TroveSpace

Design Document

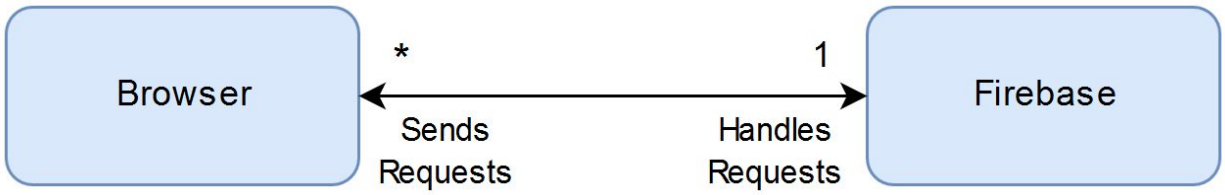
Team 13 | https://github.com/baker323/TroveSpace
Adam Baker, Kareem Elhadidi, Rei Orikata, Pritesh Kadiwala

Purpose

The system we are designing will be a web application that allows users to catalog any type of collection and interact with other collectors. We will create a single platform for users to manage multiple different collections; use crowdsourced data to create a large, peer reviewed database of items; allow users to interact and arrange trades and sales with other collectors; and show detailed reviews and average user ratings for specific items. This software will serve as a web-based replacement for outdated native apps and online sites that only focus on specific items. Collectors will be able to manage multiple types of collections all in one place and have the added bonus of an online community where they can view other people's collections and arrange trades or purchases. This will allow them to not only keep track of items that they have, but also items that they are actively seeking, helping them to build the collection of their dreams.

Design Outline

This project will have multiple clients connecting to a single database. We will be using the client-server model. Requests from clients will be sent to the database when a user wants to create an account, create a collection, add an item to their collection, and so on. Most if not all of our user stories will heavily involve client-server communication between multiple users' browsers and Firebase. When a request is received and processed by the database, our system will update the frontend to account for the change. We will be using AngularJS to produce dynamic HTML on the client side with Firebase being in charge of updating and fulfilling requests for information and user authentication.

