Report

For COMP1002 Group Project Group 88
LIU Yi 20074863d
SHU Kunxin 20081657D
WANG Xiangzhi 19082878d

Problem description

In this project, we are requested to make a system that can maintain the information about user and the articles. Besides, the project should have a functioning friend system and article post system. After this main function, we should implement some functions in our system to read the data and store in the memory, for example, find the anchor and the quote of one article. At last, we also need to identify KOL or to identity whether friends tend to quote articles posted by other. Above are the all problems we have, then we will report the solution.

Data abstraction

Data structure for store single Users and Articles:

Dynamic class: User & Articles

We use Dynamic class so that we can easily read and load the data in Archive and access the information of them.

Data structure for Store all users and articles:

Dictionary: dataPool.UserDict, dataPool.ArticleDict
List: dataPool ArticleIDs

For dict: Keys are the ids and the values are the User or Article Object. So that we can access the object through the IDs, and some information of them can easily stored by IDs.
For list: This is for View Area(explained below), Add a list to make it easier to index, so it's better to let the user select the article to view it.

Data Structure for KOLs:

List: dataPool.KOLs

We already have UserDict above, so we only need a list to store the ids of the KOLs' Ids after calculating the KOLs.

Data Structure for T, P, AdiminPassword：

Single Variable: dataPool.T, dataPool.K, dataPool.AdminPassword
They are themselves a single piece of data, so you only need to store them in a single variable and get them when you need them.

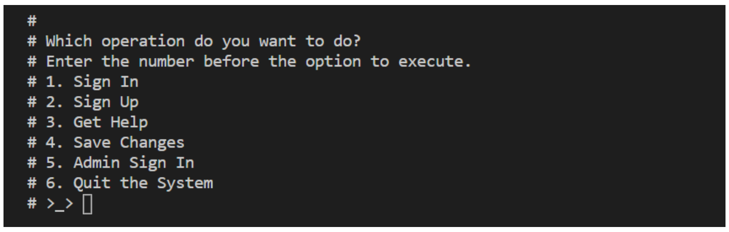
Implementation

Main Structure(module design)

We used **MVC(Model-View-Controller) Pattern** to Design our System.

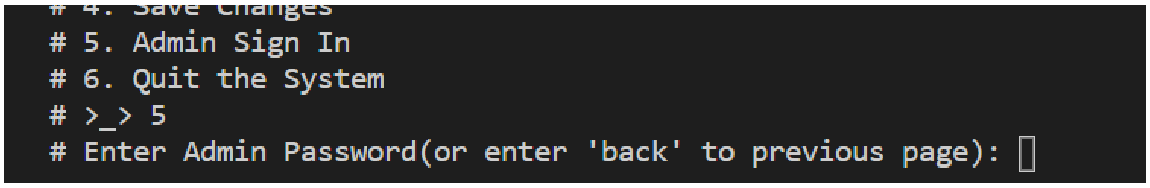
In the **View Part**, all code is used to interact with the user, including Login page, information display page, add new friends, publish new articles and so on. Almost all the interaction is done by a function called Diague() and a function called tranMess() located in the InPutControll class.

Diague has 2 arguments, one called title, which shows the effect of current dialog, so that the user will easy to know what are the system doing and what does the user can do. Another is called ops which means options, it is a list, all options can be supplied to user this time contained, the screen short cut is a typical Diague which is in the main page called by main().



```
#
# Which operation do you want to do?
# Enter the number before the option to execute.
# 1. Sign In
# 2. Sign Up
# 3. Get Help
# 4. Save Changes
# 5. Admin Sign In
# 6. Quit the System
# >_> 
```

