Project Documentation

YouBox - the website beyond YouTube and kkBOX

Contents

1 Introduction

Good music is never enough. Everyone who listens to music will always want to discover more great songs. However, there are so many different music in the world that makes recommending to every single user a hard thing to do, not to mention everyone's got a different music taste, which is evolving every day. YouTube recommends similar music to the ones you've listened, but it will never recommend you the ones you haven't been in touch with, which you might just love. However, people who have similar music taste with you will discover all sorts of music, maybe the one he heard on the street, or the one his friend recommended, and there is a great chance you will like it as well since you've got similar taste.

We think people can only identify such thing as 'music taste' by the music they like, and only human can identify it. Instead of music category defined by system like in MixerB0x, we think a person can actually stands for a music style as well. We leave the music category defined by users. So we personalize each user with the YouTube videos they have shared/posted in the website before, and let users decide whether or not to follow other users by themselves. Youtube currently suggests you videos according to your video-watched history, but lacks the feature to suggest video based on your friend's taste. In your homepage, you will see posts of other users you are following. Our website will also provide many other ways to explore music.

For those who are always craving for more music and who enjoy listening to music but don't know what songs to listen to except Taylor Swift's songs, this website will be the best place to discover the music just for you.

2 Project Design

Our website is designed for music fan and lover. Just as Instagram, a place to let photo speak out your life, our website aims to build a social network based on music clips.

2.1 User Scenario

Say you just broke up with someone, you found a touching music, your heart brims with words and thoughts of "*son of a bitch!*". Comment that under the video on YouTube? You'd probably be downrated and people replying "*what the hell is wrong with you?*", because they just don't understand. In YouBox, all your friends are here, you follow each other and share music and thoughts on life. You know in YouBox, they always stand beside you. You decided to post the video using our handy POST Modal, and leave the message using your true words and emotion. Soon after, comments with comfort and warm words arrived. You realize YouBox is indispensable and life without YouBox would be a tragedy.
You may ask, won't Facebook suffice the case above? Surprisingly, YouBox is far beyond that. YouBox is essentially designed to construct a "music" social network, a blend of social and music functionality. Besides the social part, YouBox is equipped with complete music playback feature and robust playlist management.

2.2 Social Design

For the home page, we apply the common approach - wall feed - used by Twitter, Facebook and many other social websites. Each feed contains a video, message written by the poster, and comments downside. Each user has a profile page, similar to that of facebook, displaying user's own feed. To explore new users, a recommendation system will list up users of interest. One can link to another by following him.

2.3 Music Features

Music needs and functionality are carefully considered. Addition to basic playback, start/end time option to realize A-B section playback is included. Within profile tab resides the playlist manager. A default playlist which contains video you liked, and

you can create new customized playlists. We drop the use of YouTube playlist since it lacks vital features such as looping a video and changing video order. Instead, YouBox playlist provides complete options on looping, A-B section playback, and a handy drag-and-drop interface. Not only can you change the order, but also dragging video among playlists. Life is getting easier in YouBox.
Moreover, our website possesses a rich database of various music categories up to 12 to serve every picky users. You can find music styles like “pop”, “rock” or by language “Japanese”, “English”. The database is built simply by a developer account posting 50 threads. Building such database will solve the problem that no music is presented right after the website released to public. While one may prefer using algorithm to classify music type automatically, the effort required may exceed the scope of this project. After all, accuracy of such algorithm is not guaranteed. Although posting 50 threads doesn’t sound clever, it is an one-time work, and it gives exactly “YouBox presented” best music over each categories.
The category for video newly posted by user requires special care. We decide to leave the work to user. When one creates new post, he is expected to select the correct category. Concerns about correctness may raise, and how to handle this is left as future work.

3 Project Architecture

3.1 Framework

We use Django as our backend framework. For frontend, it is a mixture of LESS compiled CSS base on bootstrap source code, jQuery for manipulating DOM objects, and other libraries realizing fancy mouse hover effects.

3.1.1 The Frontend

Our front end also relies a lot on Django framework using template tag and extend technique to organize our HTML file.
Generally, it is easier to craft out layout in HTML. Dealing with DOM inside JavaScript is tedious. We found understanding the layer structure is a tough task even using jQuery. We may apply some famous frontend framework indeed such as AngularJS, react, yet all of them require extra time on document reading. Since our website covers a large scale of functionality and our time is limited, it is an immediate choice to use bootstrap only. This means we have to handle object state ourselves(the active state switching among several tabs). As a principal, do all the

rendering work in HTML template. Only when a task needs to be triggered by user(perhaps a state change) and requires lots of database entry should we use ajax or render new DOM in JavaScript(the *Like* and *Comment* feature).
And of course the YouTube iframe API. The API allows us to initialize video and control playback by JavaScript code and HTML iframe.
The bootstrap default style is far from satisfactory. We found fancy animation response in many websites uses rather customized bootstrap or other library such as flat UI or bootstrap module. Even examples provided by some bootstrap tutorial sites use variations that requires lots of change. As a quick start, we use existing code from these examples to fulfill animation effects which improve UX significantly.
For relatively simple job such as border color, background color and font, we use a central-managed and customized bootstrap CSS, residing in the base HTML template.
Customizing bootstrap CSS is done by compiling LESS files and generate CSS stylesheet. Now, the bootstrap source is cloned and combined with our project as a git submodule. We have our LESS file containing all settings in form of @variables. Most of them are bootstrap-predefined, and we just modify them, so it does not change the framework structure. As a rookie to LESS and Node.js, we spent much time on reading tutorials and, considering extendability and maintainability, we want to do it in best practice, which requires extra research.
Local settings which should not apply to other pages, is required to reside in individual HTML file.

3.1.2 Backend Technology

Compare to frontend, the Django backend is rather simple. Given an request, the url.py will handle it and dispatch the request to the designated View. The View accesses data defined by Model, and then response client with a HTML template. Dynamic contents are passed to the template as variables. Another approach to render dynamic contents would be an ajax call in the HTML file(perhaps using a <script> tag or explicit JavaScript file). One then has to implement the corresponding url.py and View to generate response.
Django is a frequently updated framework. Many information we got in StackOverflow is out-dated and deprecated, so it is challenging to maintain such framework especially when it comes to advanced usage, such as customized user authentication, inherited model, and so on.
Also, our project uses git as version control system. This raised issues related to database. Even if the database file is excluded in .gitignore, the directory /migrations/ is not, and should not be, according to Django official document.

Certainly, when database field is modified every member should apply the change and do “manage.py migrate”. However, due to operating system, or the python version used, some developers got database-related issue while others did not. Resetting database and clean up migration directory is a quick fix, but the git will propagate such reset to others, and thus cause numerous error during the development. As a final solution, all team members were informed to reset the database.
All packages are recommended to be installed under a virtual environment, as a policy for best practice. Remember to exclude such folder from git in .gitignore.

3.1.3 Deployment to Heroku

We followed the Girl’s Django Tutorial “Deploy” section(see Reference section). In this section we will describe issues we faced, and the solution and debug techniques on Heroku.
The first issue is related to directory structure. It seems that Heroku expects the project located in the root directory and using subdirectory raises error.