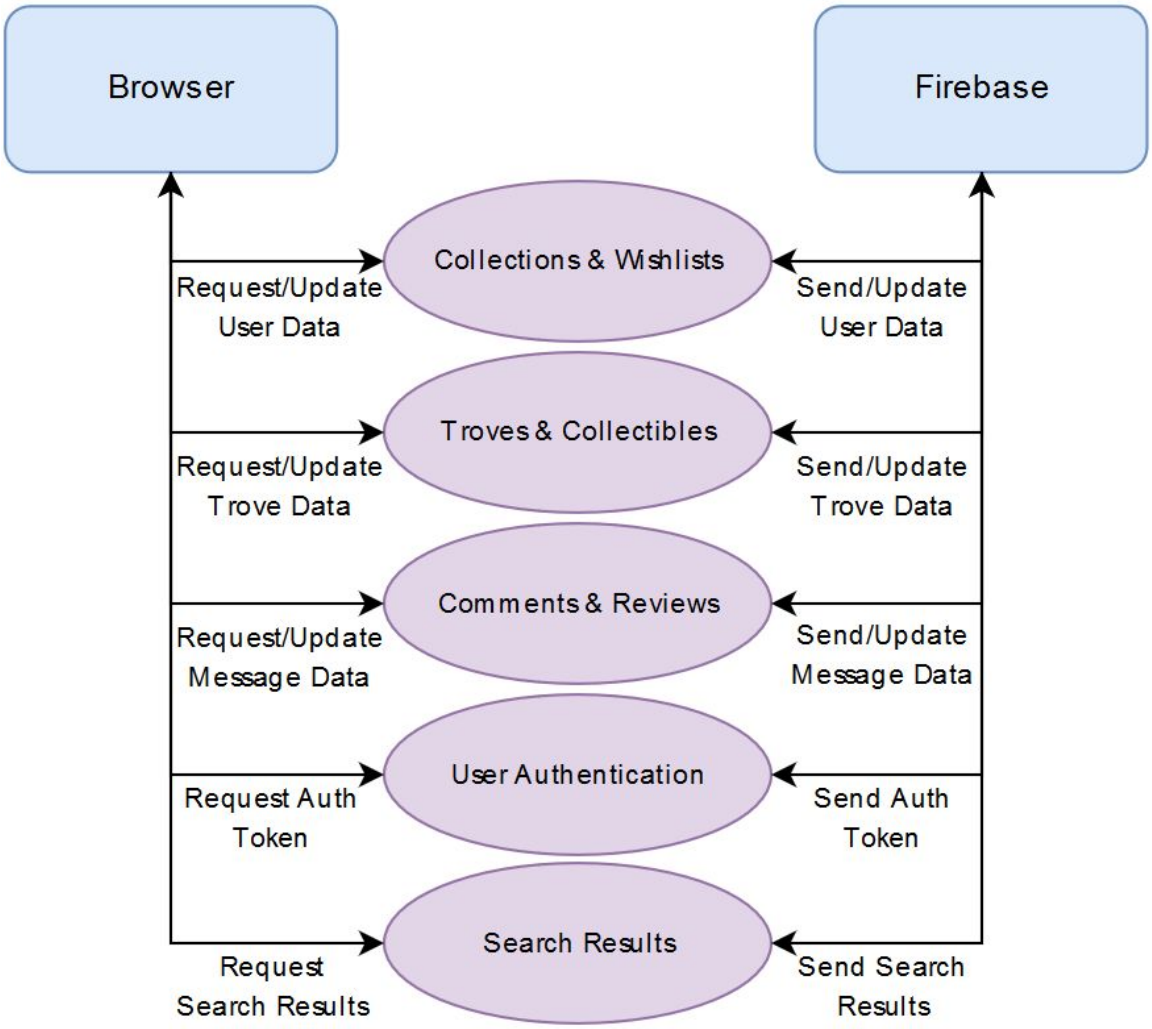


Browser

Our frontend will use AngularJS, Bootstrap, HTML5 and CSS3 running in the Chrome web browser. This will display an intuitive and visually appealing interface to the user and allow them to complete actions to manage their information stored in the database.

Firebase

All persistent information will be stored in Firebase. Each user will have a unique account with multiple collection folders and a wish list, which will each contain many collectibles of varying categories. Users will also have a list of categories and users that they follow. For our project, categories will be referred to as Troves and each will have unique attributes. The database will store all collectible items under their respective Troves with their details, ratings, and comments. Firebase will also handle user authentication as well as account creation and deletion.



Design Issues

Functional Issues

1. How do we handle editing an item so that the changes are acceptable to other users with that item in their collection?
    a. Option 1: Have users with moderator privileges approve edits.
    b. **Option 2: Notify affected users about edits and have them vote on whether the change should be reverted.**

    Decision: We decided that having a voting system would be the best course of action because it would allow collaboration among all users to decide on changes that would affect their collection. Additionally, having moderators could introduce unnecessary complexity and problems with moderators not properly performing their duties in a timely fashion.

2. How do we handle many different types of collections that may require their own unique sets of attribute fields?
    a. Option 1: Have preset collections that offer users fixed attribute fields.
    b. **Option 2: Allow users to create their own attribute fields that are tied to that specific category of collection.**

    Decision: We decided that allowing users to customize sets of attribute fields for any category would be better because it gives users greater freedom and allows for more detailed cataloging of niche collections. Having preset attribute fields for certain collections would severely limit users and needlessly prioritize those collections.

3. How should we handle situations where users create collection categories that are too similar or duplicates of other categories?
    a. Option 1: Have an option for users to mark a category as a duplicate and then database administrators review requests and delete the duplicates.
    b. **Option 2: Allow users to propose combining a category with a duplicate category through a voting system.**

    Decision: We decided that a voting system would be better because it allows users to have a say in which categories are duplicates and takes away the unnecessary hassle of having to manually approve requests. Having admins delete categories could also interfere with people who are participating in or following duplicate categories that suddenly get erased.

4. How should we handle situations where users want to delete a category of collection?
    a. **Option 1: Have an option for users to request deletion of a category and then database administrators review requests and delete the categories.**
    b. Option 2: Allow users to vote on whether a category should be deleted.

    Decision: We decided that manually reviewing requests would be better than voting because the cases where categories would need to be outright removed would be rare. Additionally, if users vote to delete a category and there are a large number of users, at least one of those users might still want to keep the category. Rather, users can freely leave or unfollow categories.

5. What search engine service will we use to implement our search functionality?
    a. Option 1: Our own JQuery search function
    b. **Option 2: Algolia**

    Decision: We decided that we would use the Algolia API as it is a service that focuses on implementing search functions which are already modified to work efficiently.

6. How will we implement the storage of pictures for a large number of collectibles?
    a. **Option 1: Use Firebase for image storage as well as database storage.**
    b. Option 2: Use a separate CDN for image storage.

    Decision: We decided to use Firebase for the sake of simplicity because Firebase supports image storage. Since we are already using Firebase to store our data, it just makes sense to use it for image storage as well.

Non-Functional Issues

1. How should we build our backend?
    a. Option 1: Python + Flask
    b. Option 2: Django
    c. Option 3: Firebase

    Decision: We decided to go with Firebase as the team has more experience with it and it is a real time database which would use data synchronization instead of usual HTTP requests. Thus, every time the database is changed, a connected client will receive the changes in no time. Furthermore, it is a cloud-hosted database that could be accessible on any system.

2. How should we build our frontend?
    a. Option 1: VueJS
    b. Option 2: AngularJS
    c. Option 3: ReactJS

    Decision: We decided to use AngularJS as the team members have experience in this frontend framework.

