



Beanstalk

Milestone 3

Terry Yang, Annie Lin,
Hiroka Tamura, Kaelan Mikowicz



Beanstalk - Share your adventure

- Bring people together around trendy spots and hidden gems!

Outline

Creating and Retrieving Posts

Likes and Comments

Username Search

Viewing and Following Users

Activity Feed

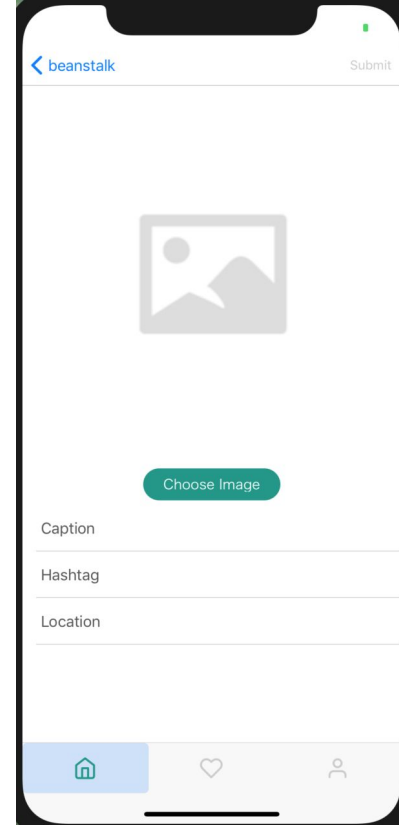
Extra Features

Live Demo

Creating a Post: Frontend

Workflow

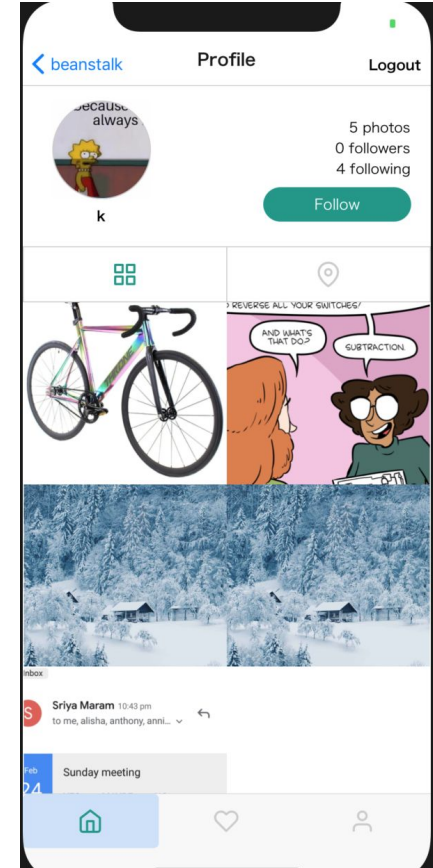
1. Click icon on Activity Feed to perform upload
2. Click “Choose Image” to pick an image from your Camera Roll
 - a. Add caption, hashtag, location
3. Click “Submit” post
 - a. Creates a form with Post information
 - b. Performs a POST request to Post endpoint



Retrieving a Post: Frontend

4. Display Posts on

- a. User Profile
 - i. After submitting a post
 - ii. Load profile
 - iii. Perform GET request to profile endpoint
- b. Activity Feed





Creating & Retrieving a Post: Backend

- Post Endpoint - `/api/Post`
 - ◆ POST - Creates a new post for auth user
- Post Item Endpoint - `/api/Post/<pid>`
 - ◆ GET - Retrieves the post contents, comments and likes for post with `<pid>`
 - ◆ PUT - Updates the caption for post with `<pid>` if post belongs to auth user
 - ◆ DELETE - Removes the post with `<pid>` if post belongs to auth user

Likes: Frontend

- Heart Icon to elicit a like action
- User is able to
 - ◆ Like a post
 - ◆ Unlike a post
- Once user clicks to like/unlike
 - ◆ Perform POST or DELETE to like endpoint
 - ◆ Render an update colors & numbers to inform users of their action