3.1.3.1 Database-related Issues

A common case is the webpage presents “Exception: ObjectDoesNotExist”. To check out the database contents, /admin/ will suffice in most cases. This requires registering models and fields in admin.py. If the /admin/ is broken as well, you may want to check out python shell message by “heroku run python manage.py makemigrations/migrate” and see if any error presents.
Additional option is to look into the PostgreSQL database on heroku. Run “heroku pg:psql” to enter PostgreSQL shell, and type “\d” to list out the table. You can look into a table, delete problematic entry, drop or truncate(e.g. clean content) a table using SQL command. Please refer to the Reference section.
To modify file system on Heroku(e.g., delete /migrations/), the common practice is to edit your local directories and push to Heroku. However sometimes you want to keep the local database untouched. You can enter bash shell on Heroku by “heroku run bash”. Here you don’t need to add prefix “heroku run” before any command so is quite convenient for migrate/makemigrations debugging as well. You may want to remove all pyc files under an app, then do “rm -rf *.pyc”.

3.1.3.2 Watson Issues

The search engine Watson requires its index table be empty before doing “python manage.py buildwatson”. So if you face error, try to truncate the problematic table. See Reference for more details.

3.1.3.3 SciPy Issues

The SciPy, numpy and sklearn packages needed by the recommendation system raise numerous issues on Heroku. Except numpy, the other two fails to operate on Heroku. It turns out that “pip install” is problematic for these packages. Our local environment did work, but Heroku failed.
After lots of research, the multiple buildpack method finally worked out. First we customize our heroku app buildpack via “heroku buildpacks:set https://github.com/ddollar/heroku-buildpack-multi.git” and add a .buildpack file, containing the 2 lines:
https://github.com/conda/conda-buildpack.git
https://github.com/heroku/heroku-buildpack-python
Here we use the conda buildpack along with heroku official python buildpack. For official python packages, specify them in “requirements.txt” as
dj-database-url==0.3.0
dj-static==0.0.6
Django==1.9.1
gunicorn==19.4.5
static3==0.6.1
psycopg2==2.6.1
For conda packages, we specify them in “conda-requirements.txt” containing the 2 lines:
scikit-learn==0.17
scipy==0.16.1
Lastly, runtime.txt should be python-2.7.10; otherwise, numpy and sklearn raise build error.

3.2 Directory Structure

./git root directory.....This root cancelled and project root moved upward
|
|__musicapp/.....Django project root folder
| |__login/
| | |__model.py.....Customizing user model
| |__musicapp/.....This is the main app of the project
| | |__urls.py.....Handles mapping of url to view
| | |__settings.py
| |__musicSite/

```
|               |               |__static/
|               |               |       |__bootstrap/
|               |               |       |__js/
|               |               |       |__img/
|               |               |       |__site/
|               |               |               |__js/.....!!
|               |               |__templates/
|               |               |               |__playground_base.html
|               |               |               |__...(most of the templates)
|               |               |__view.py.....Handle most of the webpage access/requests
|               |               |__admin.py.....Customizing /admin/ view
|               |__posts/
|               |       |__model.py.....Defines most of the database
|               |__search/.....Search Tab
|               |__account/.....Account Setting page
|               |__watson/.....Search Engine
|               |__manage.py.....Run Django server by this file
```

3.3 Project Dependency

```
# [prerequisite]
python 3  python2.10.7 preferred (Scipy requires python2 on Heroku)
pip install django
pip install dj-database-url gunicorn dj-static

# [change fields in db]
python manage.py makemigrations
python manage.py migrate
# In case the db crashed, reset it by: delete all “migrations” folder and db file. Run following command for each app, then migrate and it should work
python manage,py makemigrations {individual app name} # this creates migrations folder

# [change fields in watson search engine]
python manage.py migrate
python manage.py installwatson
python manage.py buildwatson
```

[for cosine rating recommendation]
pip install sklearn numpy scipy
for windows, numpy requires VC files. Install Visual Studio C++ and check the box "Common Tools for Visual C++ 2015" during installation
for windows, scipy and numpy may fail. Please see drewid's answer here(http://stackoverflow.com/questions/28190534/windows-scipy-install-no-lapack-blas-resources-found) . Other reference to listing out pip support version (http://stackoverflow.com/questions/28107123/cannot-install-numpy-from-wheel-format). If you face failure, try to delete the whole virtualenv and start over again. Uninstall may remain some broken component thus makes things unworkable.

[compiling LESS to CSS for customizing bootstrap]
this requires python2
do this under a python2 virtualenv. If you are using python3 venv, you should create one for python2
pip install nodeenv
integrate with virtualenv
nodeenv -p
install LESS, use -g as global. However, you may want to restart the terminal first. This will let it be under the scope of virtualenv.
npm install -g less

3.4 Release

Access released deployment on Heroku via (https://youboxapp.herokuapp.com).
See our promotion poster here (https://docs.google.com/presentation/d/1ifeyc0LSjtniFfMQ_ynaaMLLONY-AXdkGqR4AYsgGeo/edit?usp=sharing)

4 Global Component Development

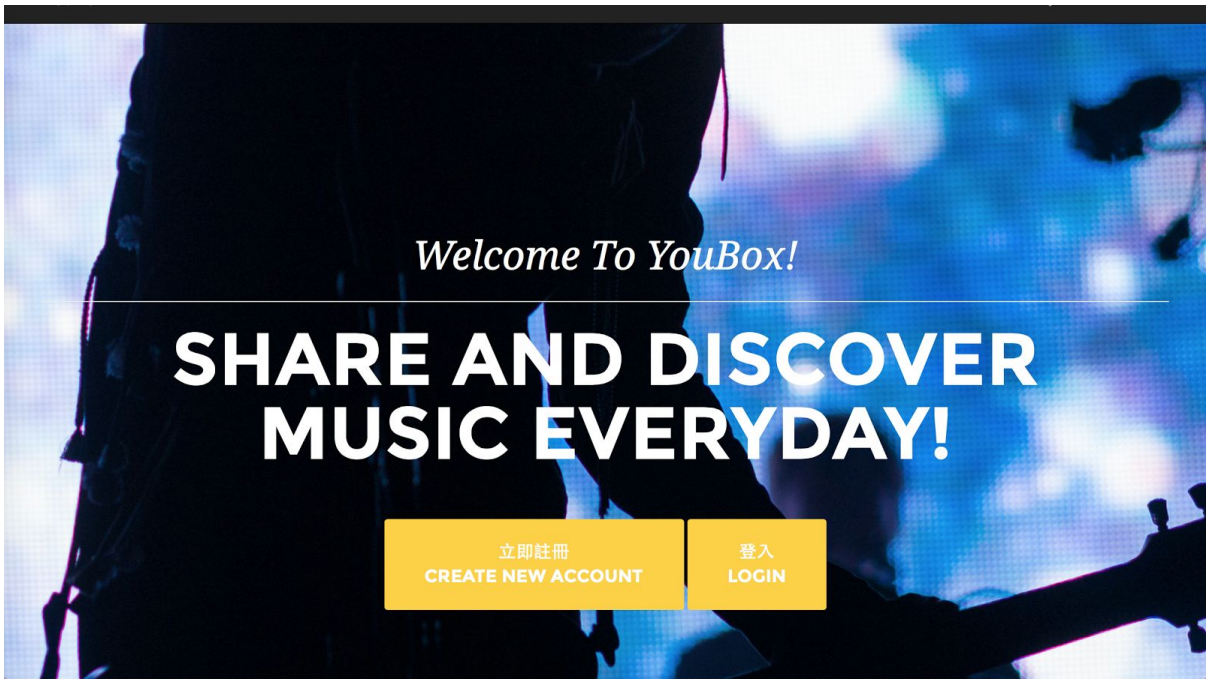
4.1 User Authentication

Including login/logout, register. In order to let the user have the default methods and fields of the User model and also obtain the customized fields that we need, we extend the default User model. Django contains a user authentication system that

we could use to implement the login and logout operations in a easy and simple way.

