**Last updated 02/11/16**

This will be a no means comprehensive outline of the project, but a rough outline, major and minor bugs, the main large scale tasks, and a changelog.

The basic application acts as an interface between the services provided by Pandora bots – the provider of the chatbot software, which is based on an AIML configuration system – and social media – currently twitter via tweepy library and it’s API.

The primary purpose of the application is to facilitate the creation of multiple chatbots, each with their own specific configuration and purposes across multiple twitter accounts. These should be easy to manage with an admin interface for those relatively technically uninclined – students, professors, etc.

Future revisions may include more management tools for the configuration of the chatbots to allow for basic addition to the bot knowledge without being entirely familiar with the AIML system of pandorabots. A suggested route for this is <https://github.com/UNC-Libraries/jquery.xmleditor> , however I would like to get input from someone who is reasonably familiar with the templating language to figure out a valid tool; this may involve creating a few basic editor prototypes tied to an aiml.

Description of functionality

**Database**

**Type: SQLITE 3** ( Has potential to be upgraded if necessary )

**Models:**

**Cbot (Chatbot Instance):**

* Created (Date)
* Title (Django title; purely internal)
* Pandora Name ( The name of the bot on Pandora)
* AIML Config ( A link to the AIML configurations in use )
* Enabled ( Is this chatbot supposed to be running/enabled )
* Chatbot Name (Twitter Name ; the name of the twitter chatbot)
* Twitter Hashtags ( The hashtags and keywords that the twitter bot will respond to aside it’s own name)
* Author (owner, only used internally)
* Twitter Token ( Twitter API config )
* Twitter Token Secret (Twitter API config)
* Twitter Consumer Key ( Twitter API config)
* Twitter Consumer Secret ( Twitter API config)

Special Notes: N/a

**Pandora Settings**

* App ID ( Pandora account API detail )
* User Key ( Pandora Account API detail )
* Host ( Pandora Account API detail )

Special notes: Only one instance allowed in database.

**AIML Config:**

* Title ( Internal name )
* Aiml\_file ( linked files for this configuration)
* Last modified
* Author (owner, only user internally)
* Is\_public ( should be visible to all)

Special Notes: any number of files can be selected.

**AIML file**

* Docfile ( the attached File Field holding the file )
* Text\_file ( a text field which replicates the contents of the file on upload, and takes precedence. Also provides the simple file editor)
* Author (owner, only user internally)
* Special Notes: N/A

**Technical notes**

If you find yourself working on the application the following might help:

The django application acts as the database and front end. The database contents are listed above.

The twitter interface(scripts/twitter\_bot.py) is responsible for taking all matching tweets, and replying if they match certain specifications. The log of tweets that meet these criteria and these responses are held in chatbot/script/chatlogs/[twittername]/tweet\_log.txt . This may change in the future, and some integration into the visible side of the app would be beneficial.

The twitter interface is to be kept online by regular cron tasks. To this end it is connected to a simple django command which checks if the bot is enabled, and if it is the cron task should pick it up and launch it.

Chatbot/scripts/process\_monitor.py contains a function for initiating the twitter script. This is where database values are passed to the appropriate twitter script.

The response that the twitter interface uses is provided by Pandora, which needs to be fed with information through it’s API. The chatbot/script/pandora\_actions.py file contains most of these files, which can be accessed through the interface or by URL. Some functionality can be found in chatbot/views.py

The particular implementation of the phusion passenger wsgi server only updates compiled python files when the file tmp/restart.txt is touched or updated in some way.

**Major To Do’s**

**Minor Priority To Do’s**

This may count as minor, but the system doesn't go to any particular effort to synchronise pandora and the web application, particularly in regards to it's model-based file system. This might be something to remedy after a successful prototype/project is launched.

**Known Bugs:**

Chatbot doesn’t always compile – leading to uninformative errors

The Pandora python script outputs strings and is not future proofed; would be ideal if it were json.

Pandora delete/file listings/etc – when there are no maps/sets/etc a dead link will appear.

User Story:

It's a Tuesday. It's raining outside. A soft pitter patter of droplets impact the window lending a soft drumbeat to the room inside. Barbara sits by her computer at peace, as although she is on fire – in the most literal of senses – she is going to do some configuration of a Teacherbot for the students of her MOOC.

Barbara simmers gently, and loads up the link provided to her by email, and logs in with her password.

She has pre-prepared a small set of files to add to the bot, and needs to urgently delete some of the existing configuration on the live bot, as it has been harassing questions up to the point that a young chinese business student decided to reinvent gunpowder and make a merciless death charge on the server room.

But this is besides the point. A plastic cup near Barbara's arm starts to melt. This is also besides the point. She reaches the first page – an index of currently activated chatbots she has permission to access, selects the appropriate chatbot and then deletes several files, followed by uploading her new files.

--- Main needs: Uploading, and deleting should be easy to do. Barbara here has been done some research into the AIML language and fortunately understands the ML very well.

--- Indicates: need for permissions to work, otherwise inevitably someone will break someone else's bot.

**Current Work:**

Added create-a-bot tab, intending to