1 likes



hiroo Yo



Likes: Backend

- Post Like Endpoint - `/api/Post/<pid>/like`
 - ◆ POST - Lets auth user like the post with `<pid>`
 - ◆ DELETE - Lets auth user unlike the post with `<pid>`



Likes: Queries

Comments:

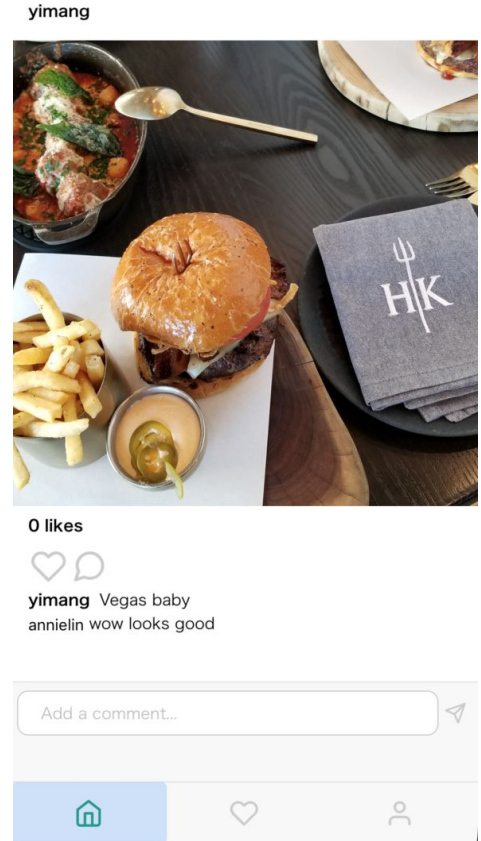
```
SELECT COUNT(UID)
FROM "Comment_Like"
WHERE commentID = %(comment_id)
```

Posts:

```
SELECT COUNT(UID)
FROM "Like"
WHERE PID = %(post_id)
```

Comments: Frontend

- Users will be able to add comments when they navigate to one Post
- Once submitting a comment
 - ◆ Perform POST request to Post/<pid>/comment endpoint
 - ◆ Store comments in an array
 - ◆ Render all previous comments and recently commented





Comments: Backend

- Comment Endpoint - `/api/Post/<pid>/comment`
 - ◆ POST - Creates a new comment for auth user on post with `<pid>`
- Comment Item Endpoint - `/api/Post/<pid>/comment/<comment_id>`
 - ◆ PUT - Updates the comment with `<comment_id>` for post with `<pid>` if comment belongs to auth user
 - ◆ DELETE - Removes comment with `<comment_id>` for post with `<pid>` if comment belongs to auth user



Comments: Queries

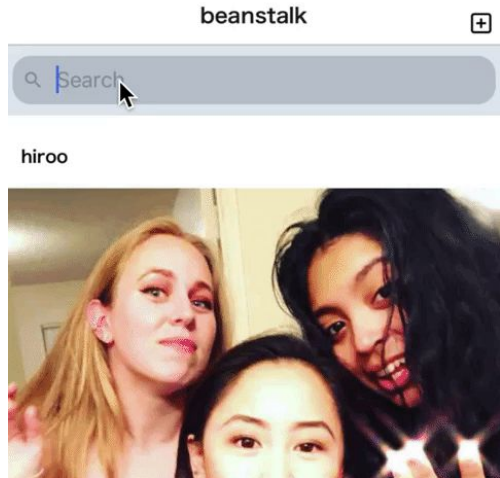
Getting all comments for a post:

```
SELECT "Comment".commentID, "Comment".comment, "Post".PID,  
FROM "Post"  
JOIN "Comment" ON "Comment".PID = "Post".PID  
WHERE "Post".PID = %(post_id)
```

Username Search: Frontend

Workflow

1. Users will locate Search Bar through the Activity Feed page
 - a. Send a GET request upon changing the search text
2. User will type desired username
3. Search Bar renders a page with usernames that match the search queries
4. User may click on result to navigate to desired profile
 - a. Pass in username prop when navigating to profile





