Title: Team Flour

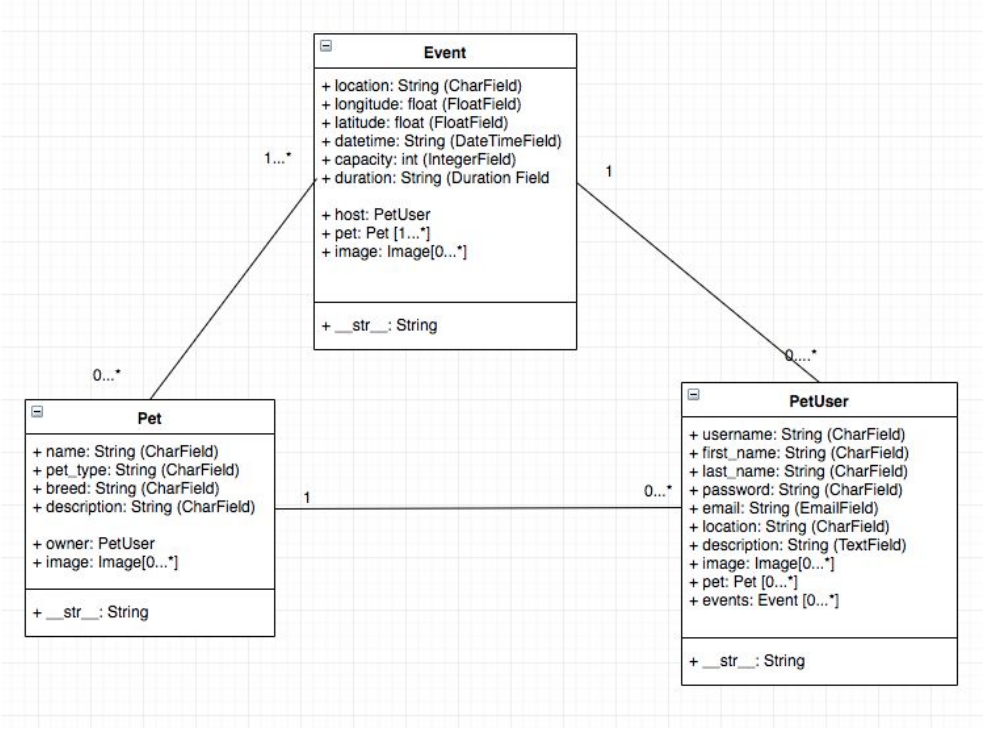
App Name: Catch

Semester: Fall 2018

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.

Team Members: Parth Nagraj, Erica Zheng, Justin Hui, Bailey Boone

Data Model OVERVIEW:



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*.

User Interface
We have a base template, navbar.html, that we use for all of our web pages. The base template contains a navigation bar on the top of the page with five tabs -- Home, Events, Map, Profile, Pets. When the user is not yet logged in, the website displays a page asking the user to either sign in or sign up. Once logged in, both the home page and the events page provide an option to create new events. The home page displays all the events and the events page displays the events the user is hosting. The map page displays a map with pins indicating where each event would be held. If you right click on one of the pins, a window would pop up with specific details of that particular event. The profile page displays account and personal information of the user and buttons to edit profile and add pet. Similarly, pets page displays information regarding your pets and a button to add pet to your account. There is a “log out” button on all pages. When clicked, it prompts the user to login/sign up page.

URL Routes/Mappings
To get started navigating on our website Catch, connect to the server using terminal commands and enter “http://localhost:8000/Catch/homePage” in the browser of your choice. This would lead you to our homepage. From there, you can login using valid credentials. All URL mappings can be looked up in Catch/urls.py.

Authentication/Authorization
Our PetUser model is extended from Django’s User Model to allow simpler account functionalities, such as login, and log out. This implementation strategy allows us to add in customized fields such as first and last names while taking full advantage of Django’s built in features, giving us flexibility and control. When the user is not yet logged in, the website displays a page asking the user to sign in. Once logged in, the user will attain authorization to edit one’s own profile and add “Pets” that will be linked to their user account. Users are not authorized to edit some other users’ profile or add pets to other users account.

Team Choice
For our team choice component we decided to add the map component where it showcases events near your location.

We implemented a map using the Google Maps API. The map features a red marker indicating the user's current location and centers the map on this point. All events are queried from the database and smaller blue markers are placed around the map based on latitude and longitude fields. When clicked the markers display the events image, name, host, and the date time.

Conclusion

In conclusion, our team has learned a lot from working on this project. There were a lot of difficulties but overall a very fulfilling experience as we all got to work on a team for a project we are all passionate about. Learning about the process of implementing the Django framework with the HTML/CSS was very difficult to wrap our heads around at first but with a lot of trial and error, we were able to have a functioning site look like the way it is supposed to. The implementation of the database was at first very hard to implement at first as we did not correctly make our data model. Once we made the correct data model we were running into a number of difficulties making the database. Eventually, we figured out how to create the database and be able to add many different data attributes. I wish we knew how to properly design a professional UI instead of a very basic looking UI. Overall amazing experience, would do this class again if given the choice.

Problems vs Success

Problems:

* Getting an image associated with the admin user and pets
* Assigning encrypted passwords with init.py
* Programatically creating a group and assigning permissions
* Editing user image profile

Success:

* Displaying events and pets based on the logged in user
* Forms that allow user to create pets and events
* Form that allows user to edit their profile

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