Clippy Project 3 Team Write Up: Gather

Overview:

The purpose of Gather is to allow users and groups to quickly organize and post location-based events. As laid out in the Project 1 submission, users start by creating and posting an event to specific individuals or groups. Events have a name, location, and time, as well as an optional image and description. Users can view upcoming events in various home, user, and group feeds and then either join or hide events. Viewing a group or another user's page will show the events shared with that group or user. This submission implements many of these goals. There are further goals such as new user creation and Google maps integration that we will continue to work on in future project submissions. We have moved away from adding comments to different parts of the application.

Team Members: Luis Quiterio, Cyrus Freshman, Connor Ingram, Michael Maio, Ryan Clayton

Github Repository: https://github.com/cfreshman/clippy

Design Overview:

As a sort of social network, this is a highly interactive application. Login/logout functionality is simple: you must be logged in to see any pages. WIthin the application, there are many ways to interact. Nearly everything in the view is tailored to the user that is currently logged in. Their profile image is displayed in the top right corner and their name is in the dropdown menu. Their groups are shown to the left, the upcoming events they've joined are shown along the top, and the main feed shows the events they can interact with (invited to, hosting, joined). Users can join/hide, leave, or edit events depending on their status. Users can search for other profiles by username using the search box in order to friend/unfriend other users. Users can edit and leave groups they're a member of. Users can create new groups and events and invite their friends/groups. Users can also edit their own profile info.

Problems/Successes:

One consistent problem we have faced is with form styling, as it is difficult/tedious/unwieldy to integrate Bootstrap with Django forms. Image uploads were another issue as well, as it turned out that there is a single form attribute that must be correctly set for forms to handle file uploads. It was also tricky getting the forms to show exactly what we wanted, like limiting the users they can choose from to their own friends. A success we've had is that the site now works almost exactly like we envisioned it!

Team Choice:

At this point, we are still looking to accomplish Google Maps integration with events. This would show a map of the location for an event in the feed without an image, and show both when viewing an event page directly. However, if this proves to be out of our reach, we plan to fall back on adding the capability for new users to sign up for the application.

Ryan Clayton: My contribution to this project includes, brainstorming about part 1 and implementing the “Group Create” and “Group Update” forms. For part 1, we needed to implement our user and group in a special way to work with our model, we talked about how to do this during the group time in class then, Cyrus implemented it after class. For part 2 I ultimately had to make an HTML form for both pages, then make 2 classes in the view.py file for the forms and then update the url mappings to display the correct views of the forms.

Luis Quiterio: My contribution for project 3 was to complete the view functions for leaving an event group and also befriending/unfriending other users. This task consisted of creating url routes in the HTML that routed to corresponding view functions in views.py. Each view function was then responsible for retrieving and manipulating model objects based on different actions that may occur on the UI.

Cyrus Freshman: I added authentication and authorization support for the application, which mostly consisted of creating the login/logout page and requiring that the user be authenticated for every other page. I also added the viewer’s profile image and name to the navbar and dropdown menu. I added interactivity to the event cards, allowing users to join/hide, leave, and edit events. I added functionality to the search box, allowing users to find and friend/unfriend other users.

Connor Ingram: For project 3, I was responsible for adding forms to the account settings/preferences page. As all forms go, this involved a tight interaction between the forms, urls, views, and the Profile model, as well as displaying it through the html template. I implemented this through a generic UpdateView on the Profile model, and given more time past this project will continue to work on the interaction between the two.

Michael Maio: My contribution for this project was creating the “EventCreate” and “EventEdit” forms. This included creating classes in views.py to successfully render the forms, as well as updating their URL mappings, and creating the templates for the forms. I decided to set the host of an event to the default of the current user creating the event and did not include every field from the Event model because some could not be entered in the form; such as the “hidden” or “joined” fields.