Username Search: Backend

- Username Search Endpoint - `/api/User/search?query=<query_string>`
 - ◆ GET - Retrieves a list of likely usernames for `<query_string>`



Username Search: Queries

Retrieving likely usernames:

```
SELECT "User".username,  
FROM "User"  
WHERE "User".username LIKE "query_string%"
```



Viewing and Following Users: Frontend

Viewing

- Usernames will navigate to a User's Profile
 - ◆ Pass in a username prop
- Perform GET request to a specific username

Following

- Each profile that isn't yours will have a follow button
- Upon Follow or Unfollow
 - ◆ Perform a POST or DELETE request to the follow endpoint



Viewing Users: Backend

- User Profile Endpoint - `/api/User/profile/<username>`
 - ◆ GET - Retrieves User's number of photos/followers/following, posts, username and profile picture
 - ◆ PUT - If owner of profile, updates profile picture and other changeable fields like first name and last name



Following Users: Backend

→ Follow Endpoint - `/api/User/follow/<username>`

- ◆ POST - Lets the auth user follow the user specified by `<username>`. Adds an entry into the Follow table. Updates the FollowAggregation table.
- ◆ DELETE - Lets the auth user unfollow the user specified by `<username>`. Deletes an entry from the Follow table. Updates the FollowAggregation table.



Denormalization

Following Aggregation to be
part of Users table

id	followers	following
1	2	0
3	199	200

id	username	followers	following
1	annie	2	0
3	terry	199	200

Activity Feed: Frontend

Workflow

1. Home tab will display the Activity Feed
2. Upon logging in:
 - a. Perform GET request to the User's Home endpoint
3. What we render:
 - a. All posts from users you follow
 - b. Posts in reverse chronological order





Activity Feed: Backend

→ Activity Feed Endpoint - `/api/User/Home`

- ◆ GET - Retrieves a list of posts from users followed by the auth user plus posts by the auth user. For each post, returns the post image, username of poster, number of likes on post, and whether the auth user has liked the post

