Getting Connected – Tools to Support Community Growth in Open Source Projects

Justin Bruntmyer
Hai Nguyen
Megan Goossens

4 December 2015

1. FRONTSPIECE

1.1 Date of Issue and Status

The design of the project was issued on October 4th, 2015. The status of the project was already in production with a prototype.

1.2 Issuing Organization

The Apache Software Foundation originally began development of the project and issued the proposal to Oregon State University for continued development. This entails to contributing to Apache Software Foundation's existing software platform to create tools that identify and support community leaders and members.

1.3 Authorship

Current developers contributing to the community tool include Ross Gardler, Justin Bruntmyer, Hai Nguyen, and Megan Goossens. Ross Gardler, president of Apache, is the original developer for the project.

1.4 Change History

2. INTRODUCTION

2.1 Purpose

The primary purpose of this document is to present a detailed description of the design elements of the Getting Connected: Tools for the Open Source Community project. This will guide the group in the design of the application.

2.2 Scope

This document will provide details on the design of the various technologies of the web application, in particular the Django framework, PostgreSQL, HTML user interface, and the user & group authentication system provided by Django.

The user will have the ability to view the community leaders that are posting events on social media sites based on tags fetched by the tools. This will allow the user to see the activity and contact information of the community leader to potentially contact if the user is interested in contributing. The user will be able to view a summary of the event as well as have a link to the source of the event hosted by the community leader.

2.3 Context

This project will be free to access for everyone. Development and maintenance will have not cost as this project is open source. Future development plans will be based on the features that do not make it in the 1.0 release of the application. There are several stretch goals that will potentially be incorporated. These features are not covered in this document.

2.4 Summary

This document will go over the design for the aspects of the project including web framework, database design, user interface, and authentication system.

3. REFERENCES

[1]*IEEE Standard for Information Technology--Systems Design--Software Design Descriptions*, 1st ed. IEEE, 2009.

[2] Comdev1-us-west.apache.org, 'Community Development: Events', 2015. [Online]. Available: http://comdev1-us-west.apache.org/events/. [Accessed: 03- Dec- 2015].

[3] Djangoproject.com, 'The Web framework for perfectionists with deadlines | Django', 2015. [Online]. Available: https://www.djangoproject.com/. [Accessed: 03- Dec- 2015].

[4] W3.org, 'HTML5', 2015. [Online]. Available: http://www.w3.org/TR/html/. [Accessed: 03- Dec- 2015].

[5] Postgresql.org, 'PostgreSQL: The world's most advanced open source database', 2015. [Online]. Available: http://www.postgresql.org/. [Accessed: 03- Dec- 2015].

4. GLOSSARY

Term	Definition
API	Application Programming Interface
Query	Request sent to a database
Query Parameter	Argument sent to a database to refine a search
Community Leader	Someone who is in charge of a project or organization
Tags	Keywords - e.g. "apache," "open source," etc.
HTML	HyperText Markup Language
Django	Web framework
PostgreSQL	Database management system
Meetup	Meetup.com

5. BODY

5.1 Identified Stakeholders and Design Concerns

For the stakeholders, the tools support and help the Apache Software Foundation. Specifically, Ross Gardler is the main stakeholder that is part of the design.

Concerns regarding the design of the project include completions of certain implementations with the requirements listed. Troubleshooting problems or possible encounters that may appear during implementation are listed as such.

5.2 Algorithm Viewpoint

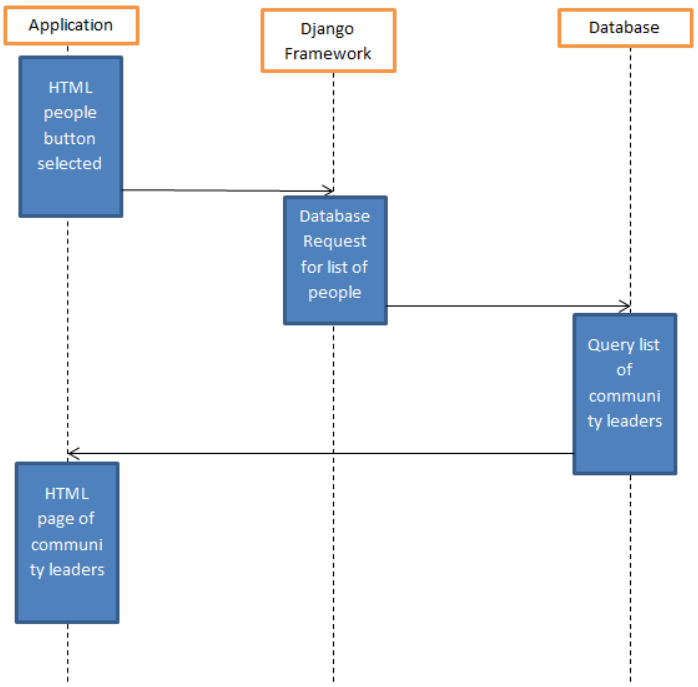
5.2.1 Fix the "People" page where profiles are not implemented

