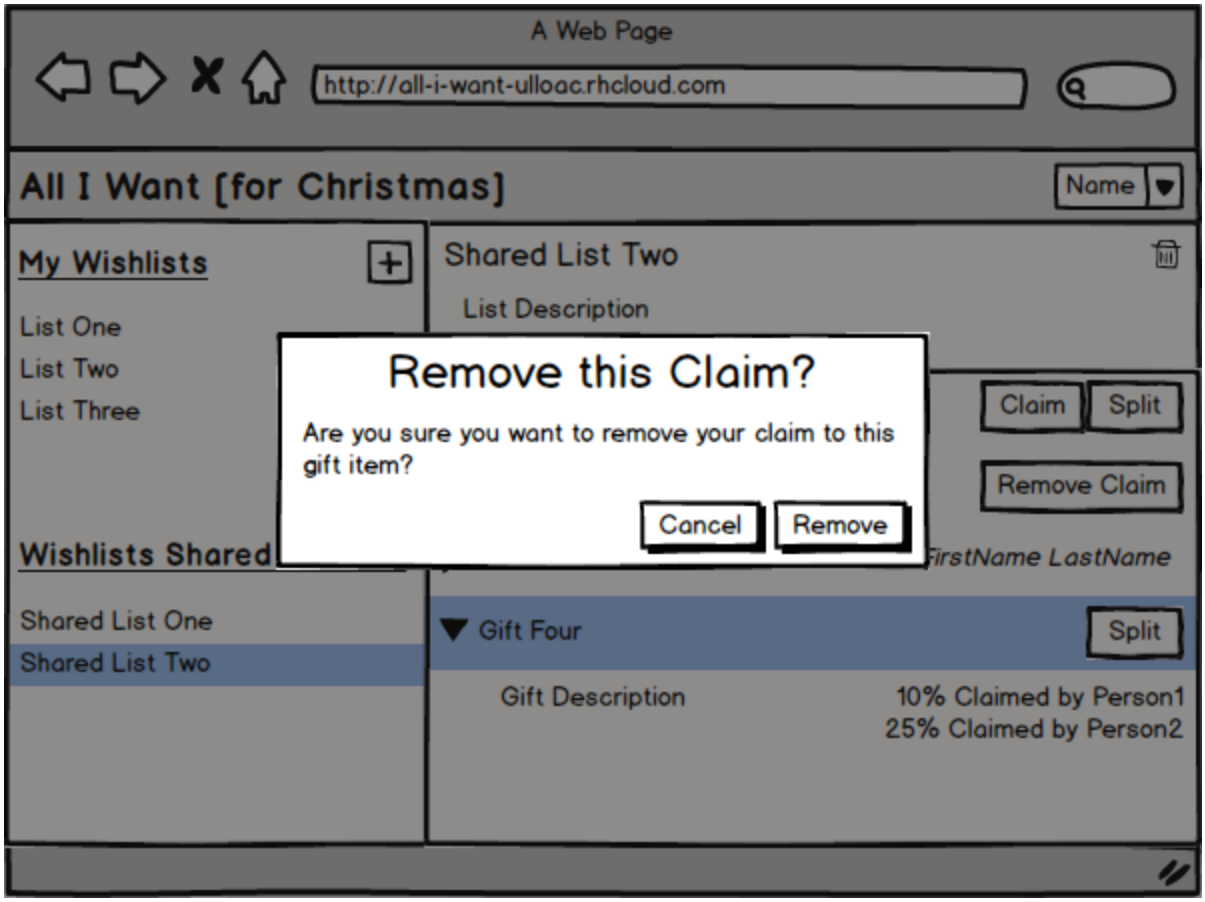


Share Wishlist Modal:

