<u>TEAM MEMBERS:</u>
Bailey Boone - baileyboone
Erica Zheng - pinebay
Justin Hui - jhui04
Parth Nagraj - pdnagraj

<u>OVERVIEW:</u>
The team name: Flour

The application name: Catch

Brief Overview: A pet-based social media platform where people can publically announce a location and a window of time where other users can play with their pet. It's similar to other common social media platforms such as Facebook, Instagram, Snapchat, except that it has a real-time aspect.

Changes from the last project:
1. Have our final template.
    a. Homepage
        i. Made it so that the home page has 4 tabs: hosting, going, interested and following
        ii. Each of these tabs will go through the database to get user-specific information. As of now, it displays all current events from the database.
    b. Events page
        i. As of now, we have displayed all the events on this page from the database.
    c. Map page
        i. Have implemented a map and showcased events
    d. Profile page
        i. Shows all the profiles from the database
    e. About page
2. Our Django database has been implemented
    a. Admin page
        i. Different implemented databases in Catch:
            - Events
            - Users
            - Pets
    b. Implemented Data Model
3. Populated our database with our mock data
    a. Events
        i. Name
        ii. Location
        iii. Date
        iv. Time

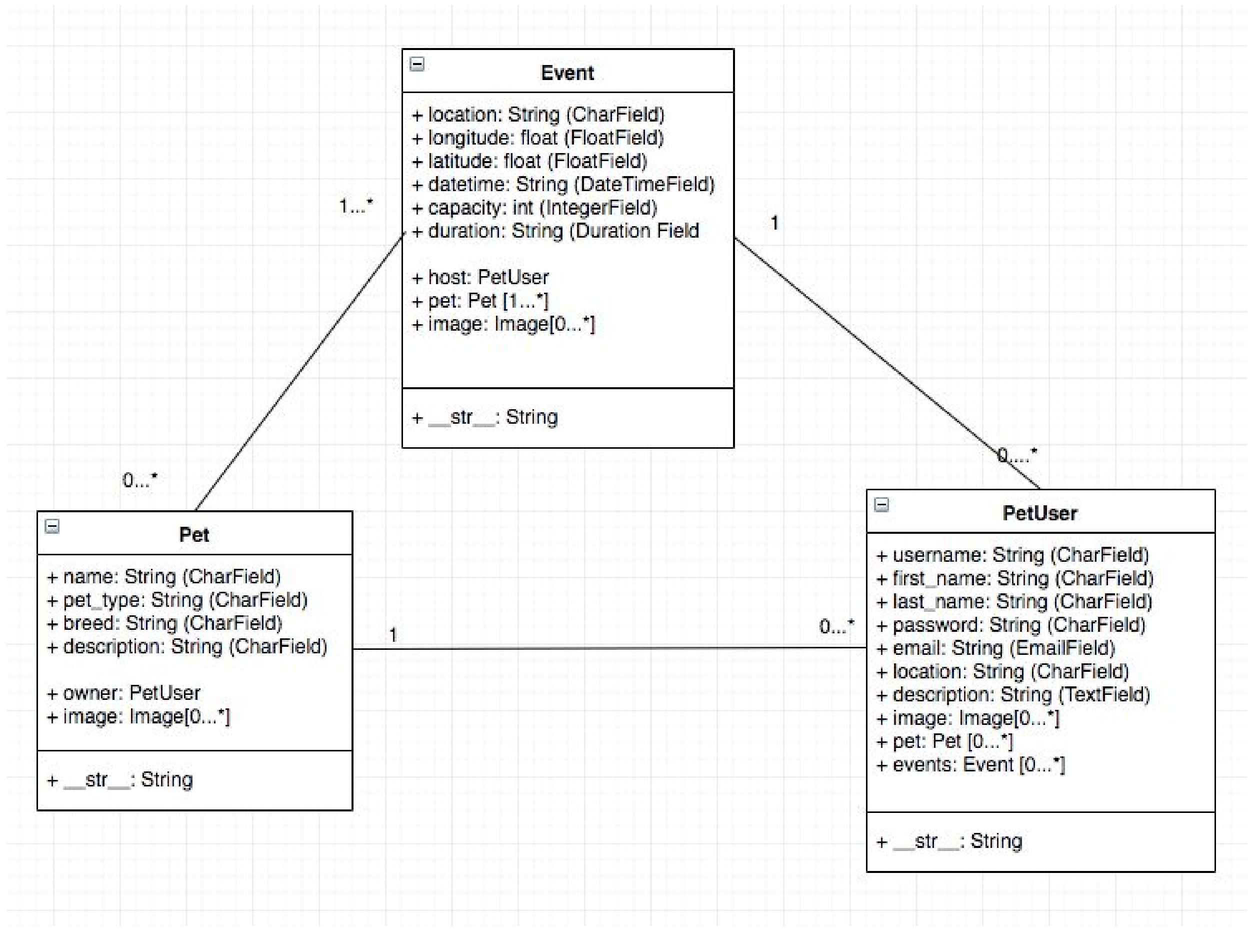
v. Capacity
vi. Description
vii. (Duration - thinking of changing that)