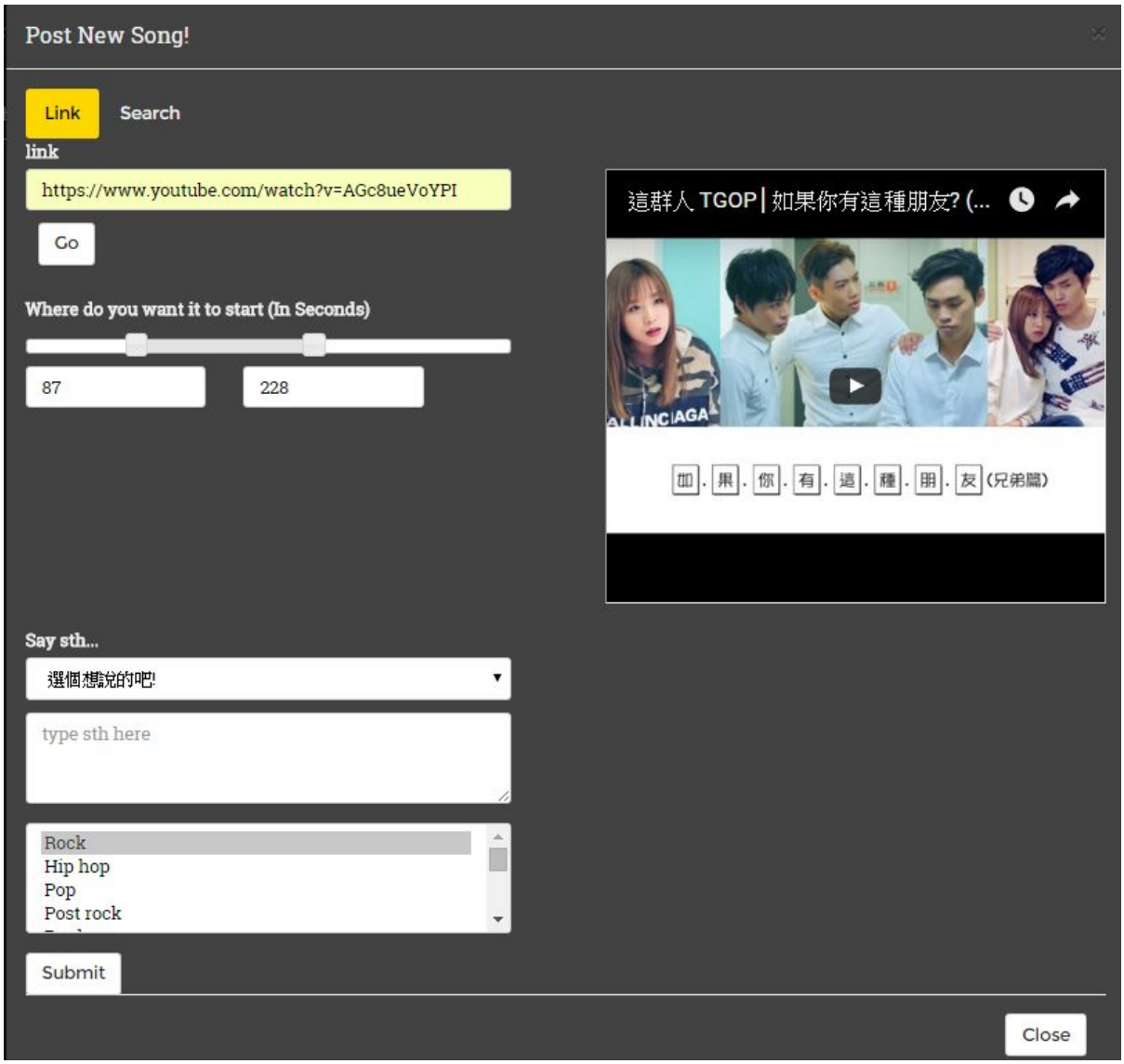
4.2 Welcome Page

This is the first page to be seen in our website. The purpose of a welcome page is for people to understand what this website is all about and how it works, and most importantly, make them want to register and become an user in our website.
I created the page with bootstrap. Except for the navigator and footer part which are extended from a base html just like all the other pages do, the main part of the page is a header and multiple sections.
In header, there are a title, an brief introduction in a sentence, a register button which lead to the registration page, and a login button which lead to the login page.
The following sections are a brief introduction of what you can do in the website, a brief tutorial of how you can perform acts to use our website, and an introduction of the team.