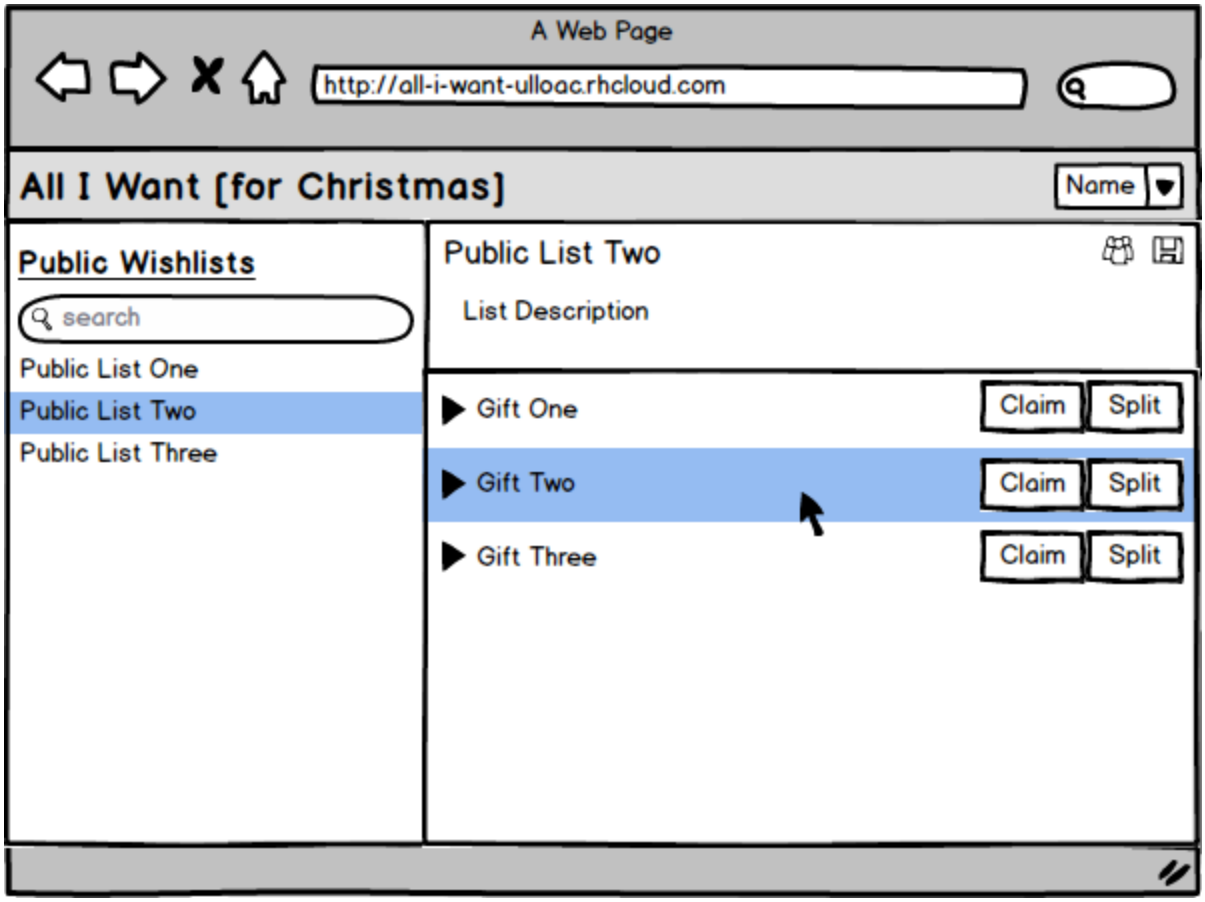


Split Claim Modal:

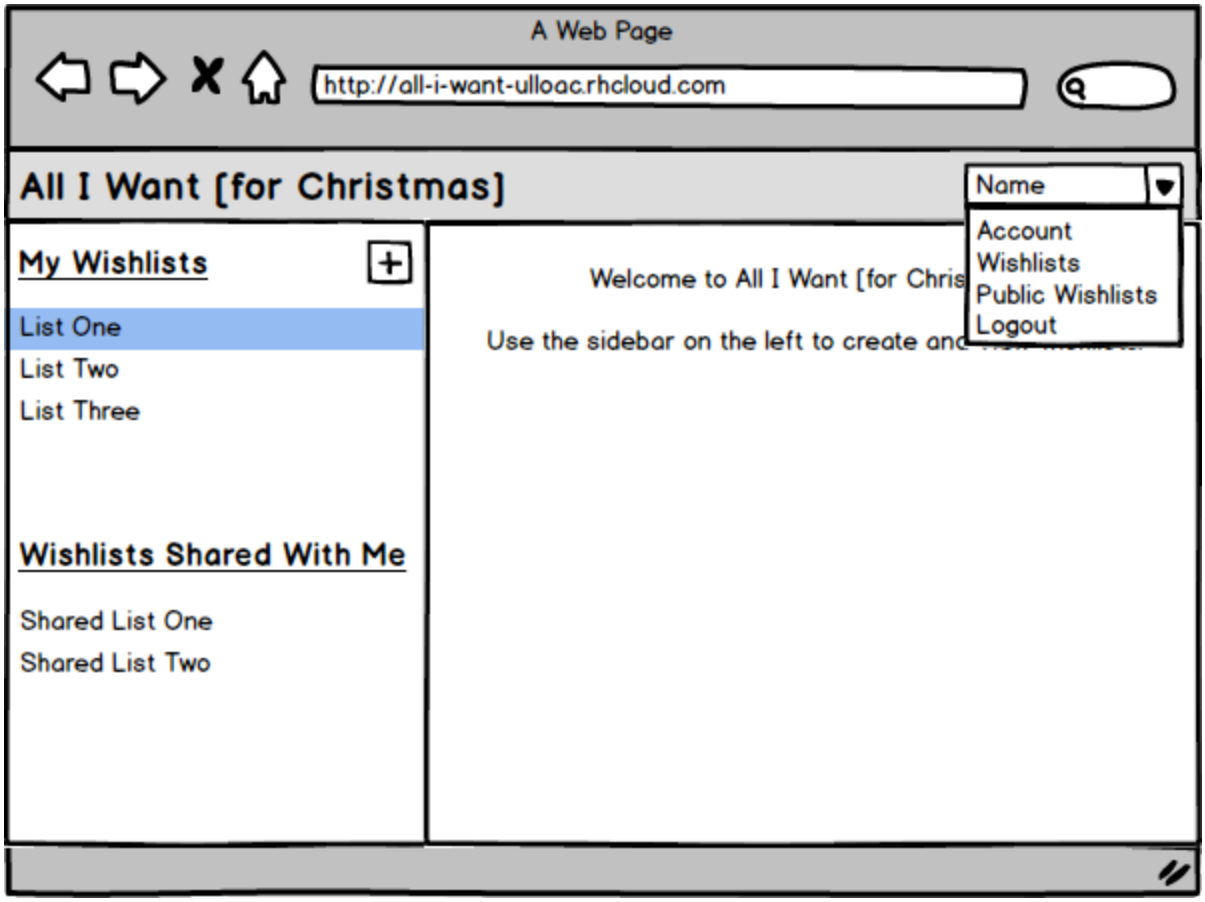
Remove Claim Confirmation Modal:



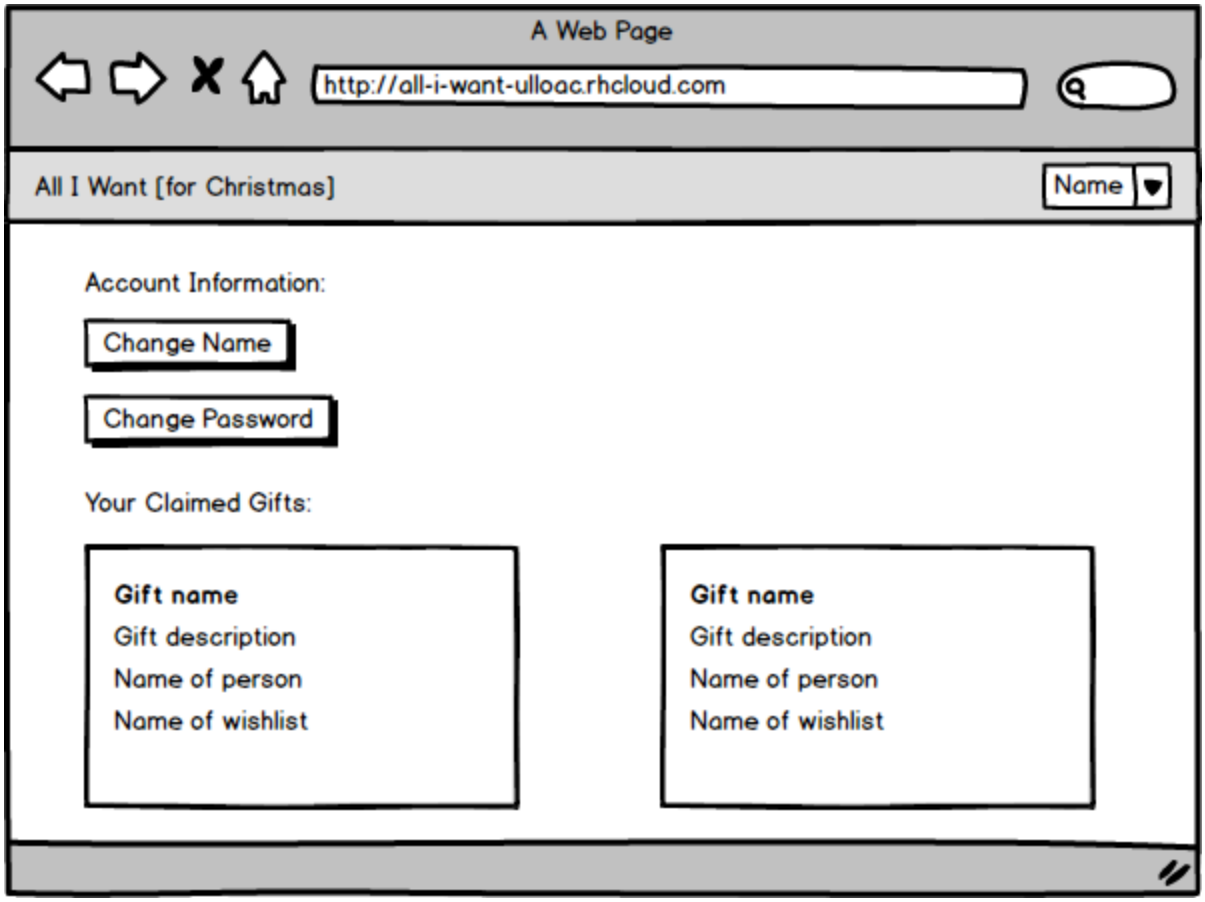
Public Wishlists Navigation Page



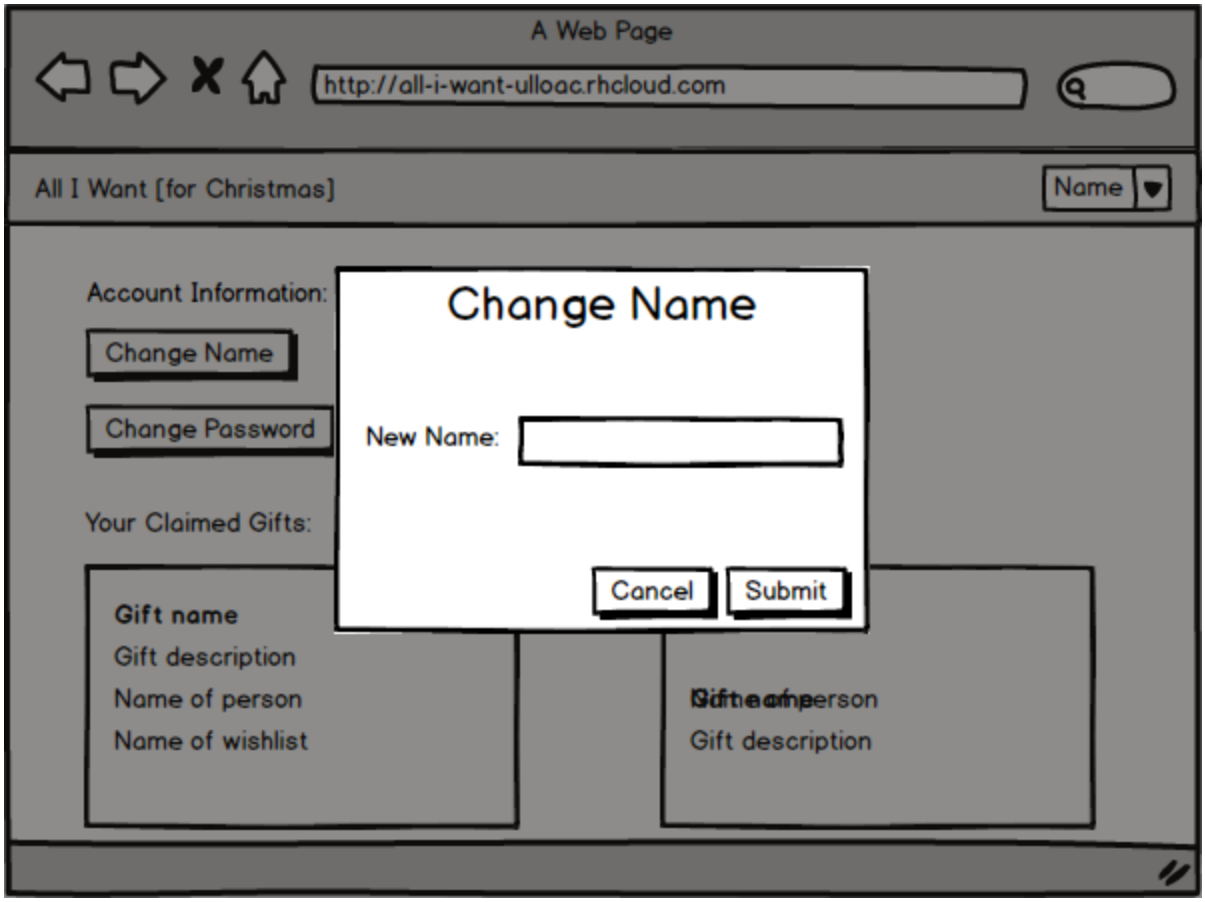
Open Navigation Dropdown:



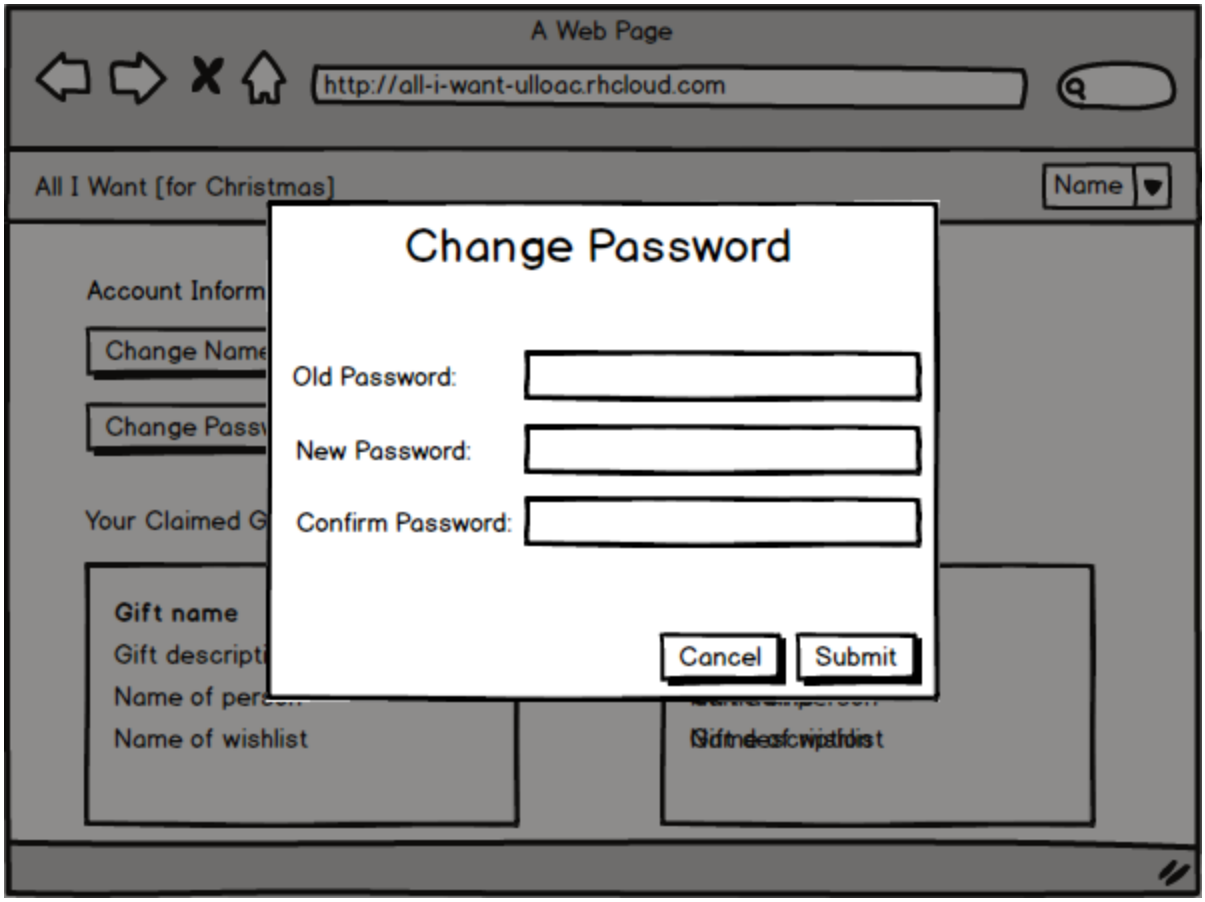
Account Info Page:



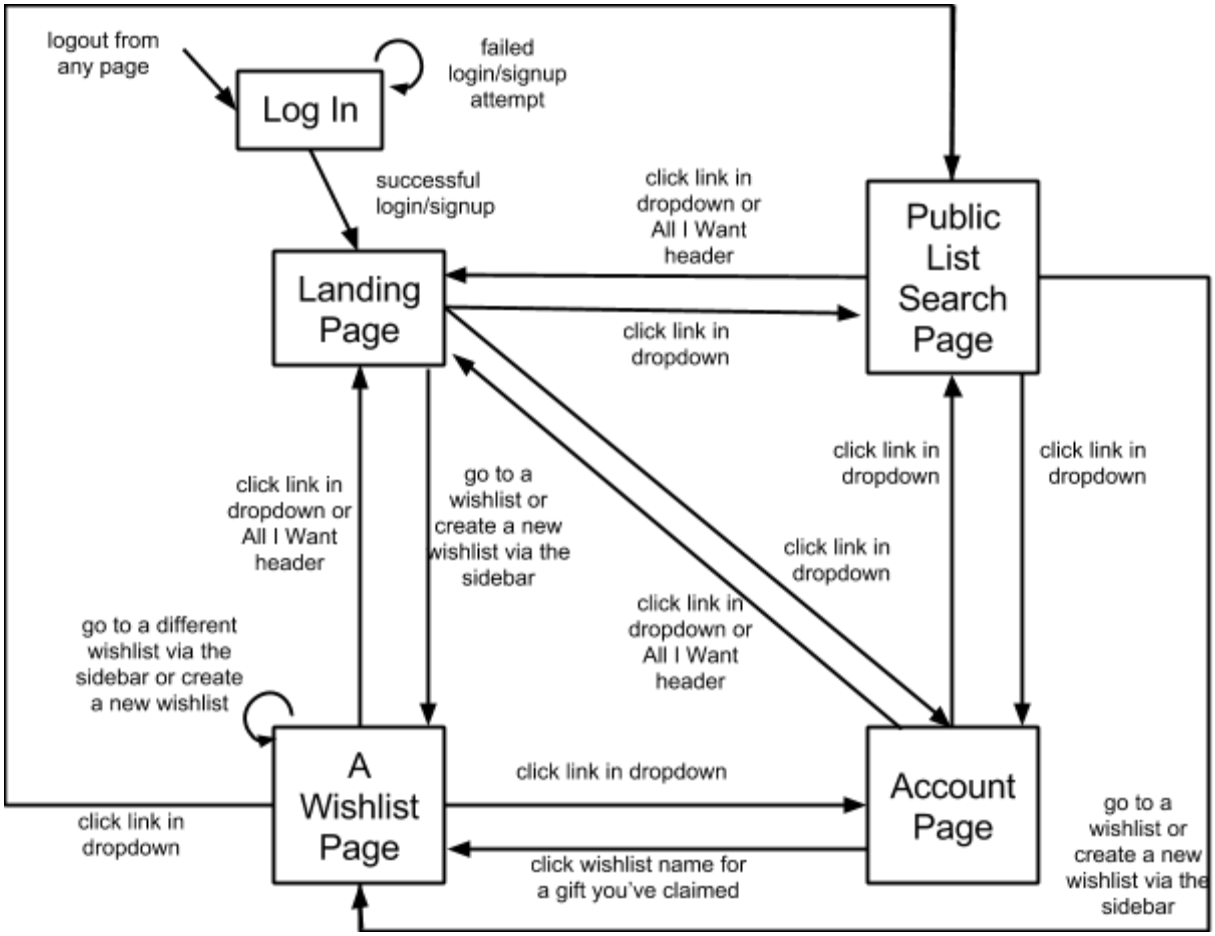
Edit User Name Modal:



Edit User Password Modal:



Page Flow Diagram



Design Challenges

Partially claimed gifts.

- If a gift is partially claimed, but a later user wants to claim the whole thing, should we allow them to override the split claim?
    - Option 1: Allow overriding a split claim
        - Prioritizing full claims makes sense to some degree, as it's ensuring the gift will be given (to the best of our ability)
    - Option 2: Don't allow overriding a split claim
        - Everywhere else claims are ruled by first-come-first-served. If someone has gotten there first and said they want to chip in to this, you shouldn't be able to take that away from them - but you can always contribute the rest

