Beanstalk

Milestone 2

Terry Yang, Annie Lin,
Hiroka Tamura, Kaelan Mikowicz

Beanstalk - Share your adventure

- Bring people together around trendy spots and hidden gems!

Outline

Frontend: UI / UX Design

Database Schema

Backend: API Endpoints and Queries

Live Demo

Frontend:
UI / UX Design

App Personality

App Persona

User Experience Design

User Interface Design

App Personality: Brand Traits

- ➔ A friendly, modern and accessible app

Simple but not unsophisticated

Trustworthy but not dull

Kind but not passive

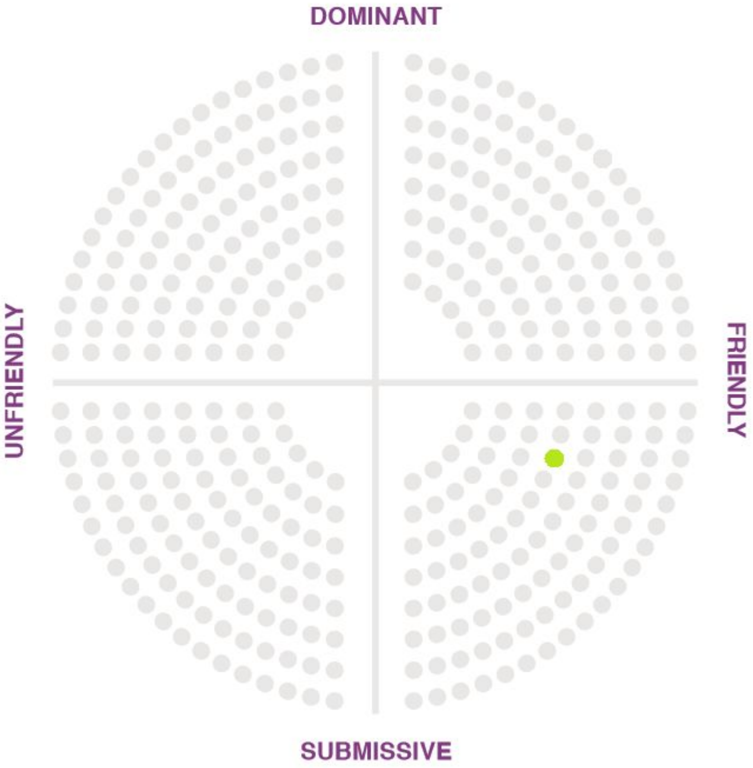
Hip but not exclusive

Adventurous but not aggressive

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.
- Slightly more formal than conversational, but still human.

Personality Map



App Personality: Visual Lexicons

- Color: White with soft emerald-green accents.
  - Flat colors and little textures that parallel app's personality of being simple/non-aggressive.
  - Green that is kind to the eyes as well as refined and functional.
- Text: Sans-Serif font that portrays the app's feminine tone; clean and professional.
- Reduce Clutter: Clean interface that embraces the white space and effectively leads users' eyes in navigation.

User Persona

Molly (21)

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

Alan (25)

- A big travel nut
- A major foodie

Cathy (18)

- Fancy pants
- Likes to boast her riches

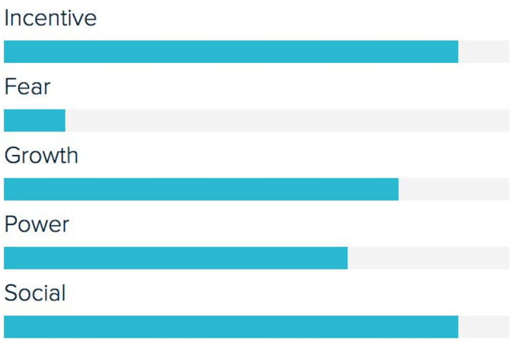
Vishal (30)

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

User Persona: Molly

“Is it picture-worthy?”

Motivation



Goals:

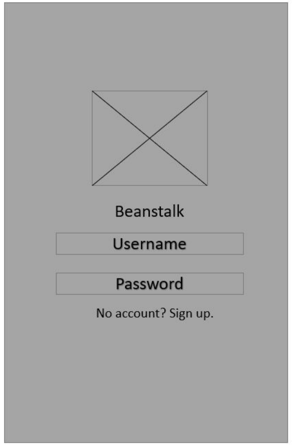
- Wants to find the best food in a certain area.
- Wants others to see what she has been up to and where she has traveled.
- Is curious about her friend's lives.