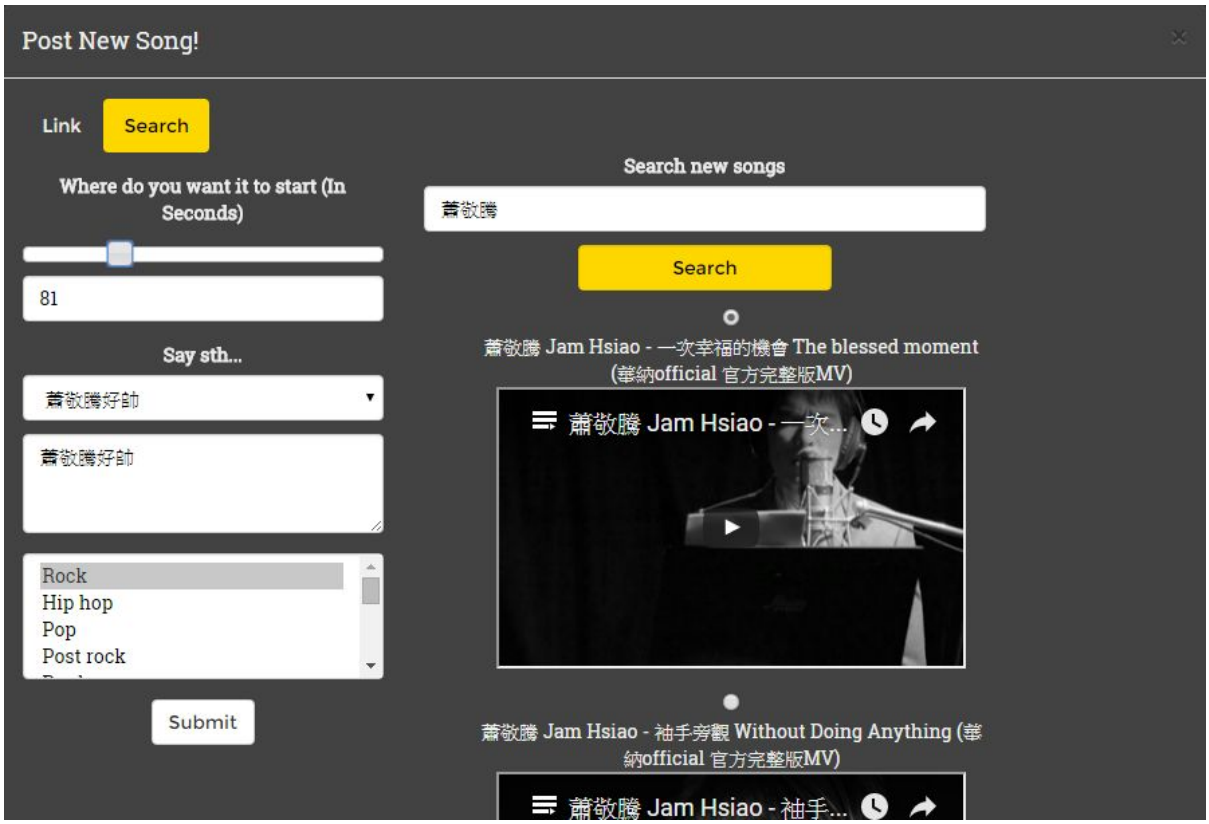
4.3 Posting Threads - The POST Modal

Posting a message is an essential function for a social network. However,how it works makes an influence on user's willing to use this app.So base on design thinking , we improved the function and efficient continuously.The post function separates into two parts , Link and Search .



In function Link , we provide link pasting to get the specific song that user would like to share. After user paste a link ,we use a regular expression to get the video ID in the URL and then save it in the database. Besides , we also get the title of the video by YouTube api implement. There is a customized function that user is able to choose the start time (end time) in the video by sliding the slider . That is , the posted videos are not only the videos in YouTube but also videos containing provider's thought. In order to help user with adding message easier ,we provide several default message options so that user don't have to type some words like "哈哈 這好好笑" "看了好想哭" .

Finally, user can choose a category that the video belonging to. Base on this sorting, YouBox would run an algorithm to do recommendation, also user can find the songs in certain category easily.



The other function Search gives a different way for user to find the video. After vedios have been load, choose the video and YouBox would get it's title and video ID.The rest of steps are the same as the steps in Link function.
In this part ,we use regular expression to get rid of some special characters in order to avoid the bug that happening in the playlist page

4.4 Navigator

The navigator is implemented with the <nav> tag to define a set of navigation links. Then we used the class, "navbar", in bootstrap to enhance the appearance of the Navigator, and assign the relative html it needs to forward to.

The Hi username! link orients user to the Account Setting page. Currently only profile image is editable, and users are limited to the six default images. Changing

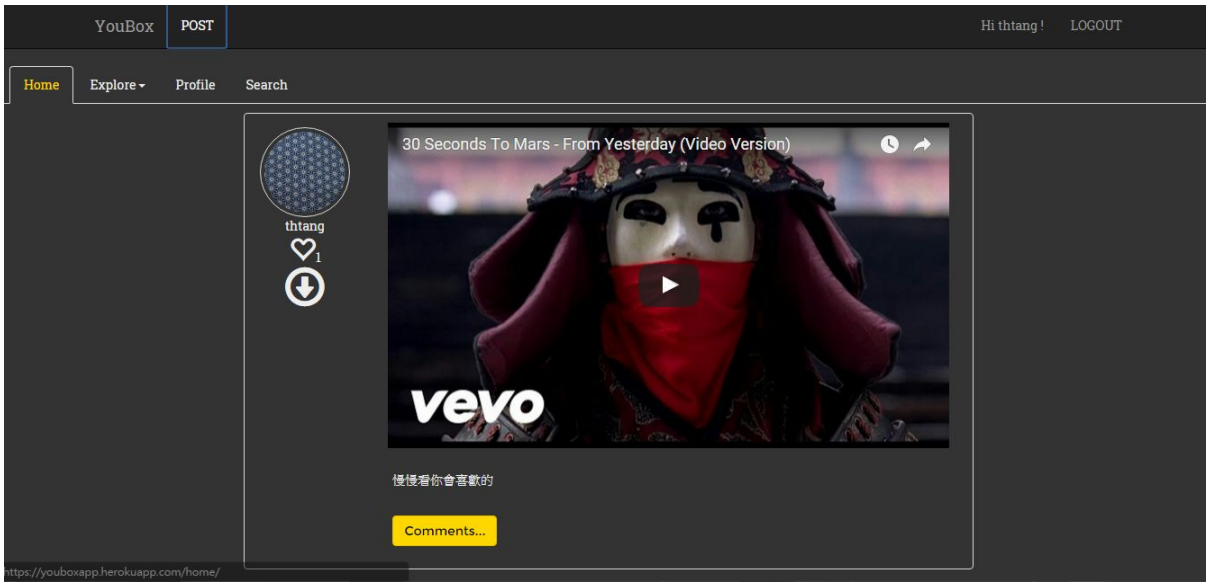
password, email, upload an image and other settings, are left as future work.

4.5 The Base Template

The base template contains the navigator and the post method. It defines the base layout and the main appearance of our website. With the base template, other html documents could simply extend from it and concentrate on designing the functions of each individual web page without worrying the unity of the format.

5 Home Tab Development

The Home Tab mainly do one single thing: render posts. It is easier to implement by hardcoding the DOM in HTML. However for reusability, we may need JavaScript function for encapsulation. Consider the explore user page and profile page, they need to render posts as well. Also, unifying the style benefits UX. Thus we start on building a Render Post API which is recallable elsewhere in other pages.



5.1 Constructing A Post Thread

The body of a single post uses the bootstrap grid system with layers of rows and columns. It is splitted into three part: (1) the profile block displaying user profile

photo, like and download. (2) video block for youtube iframe player. (3) post contents and the comment accordin collapse. (1) and (2) are rather simple, however (3) we will need extra consideration. The comment part would need additional care, since comments will increase the post height rapidly and make the layout out of control. Current solution is to use collapse accordin. The button “Comments...” will serve as a trigger to toggle the collapsed part. The shortage is user can’t see any comments until he/she pressed the button, one even doesn’t know if there’s any comment inside! The best answer perhaps is to show (at most)2 latest comments at first, and give a “more comments” link. As the link pressed, pop out all the comments. This requires a robust mechanism to handle each user action, so the order and number of the comments won’t mess up. An compromised workaround is to give a number indicating number of comment, so when no comment one saves his/her time to check it out. This is left as future work.

