Sprint5 Report

Team17 shallWeGame

So far, we've implemented most of the features of all pages.

---

1. Difficulties

a) Technically

i) In developing SNS applications that have a lot of interaction between users and services, we struggled to make them work in a stable environment. Therefore, we tried to ensure that services deployed stably in the EC2 environment can communicate reliably by the format specified by back-and-front-end through NGINX. The code written in the initial implementation phase was corrected by finding bugs that did not operate correctly during re-distribution due to the process of refreshing pages or modifying Frontend codes after being distributed.

ii) The most strenuous part of back-end implementation is data management. Due to the inevitable nature of SNS, it accumulates a lot of image data and requires frequent searching and updating. Two methodologies were considered for storing data for newly written posts in EC2's deflated applications, and for storing images on S3 when POST on back-end and obtaining URLs to access them.

iii) The first method is to save the image on s3 at the time the user uploads and posts the image on the front-end and sends the reqest in the model to the back-end and reflect it in the database. The second method is to forward the user-uploaded file from the front-end to the file and process the uploads of S3 and DB at once on the back-end. The first attempt was made to divide resources and simplify coding, but later turned to a second method to separate security issues and conceptual behavior found, and the implementation is being completed.

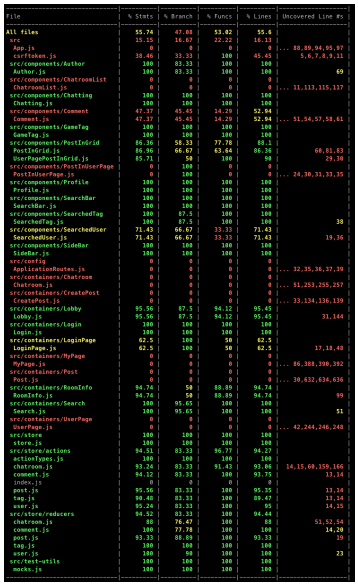
iv) Afterwards, when the initial data of various directions collected through crawling is inserted into the s3 server and the DB is completed, it is expected that the newly posted data will be continuously reflected in ML based on the results of machine learning conducted so far.

b) Organization

v) As we lives in the SNS era, we felt the need to reflect the "accumulated UI experience" that each user has in front-end implementation. For example, there are 'common functions' that users expect on the 'My Page' page, and we felt that we should reduce the experience that goes against the user's intuition by tracking and implementing these user experiences in reverse. To this end, we added the ability to provide a larger view by clicking on the post on the Post page. This has one more expected effect as well as intuitive behavior of UI/UX, which can act as an important feature in ML. In order to provide personalized recommendation services, each user needs a standard that must be applied to them, and the team members agreed that the indicator that reveals the user's interests most clearly is "viewed posts" that users have seen through clicking.