Frustrations:

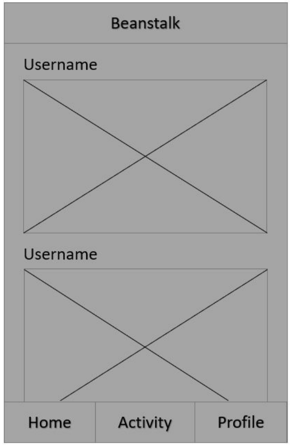
- Not being able to find cool spots her friends are going to
- Not knowing the newest trendy spots near by
- Trying to plan for her travels but not knowing where to go

UI Design: Low Fidelity

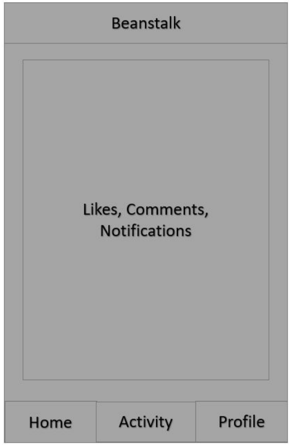
Login



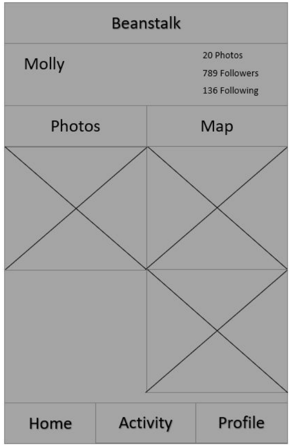
Home