3. What service should we use to host our website?
    a. Option 1: GoDaddy
    b. Option 2: AWS
    c. Option 3: GitHub

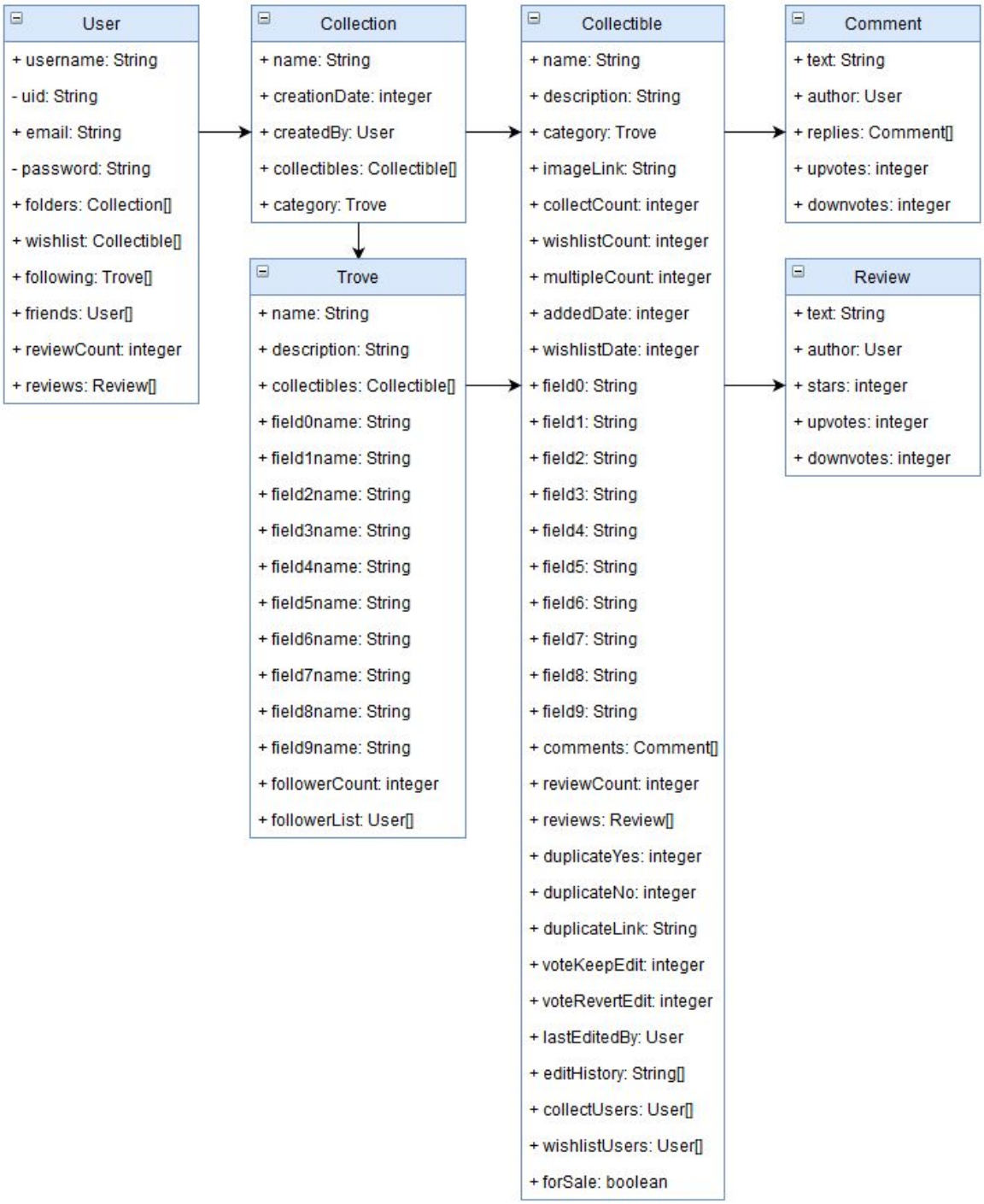
    Decision: We decided to host our website on GitHub because GitHub does not charge for hosting one site.

4. What CSS framework would we use?
    a. Option 1: Bootstrap
    b. Option 2: Foundation
    c. Option 3: Bulma

    Decision: We decided to go with Bootstrap framework as the team knows how to use Bootstrap and it is one of the most popular frameworks used with AngularJS. Thus, we will have a larger community with various solutions to problems we might encounter.

Design Details

Class Diagram



Class Descriptions

User

- This contains the user's login credentials as well as all of their personal data.
- Login credentials include username, password, email, and a unique identifier.
- Personal data consists of folders of collections, a wishlist of collectibles, a list of all followed Troves, and a list of all users who were added as a friend.
- There are also reviews and a review count for others to rate trades or purchases.

Collection

- This contains a list of collectibles as well as the Trove to which they belong.
- A collection includes a name, the creation date, and the user that created it.

Collectible

- This is the main collectible class that represents an item in a collection.
- A collectible includes a name, image, description, and Trove to which it belongs.
- Also stored are fields for personal collections and a dynamic set of custom fields, represented above by fields 0 through 9, which will be different depending on the Trove.
- The fields for personal collections include dates the item was added to a collection or wishlist, a count to signify how many of that item are in a user's collection, and a tag that shows whether the item is for sale.
- Lists keep track of all users that have added the item to their collection or wishlist.
- Sections for comments and reviews with a review count to calculate the average rating.
- Vote tallies for accepting edits of the collectible as well as whether it is a duplicate of another existing item or not. Edit history is also stored so changes can be rolled back.