5.2 API Consideration

The final function for use should do every effort to eliminate dependency, so other developers can use it without extra settings. And this is important, because usually they hate reading more documents, and it makes things easier and better maintainability, thus less bug.
The first required argument is an id. We use jQuery to build the whole structure. Bootstrap grid system requires a container at the top level layer. Considering the scenario that a developer now wants to render out posts. He/she should specify where to put the posts in, so this is the first argument to pass. We have many choices here, we can simply let the developer pass in the id of an arbitrary DOM, and posts are filled in. Remember there’s a dependency that the top layer should be a container class object. To follow the bootstrap rules, we acquire developers to put a in their HTML, and give an id like post_container. Posts will be appended in the container. However, the size of the rendered post depends on the context code, that is, where the post_container resides. If post_container is an immediate child of <body> then one may expect the same layout as Home tab. Otherwise, the rendered post will try to scale down according to the parent of post_container. Generally, developer should handle this, and using bootstrap grid system to contain post_container is recommended.
The second input should be a filter, telling the function the desired posts to be taken out. This may raise some security issues though, since we haven’t check authentication of the ajax request yet. In future work, we may let user choose to limit users that can see his/her post, and authentication of render_post would need to be added.

5.3 The Render Post API

Call the function as

render_post(Container ID, Filter Object);

Dependency is listed below.

- Place a container where you want to the posts to be.
- The post uses youtube iframe API, so no extra Youtube API loading setting in Javascript is required. While the home tab template we call render_post in the function onYouTubeIframeAPIReady(), this is not a prerequisite.

Details on the arguments below.

- HTML Container ID: you can simply pass in “post_container”, just make sure it is the id of the container div.
- Filter object: giving conditions to retrieve specific posts. Particularly, we use field name same as the database. See the following examples.

We want to retrieve posts made by user “iriver” and belong to category “pop”, and we only want 2 posts. Then declare the filter objects as below:

----------Example----------

```
var filter = {
   "username":"iriver",
   "limit":2,
   "category":"pop",
};
...
render_post("post_container", filter);
```

---------Example end----------

Notice that if you don't specify the “limit” key, default will be 1. If the matched result is less than 2, it will show them as much as possible. As best practice, you should always determine the number of posts you desire to render.

5.4 Advanced Options of Filter Object

- Say you want to retrieve posts of several categories at once, so the condition is “pop OR Japanese”. Construct the filter as below.

```
var filter  =  {
   “or”:{
      “category” : ["pop", "Japanese"],
   }
```

```
}
```

Generally, if your condition is an OR operation, specify the field in the “or” key. Otherwise it’s an AND operation, then specify it at the first level. However there’s one exception for user name. Please read the following.

- Notice that user name needs to be handled explicitly. If you want to OR several users, this is done outside of the “or” key.

```
var filter  =  {
    "username":["user1", "user2", "user3"],
}
```

- To sort the result, say, posts with higher like count rendered first(upward),

```
var filter = {
    "sort_by" = "-like"
}
```

To sort as increasing order,

```
var filter = {
    "sort_by" = "like"
}
```

- Greater/less than condition to set value constraints. Below is an example that retrieves posts of like count greater than 2.

```
var filter = {
    "likes__gt": 2,
}
```

Use double underscore and the suffix to specify more value constraints:

Suffix	Expression	Constraints
gt	>	greater than
lt	<	less than
gte	>=	greater or equal to
lte	<=	less or equal to

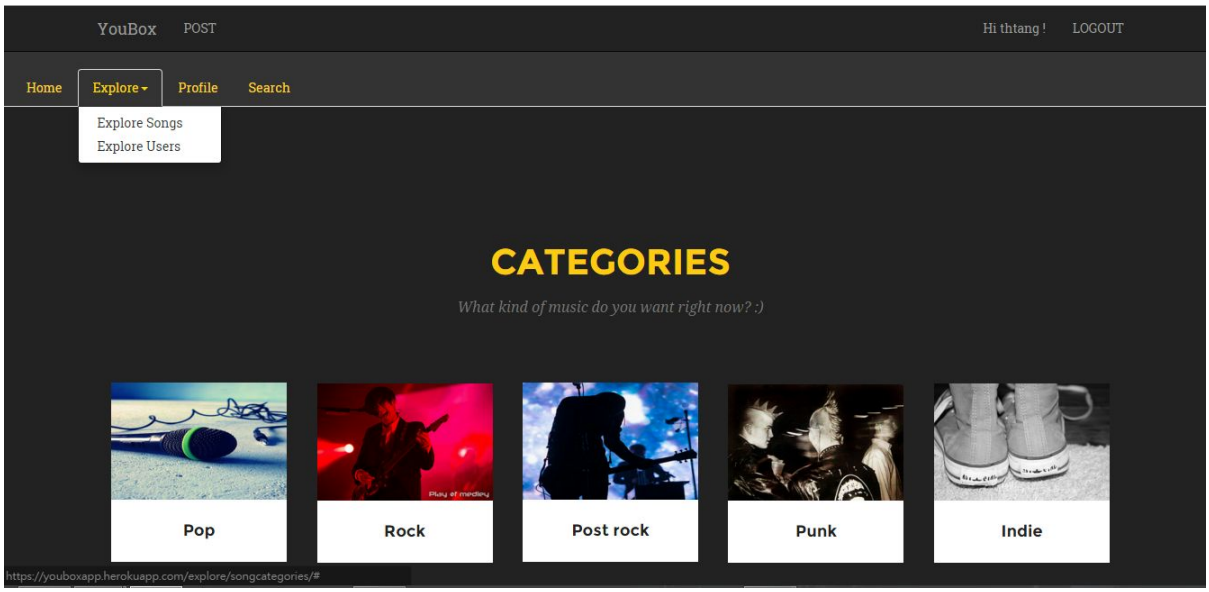
- Retrieve posts that are liked/not liked by specific user as the below code. Currently you can only specify one single user. Specifying multiple users’ liked posts is left as future work.

```
var filter = {
   "limit": 5,
```

```
    "is_like" : {
        "username": "sandy",
        "value" : "true",
    },
};
```

6 Explore Tab Development

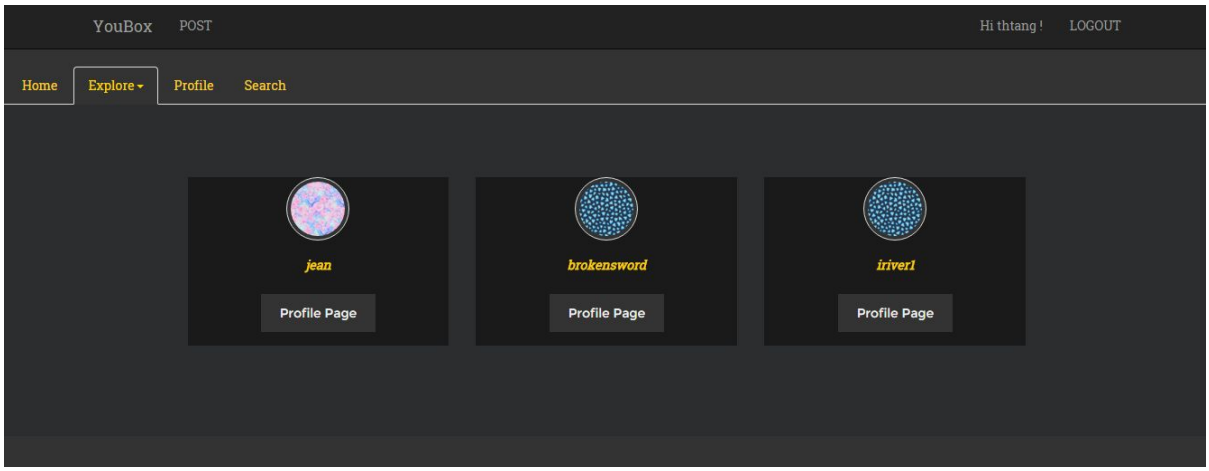
Users can explore popular songs shared by all users, not just the ones followed, organized in categories.
Also, users can explore the users not yet followed by themselves by our recommendation. So they can follow those who have similar music taste with them.