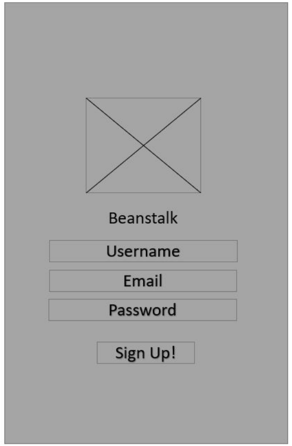
Activity Feed



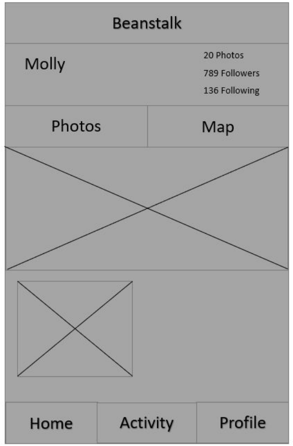
Profile



Sign Up

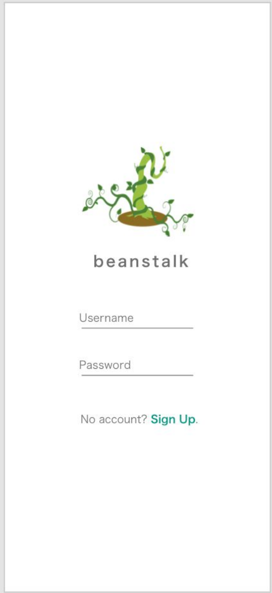


Map

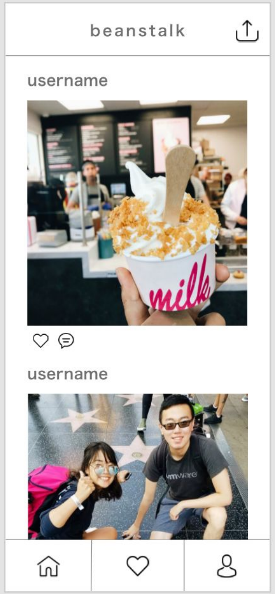


UI Design: High Fidelity

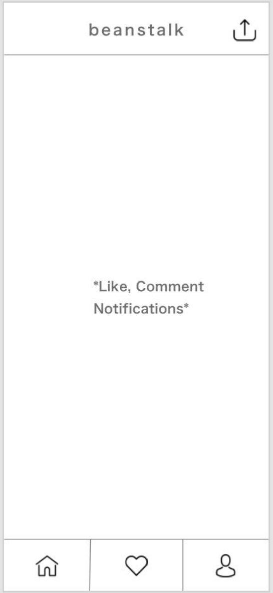
beanstalk



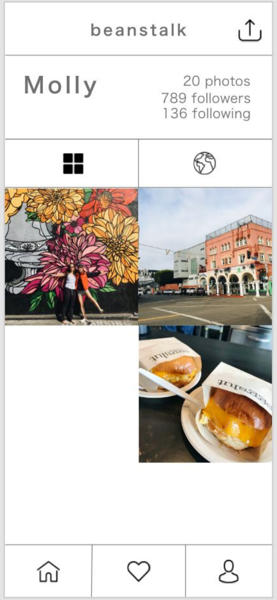
Home



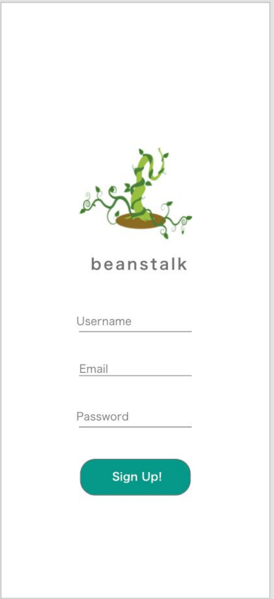