Trove

- A Trove is a category that contains a specific set of collectibles and can be followed.
- Troves contain a name, description, and all collectibles that are under that category.
- There is a count of all users that follow each Trove as well as a list of all such users.
- A dynamic set of customizable fields allow users to personalize specific details for any type of collectible, which will be different for each Trove.

Comment

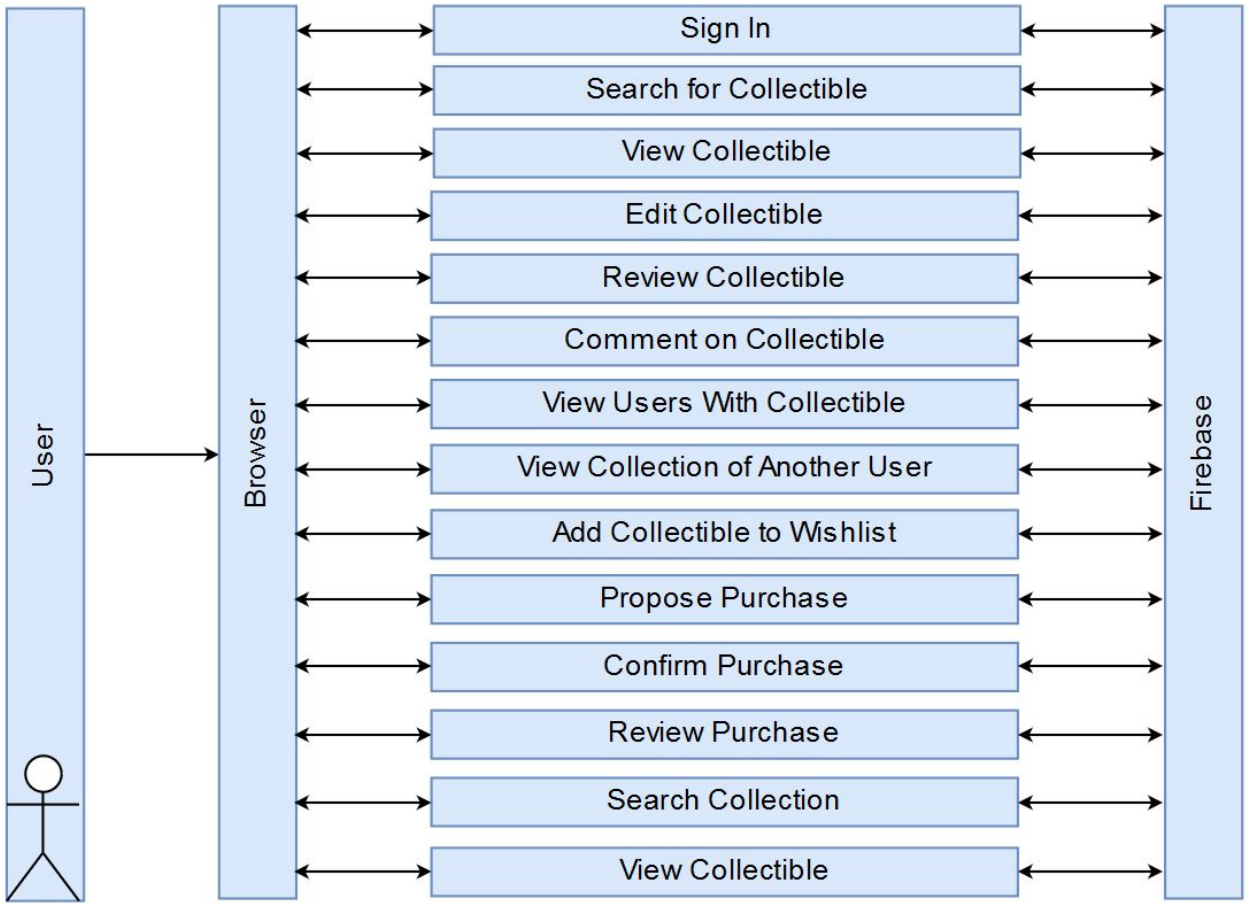
- This contains the text of the comment as well as the user who posted it.
- Comments also include the ability to reply, upvote, and downvote.

Review

- This contains the text of the review as well as the user who posted it.
- Reviews also include a rating out of five stars as well as upvotes and downvotes.