As a high priority, fixing any type of design bug that was already implemented in the prototype is important. The page listed currently takes a very long time as a link from extracting all data from the PostgreSQL database causes difficulty in the time frame. Possible ways to approach in this solution include:

- Taking a look at how the data is sorted
- Looking at the users list and what is included in "People"
- Changing the algorithm in the time frame to extract the data

5.2.2 Design View

Our fixing the people page viewpoint shows how the application will be interacting with the Django framework along with the database. We know that the button on the webpage will be an HTML button that will be selected by the user. Once selected the Django framework will be activated to query the database for the current list of community leaders. Once the database finishes the query it will send the list back to the HTML page to be displayed for the user.



5.2.3 Design Rationale

A look at how the various steps for loading the people page is necessary due to the fact that it is not obvious as to why the page is not loading properly. As there are several possible sources for the slowness, all of them must be examined to be able to determine the cause or causes. Once the problem has been pinpointed, the next step is to work on optimizing it, whether that involves changing how the data is stored, changing the query that fetches the data, or changing the algorithm that processes and presents the fetched data.

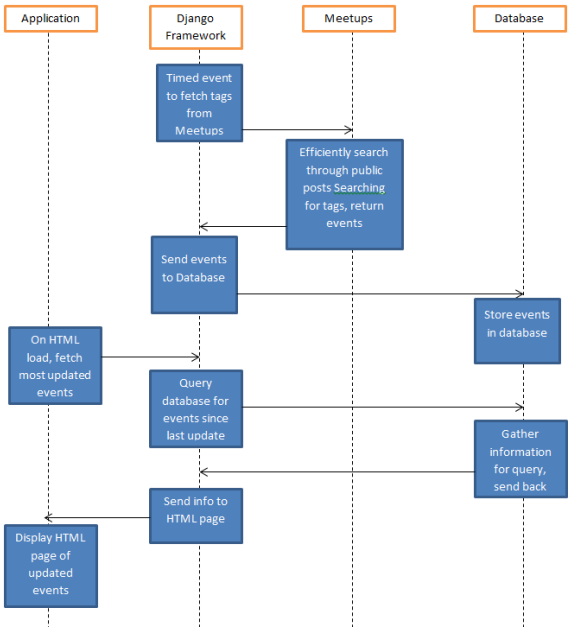
5.2.4 Improve searching algorithm of tool for more relevant events

The way the events are listed and how relevant they are to the user is important. With the prototype, events currently being extracted have the ability to not even be related to open source or programming itself. Implementing a new searching algorithm will be difficult and could cause a lot of troubleshooting listed below:

- Improving methods of recognizing relevant tags and keywords
- Removing any event that has no relation to open source projects
- (Negative search words)
- Analyze existing non-applicable groups

5.2.5 Design View

Our improvement with the searching algorithm viewpoint is shown below and mainly shows how the application will work with an efficient algorithm for searching the events. With the correct algorithm the Django framework will periodically query the meetups public posts searching for tags that have to do with the open source projects community. Once it returns those events they will be sent to the database and be stored for the HTML request will be sent to the
Django framework where the query will be sent to the database to show the list of events on the application via HTML.



5.2.6 Design Rationale

There are two parts to this viewpoint: The fetching of events from Meetup, and showing the events to the user. Getting the events from Meetup will be a regular, scheduled event that will happen at appropriate intervals. Constant fetching from Meetup, or fetching every time the events page is loaded, is impractical and would slow the website down considerably. Once the most recent set of events has been stored in the database, it is a relatively simple matter to fetch them from the database when the events page is loaded.

