

Project4 Tweet Clone Part1

UFID: 83816537 Yi-Ming Chang

System Structure

1. Server Client communicatie by JSON message
2. Use Json file as a database
3. Use Actor message passing as an API and also Websocket
 - a. API
 - Client will call server API by passing the JSON string
 - Server will deserialize the message and we response according to the action.

```
let sendRequest: MessageType = {  
  action = "SUBSCRIBE"  
  data = Json.serializeEx JsonConfig inputData  
}
```

- b. Websocket
 - We'll broadcast the new tweet post to all the subscribers.
 - Simple mapping the actor path with the subscribers userId and push the message base on the paths.

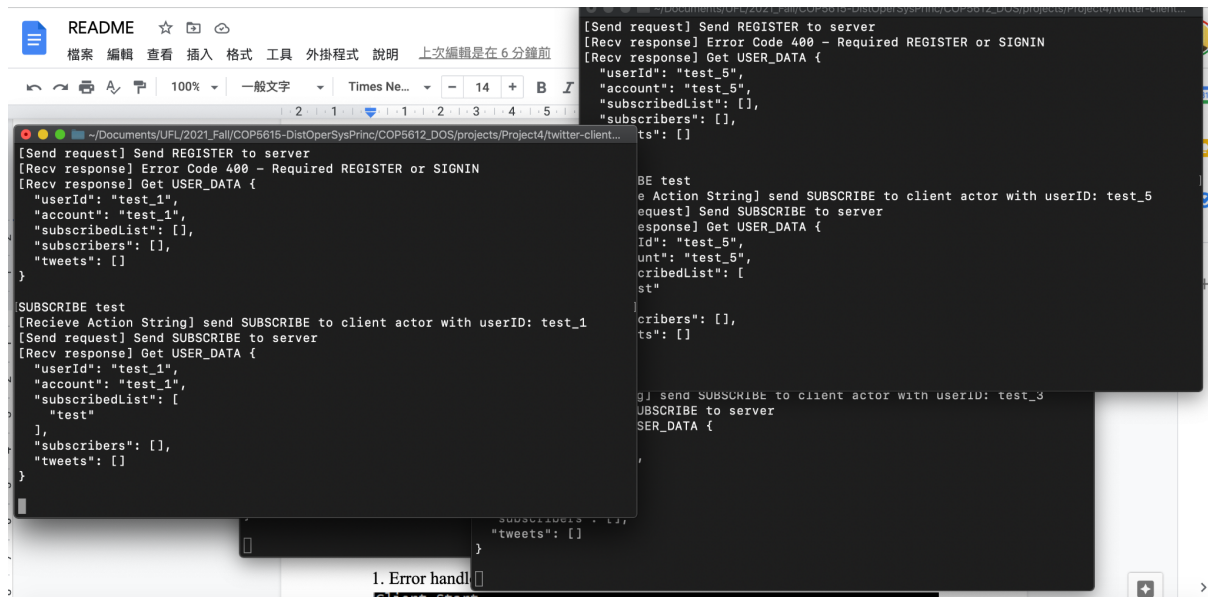
Down below is the Actions we made for Client

1. CONNECT {user_id}
 - a. Server will return user data and Client will store the data
 - b. Invalid user_id will be handled (ex. Error code 401 or 403)
 - Required REGISTER
2. REGISTER {account}
 - a. Return a unique user_id for user to CONNECT
3. SUBSCRIBE {user_id} {target_user_id}
 - a. User will be add to target user's subscribers
 - b. User subscribed list will add the target user

4. TWEET {user_id} {content}
 - a. Users will post the new content.
 - b. Server will broadcast the new tweet obj through the subscribers.

SYSTEM simulator

We use a shell script to trigger multiple client users. Our goal is to test all the functions and try to broadcast to as many users as we can.



```
[Send request] Send REGISTER to server
[Recv response] Error Code 400 - Required REGISTER or SIGNIN
[Recv response] Get USER_DATA {
  "userId": "test_5",
  "account": "test_5",
  "subscribedList": [],
  "subscribers": [],
  "tweets": []
}

[Send request] Send SUBSCRIBE to server
[Recv response] Error Code 400 - Required REGISTER or SIGNIN
[Recv response] Get USER_DATA {
  "userId": "test_1",
  "account": "test_1",
  "subscribedList": [],
  "subscribers": [],
  "tweets": []
}

SUBSCRIBE test
[Receive Action String] send SUBSCRIBE to client actor with userID: test_1
[Send request] Send SUBSCRIBE to server
[Recv response] Get USER_DATA {
  "userId": "test_1",
  "account": "test_1",
  "subscribedList": [
    "test"
  ],
  "subscribers": [],
  "tweets": []
}

[Send request] Send SUBSCRIBE to server
[Recv response] Error Code 400 - Required REGISTER or SIGNIN
[Recv response] Get USER_DATA {
  "userId": "test_5",
  "account": "test_5",
  "subscribedList": [],
  "subscribers": [],
  "tweets": []
}

[Send request] Send SUBSCRIBE to server
[Recv response] Error Code 400 - Required REGISTER or SIGNIN
[Recv response] Get USER_DATA {
  "userId": "test_3",
  "account": "test_3",
  "subscribedList": [],
  "subscribers": [],
  "tweets": []
}

[Send request] Send SUBSCRIBE to server
[Recv response] Error Code 400 - Required REGISTER or SIGNIN
[Recv response] Get USER_DATA {
  "userId": "test_3",
  "account": "test_3",
  "subscribedList": [],
  "subscribers": [],
  "tweets": []
}
```