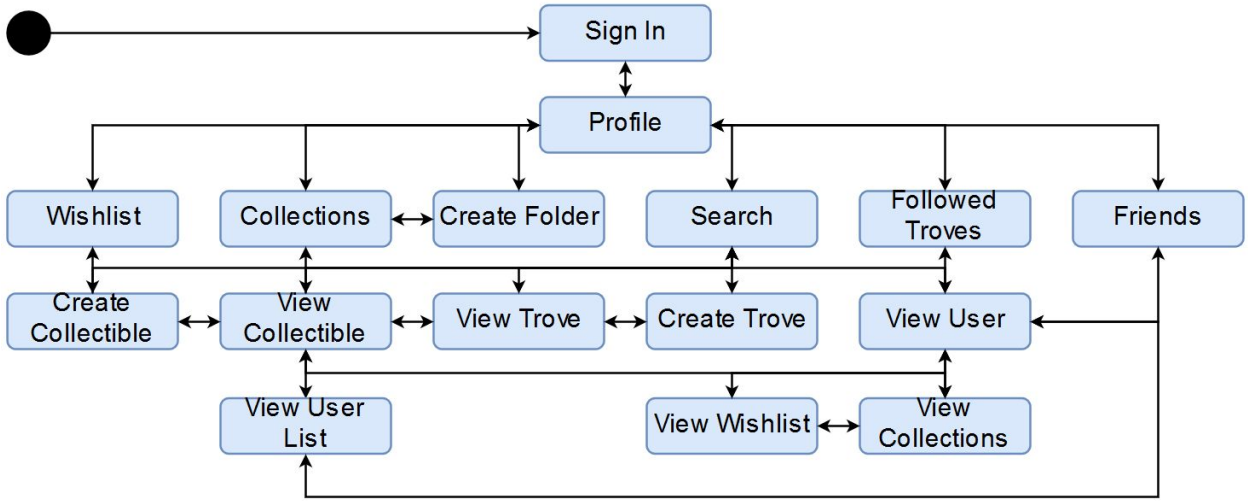
Sequence Diagrams

Sequence of events for finding and adding an item to a collection



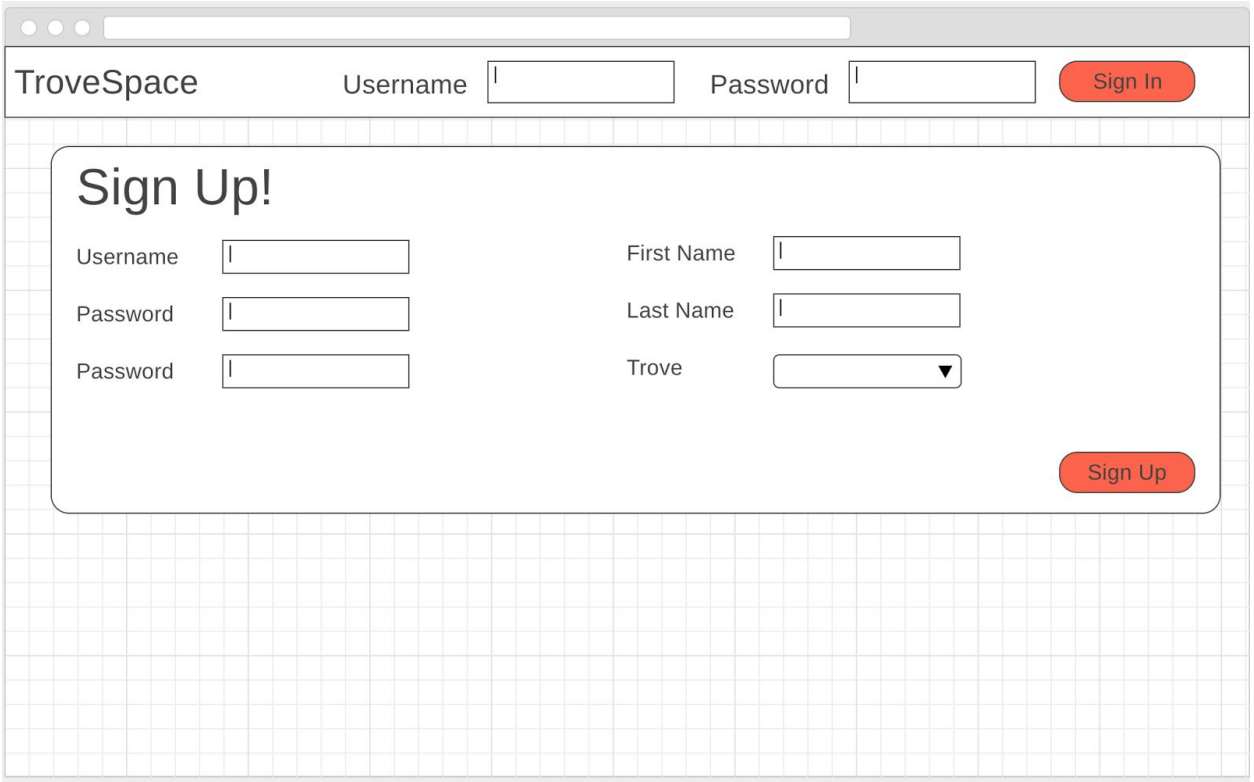
Sequence of events for interacting with a collectible and another user

