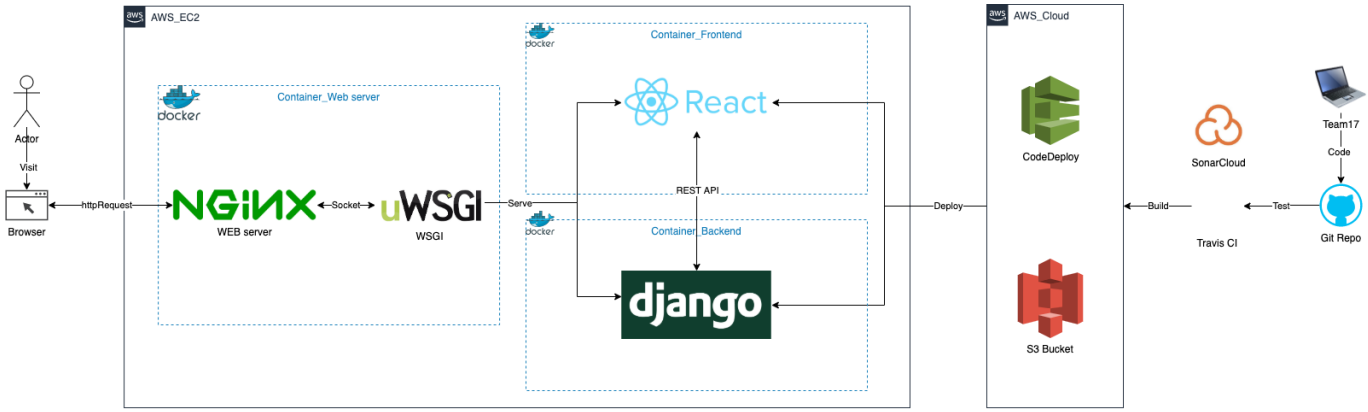
Document Revision History

Rev 1.0 2020-10-27 - initial version

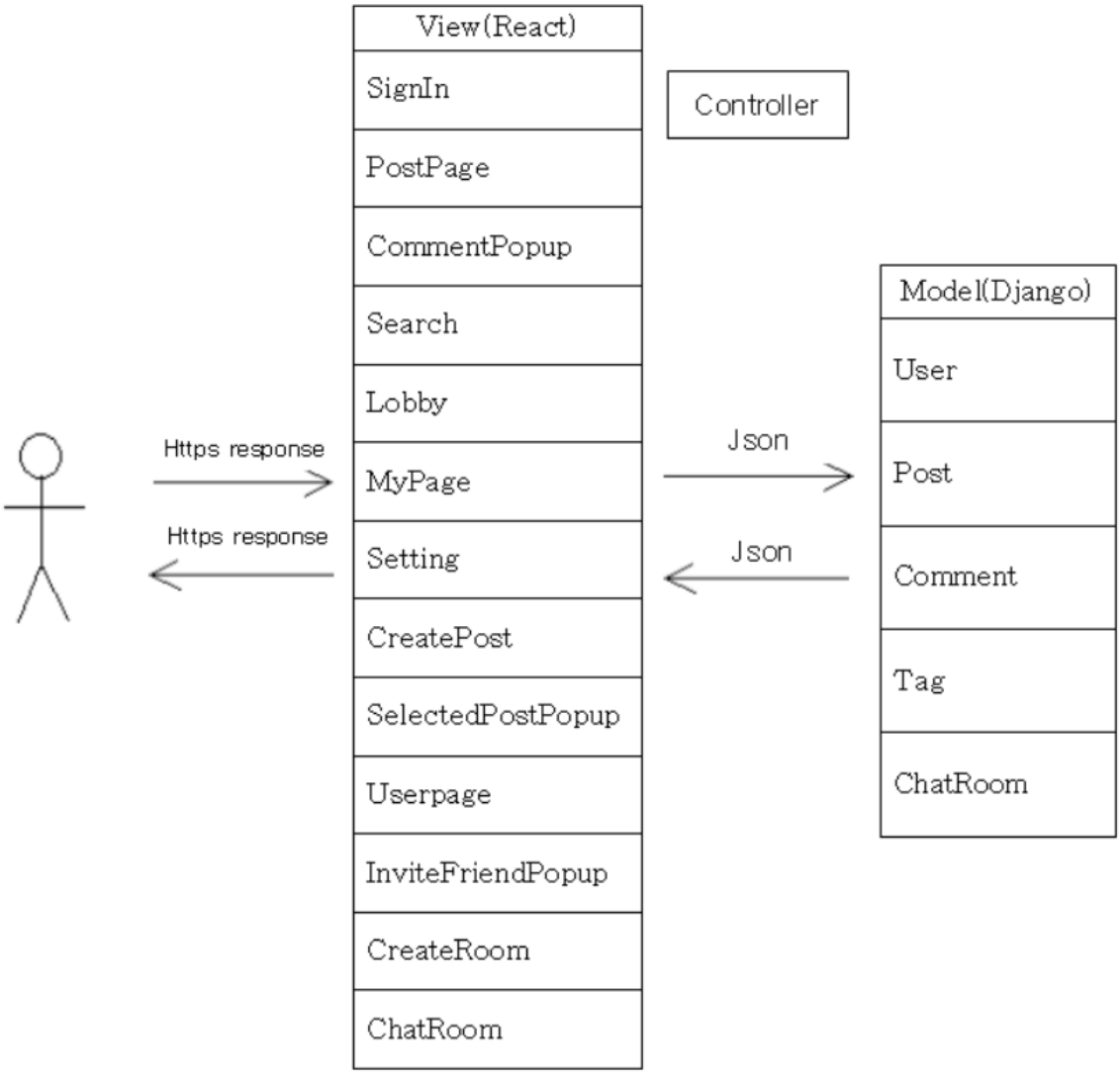
- 1.0 : initial version
- 1.1 : Add bird's eye view
- 1.2 : Update docs hierarchy
- 1.3 : Update model, view details
- 1.4 : Revise about chatroom discord
- 1.5 : Revise overall feature