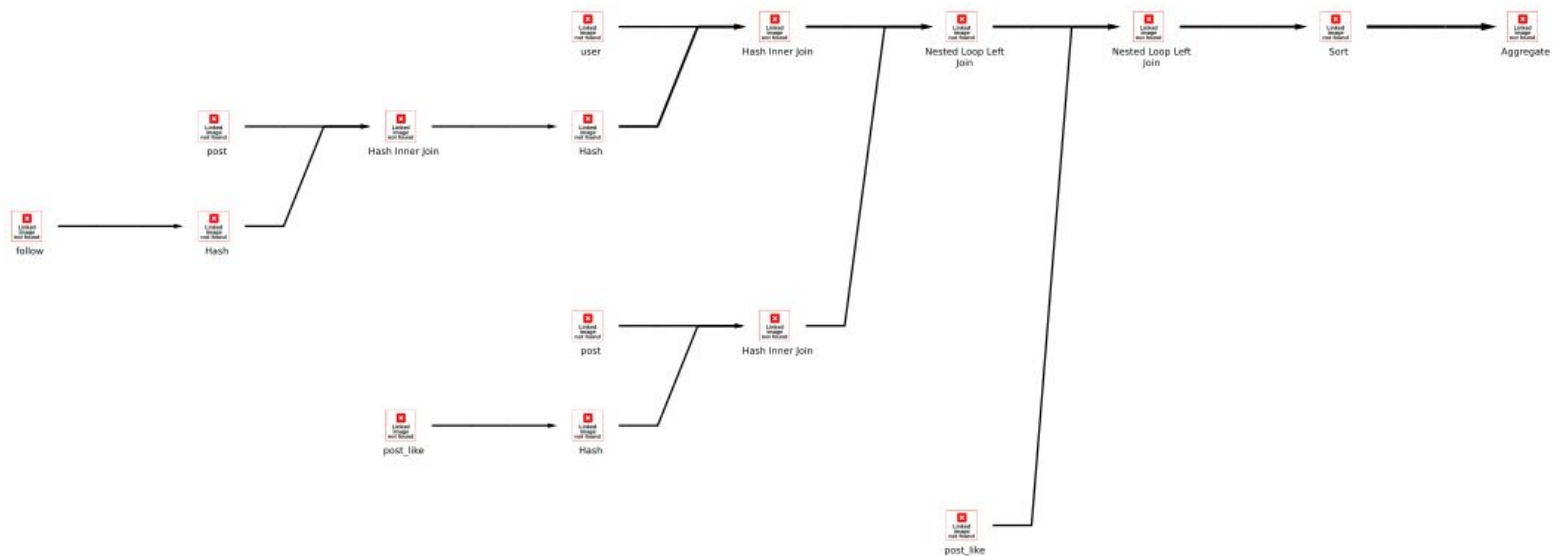


Activity Feed: Queries

Naive method:

```
SELECT post.id, post.caption, post.photo, user.username, like_exists.pid, COUNT(post_like.uid)
FROM post
JOIN user ON user.id = post.uid
LEFT OUTER JOIN post_like ON post_like.pid = post.id
JOIN follow ON follow.follower_uid = auth_user.id AND follow.following_uid = user.id
LEFT OUTER JOIN ( SELECT post_like.pid
                   FROM post_like
                   JOIN post ON post.pid = post_like.pid
                   WHERE post_like.uid = auth_user.id) AS like_exists
ON like_exists.pid = post.pid
GROUP BY post.id, post.caption, post.photo, user.username, like_exists.pid
```

Explain Query





Optimizations and Improvements

- Remove subquery just to get whether the current user is liking a post
 - Use a separate endpoint to speed up the initial loading
- Remove join to count post likes by aggregating it in post table
- Originally storing the entire image binary in database
 - Low bandwidth from database to server
 - Query returns array of full sized photo binaries, python ran out of memory
 - Photo is now a UUID to a folder hosted on backend
- Cache database results per user
 - Only refresh the cache based on time



Indexing

- Hash index on comment pid so that it is fast to retrieve all comments for a post
- Hash index on post_like uid so it is fast to check whether a user likes a post
- Hash index on comment_like so it is easy to check if a user likes a comment

Extra Features

- Profile Picture
- Editing a Post:
 - ◆ Deletion
 - ◆ Edit caption (in progress)