Activity Diagram

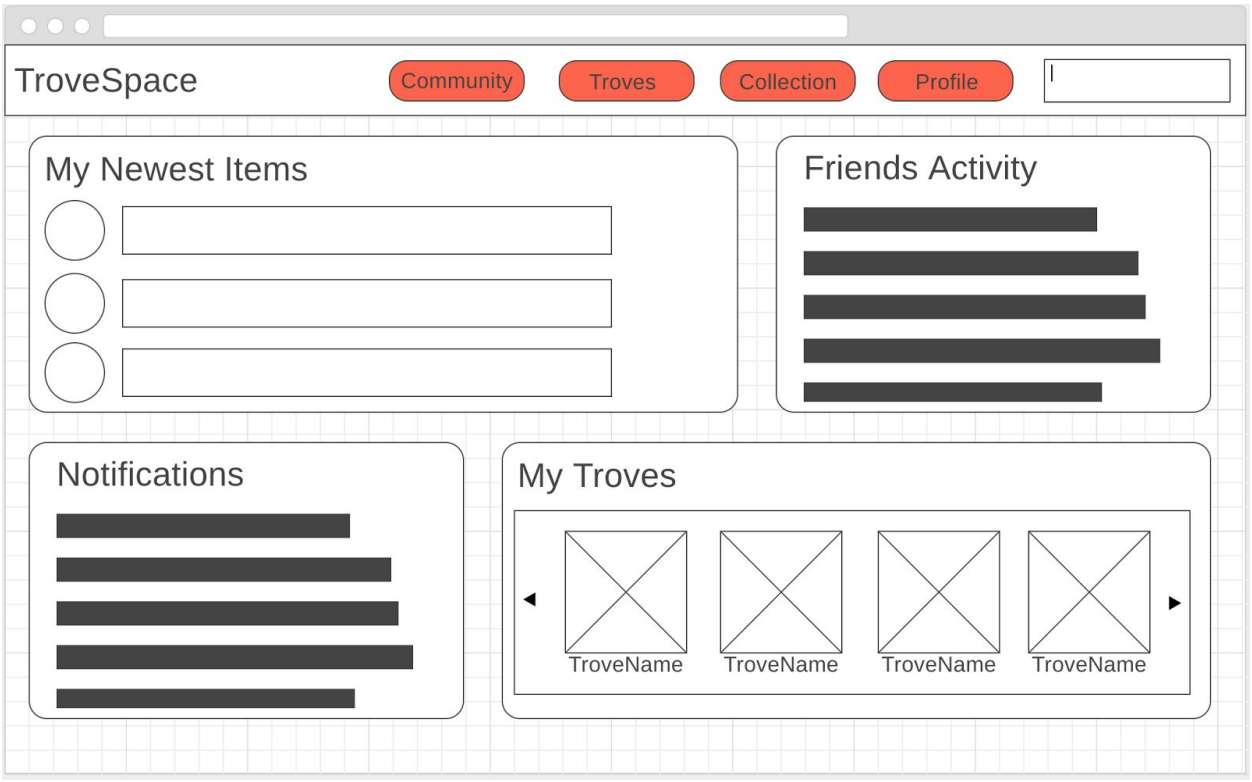


UI Mockups

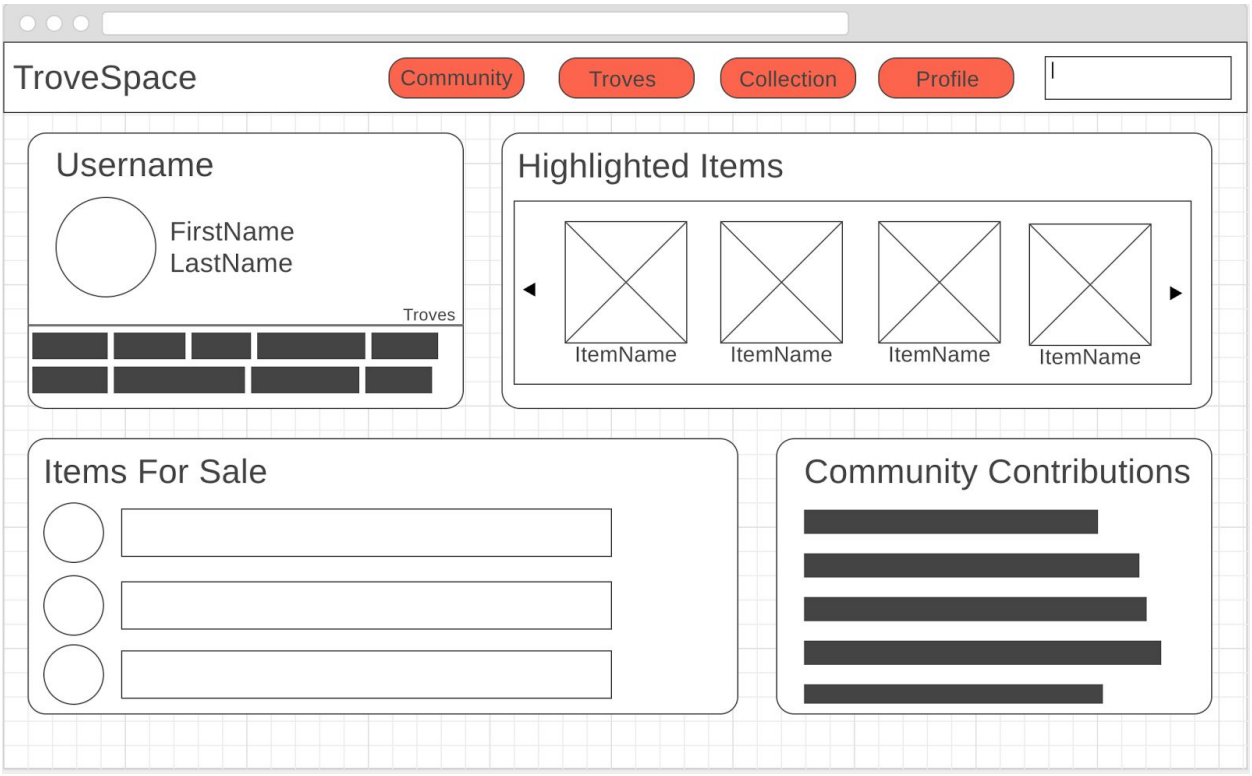