2. Progress

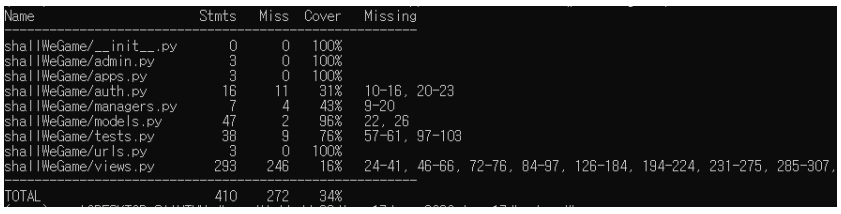
As briefly mentioned earlier, features for various user interactions were added. Specifically, it implemented actions for the shall we buttons including redirection, the creation of a user page that presses the user's profile image to enter, and a grid view of the posts written by the user in my page. It implemented a function to create a chat room between users that corresponds to a key function of our service. If the user does not explicitly select logout, there is no logout processing due to the expiration of the session and improvement is still needed. Successfully deployed applications on AWS EC2 servers. Although it is still implementing the ability to send and receive data with S3, it has finished testing all features that operate in the local environment.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	55.74	47.08	53.02	55.6	
src	15.15	16.67	22.22	16.13	
App.js	0	0	0	0	... 88,89,94,95,97
csrftoken.js	30.46	33.33	100	45.45	5,6,7,8,9,11
src/components/Author	100	83.33	100	100	
Author.js	100	83.33	100	100	69
src/components/ChatroomList	0	0	0	0	
ChatroomList.js	0	0	0	0	... 11,113,115,117
src/components/Chatting	100	100	100	100	
Chatting.js	100	100	100	100	
src/components/Comment	47.37	45.45	14.29	52.94	
Comment.js	47.37	45.45	14.29	52.94	... 51,54,57,58,61
src/components/GameTag	100	100	100	100	
GameTag.js	100	100	100	100	
src/components/PostInGrid	86.36	58.33	77.78	88.1	
PostInGrid.js	86.96	66.67	63.64	86.36	60,81,83
UserPagePostInGrid.js	85.71	50	100	90	29,30
src/components/PostInUserPage	0	100	0	0	
PostInUserPage.js	0	100	0	0	... 24,30,31,33,35
src/components/Profile	100	100	100	100	
Profile.js	100	100	100	100	
src/components/SearchBar	100	100	100	100	
SearchBar.js	100	100	100	100	
src/components/SearchedTag	100	87.5	100	100	
SearchedTag.js	100	87.5	100	100	38
src/components/SearchedUser	71.43	66.67	33.33	71.43	
SearchedUser.js	71.43	66.67	33.33	71.43	19,34
src/components/SideBar	100	100	100	100	
SideBar.js	100	100	100	100	
src/config	0	0	0	0	
ApplicationRoutes.js	0	0	0	0	... 32,35,36,37,39
src/containers/Chatroom	0	0	0	0	
Chatroom.js	0	0	0	0	... 51,253,255,257
src/containers/CreatePost	0	0	0	0	
CreatePost.js	0	0	0	0	... 33,134,136,139
src/containers/Lobby	95.56	87.5	94.12	95.45	
Lobby.js	95.56	87.5	94.12	95.45	31,144
src/containers/Login	100	100	100	100	
login.js	100	100	100	100	
src/containers/LoginPage	62.5	100	50	62.5	
LoginPage.js	62.5	100	50	62.5	17,18,48
src/containers/MyPage	0	0	0	0	
MyPage.js	0	0	0	0	... 86,388,390,392
src/containers/Post	0	0	0	0	
Post.js	0	0	0	0	... 30,632,634,636
src/containers/RoomInfo	94.74	50	88.89	94.74	
RoomInfo.js	94.74	50	88.89	94.74	99
src/containers/Search	100	95.65	100	100	
Search.js	100	95.65	100	100	51
src/containers/UserPage	0	0	0	0	
UserPage.js	0	0	0	0	... 42,244,246,248
src/store	100	100	100	100	
store.js	100	100	100	100	
src/store/actions	94.51	83.33	96.77	94.27	
actionTypes.js	100	100	100	100	
chatroom.js	93.24	83.33	91.43	93.06	14,15,60,159,166
comment.js	94.12	83.33	100	93.75	13,14
index.js	0	0	0	0	
post.js	95.56	83.33	100	95.35	13,14
tag.js	90.48	83.33	100	85.47	13,14
user.js	95.24	83.33	100	95	14,15
src/store/reducers	94.52	83.33	100	94.44	
chatroom.js	88	76.47	100	88	51,52,54
comment.js	100	77.78	100	100	14,20
post.js	93.33	88.89	100	93.33	19
tag.js	100	100	100	100	
user.js	100	90	100	100	23
src/test-utils	100	100	100	100	
mocks.js	100	100	100	100	

3. Testing

We use Jest in frontend and django.test in backend for unit test. We've done it for some units implemented in the frontend and attained 55.74% coverage. In frontend testing, we are having difficulty in testing external library: antd, styled-components, and react-photo-gallery. Also, in the backend, we are still having difficulty in user login, which leads to 34% coverage.



Name	Stmts	Miss	Cover	Missing
shallWeGame/__init__.py	0	0	100%	
shallWeGame/admin.py	3	0	100%	
shallWeGame/apps.py	3	0	100%	
shallWeGame/auth.py	16	11	31%	10-16, 20-23
shallWeGame/managers.py	7	4	43%	9-20
shallWeGame/models.py	47	2	96%	22, 26
shallWeGame/tests.py	38	9	76%	57-61, 97-103
shallWeGame/urls.py	3	0	100%	
shallWeGame/views.py	293	246	16%	24-41, 46-66, 72-76, 84-97, 126-184, 194-224, 231-275, 285-307,
TOTAL	410	272	34%	

4. Contributions

- - Hyesun Kwak : Implement UserPage, Chatroom and shallWe, Handle redirections and routings
- - Dongjoo Lee : contributed to css design for various parts in front and implementation. For application deployment, implemented EC2, S3 server configuration.
- - Haesun Jeong: In ML, tried to tokenize with phrase, khaiii, mecab and use bigram -ed inputs to calculate cosine similarity (using doc2vec) and jaccard similarity.
- - YoungWoo Song: post creating, chatroom connection, posts in grid form

5. Submission Branch for Sprint4

- Sprint5Testing