5.3 Context Viewpoint

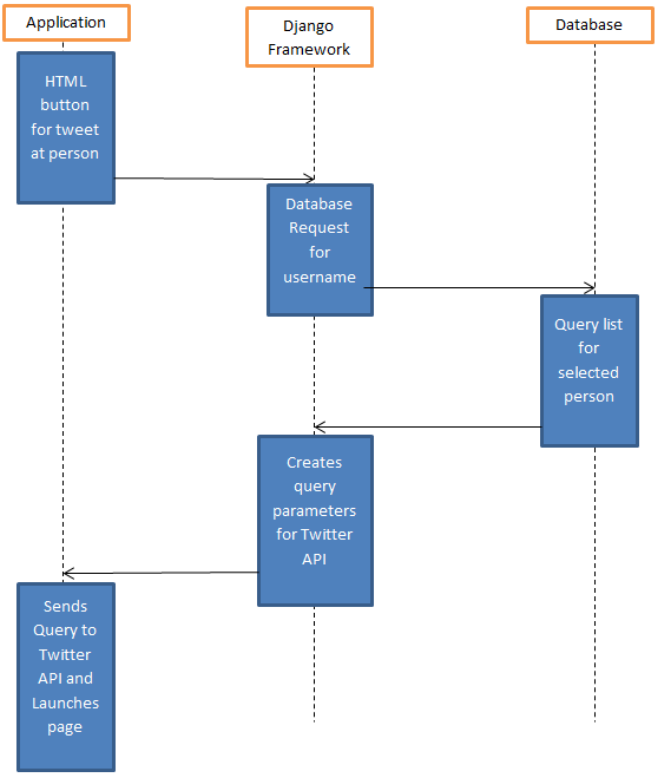
5.3.1 Tweet at a person listed in the database

Implementing another social media aspect other than just meetups.com is important for the project. This tool looks to connect every various source of data and also be able to give the user a tool to communicate with either a community leader, or a current user working on the project. Currently the requirement priority is listed as low, but possible issues for this requirement include:

- The timeframe of learning how to implement different social media
- Implementing the feature itself integrating with already registered twitter accounts

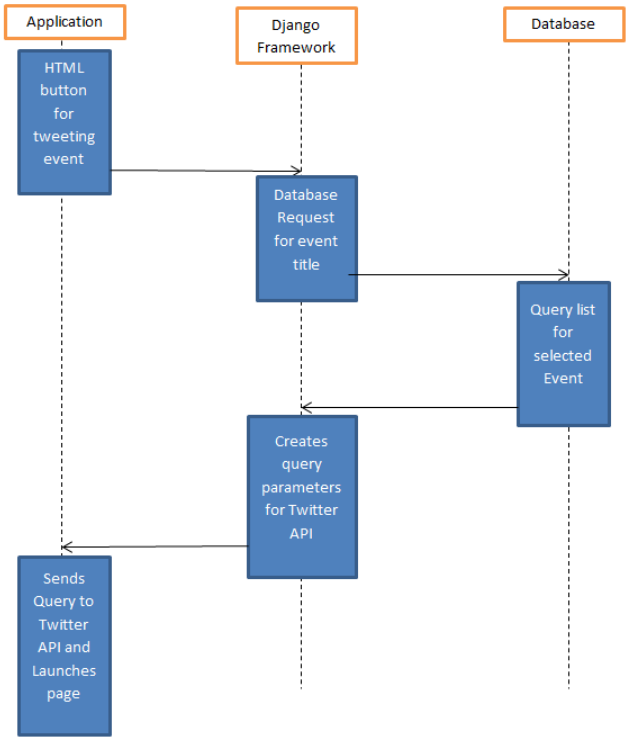
5.3.2 Design View

Our viewpoint for the ability to tweet at a community leader is shown below and has details regarding communication between the application, Django framework and the database. There will be an HTML button that can be pressed by the user and when this is pressed the Django framework will query the database for that community leaders twitter user name which will then be sent back to the Django framework. From here the Django framework will create the necessary query parameters that will be sent to the Twitter API. Once this is done the user should see a new page with the Twitter API already having the username ready and they can just enter the tweet they would like to send.



https://twitter.com/intent/tweet?query

The below viewpoint is the same process but instead of tweeting directly at a person the user is tweeting to a group.



5.3.3 Design Rationale

Twitter has a way to create tweets for you with the information you wish to be contained in them. Using their API, a pre-constructed tweet can be constructed. Every person and group's Twitter handles will be

stored in the database, so a query will need to be made to retrieve that so that it can be added to the tweet, along with whatever default message is decided upon.