- Each Client will subscribe to a specific user (userId = test)

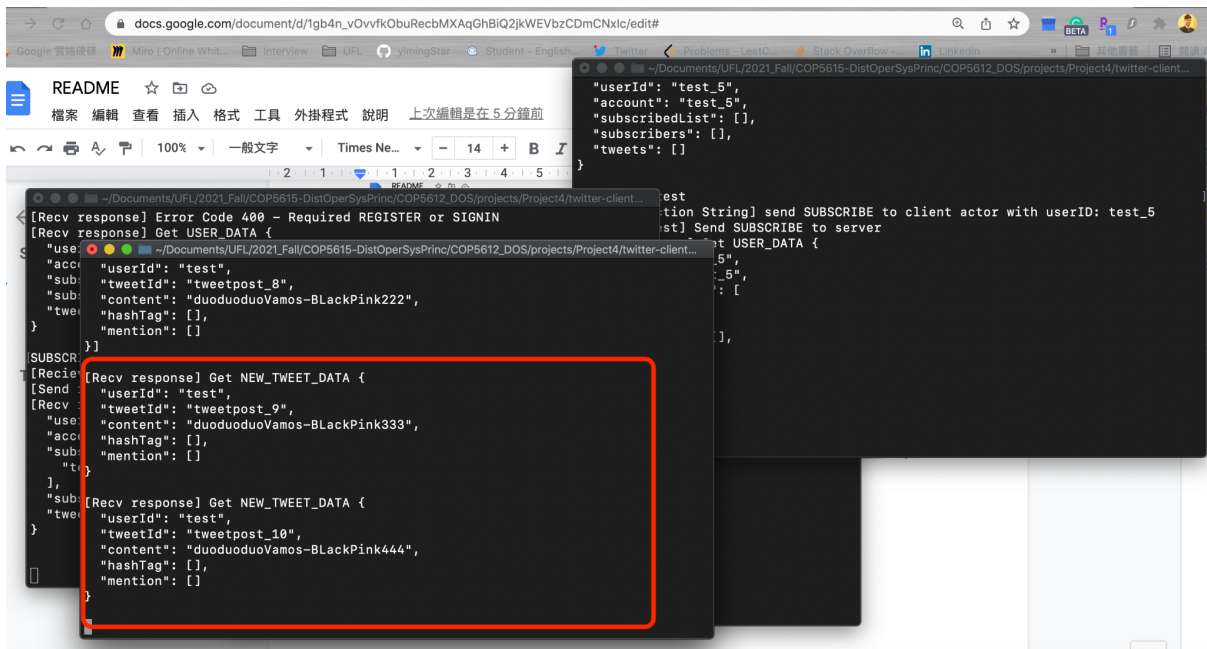
Let test user start to tweet

```

TWEET duoduoduoVamos-BlackPink444
[Recieve Action String] send TWEET to client actor with userID: test
[Send request] Send TWEET to server
[Recv response] Get USER_DATA {
  "userId": "test",
  "account": "testdeveloper",
  "subscribedList": [],
  "subscribers": [
    "tweetUser_1",
    "tweetUser_2",
    "tweetUser_3",
    "test_5",
    "test_3",
    "test_2",
    "test_1",
    "test_4"
  ],
  "tweets": [
    "tweetpost_10"
  ]
}

```

Check the multiple users did receive the new message.



CONNECT Demo

1. Error handle - missing {user_id}

```
Client Start
CONNECT
[Recieve Action String] send CONNECT to client actor with userID:
[Send request] Send CONNECT to server
[Recv response] Error Code 400 - Required REGISTER or SIGNIN
```