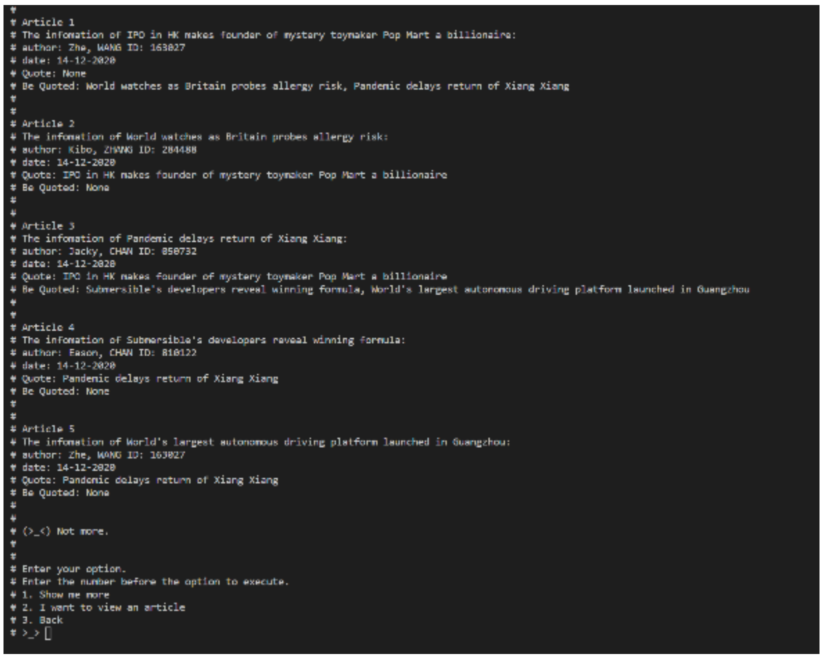
The user only needs to enter the number ahead the Operation, the system will jump to corresponding pages to operate instead of entering long and boring Key words to tell the System what you want do which is very easy to enter the wrong words causing a negative impact on the user experience. Even an illiterate child can operate the system easily. In addition, Diague keep receiving ‘back’ keyword, if the user enters ‘back’, the System will back to previous page. In fact, this is easy to handle, as long as System end the current dialog loop. The ‘back’ function is not in 1.0 version, we added it because we find it is impossible to end the operation if we chose but want to go back on it. Only when the operation competed, we can go back. We found this annoying, so we added the back function. An example of ‘back’ function is shown below.



```
# 4. Save Changes
# 5. Admin Sign In
# 6. Quit the System
# >_> 5
# Enter Admin Password(or enter 'back' to previous page): 
```

The Diague() was inspired by the dialog box I carried in the Swing package when I was writing the Java GUI. Java GUI dialogs can support editing options to edit the display, so at the time of design, I was also wondering if I could design a dialog on the **command line(CLI)** that would return the index of the option and be highly customized.

The **View Area**. The view area provides users with the opportunity to view other people's articles and their own articles. This function is similar to the browsing area of some social software, so that the function of the whole social system can be well reflected.

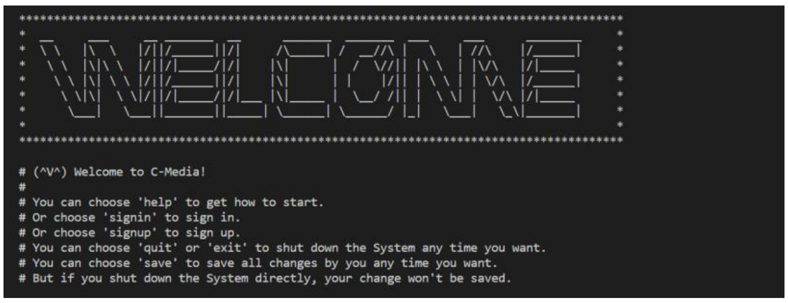


```
#
# Article 1
# The infomation of IPO in HK makes founder of mystery toymaker Pop Mart a billionaire:
# author: Zhe, WANG ID: 163027
# date: 14-12-2020
# Quote: None
# Be Quoted: World watches as Britain probes allergy risk, Pandemic delays return of Xiang Xiang
#
#
# Article 2
# The infomation of World watches as Britain probes allergy risk:
# author: Kibo, ZHANG ID: 204480
# date: 14-12-2020
# Quote: IPO in HK makes founder of mystery toymaker Pop Mart a billionaire
# Be Quoted: None
#
#
# Article 3
# The infomation of Pandemic delays return of Xiang Xiang:
# author: Jacky, CHAN ID: 050732
# date: 14-12-2020
# Quote: IPO in HK makes founder of mystery toymaker Pop Mart a billionaire
# Be Quoted: Submersible's developers reveal winning formula, World's largest autonomous driving platform launched in Guangzhou
#
#
# Article 4
# The infomation of Submersible's developers reveal winning formula:
# author: Eason, CHAN ID: 010122
# date: 14-12-2020
# Quote: Pandemic delays return of Xiang Xiang
# Be Quoted: None
#
#
# Article 5
# The infomation of World's largest autonomous driving platform launched in Guangzhou:
# author: Zhe, WANG ID: 163027
# date: 14-12-2020
# Quote: Pandemic delays return of Xiang Xiang
# Be Quoted: None
#
#
# (>_<) Not more.
#
#
# Enter your option.
# Enter the number before the option to execute.
# 1. Show me more
# 2. I want to view an article
# 3. Back
# >_> 
```

User can view the information of articles in View Area, also, user can view the detail of the article.