Sign in/up



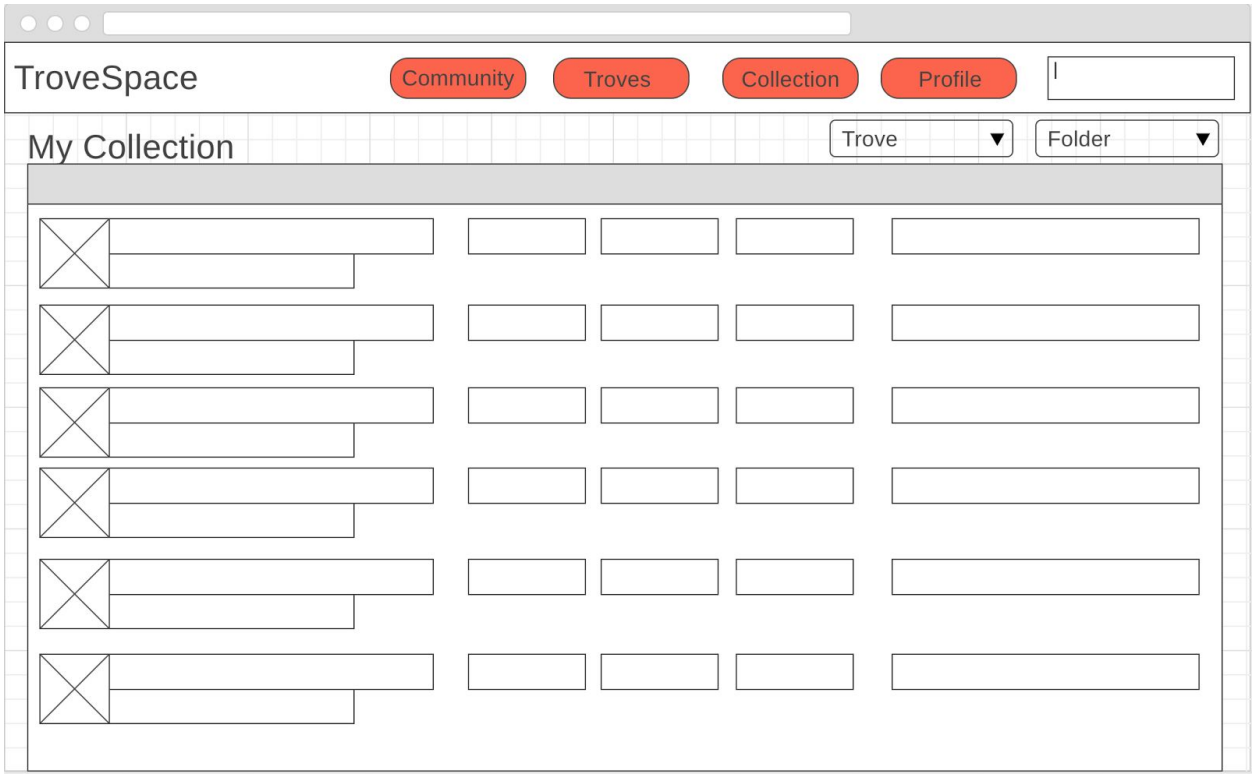
My Profile



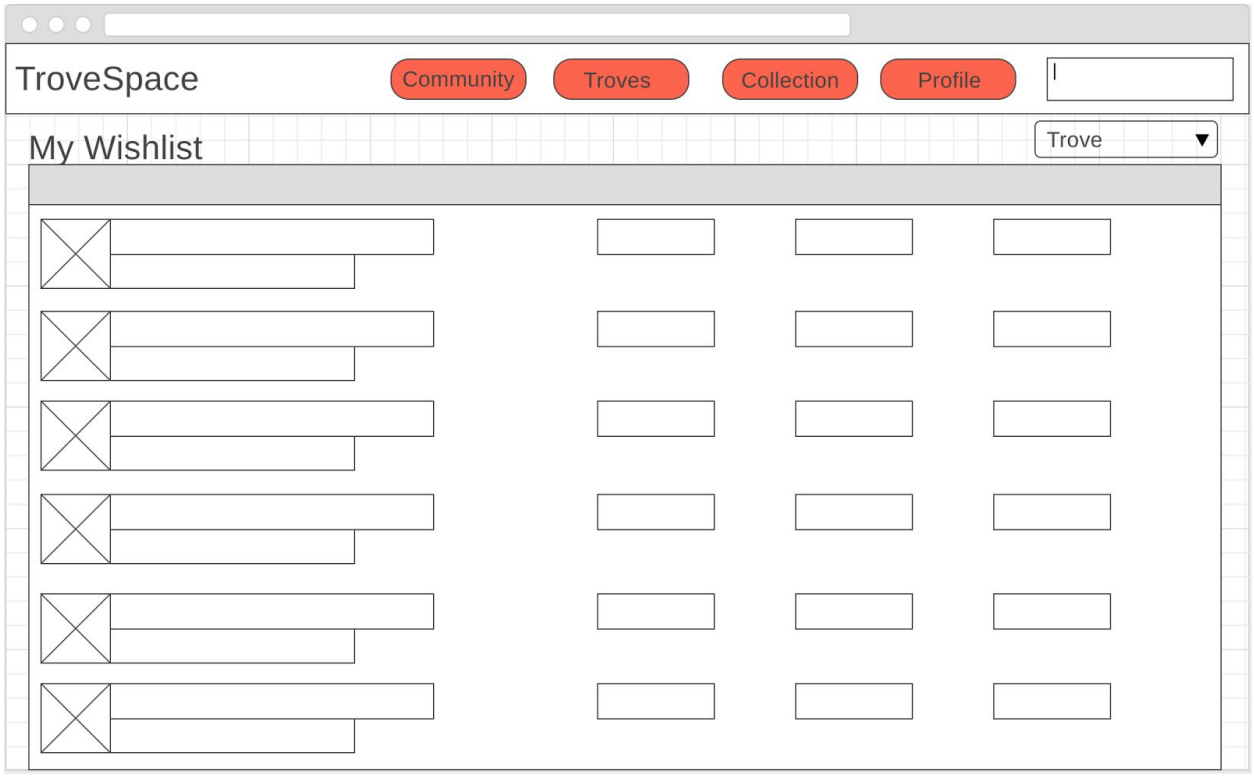