System Architecture

Bird's eye view



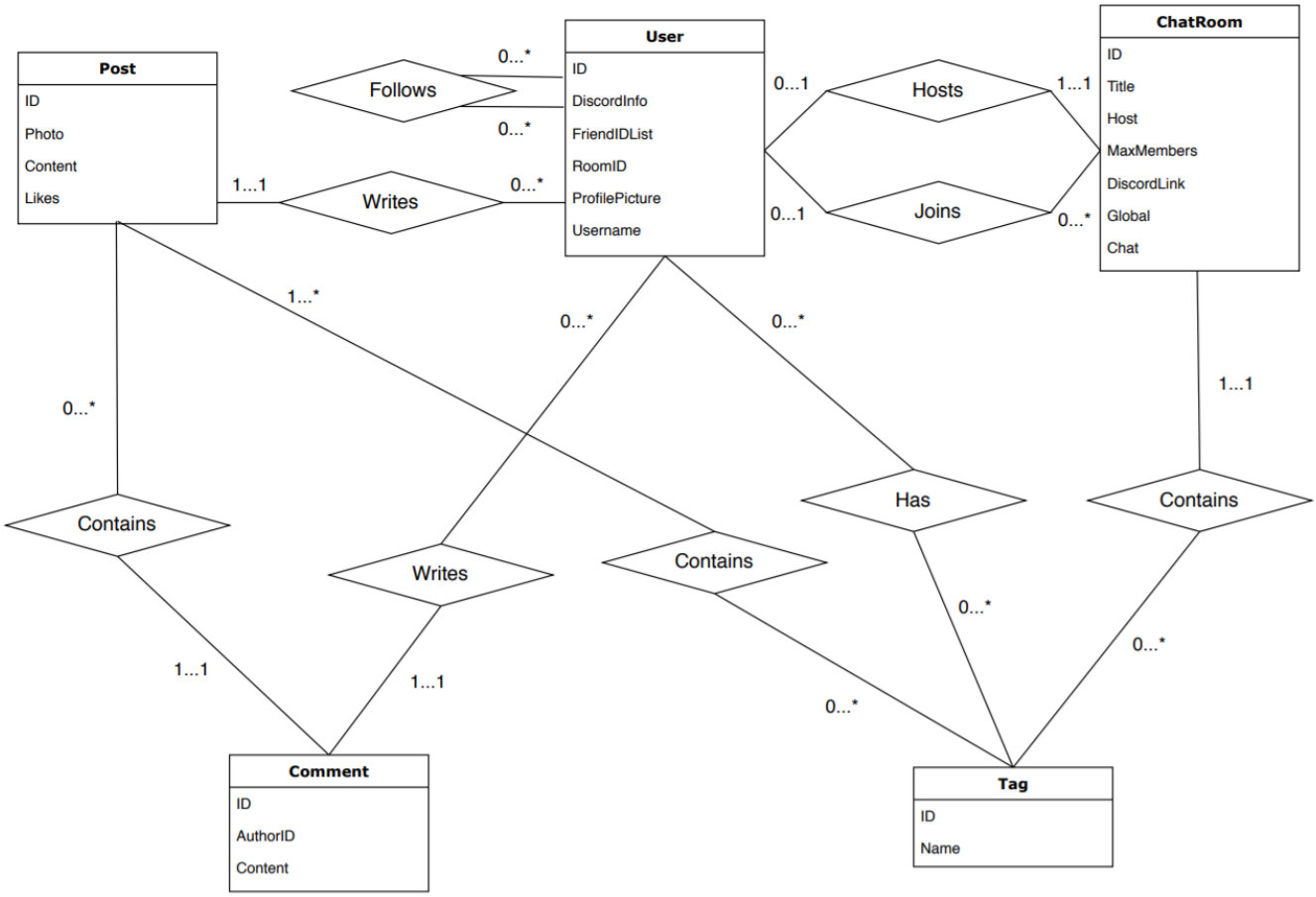
MVC pattern



We have 13 views and 5 models.