Profile

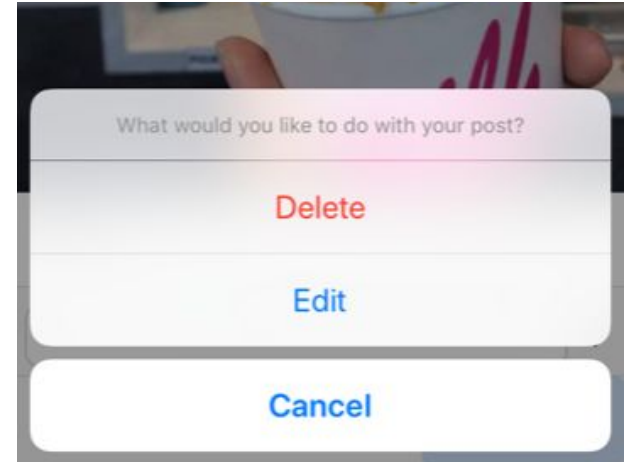
Logout



Annie

1 photos
2 followers
3 following

Edit Profile





UI / UX Design

Personality Implementation

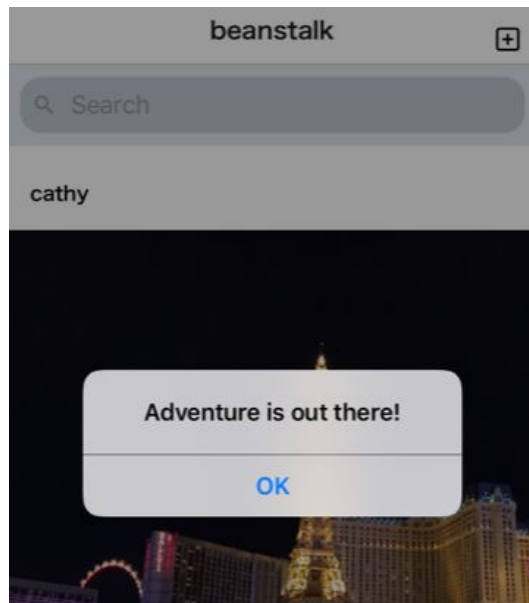
Persona Simulation

Extras

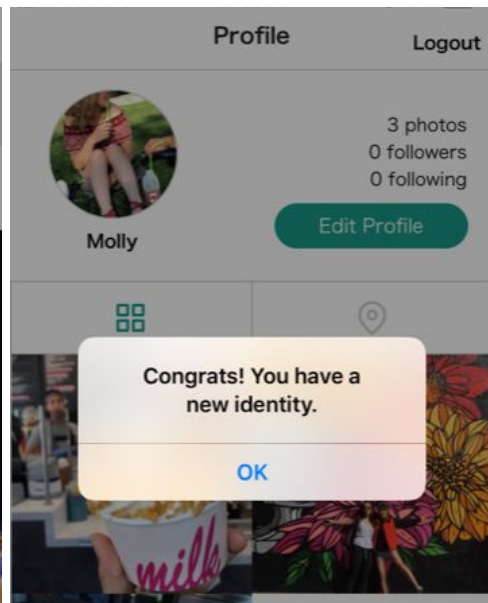
App Personality: Voice

- Interacts with users in a helpful yet playful tone. Upbeat, sweet and feminine.
- Every user is a daring adventurer: the app acts as an assistant/sidekick in their journeys.

User posts a picture:



User edits profile info:



App Personality: Visual Lexicons

- Color: White with soft emerald-green accents.
- Text: Sans-Serif font that portrays the app's feminine tone; clean and professional.



beanstalk

Username

Password

Log In

< beanstalk



2 likes



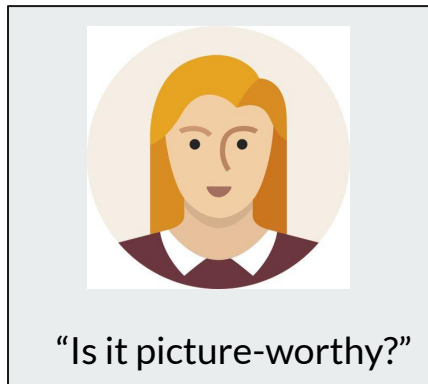
cathy New Chanel

alex you're so elite 🤔💎

wow I wish I could be like u!

1 2 3 4 5 6 7 8 9 0

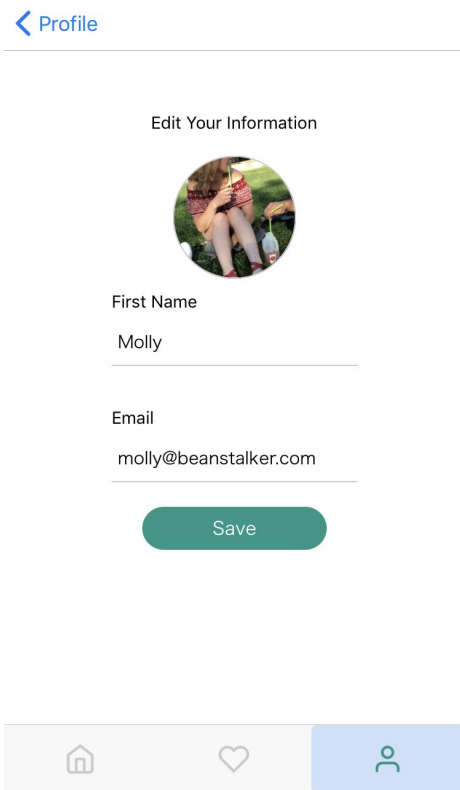
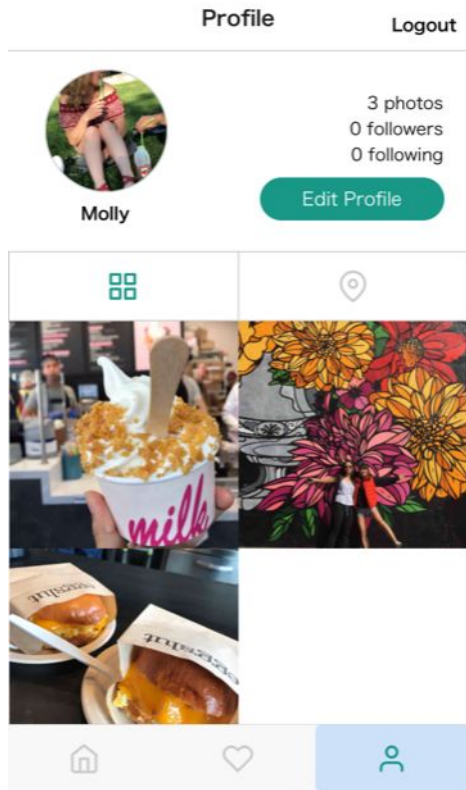
User Persona: Molly