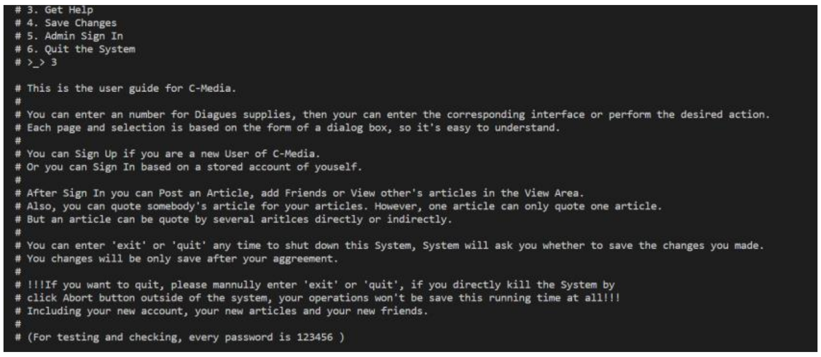
There are a few other **small elements** worth mentioning.

The welcome Page.



```
# (^V^) Welcome to C-Media!
#
# You can choose 'help' to get how to start.
# Or choose 'signin' to sign in.
# Or choose 'signup' to sign up.
# You can choose 'quit' or 'exit' to shut down the System any time you want.
# You can choose 'save' to save all changes by you any time you want.
# But if you shut down the System directly, your change won't be saved.
```

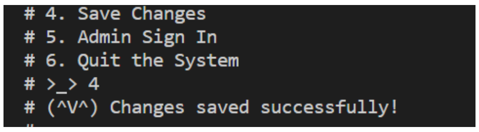
Welcome's artistic font design was done on our own, without using web resources. In addition, Some important tips and instructions are also included on the welcome page. More details of User Guide are shown in help page:



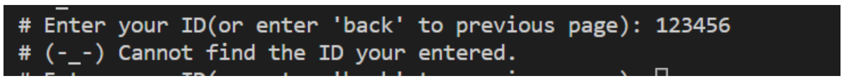
```
# 3. Get Help
# 4. Save Changes
# 5. Admin Sign In
# 6. Quit the System
# >_> 3

# This is the user guide for C-Media.
#
# You can enter an number for Diagues supplies, then your can enter the corresponding interface or perform the desired action.
# Each page and selection is based on the form of a dialog box, so it's easy to understand.
#
# You can Sign Up if you are a new User of C-Media.
# Or you can Sign In based on a stored account of youself.
#
# After Sign In you can Post an Article, add Friends or View other's articles in the View Area.
# Also, you can quote somebody's article for your articles. However, one article can only quote one article.
# But an article can be quote by several aritlces directly or indirectly.
#
# You can enter 'exit' or 'quit' any time to shut down this System, System will ask you whether to save the changes you made.
# You changes will be only save after your aggreement.
#
# !!!If you want to quit, please mannully enter 'exit' or 'quit', if you directly kill the System by
# click Abort button outside of the system, your operations won't be save this running time at all!!!
# Including your new account, your new articles and your new friends.
#
# (For testing and checking, every password is 123456 )
```

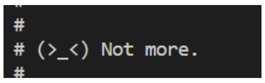
Hints that cover emoticons:



```
# 4. Save Changes
# 5. Admin Sign In
# 6. Quit the System
# >_> 4
# (^V^) Changes saved successfully!
```



```
# Enter your ID(or enter 'back' to previous page): 123456
# (-_-) Cannot find the ID your entered.
```



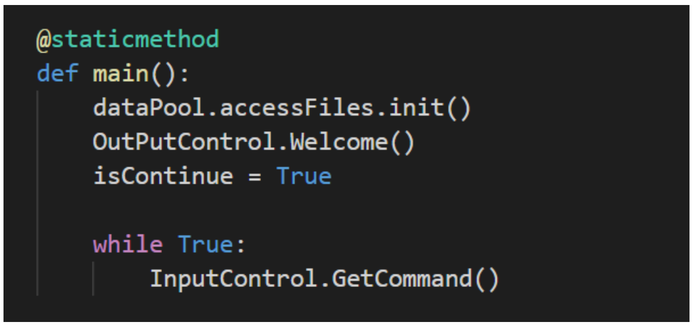
```
#
# (>_<) Not more.
#
```

A total of three expressions were presented in successful, illegal and unsuccessful situations.