Model

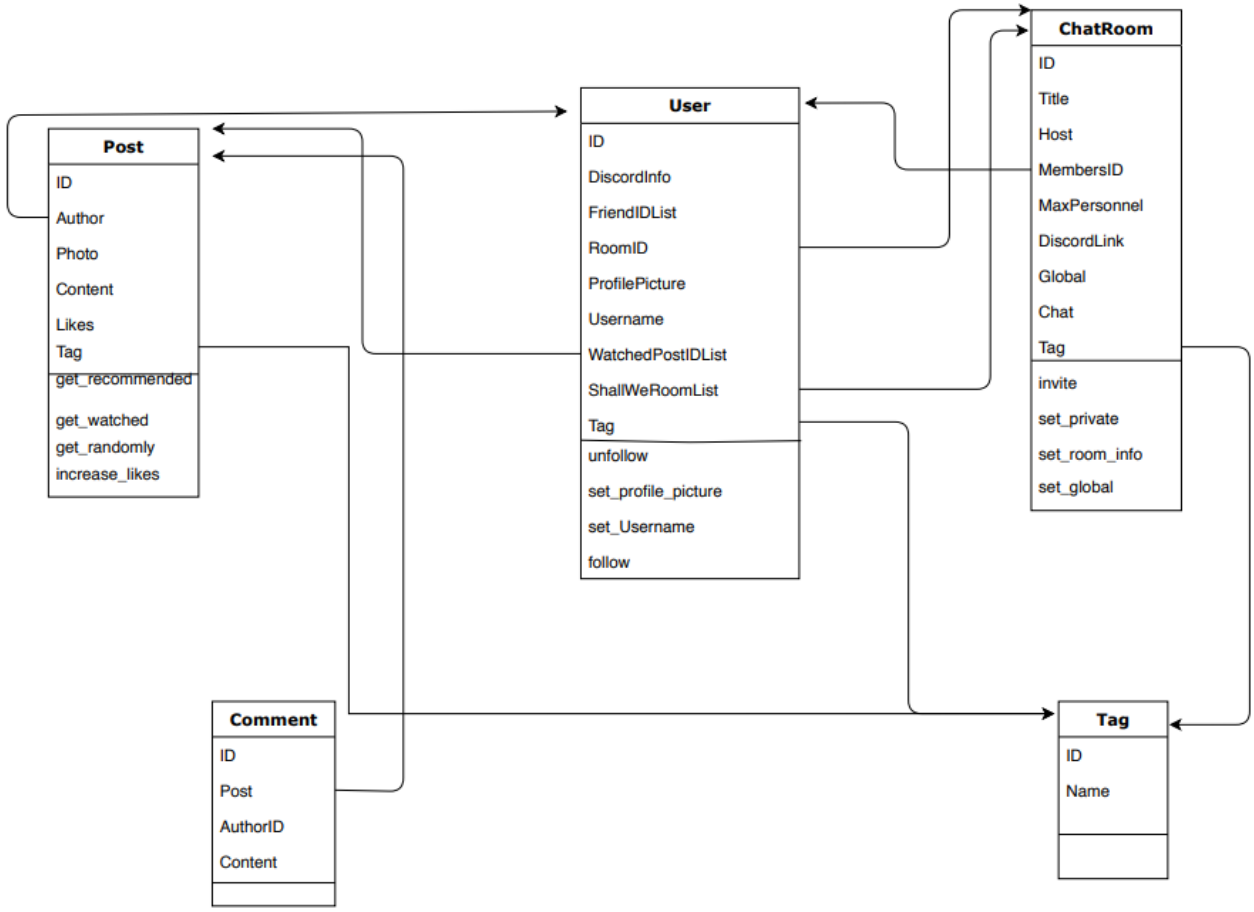
Here is E-R (Entity Relationship) diagram for model design.