ActivityFeed



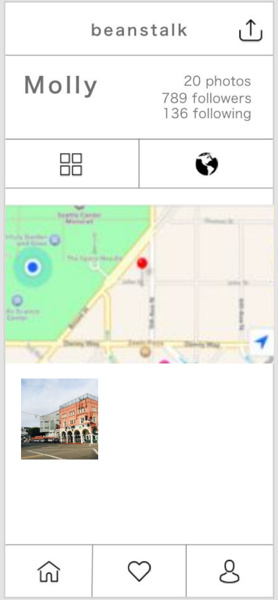
Profile



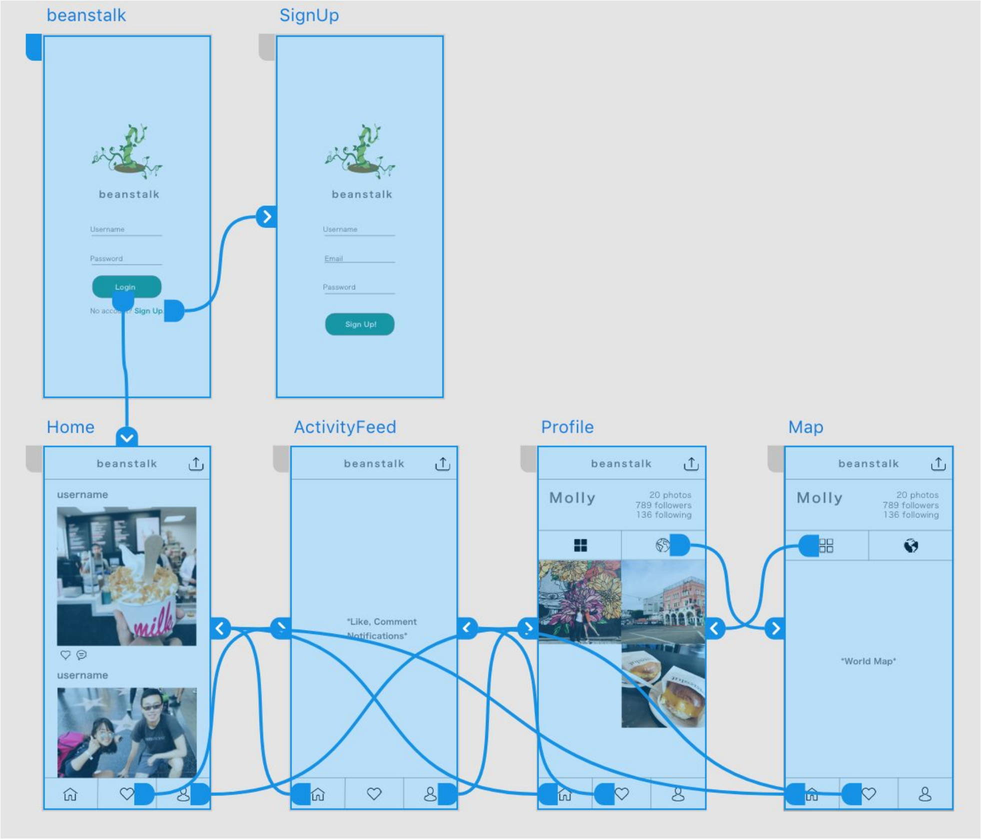
SignUp



Map



UX Design



Usability

- Simple transitions
- Minimalistic experience
- Fulfills goal of users

Recognizable

- Familiar Icons

Visibility of Feedback

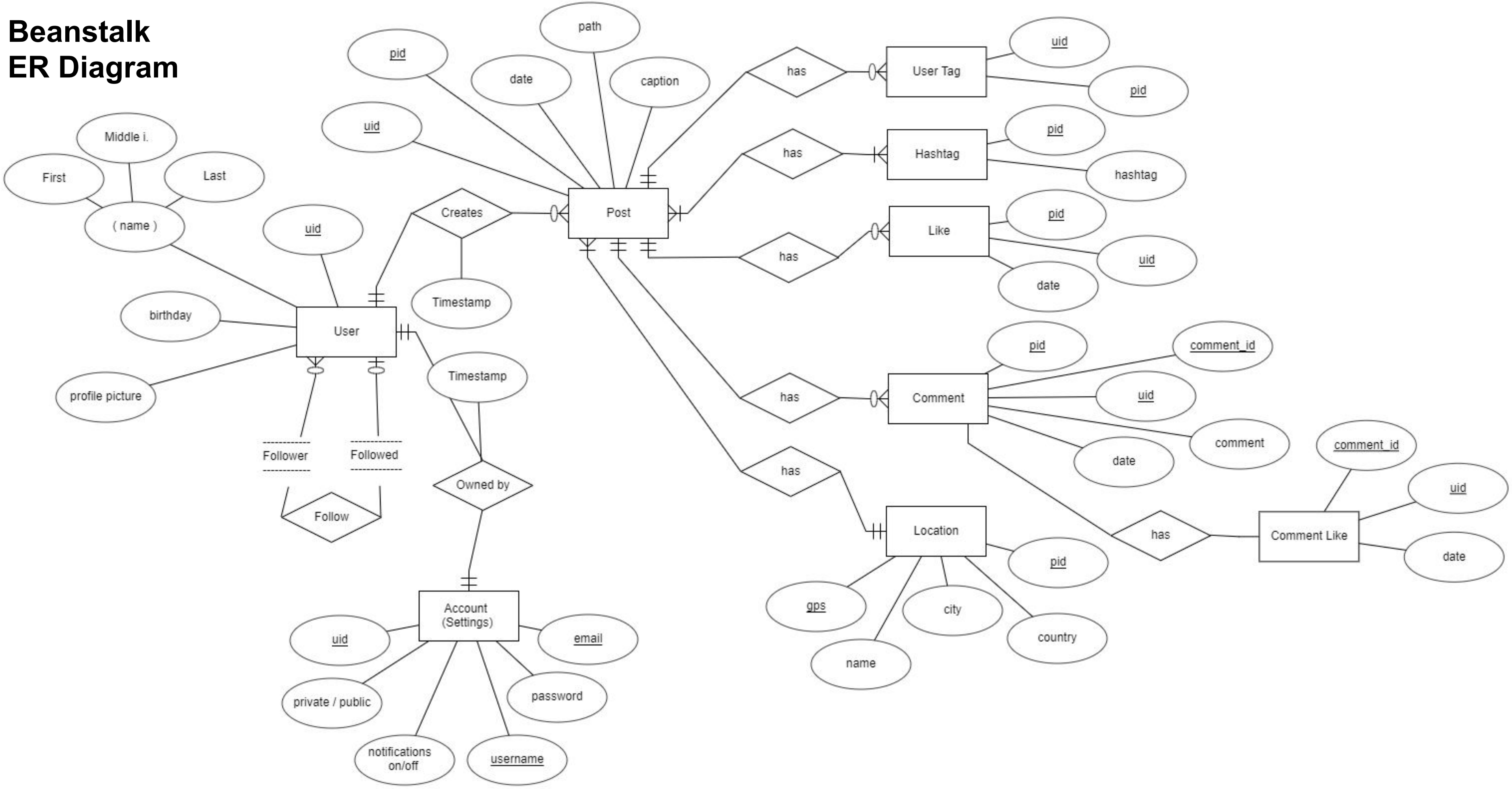
- Users will be informed of what’s going on in the app