    - Solution: Option 2. The second user can claim the remaining percentage or give a new unclaimed gift; we won't give override power to full claims over split claims
- How does a user select what percentage of a gift they want to claim?
    - Option 1: Allow user to specify a percentage
        - Pro: Allows maximum flexibility
        - Con: Allowing for users to specify a percentage can lead to a strange unclaimed percentages (e.g. 33%+33%+33% is still not fully claimed)
    - Option 2: Split gift evenly among contributors
        - Pro: Easiest to implement and will always cause the gift to be fully claimed
        - Con: It might be problematic that a gift is always fully claimed, because if only one person goes in to split, they will end up buying it all. Also no flexibility (some people might be willing to pay more than others).
    - Option 3: Specify incremental percentages
        - Pro: Allows a large degree of flexibility while preventing previous cons
        - Con: Slightly limited flexibility
    - Solution: Option 3. We will allow users to split a claim in 5% increments up to the remaining percentage, since it offers all the practical freedom we believe a user will need without the implementation challenges of the other options.
- Should users be able to unclaim split claims?
    - Option 1: Allow unclaiming of split claims
        - Pro: Allows flexibility if a user changes his/her mind
        - Con: This could cause a previously 100% claimed gift to become unclaimed at the last minute and things would get messy
    - Option 2: Don't allow unclaiming of split claims
        - Pro: Everyone will always know how much of a gift is claimed
        - Con: Doesn't allow someone to back out - maybe they changed their mind, or can no longer afford to split the gift. Limits user freedom
    - Option 3: Allow unclaiming only if the gift is not fully claimed
        - Pro: Allows a user a fair amount of flexibility while not affecting other people counting on you to chip in to a gift
        - Con: Limits user freedom slightly, since after some amount of time they can be locked in to buying a gift
    - Solution: Option 3. Users can unclaim a split gift until 100% of the gift has been claimed, at which point the user is locked in to their claim. This is because once someone buys the gift knowing that the cost is 100% covered you shouldn't be able to back out. But we also want users to have as much freedom as possible for changing their minds, so we allow them to back out until it's fully claimed.
- Should we allow split claims on public wishlists?
    - Option 1: Allow split claims on public wishlists

        - Pro: Retains a core concept of our site, that being split claims. There are definitely use cases; for example, a charity wanting to raise funds for a large purchase.
        - Con: May introduce complications for the group actually arranging the purchase
    - Option 2: Don't allow split claims on public wishlists
        - Pro: Eliminates complications in coordinating between claimant groups
        - Con: May limit the gifts that can be received, since one person has to pay for each gift in full.
    - Solution: Option 1. Split claims will still be allowed on public lists, because it follows from the concepts of our site and arranging the purchases is really outside of the scope and purposes of our site.

Claiming Gifts

- Do we want to allow users to unclaim a gift after they've claimed it?
    - Option 1: Allow users to unclaim a gift
        - Pro: It's a functionality that users would probably expect to have, and people should have the flexibility to change their minds.
        - Con: Other users would see that the gift was claimed, not claim it as a consequence, and if it then becomes unclaimed, the gift won't get given.
    - Option 2: Don't allow user to unclaim a gift
        - Pro: If a gift is claimed, it will definitely be given.
        - Con: Doesn't allow users to change their mind
    - Solution: Option 1. Users can unclaim gifts, since the app would seem incomplete without this functionality and we want to allow for flexibility as well.
- Should users have a list of gifts they have claimed?
    - Option 1: Users have a list of claimed gifts
        - Pro: It's an easy reference for users, and they would probably expect to be able to view all their claims in one place.
        - Con: Might be duplicate information between this and email confirmations
    - Option 2: Users don't have a claimed gifts list
        - Pro: Simpler to implement, less repetitive information
        - Con: Might be missing functionality expected by users
    - Solution: Option 1. The added complexity is not significant, and it is useful for users to have a central reference for their claims.
- When do gifts disappear from a user's claimed gifts list?
    - Option 1: Gifts are never removed
        - Pro: Users can always view gifts they've claimed
        - Con: Easy for the view to become cluttered, and users probably don't care about gifts given multiple years ago.
    - Option 2: Gifts are removed after a certain amount of time
        - Pro: easy way to clean up the claimed gifts list and help keep it relevant

        - Con: Difficult to determine how long gifts should be kept, and this time might change depending on the situation (Christmas list vs. birthday list vs. public list, for example). Adding a time scheduler would add significant complexity for minimal benefit.
    - Option 3: Allow users to remove gifts whenever they wish
        - Pro: Maximum flexibility for the user - they see only what they want to
        - Con: Requires more work for the user to delete every claim
    - Solution: Option 3. Allowing users to remove gifts at their discretion gives them maximum flexibility with minimum complexity.