Rectangle represents entity sets, diamond represents relationship sets, and numbers on line represent cardinality constraints (min...max). Rectangle includes attributes inside, and underlined attributes are primary key.

We would make all of users to set up tag right after sign up, but there is a moment that user doesn't have any tag. So, we set the number of minimum tag user have to 0, not 1.

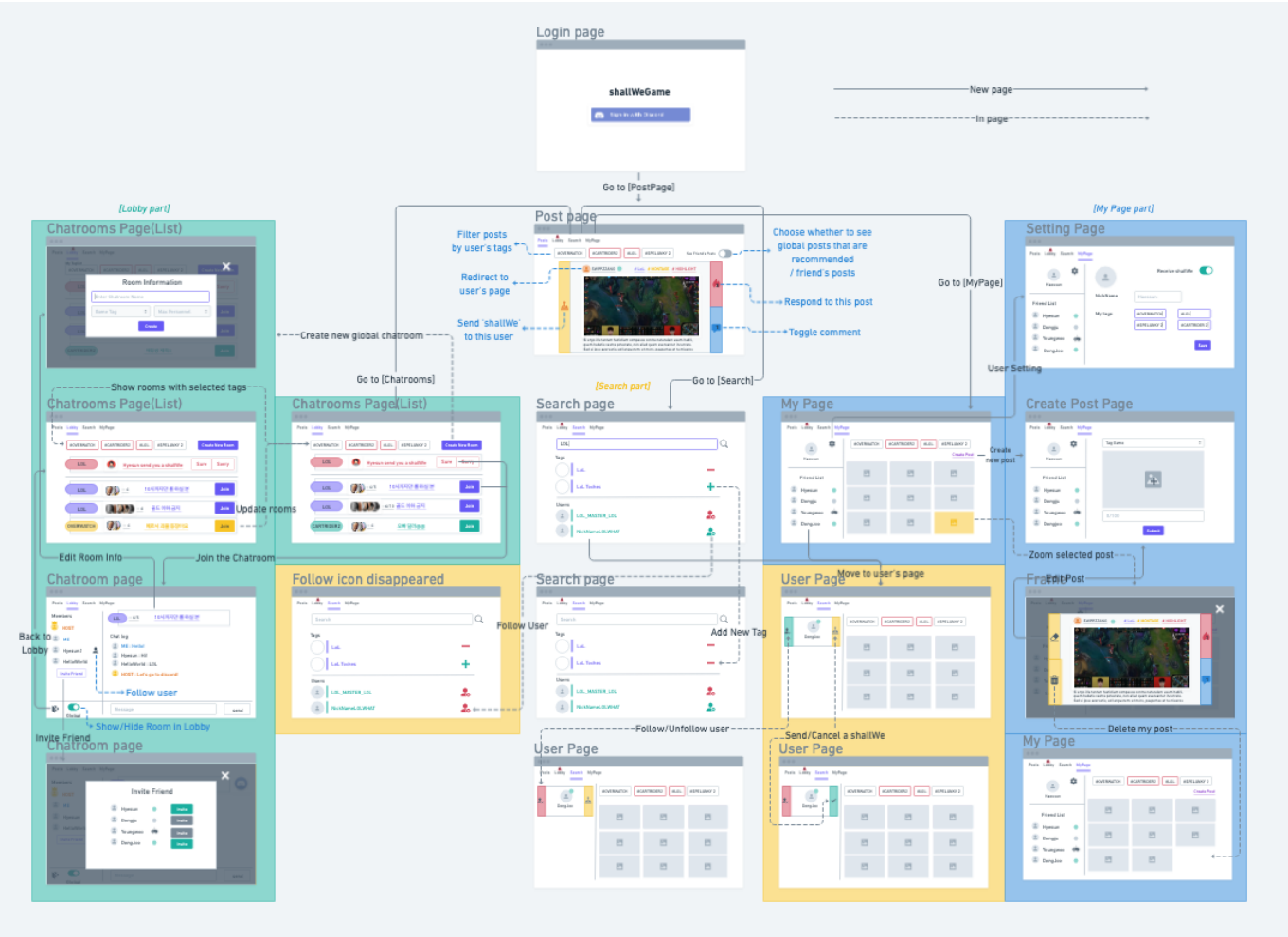
Here is relation schema diagram on E-R diagram. It would be used by the structures of Django models. Methods in lower box are core methods of each model.