6.1 Category and Explore User Page

For the explore-songs-by-category page, all the categories are listed as blocks. We categorize songs by three separate features, music type, language, and the mood of the song. There is a column in database called category, it will store whatever category user selected when posting in the format of a string.
When you click on one of the blocks, it will redirect to the web page that listed the popular songs, sorted by like counts, of that category. I implemented this part by, first, create a html file shared by all categories' page, and, second, filter all the posts of a specific category and render them as a list to the html file, and, finally, use Django template tag to sort the list of posts by a feature, like count, then display the posts in the style of the ones in homepage.

As for the explore-users page, there are users that haven't been followed by you and are recommended by our recommendation system.
Each user is displayed as a block with its username, user picture, its five most popular posts' title, and a button that redirects to its profile page to check out all of its posts and to follow that user if you want. I implemented this part by, first, get the data of all the users except for the user itself and the ones already followed then render it to the html file of explore-user page, then, use recommendation system to get the order of the users to display, finally, display the users with Django template tags.



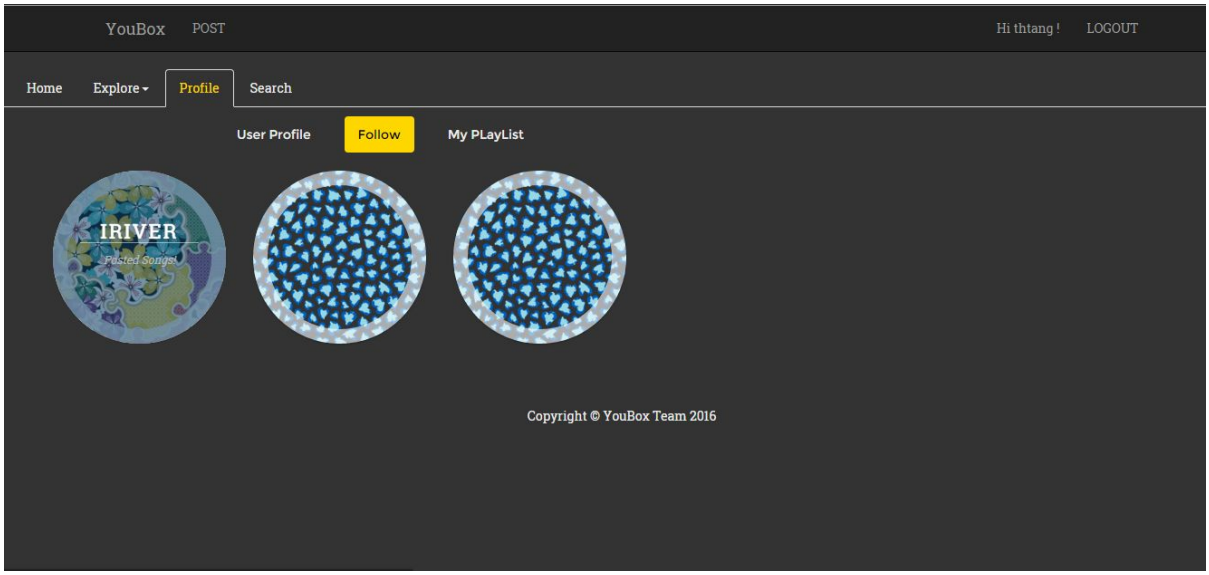
6.2 The recommendation System

The algorithm is based on calculating the similarity between two user. Here we define two users as similar if they have the same taste in music. Depending of the category of the music that each user has posted, we create a list of array, where the number of posts for the category represents the value of one element in the array. After obtaining a matrix of the users' taste for each category of music, we take the target user's row in the matrix and calculate its cosine similarity with the other rows in the matrix. Then we could rank the obtained value in a decreasing manner and get the list of users that are similar to you accordingly. The users that you have already followed would be taken out of the list and the rest would be displayed on the screen.

7 Profile Tab Development

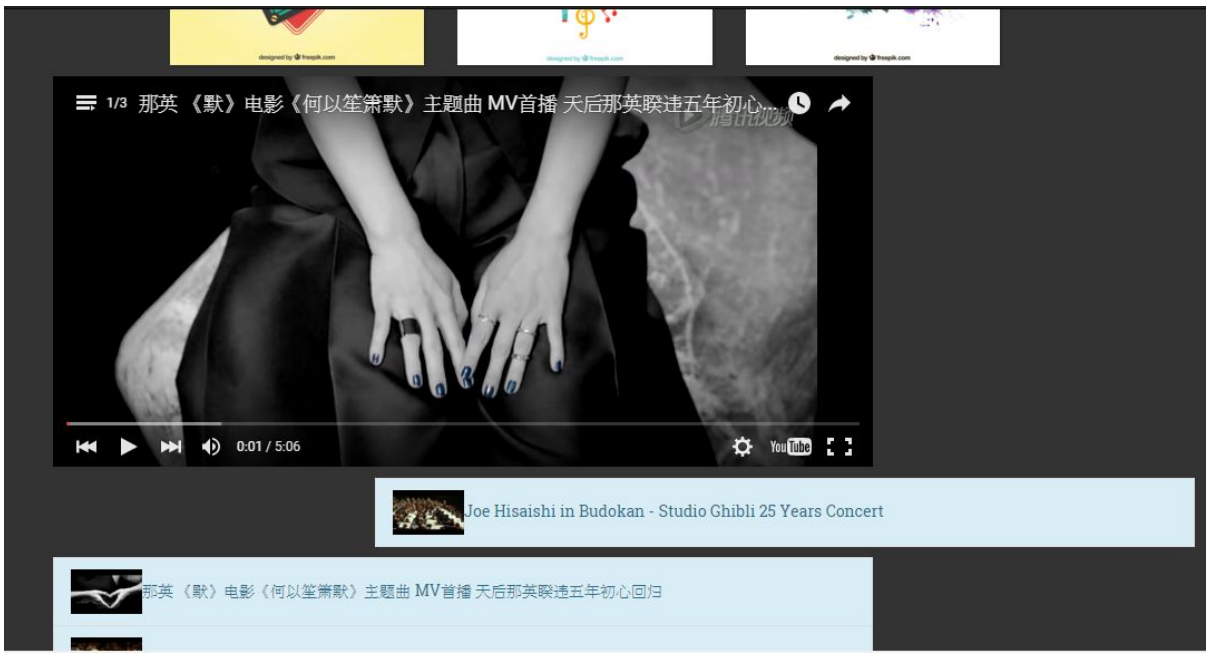
The Profile Tab targets on user profile management, the user will first see what he/she has shared and then access the Follow page in order to check or link to the

the user that has been followed.The amazing page, My Playlist, offers the user a good experience for handling his/her own playlist.