Wishlist and Gift Info

- Should wishlists require address information for where to send gifts?
    - Option 1: Require addresses on wishlists
        - Pro: Makes it easy for claimants to send gifts, helps facilitate the receiving of gifts, may be necessary information to have in order to deliver gifts
        - Con: Adds privacy concerns, and users may not want their address shared. May not be necessary, if gifts are for friends or family.
    - Option 2: Allow addresses on wishlists, but don't require
        - Pro: If a user thinks it's necessary to include, they can include their address and receive the same benefits as in option 1.
        - Con: If a user doesn't list an address, and a claimant needs an address, it adds difficulty in delivering gifts.
    - Solution: Option 2. The wishlist description can contain address (and will suggest this as possibly important), but will not be required. This solves privacy issues but allows addresses to be included as deemed necessary. Also, it is not the scope of our app to facilitate delivery of gifts, so it shouldn't be required.
- Should gifts include links to the products?
    - Option 1: Require links
        - Pro: Makes it easier for a claimant to see what the user wants, and help ensure that the exact thing a user wants is always given
        - Con: Requires that all gifts be purchasable online, which might not always be the case. Doesn't allow for generalized gifts (e.g. "a sweater" vs. "this particular sweater").
    - Option 2: Allow links, but don't require
        - Pro: Allows the pros of option 1 to occur if deemed relevant by the user.
        - Con: If users aren't required to submit links, they might not include links when they should
    - Solution: Option 2. Gift descriptions will be allowed to contain links, but won't be required to. We don't want to exclude gifts that had no specific products in mind (mystery novels, women's size medium sweaters, etc), but we want to allow links when they can be helpful, like when someone is asking for a specific item. We don't want to put the burden of finding a link to every single item on the user either.

- Should gifts include prices?
    - Option 1: Gifts include prices
        - Pro: Gives users more information before they claim a gift
        - Con: Users might not know the price range, items may not have a particular price, or there might be a large price range, which would make this feature inaccurate.
    - Option 2: Gifts don't include prices
        - Pro: Allows for the pros of option 1, if a user would like to include a price. Doesn't require the user to put in information they don't know or information that isn't relevant.
        - Con: Users have less information when looking to claim a gift, or have to do more work themselves to find the information
    - Solution: Option 2. Gifts are allowed to include prices, but only will when the user deems it necessary. Also, due to the previous design decision, we don't want to exclude gifts that might not have a particular price.

Implementation Challenges

Authentication Package

- Do we want to use passport or bcrypt to encrypt stored passwords?
    - Option 1: Use passport
        - Pro: Built-in encryption scheme
        - Con: Never used by any group members previously
    - Option 2: Use bcrypt
        - Pro: We can reference example code provided by staff using bcrypt
        - Con: One of our team members' previous teams had issues with bcrypt and openshift
    - Solution: Option 2. We were not able to debug errors with passport even after office hours at which point we switched to bcrypt.

Deploying to heroku vs openshift

- Do we want to deploy on heroku or openshift?
    - Option 1: Use heroku
        - Pro: No known errors with bcrypt
        - Con: No team member had previously deployed on to heroku. Heroku also requires more specific structure and filenames to work.
    - Option 2: Use openshift
        - Pro: All team members have previously used openshift successfully
        - Con: Openshift servers were having issues with timeouts. Possible issues with bcrypt.

- Solution: Option 2. Attempts to deploy on heroku didn't work after multiple hours of debugging. Openshift servers started working again and so we deployed on openshift in the end.

Database Decisions

- How can we populate an array with values retrieved database calls and avoid asynchronous issues?
  - Option 1: Use the async library
    - Pros: Lets us write asynchronous code as though it executes synchronously. Adds minimal complexity to the code.
    - Cons: External library adds complexity
  - Option 2: Nest callbacks
    - Pros: Pure javascript/ajax, doesn't require external libraries
    - Cons: Ugly and difficult to write
  - Solution: Option 1. The async library is very simple and easy to use, adds functionality we need, and only a few lines of additional code.
- How can we retrieve information about whether a gift has been fully claimed?
  - Option 1: Keep an extra attribute in the gift collection with a boolean for whether or not the gift has been fully claimed.
    - Pros: Makes loading the main wishlists page faster (because we don't have to loop through and retrieve all the claims on all of the gifts to calculate the sum of their percentages).
    - Cons: The extra attribute is redundant information already represented within the claims' percentages, which introduces the potential for inconsistencies if we don't keep both pieces of information in sync. This could be mitigated in future implementations by researching and applying Mongoose's virtual (automatically precomputed) fields.
  - Option 2: Retrieve all the claims for each gift on the wishlist and include the sum of their percentages for each gift with the result.
    - Pros: Doesn't introduce any redundancy into the data model, and we probably won't reach the scale (or the number of gifts/wishlist) that would make this a huge inefficiency. Also, when we introduce split claims (beyond MVP), we may need to retrieve all of the claim information anyway.
    - Cons: Is still slower than retrieving just a precomputed attribute on each gift.
  - Solution: For now, we chose option 1, and will attempt to make use of Mongoose's virtual fields to mitigate the potential drawbacks of redundant information in the database. However, we'll reconsider this decision when implementing split claims, in case we end up needing all of the claim information for gifts anyways.