Other User's Profile



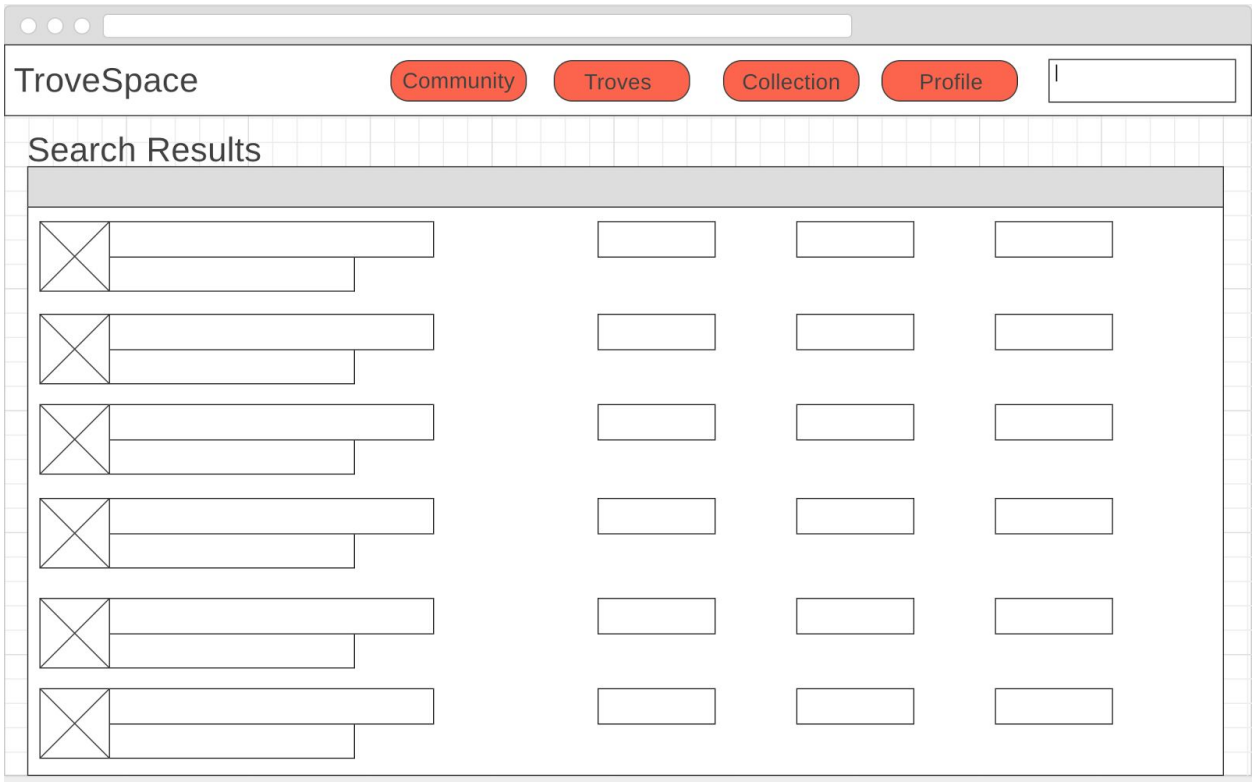
Collection



Wishlist



Search Results



Collectible Page