7.1 The Wall, Status and Follow

In order to make Post management easier , profile page shows the Post wall for user to reply comments without taking time to find his/her own post in Home page.The Follow page gives links for followed other user ,therefore user can access to other profile easily.We used render_post function to get the filtered data from database and show it.



7.2 Playlist

In the page, ”My Playlist”, different playlists would be provided. The playlists are divided into three types for now. The first one - “My Playlist” creates a playlist that contains the music that the user have ever posted. The second one - ”Favorites” includes music that the use have liked before, and for the last one- “Customized Playlist” lets the user add any songs in any order in a whole new playlist.

Considering the fact that users would like to listen to songs that he/she posted and have recently liked, we came up with the idea of creating playlists for users to enjoy music without having to click repetitively whenever a song has stopped. With playlists created automatically for them, the users can relax and listen to the music they love. Also, we have the feature of adjusting the sequence of the music to be played by simply dragging and dropping the elements in the playlist. This function could let users to decide the order of the songs anytime they want. Different from recent music applications, this drag and drop feature is much more simply to operate and provides the users more flexibility when creating playlists.

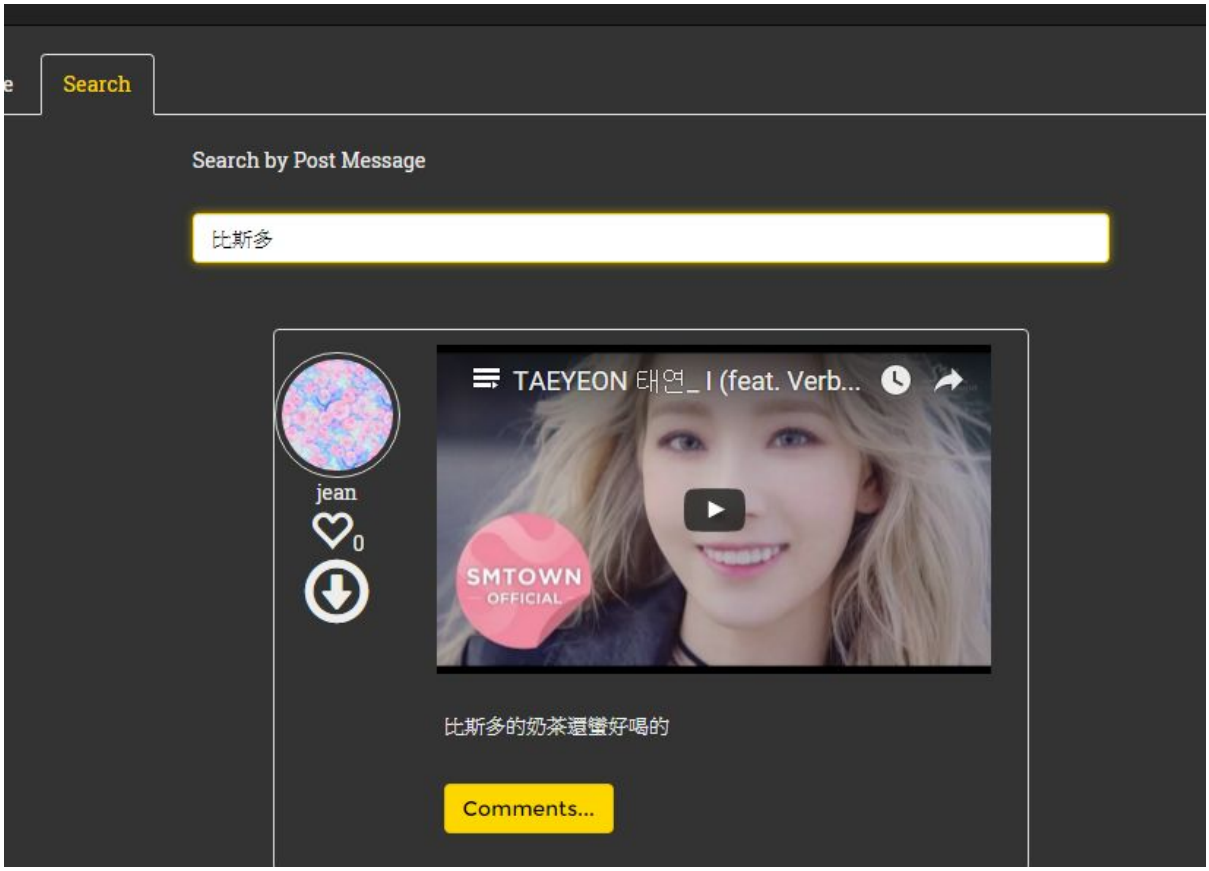
The “Customized Playlist” is provided for those that don’t like the default playlist that we have created. We give the users more choices and let them choose the songs that they want to contain in their playlists.

We encountered some difficulties while implementing the drag and drop function. The sequence of the songs after the drag and drop is operated are all messed up and couldn’t be updated. Even if the list have been updated, the sequence shown in the playlist is still out of order. Also, because we have three kinds of playlists to

implement, but only have one function to control the creation of the playlist, the songs shown in the playlist would sometimes refer to different list of songs.

To deal with the above difficulties, we used a selection variable to control the playlist that is selected and assign the relative videolist to the playlist accordingly. Then the order of the songs in the list is updated each time the elements are dragged and dropped. Then we obtain the right song in the right order referring to the id of the element. Also, to deal with the problem of the playlist crashing down or disappearing after two or more clicks continuously, we destroy the playlist each time the order is changed or when another playlist is chosen .

8 Search Tab Development



The search tab aims to serve for our database instead of YouTube's. Currently we implemented post search based on post message. No authentication applied so you may search out any user's post which may raise security concerns. Another feature that may be useful is a user ID search. Just like facebook, when you want to add a specific person as friend, you will need to search for him first. This is left as future

work. Current workaround will be accessing the url “/profile/username/1” to reach one’s profile.

8.1 Implementing Search

We have several choices here. (1) Use google search service and simply embed the search bar, (2) Use Regular Expression to implement from scratch, (3) Use a search engine. (4) Use SQL command provided by database language. The easiest way is (1), but it may not fit our needs because the google crawler should be based on hyperlink, and thus traverse a website tree. However, in our website user posts are not retrieved by hyperlink. Instead, post information is gained by ajax request or passed to HTML template by Django View. Therefore the google crawler may not be able to access them.
While (4) is also a good option, functionality is dependent to specific database language. Our local database is SQLite, while on Heroku we use PostgreSQL. Obviously the best option is (3) and use existing open-source’s work. However we soon found this a field that requires a lot of research. People suggest using a search manager for Django, the Haystack, and then apply a search engine such as Solr and Elasticsearch, each of which are pretty new to us. As we inspect closer, Solr requires additional dedicated server, so does other mainstream search engines. This makes thing more complicated, because ultimately the project should be deployed on Heroku. Heroku provides search engine add-on, and may eliminate compatibility issues. However, the functionality will be limited on Heroku and we cannot test the search engine in local. Also the documentation is rather simple. Where to place the required code is not mentioned and many other details skipped. Therefore, we dropped the attempt to find a mainstream search engine, yet this is left as future work.
We ended up with a lightweight search engine “Django watson”, distributed as a Django App, and can be easily added into an existing Django project. Installation is covered in the “Project Dependency” section. It provides full-text search, which means it can search a keyword across fields of a table.