b. Mock Images
c. Pet Users
    i. Mock Pets pet users
d. Pets
    i. Mock Data on pets

VIDEO LINK

- https://www.youtube.com/watch?v=ZSpTD-SxA20&feature=youtu.be

DESIGN OVERVIEW


Event
+ location: String (CharField)
+ longitude: float (FloatField)
+ latitude: float (FloatField)
+ datetime: String (DateTimeField)
+ capacity: int (IntegerField)
+ duration: String (Duration Field
+ host: PetUser
+ pet: Pet [1...*]
+ image: Image[0...*]
+ __str__: String
Pet
+ name: String (CharField)
+ pet_type: String (CharField)
+ breed: String (CharField)
+ description: String (CharField)
+ owner: PetUser
+ image: Image[0...*]
+ __str__: String
PetUser
+ username: String (CharField)
+ first_name: String (CharField)
+ last_name: String (CharField)
+ password: String (CharField)
+ email: String (EmailField)
+ location: String (CharField)
+ description: String (TextField)
+ image: Image[0...*]
+ pet: Pet [0...*]
+ events: Event [0...*]
+ __str__: String
1...*
0...*
1
0....*
1
0...*

We implemented three entities in *models.py*: *User, Pet,* and *Event*, as depicted in the data model diagram above. Each *User* is identified by his or her username. From *User* to *Event* and from *User* to *Pet* exist a one-to-many relationship. Because not every *User* owns a *Pet*--some simply want to participate in events, a *User* can be associated with many *Pets/Events* or none at all. Each *Pet* is identified by his or her name and owner's username. Each *Pet* object also has exactly one owner(one-to-one relationship) and at least one *Image*(one-to-many relationship). *Users* can host as many *Events* in which their pets participate. Currently, we designed the models such that each *Event* can be hosted by only one *User*, but we may include an event collaboration feature in the future.

We defined five URL patterns that lead the users to our homepage, events page, map page, profile page, and about page. In the left column of our home page is a box containing four general event filters for events the user has shown interest in: hosting, going, interested, and following. All events in our database are displayed in the homepage in addition to the events page. Also in the events page is a more specific event filter for users to find events they would like to participate in. The map page gives us a visual to where the events are located. The profile page showcases the profiles of our users and their pets. We defined functions in *views.py* so that upon request of accessing the different web pages, the appropriate HTML file will be loaded with the corresponding context. For example, if a user requested to access our profile page, according to the profile function we would retrieve all instances of Pet and User(this is our context) and load *profilePage.html*.

PROBLEMS/SUCCESSES

Problems:

- Populating the database
- Look and feel of the website, how the UI is going to look like figuring that out was a bit of problem
- Figuring out the different tabs for our website
- Implementing the Django framework
    - There were some problems with sign-in that had to be resolved
    - We did not have a concrete idea on how our framework should look like
- Had some problems delegating tasks
- Had some problems getting our server running from vagrant
- Had some idea conflicts within the team
- Hard finding times to meet up as a group
- Hard finding times to meet up with TA

Success:

- We were able to populate the database and get it working

- We were able to figure out how we wanted the site to look like as we were having some conflicting ideas within the group, we just tried out different looks and we used one that felt most comfortable to all of us
- We were able to have the Django framework up and running
- We were able to delegate tasks and give everyone a sizeable amount of work based on each’s expertise