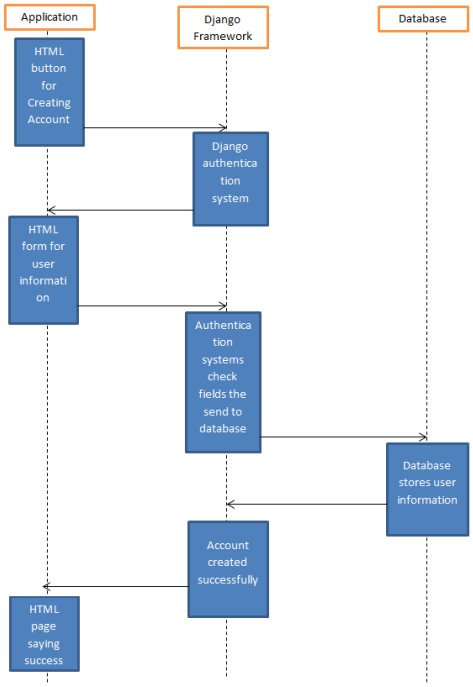
5.4 Information Viewpoint

5.4.1 Add user accounts and track when a user has tweeted an event

In hopes to increase social media feed, the tool looks to integrate movement with community leaders and help users consistently track their activity. As a medium priority, the most concerning implementation is to correctly link the account with the correct user and to distinguish their own Twitter accounts with the correct user account on the website. (Hoot Suite).

5.4.2 Design View

Our viewpoint for creating user accounts is shown below with the communication between the application, Django framework, and the database. Here the user will select the create account HTML button which will begin the actions of the Django authentication system for creating accounts. This will bring a form to the application and present using HTML to the user who will enter in the required fields. Once submitted the Django authentication system will check the fields and then send it to the database for account creation. Once this is successful the application will display a message stating the account has been created successfully.



5.4.3 Design Rationale

As the application is build in Django, we will be using Django's authentication system for user accounts. It is the best choice for this project because it is simple and already implemented within the Django platform.

5.5 Interface Viewpoint

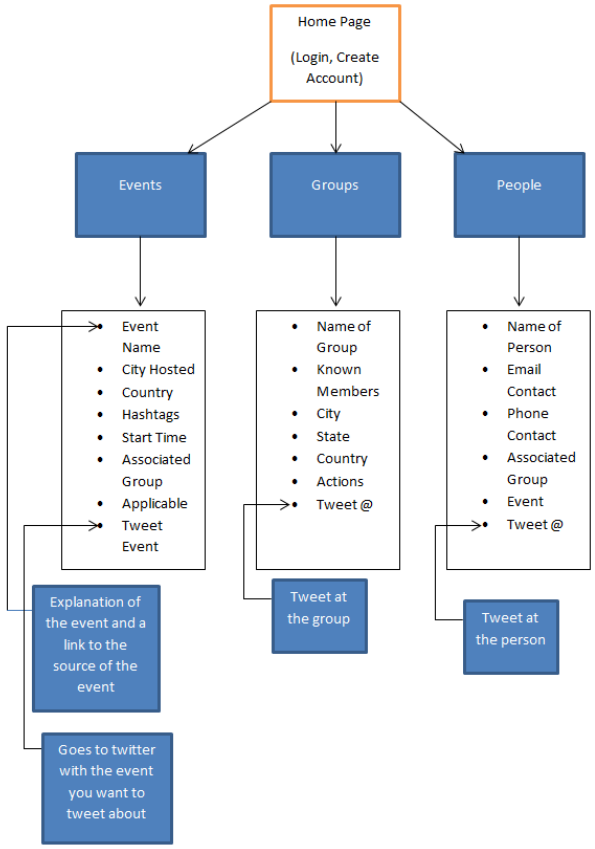
5.5.1 Improve viewing method to examine the event itself

Improving the way users view the website is important to make it look more pristine and clean. All information should be clearly sectioned and any relevant information that the user would like to know should be put in a convenient viewing form.

- Categorizing certain data could turn to an issue in organizing the data
- Some data will be missing and handling that missing information in the event should still give the user enough information to look for it

5.5.2 Design View

Our viewpoint for the improve viewing method requirement that we will be implementing is shown below. This shows the layout of how the information should be spread out through the website. This is a basic look at how each location will store data for the user to easily navigate through in order to find what they are searching for. There are also a couple of links that are described for the tweet at someone or tweet at an event.



5.5.3 Design Rationale

The goal of this viewpoint is to make it as easy and intuitive as possible to view and interact with people and events. It is important that the relevant data be shown in such a way that makes it easy to find what the user needs, as well as easy access to the original information.