Molly (21)

Goals: Wants to be popular and show that she lives a cool life as well as stalk others'.

- Lives in LA
- Loves LA
- Loves avocado toast and pretty lattes
- Always looking for the next instagrammable spot to show her friends



User Persona



Alex (25)

- A big travel nut
- A major foodie

Profile

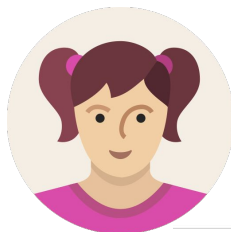
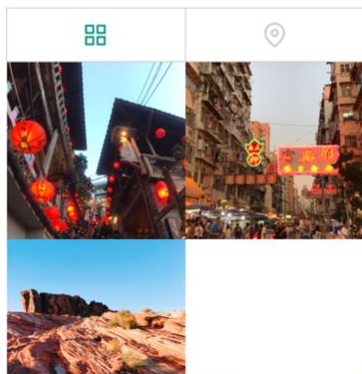
Logout



Alex

3 photos
0 followers
0 following

Edit Profile



Cathy (18)

- Fancy pants
- Likes to boast her riches

Profile

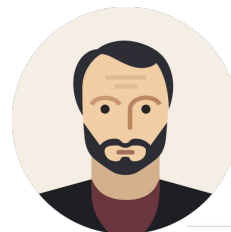
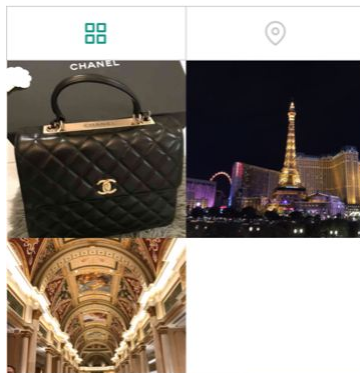
Logout



Cathy

3 photos
0 followers
0 following

Edit Profile



Vishal (30)

- New Yorker who loves a good drink
- A little bit of a party animal

Profile

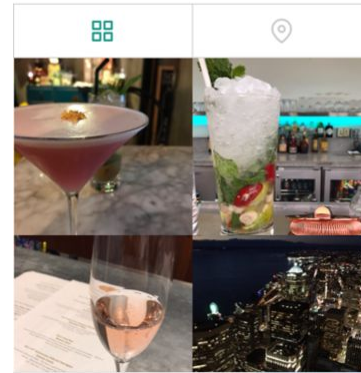
Logout



Vishal

4 photos
0 followers
0 following

Edit Profile



Extra UX components

KeyboardAvoidView:

Shifts the view by either padding or positioning so that user is able to get a full view when inputting text.