The **view** part contains the need to display all the functions of the system, but because it is too much content and can be felt directly during the actual operation, there is no need to elaborate here.

In the **Controller Part**, main() is located. In fact, the Controller section simply starts the system off and provides the main menu interface in a circular form, i.e. The login and registration interface, which will only be displayed after the system is opened or the user logs out. Considering that the system is very interactive, the main pace is controlled by the View section.

The main() function is shown below.

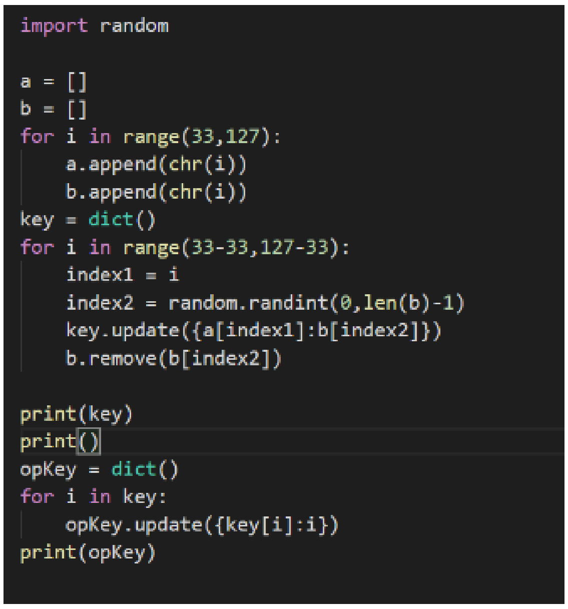


```
@staticmethod
def main():
    dataPool.accessFiles.init()
    OutPutControl.Welcome()
    isContinue = True

    while True:
        InputControl.GetCommand()
```

In the **Model Part**. Class dataPool is user for store the reading file information, we let the system once started, read all article and user information in the archive, This is because for convenience score (change of TA computer may not install python MySQL package or SQLite package, their installation and configuration in particular trouble) we use the TXT file instead of MySQL or SQLite database to store data, and find TXT packed in python data will be prone to bugs and inefficient, so we take the form of a removed. When the user opts out of the program, the function in charge of archiving will package all the important data in dataPool and overwrite the previously archived TXT file to ensure that the data is up to date. Model also provides encryption and decoding of characters(Generate the key by randomly matching the printable characters on the command line), as well as a random generator for the ids of articles and users Detail are in our python file's comments, the report will not fill up so much(5 pages max).

The code for random key generation is as follows:



```python
import random

a = []
b = []
for i in range(33,127):
    a.append(chr(i))
    b.append(chr(i))
key = dict()
for i in range(33-33,127-33):
    index1 = i
    index2 = random.randint(0,len(b)-1)
    key.update({a[index1]:b[index2]})
    b.remove(b[index2])

print(key)
print()
opKey = dict()
for i in key:
    opKey.update({key[i]:i})
print(opKey)
```

Requirement satisfaction

Query Functions:

We did not write the query functions because when it comes to User and Article, we used Object-Oriented Programming. So the query function seems not necessary. But code in our .py file fulfilled those query functions. So we write them below directly write them below based on our own data structures to prove we can do this.

For isFriend(X,Y): return Y.id in X.friendIDS

For isDirectSource(A,B): return B.quote == A.id

For isSource(A,B):

```
If B.quote == A.id: return True
while B.quote != A.id:
    B = dataPool.ArticleDict[B.quote]
    If B.quote == A.id: return True
return False
```

For Anchor(A):

```
K = []
```

```
        while A.quote != “”:
            K.append(dataPool. .ArticleDict[A.quote])
            A = dataPool. .ArticleDict[A.quote]
        return K
    For DirectReport(A): return A.beQuoted
    For Report(A):
        K = []
        For ID in dataPool. .ArticleDict:
            If isSource(A, dataPool. .ArticleDict[ID]):
                K.append(dataPool. .ArticleDict[ID])
        return K
```

For **Useful auxiliary functions:** we have the same case, and we has functions or statements which have the same effect. Also we have fulfilled setting of T, P , Administration management and selecting KOLs, details in codes and comments.