8.2 Evaluation

The performance and accuracy of watson is great for english text. Search for “could” will give the result “couldn’t”. But search for “ed” won’t match a post containing the word “loved”, neither will “est” yield “best”. Still, “bes” will match “best”, so we can see the search engine matches “prefix” of words.

However search of chinese text gives few results. Given the post message “比斯多的奶茶還蠻好喝的”, searching for “比”, “比斯多” and “比斯多的奶茶蠻” matched, but “斯多”, “的” did not. Therefore the guess that it uses a prefix approach is confirmed. The problem is the search engine simply recognizes a space as delimiter among words, yet in Chinese this rule does not follow. The necessity for changing a search engine supporting Asia words is then raised, and we leave it as future work.
Another shortage of current search engine, yet meanwhile its strength, is the nature of full-text search. Sometimes one only wants to search by post message, or search by category. Currently, the text base of search is mixed with post message, video title, category and video url. Keywords would be compared with these fields. An option for user to choose a field to search may be preferable, and again is left as future work.
Notice that one cannot search by username, nor like count or start time. To view posts of a specific user, simply access his profile page. As mentioned previously, user ID search is left as future work.
Other shortage of current search engine includes limited scope of search. Currently we can only choose one table for indexing. For multiple indexing, it requires extra research on the watson usage, or applying other search engine, and thus left as future work.

9 Future Work

While our website already contains the essential function as a social network, we still have to make an effort on its efficiency and fluency. However, the free program of the server , Heroku , has its own limits , in that some advanced function couldn’t develop easily and deploy on Heroku. To deal with that problem , we tend to purchase a premium plan to take responsibility on large amount of API calls and also enhance the site speed.
Recently,we have received some feedback of our website. First of all, a random recommendation system is required to make the website more attractive. YouTube implements some algorithms to recommend videos, but what we want to do is to build a purely random recommend. Users may find some interesting videos that they would never watch by searching via keywords in their mind. Secondly , it’s better if we simplify the category sorting because the more operations users have to do, the less fond of user experience YouBox would get.

10 Developers

Shaung Cheng

Department of Computer Science, NTHU 100062233
Responsible for integration and deployment of the project. LESS CSS management, unifying website design. Home tab, post render API. Account Setting page. Search tab and search engine, 28%

Tsun Hsien Tang

Department of Industrial Engineering and Engineering Management , NTHU 102034038
Implements profile tab, playlist which includes cutting-edge functionality such as drag-and-drop and customized playlist. Offer different approaches for post like pasting a link or search music from youtube directly in POST modal, 24%

Jean Lee

Department of Computer Science, NTHU 102062305
Django authentication backend including login and registration. Follow, like feature, POST modal, and navigator. Recommendation algorithm applying cosine similarity approach. Linking profile tabs and the control between different playlists, 24%

Yi Ching Lu

Department of Computer Science, NTHU
102062127
Came up with the concept and purpose of this website.
Responsible for Explore Tab, which includes the explore-songs-by-category page and the explore-users page. Build category database.
Theme design of the entire website. Welcome page implementation, 24%

11 Reference

11.1 Library and API Documents

- Get Start with Django, girls django, https://djangogirlstaipei.gitbooks.io/django-girls-taipei-tutorial/content/index.html
- A UI drawer, good tools for defining the prototype http://webdesignledger.com/13-super-useful-ui-wireframe-tools
- A online play ground, you can add custom CSS, JS by url here to try out new components provided by bootstrap or others. https://jsfiddle.net
- Icon library provided by bootstrap. Glyphicon. http://www.w3schools.com/bootstrap/bootstrap_ref_comp_glyphs.asp
- Icon library using Font-Awesome. http://fortawesome.github.io/Font-Awesome/icon/heart-o/
- Python Regular Expression, https://docs.python.org/2/library/re.html
- W3C Bootstrap, http://www.w3schools.com/bootstrap/default.asp
- Search Engine django watson, https://github.com/etianen/django-watson/wiki/registering-models
- Django QuerySet, the F object, https://docs.djangoproject.com/en/1.7/ref/models/queries/
- Youtibe IFrame API, https://developers.google.com/youtube/player_parameters
- Bootstrap Live Customizer, http://bootstrap-live-customizer.com/
- Bootstrap Components List out, https://kkbruce.tw/bs3/Components

11.2 Issue Solution

- How to align the content within a column of bootstrap grid system? Use a ".text-center" in the parent object, say, the .row object
- limiting query result amount, https://docs.djangoproject.com/en/dev/topics/db/queries/#limiting-querysets
- Using OR operation in QuerySet, https://docs.djangoproject.com/en/dev/topics/db/queries/#complex-lookups-with-q-objects

- Deploying to Heroku, https://djangogirlstaipei.gitbooks.io/django-girls-taipei-tutorial/content/django/deploy.html
- Heroku Collaborate, https://devcenter.heroku.com/articles/sharing
- Displaying admin page, https://djangogirlstaipei.gitbooks.io/django-girls-taipei-tutorial/content/django/admin.html
- Customizing View of /admin, https://docs.djangoproject.com/en/1.9/ref/contrib/admin/, search for "list_display"
- Check if a user is logged in, http://stackoverflow.com/questions/12615154/how-to-get-the-currently-logged-in-users-user-id-in-django
- How to use JOIN in django, http://stackoverflow.com/questions/28510108/django-table-join
- How to deal with _id suffix on Foreign Key field, http://stackoverflow.com/questions/8223519/preventing-django-from-appending-id-to-a-foreign-key-field, http://stackoverflow.com/questions/13116130/django-suffix-foreignkey-field-with-id,
- Fix shitty migrations, http://stackoverflow.com/questions/23755523/how-to-reset-migrations-in-django-1-7
- Compiling LESS, install Node.js as a prerequisite, https://pypi.python.org/pypi/nodeenv
- Addning submodule, http://brianflove.com/2014/08/21/bootstrap-git-submodule-workflow/
- Installing SciPy with buildpacks, http://stackoverflow.com/questions/24565799/deploying-to-heroku-with-anaconda, official conda buildpack: https://elements.heroku.com/buildpacks/conda/conda-buildpack
- Heroku: solution to a H10 Application error on heroku server, HunterMeyer's answer worked, http://stackoverflow.com/questions/13496827/heroku-deployment-error-h10-app-crashed
- Heroku: run bash console by heroku run bash

- Heroku: view db on heroku, http://stackoverflow.com/questions/15953390/how-to-view-current-database-schema-for-heroku-app-in-terminal
- Heroku: db command on heroku, http://stackoverflow.com/questions/14126171/how-to-drop-a-single-postgres-table-in-a-ruby-on-rails-heroku-app
- Watson install error: when reporting search_tsv has NULL values after install watson, clean(truncate) the table watson_search..., then do again and it might work.
- Heroku db backup, https://devcenter.heroku.com/articles/heroku-postgres-backups