Brian Black

Mychal Calderon

Lexi Christiansen

Final Project write-up

Statement of the Problem:

Create a flower shop intro level web based event system which would allow for an interface of flower shops and delivery companies work together to create a large network of flower deliveries. The problem we are trying to solve is how to have multiple flower shops deliver to multiple homes through multiple delivery services. This modular way of connecting companies with their customers could then be used by many other companies as a delivery API

API’s we chose

Foursquare and Twilio for quick, real-time updates about where deliveries are and when deliveries have been completed. It allows the users to find and receive data in a quick and easy fashion through simple push of data to consumers and companies.

Events Generators, Consumers

Events

We chose two different types of event domains for now: rfq (for request for quote) and delivery. The rfq events are for any talk between a flower company and a delivery company. Through these events we discuss bids for delivery and delivery times. The delivery event domain is for discussing where a delivery is at any given time. We can notify the customer whenever there is an updated status in the delivery route such as picked up, in transit, or arrived. An outline for each event is listed here.

* Delivery:picked\_up – the delivery is out of the shop and on the way to the customer
* Delivery:complete – the delivery has successfully reached the customer
* rfq:bid\_avaliable – Send a request to the flower shops that there are drivers ready to deliver products
* Rfq:bid\_awarded – Send an update to notify a driver that they won the delivery bid
* Rfq:delivery\_ready – Send an update to the driver that won the bid that the delivery is ready to be picked up

Diagram

Method

We went with the micro-framework flask which allowed for a python wrapped website with the PyPub Api for using event passed programming with python and a Sqlit3 database.

Flask was our choice of framework because python allows for simple scripts to accomplish large work in a short amount of time. For example, a simple script would be able to pair a flower delivery with a delivery company that works closest to the consumer. This simple setup also allowed for easy navigation between web pages and allowed for new features or pages to be added quickly and easily.

The PyPub subscriptions allows different components to subscribe to events in a fashion similar to KRL with listeners and receivers. We made our listeners generic enough that it would be useful for outside users to send requests and made our event calls simple enough that they would be found by either our flower shop sites or our delivery company sites.

We chose Sqlit3 as our database as it is lightweight and good for a small/module system. The main focus of our group was a way to join multiple shops and delivery companies to better serve the consumer. Our database did not require a large amount of data about the flower shops or the delivery companies. Only enough to know where to place a bid and what company accepted the bid.

Analysis of Events

These events were a great idea because they allowed for a non-intrusive way of keeping information flowing through simple push commands to consumers. The passive arrival of event calls allowed for a more streamlined external API calls as well as being unobtrusive when being called by other systems.