Ice ice baby

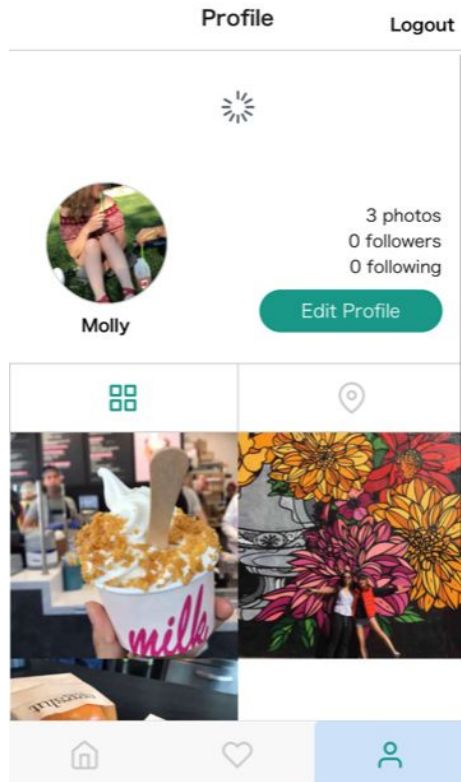
#milkbar

Milk bar, Los Angeles



RefreshControl:

Page reloads when the user pulls the screen downward to ensure update visibility.




Extra UX components

Disabling Buttons:

Making it visually apparent whether a button can be clicked or not. This can help avoid double posting.

[< beanstalk](#)

Submit




Choose Image

Caption

Hashtag

[< beanstalk](#)

Submit



Choose Image

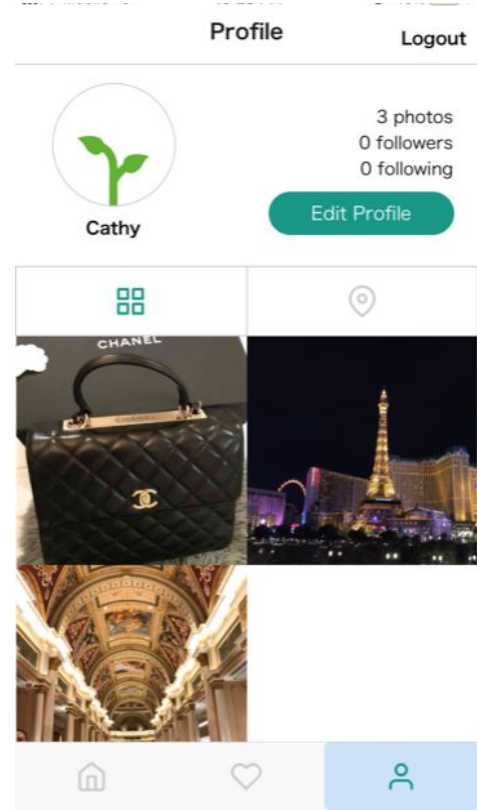
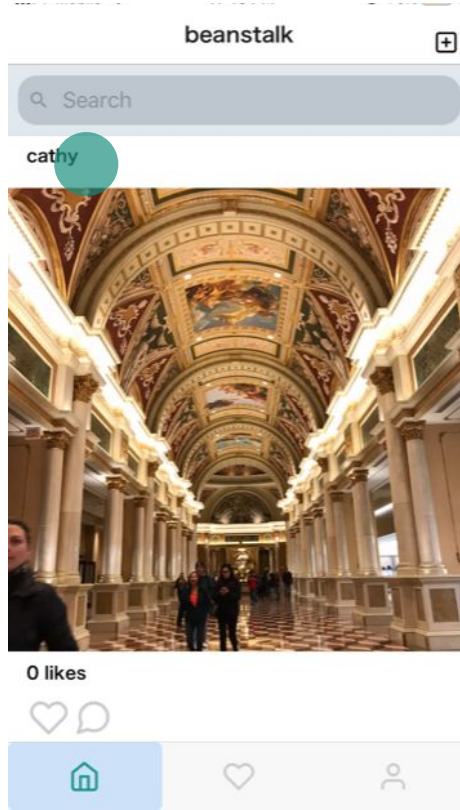
Caption

Hashtag

Extra UX components

Navigation by username:

Allowing users to easily navigate between profiles with a touch of a button at the usernames on the activity feed or posts.



Live Demo

Project Goals

Registration

User Login

Authentication

User Profile Editing

Questions?