Database Schema

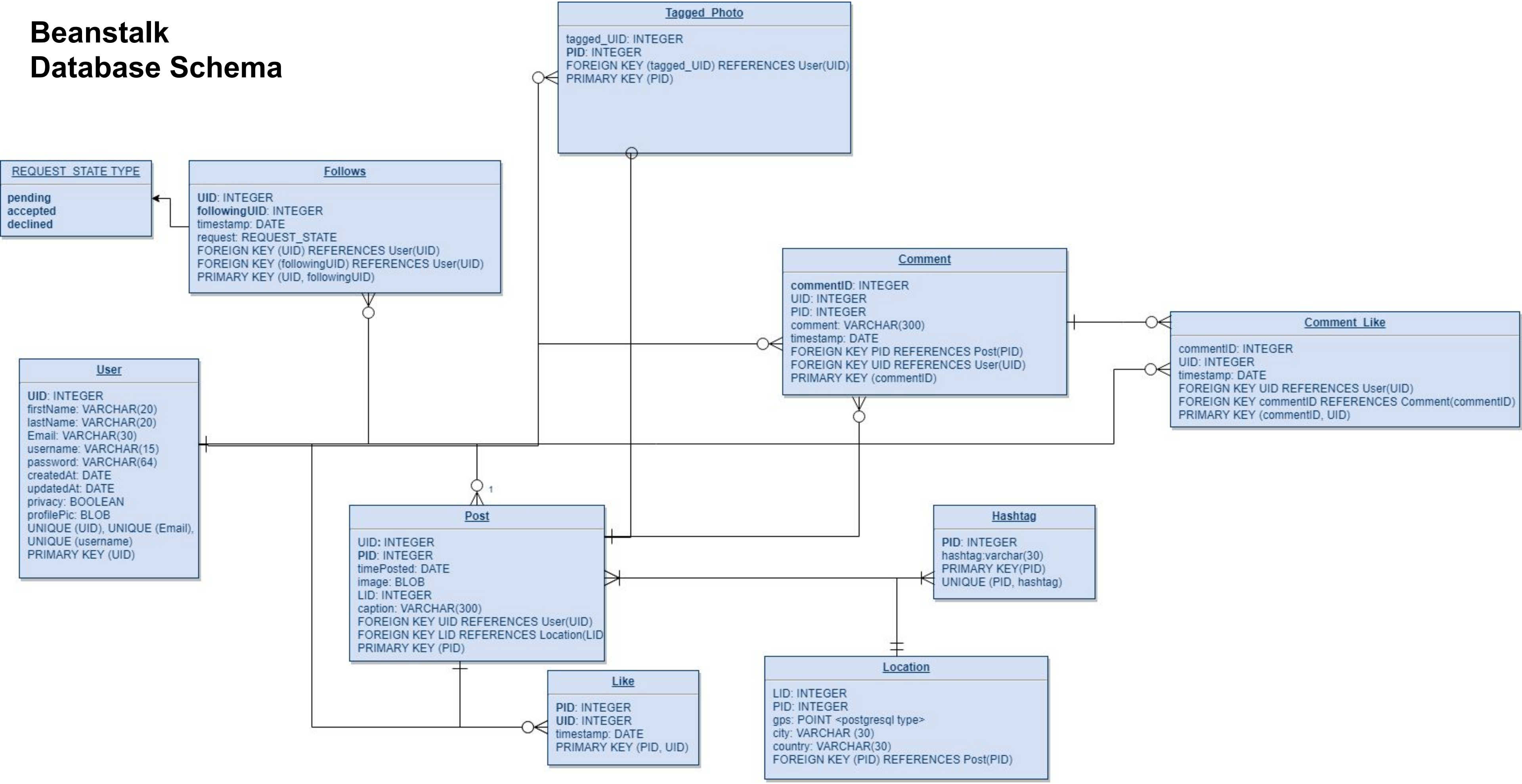
ER Diagram

Database Schema

Beanstalk
ER Diagram



Beanstalk
Database Schema



Tagged_Photo

tagged_UID: INTEGER
PID: INTEGER
FOREIGN KEY (tagged_UID) REFERENCES User(UID)
PRIMARY KEY (PID)