2. Connect with ID - get userData, ownTweetsList, and browseTweetsList

- browseTweetsList is the page for user to browse others tweets
- ownTweetsList will build users home page

```
CONNECT tweetUser_1
[Recieve Action String] send CONNECT to client actor with userID: tweetUser_1
[Send request] Send CONNECT to server
[Recv response] Get USER_DATA {
  "userId": "tweetUser_1",
  "account": "tommy999",
  "subscribedList": [
    "test"
  ],
  "subscribers": [],
  "tweets": [
    "tweetpost_1",
    "tweetpost_3",
    "tweetpost_6"
  ]
}
```

```
[Recv response] Get OWN_TWEET_DATA [{
  "userId": "tweetUser_1",
  "tweetId": "tweetpost_1",
  "content": "I am testing this system",
  "hashTag": [
    "#testingBoy",
    "#Just Do IT"
  ],
  "mention": []
}, {
  "userId": "tweetUser_1",
  "tweetId": "tweetpost_3",
  "content": "Go Vamost",
  "hashTag": [
    "#testingBoy",
    "#Just Do IT"
  ],
  "mention": []
}, {
  "userId": "tweetUser_1",
  "tweetId": "tweetpost_6",
  "content": "Duo Duo Duo",
  "hashTag": [
    "#BlackPink",
    "#Just Do IT"
  ],
  "mention": []
}]
```

```
[Recv response] Get BROWSE_TWEET_DATA [{
  "userId": "tweetUser_1",
  "tweetId": "tweetpost_1",
  "content": "I am testing this system",
  "hashTag": [
    "#testingBoy",
    "#Just Do IT"
  ],
  "mention": []
}, {
  "userId": "tweetUser_2",
  "tweetId": "tweetpost_2",
  "content": "I am testing this system",
  "hashTag": [
    "#testingBoy"
  ],
  "mention": []
}, {
  "userId": "tweetUser_1",
  "tweetId": "tweetpost_3",
  "content": "Go Vamost",
  "hashTag": [
    "#testingBoy",
    "#Just Do IT"
  ],
  "mention": []
}, {
  "userId": "tweetUser_2",
  "tweetId": "tweetpost_4",
  "content": "Doing my project on thanks giving @testdeveloper1",
  "hashTag": [
    "#ThanksGiving",
    "#DOS"
  ],
  "mention": [
    "@testdeveloper1"
  ]
}]
```

REGISTER Demo

```
REGISTER IamProgrammer
[Recieve Action String] send REGISTER to client actor with account: IamProgrammer
[Send request] Send REGISTER to server
[Recv response] Get USER_DATA {
  "userId": "tweetUser_4",
  "account": "IamProgrammer",
  "subscribedList": [],
  "subscribers": [],
  "tweets": []
}
```

SUBSCRIBE Demo

1. User tweetUser_4 has subscribed to tweetUser_1

```
SUBSCRIBE tweetUser_4 tweetUser_1
[Recieve Action String] send SUBSCRIBE to client actor with userID: tweetUser_4
[Send request] Send SUBSCRIBE to server
[Recv response] Get USER_DATA {
  "userId": "tweetUser_4",
  "account": "IamProgrammer",
  "subscribedList": [
    "tweetUser_1"
  ],
  "subscribers": [],
  "tweets": []
}
```

2. tweetUser_1 connected and found a new subscribers has popped up

```
CONNECT tweetUser_1
[Recieve Action String] send CONNECT to client actor with userID: tweetUser_1
[Send request] Send CONNECT to server
[Recv response] Get USER_DATA {
  "userId": "tweetUser_1",
  "account": "tommy999",
  "subscribedList": [],
  "subscribers": [
    "tweetUser_4"
  ],
  "tweets": [
    "tweetpost_1",
    "tweetpost_3",
    "tweetpost_6"
  ]
}
```

TWEET

1. Get user data with a empty tweets list

```
CONNECT test
[Recieve Action String] send CONNECT to client actor with userID: test
[Send request] Send CONNECT to server
[Recv response] Get USER_DATA {
  "userId": "test",
  "account": "testdeveloper",
  "subscribedList": [],
  "subscribers": [
    "tweetUser_1",
    "tweetUser_2",
    "tweetUser_3"
  ],
  "tweets": []
}
```

2. Tweet and renew the user tweet list

```
TWEET test duoduoduo-BlackPink
[Recieve Action String] send TWEET to client actor with userID: test
[Send request] Send TWEET to server
[Recv response] Get USER_DATA {
  "userId": "test",
  "account": "testdeveloper",
  "subscribedList": [],
  "subscribers": [
    "tweetUser_1",
    "tweetUser_2",
    "tweetUser_3"
  ],
  "tweets": [
    "tweetpost_7"
  ]
}
```

3. tweetUser_1 have subscribed to test user. So, eventually, he/she will receive the new post tweet.

```
CONNECT tweetUser_1
[Receive Action String] send CONNECT to client actor with userID: tweetUser_1
[Send request] Send CONNECT to server
[Recv response] Get USER_DATA {
  "userId": "tweetUser_1",
  "account": "tommy999",
  "subscribedList": [
    "test"
  ],
  "subscribers": [],
  "tweets": [
    "tweetpost_1",
    "tweetpost_3",
    "tweetpost_6"
  ]
}
```

```
[Recv response] Get NEW_TWEET_DATA {
  "userId": "test",
  "tweetId": "tweetpost_7",
  "content": "duoduoduo-BlackPink",
  "hashTag": [],
  "mention": []
}
```