5.6 Resource Viewpoint

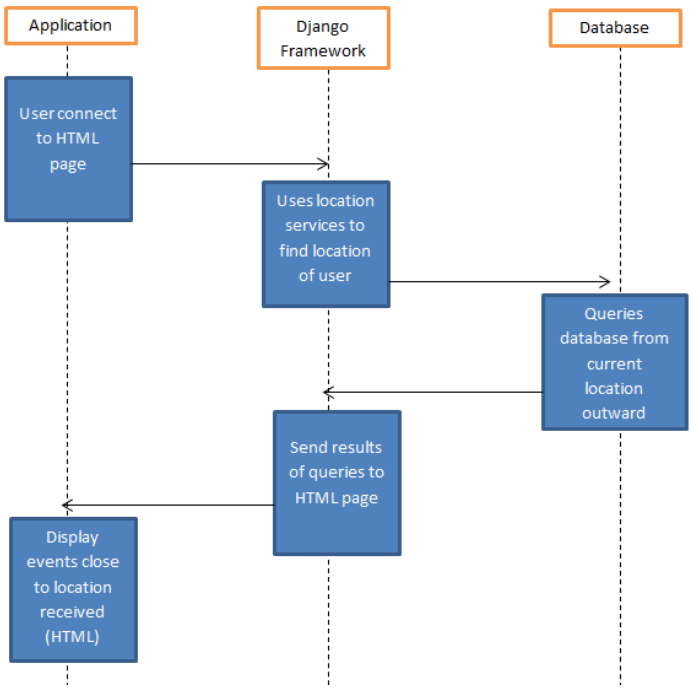
5.6.1 Implement system of improved sorting of finding events by nearby location

Listed as one the more difficult requirements to complete, the task is given a large amount of time to complete in order to get it fully functional. It is important and more relevant for the user to view events that can easily be travelled to by their location. Developing a way for the application to view and sort those locations could cause some concerns.

- Creating different approaches in sorting locations of the events currently listed and detecting the location of the user
- Creating a database of all locations and their immediate distance from each other

5.6.2 Design View

Our viewpoint for the location nearest the user functionality is shown below. We plan to make use of how most internet browsers have a location setting which can tell where a user is accessing the website from. From this information we will query the database and look for upcoming events that are close to the given location and return those events to the application to be displayed. This is our most difficult challenge and one that will take a lot of testing.



5.6.3 Design Rationale

This viewpoint will require manipulating location data. Once the user's location has been determined, either through location services or through manual entry, we will have to perform a search through the database to find all events within a certain distance of that location. This will be the most difficult part of the entire task. Once the events have been found it will be relatively simple to show them to the user, sorted by distance, so that the user can view those closest to them easily.

5.7 Composition Viewpoint

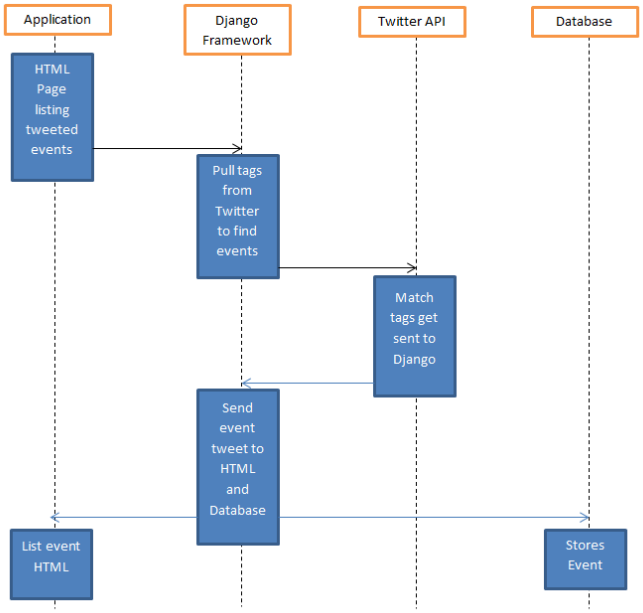
5.7.1 List tweets about events and/or people via the app and without

Currently the prototype lists only official events found on meetups.com. It is important to also view events not listed on that website and that are posted via Twitter. Design concerns include:

- Designing a way to distinguish between a regular tweet, and a tweet with an event
- Linking twitter accounts with the right user and the matching account
- People who tweet about events, but are not a part of the application
- Search via API all about the meetup page and linked towards events
- Create entry in database about a certain person tweeting at a certain time
- Want to be able to see who tweeted, and able to retweet that tweet

5.7.2 Design View

Our viewpoint for the tweeted events requirement is shown below with the communication between the application, Django framework, Twitter API, and the database. We plan to have the application communicate with the Twitter
API such that the items that we find with the associated tags we will use will be displayed on the application and stored in the database.



5.7.3 Design Rationale

Once again the Twitter API will be useful in accomplishing a task. The API will be used to find tweets that have hashtags that are related to open source events. Once those have been acquired, they will be presented to users on the website as well as stored in the database for future reference.