REQUEST_STATE TYPE

pending
accepted
declined

Follows

UID: INTEGER
followingUID: INTEGER
timestamp: DATE
request: REQUEST_STATE
FOREIGN KEY (UID) REFERENCES User(UID)
FOREIGN KEY (followingUID) REFERENCES User(UID)
PRIMARY KEY (UID, followingUID)

Comment

commentID: INTEGER
UID: INTEGER
PID: INTEGER
comment: VARCHAR(300)
timestamp: DATE
FOREIGN KEY PID REFERENCES Post(PID)
FOREIGN KEY UID REFERENCES User(UID)
PRIMARY KEY (commentID)

Comment_Like

commentID: INTEGER
UID: INTEGER
timestamp: DATE
FOREIGN KEY UID REFERENCES User(UID)
FOREIGN KEY commentID REFERENCES Comment(commentID)
PRIMARY KEY (commentID, UID)

User

UID: INTEGER
firstName: VARCHAR(20)
lastName: VARCHAR(20)
Email: VARCHAR(30)
username: VARCHAR(15)
password: VARCHAR(64)
createdAt: DATE
updatedAt: DATE
privacy: BOOLEAN
profilePic: BLOB
UNIQUE (UID), UNIQUE (Email),
UNIQUE (username)
PRIMARY KEY (UID)

Post

UID**:** INTEGER
PID: INTEGER
timePosted: DATE
image: BLOB
LID: INTEGER
caption: VARCHAR(300)
FOREIGN KEY UID REFERENCES User(UID)
FOREIGN KEY LID REFERENCES Location(LID
PRIMARY KEY (PID)

Hashtag

PID: INTEGER
hashtag:varchar(30)
PRIMARY KEY(PID)
UNIQUE (PID, hashtag)

Like

PID: INTEGER
UID: INTEGER
timestamp: DATE
PRIMARY KEY (PID, UID)

Location

LID: INTEGER
PID: INTEGER
gps: POINT <postgresql type>
city: VARCHAR (30)
country: VARCHAR(30)
FOREIGN KEY (PID) REFERENCES Post(PID)

Backend: API Endpoints and Queries

API documentation

Database Queries

Backend

- Last time: create, read, update and delete (CRUD) functionality for users relation
- Current usage: developer backdoor for root access to database through HTTP
  - GET - /api/User - Retrieve all users
  - PUT - /api/User - Update an user
  - DELETE - /api/User - Delete an user

Backend

- ➔ New API endpoints for user registration and login (returns authentication token upon success)
  - ◆ POST - /api/User/register - Register with username, email, first name, last name, and password
  - ◆ POST - /api/User/login - Login with username and password

Backend

- New API endpoints for user profiles (requires an authentication token to access)
    - Token is JWT HMAC secret encoded. Sent as “Authorization” header
- Own user profile (decoded authentication token matches the <username>)
    - GET - /api/User/profile/<username> - Get privileged info for the user’s profile
    - PUT - /api/User/profile/<username> - Update fields for the user’s profile
- Other user profile (decoded authentication token does not match the <username>)
    - GET - /api/User/profile/<username> - Get limited info for the user’s profile depending on privacy settings

User Registration - POST

```
INSERT INTO "user" (username, email, password_hash, first_name,
last_name, privacy, created_at, updated_at, profile_pic)
VALUES (%(username)s, %(email)s, %(password_hash)s, %(first_name)s,
%(last_name)s, %(privacy)s, %(created_at)s, %(updated_at)s, %(profile_pic)s)
RETURNING "user".id
```

User Login - POST

```
SELECT "user".id, “user”.password_hash
FROM "user"
WHERE "user".username = %(username_1)s
LIMIT 1
```

User Profile - GET

```
SELECT "user".id, "user".username, "user".email, "user".first_name,
"user".last_name, "user".privacy "user".profile_pic
FROM "user"
WHERE "user".id = %(id_1)s
LIMIT 1
```

User Profile - PUT

```
UPDATE "user"
SET updated_at=%(updated_at)s, <arg=value>
WHERE "user".id = %(user_id)s
```

Counting Followers

Following me:

```
SELECT COUNT(followingUID)
FROM “Follows”
WHERE UID = %(user_id)
```

I’m Following:

```
SELECT COUNT(UID)
FROM “Follows”
WHERE followingUID = %(user_id)
```

Counting Likes

Comments:

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

Posts:

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

Relational Queries

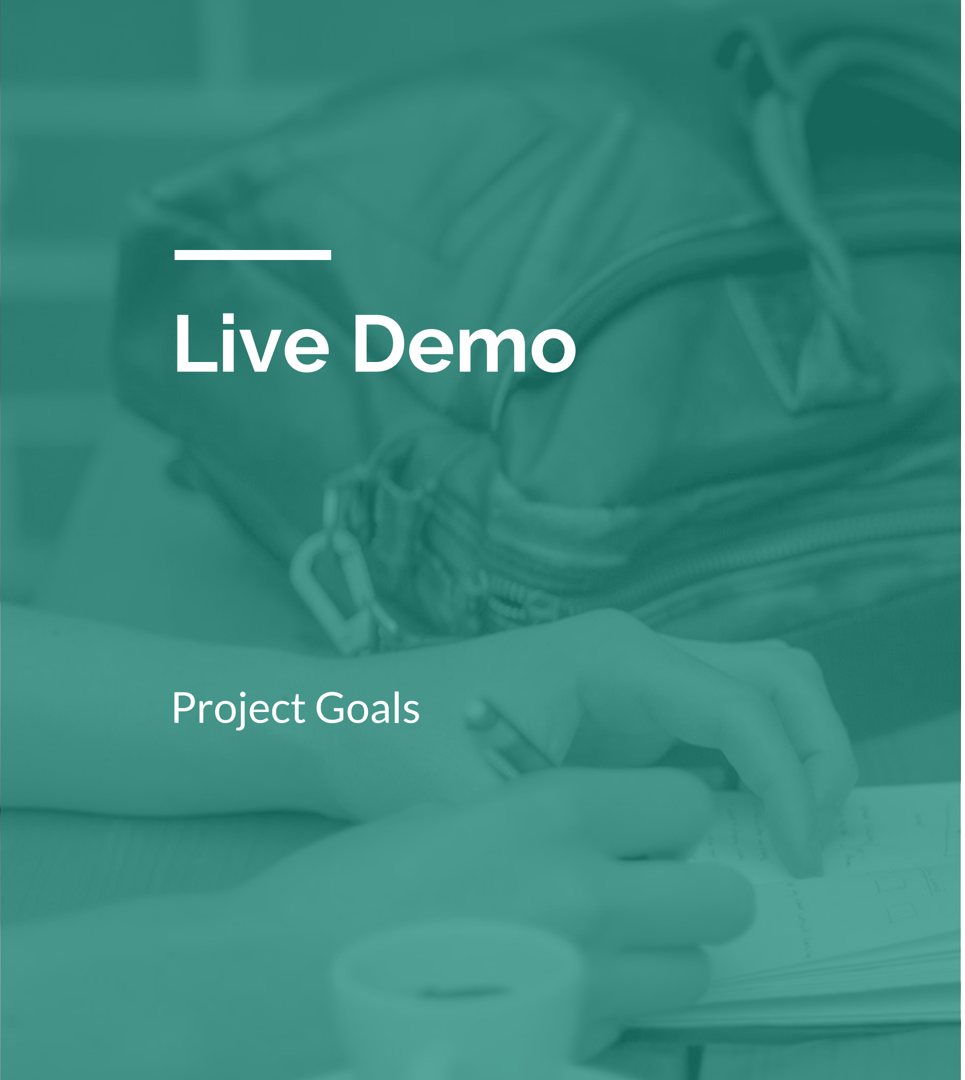
Getting all comments for a post:

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

Relational Queries

Get posts around a gps point using PostGIS:

```
SELECT “Post”.PID, “Location”.LID
FROM “Location”
JOIN “Post” ON “Location”.PID = “Post”.PID
WHERE ST_Distance_Sphere(“Location”.gps, ST_Make_Point( %lon, %lat )) <
10 * 1000
```



Live Demo

Project Goals

Registration

User Login

Authentication

User Profile Editing

Questions?