Rectangle represents relation schema. Schema attributes are in the upper box, and core methods are in the lower box. Underlined attributes are primary key, and arrowed attributes are foreign key.

View

The user interface for view design is shown below.



The functionality and the requirement for each page are as follows:

Login Part

1. Login Page (‘/login’)

- If the user clicks ‘Login with Discord’ button, login is handled by Discord API.
- If the user is new, get user info(‘username’ and ‘avatar’) from Discord account.

Posts Part

2. Posts Page (‘/post’)

- Post has ‘shallWe’, ’like’, ‘comment’ buttons.
    - If the user is in a room, ‘shallWe’ button is disabled.
    - If the user clicks ‘shallWe’ button, the author of the post receives shallWe.
    - If the user clicks ‘like’ button, the number of likes of the post increments by 1.
    - If the user clicks ‘comment’ button, shows Comment.
    - If the user clicks ‘avatar’ of the author, redirect to UserPage of the author.

3. Commen

- Show comments of certain post.
- Get ‘comment’ as user input.

- If user clicks ‘submit’, comment is saved.

Lobby Part

4. Lobby Page (‘/lobby’)

- Show shallWe’s, and rooms with chosen game tags
- If the user clicks ‘Create New Room’ button, redirect to CreateRoom page.
- The list of shallWe’s are shown above the list of rooms.
    - ShallWe has ‘Sure’ and ‘Sorry’ button.
        - If the user clicks ‘Sure’ button, redirect to the room.
        - If the user clicks ‘Sorry’ button, the shallWe disappears.
    - Room has ‘Join’ button.
        - If the user clicks ‘Join’ button, redirect to the room.

5. Create New Room ('/RoomInfo')

- Get ‘chatroom name’, ‘tag’, ‘max personnel’ as user input.
- If the user clicks ‘Create’ button, new chatroom is created and redirect to the chatroom.

6. Chatroom Page (‘/lobby/:id’)

- Members are listed in left.
    - If the user clicks ‘follow’ button of a member, he/she is added as friend.
- Members can chat in this page.
- Host of the room can set ‘global’ toggle, ‘chatroom name’ and ‘max personnel’.
    - If ‘Global’ toggle is set, the room is shown in lobby.

Search Part

7. Search Page (‘/search’)

- Search games or users
- If the user clicks ‘search’ button, show list of searched games and users by getting ‘searched text’ as user input.
    - Searched game has ‘delete’ button if the user has corresponding tag, and has ‘add’ button if not.
    - Searched user has ‘unfollow’ button if he/she is a friend of the user, and has ’follow’ button if not.
    - If the user clicks ‘avatar’ of searched user, redirect to UserPage of searched user.

8. User Page (‘/user/:id’)

- Show his/her avatar with introduction and posts with chosen game tags.
- If the user clicks a post, shows ViewSelectedPost (redirect).
- If the user clicks ‘follow’ button, he/she is added to the user’s friend.
- If the user clicks ‘shallWe’ button, shallWe is sent to him/her.

9. ViewSelectedPost in User Page ('/post/:id')

- Show details of certain post
- Function as same as post in Posts page.

MyPage Part

10. My Page (‘/user/:id’)

- Show my avatar, friend list, and my posts with chosen game tags.
- If the user clicks a post, shows ViewSelectedPost (redirect).
- If the user clicks ‘Create Post’ button, redirect to CreatePost page.

11. ViewSelectedPost from My Page ('/post/:id')

- Show details of certain post
- Function similarly as post in Posts page, but no ‘shallWe’ button.
    - If the user clicks ‘delete’ button, the post is deleted.

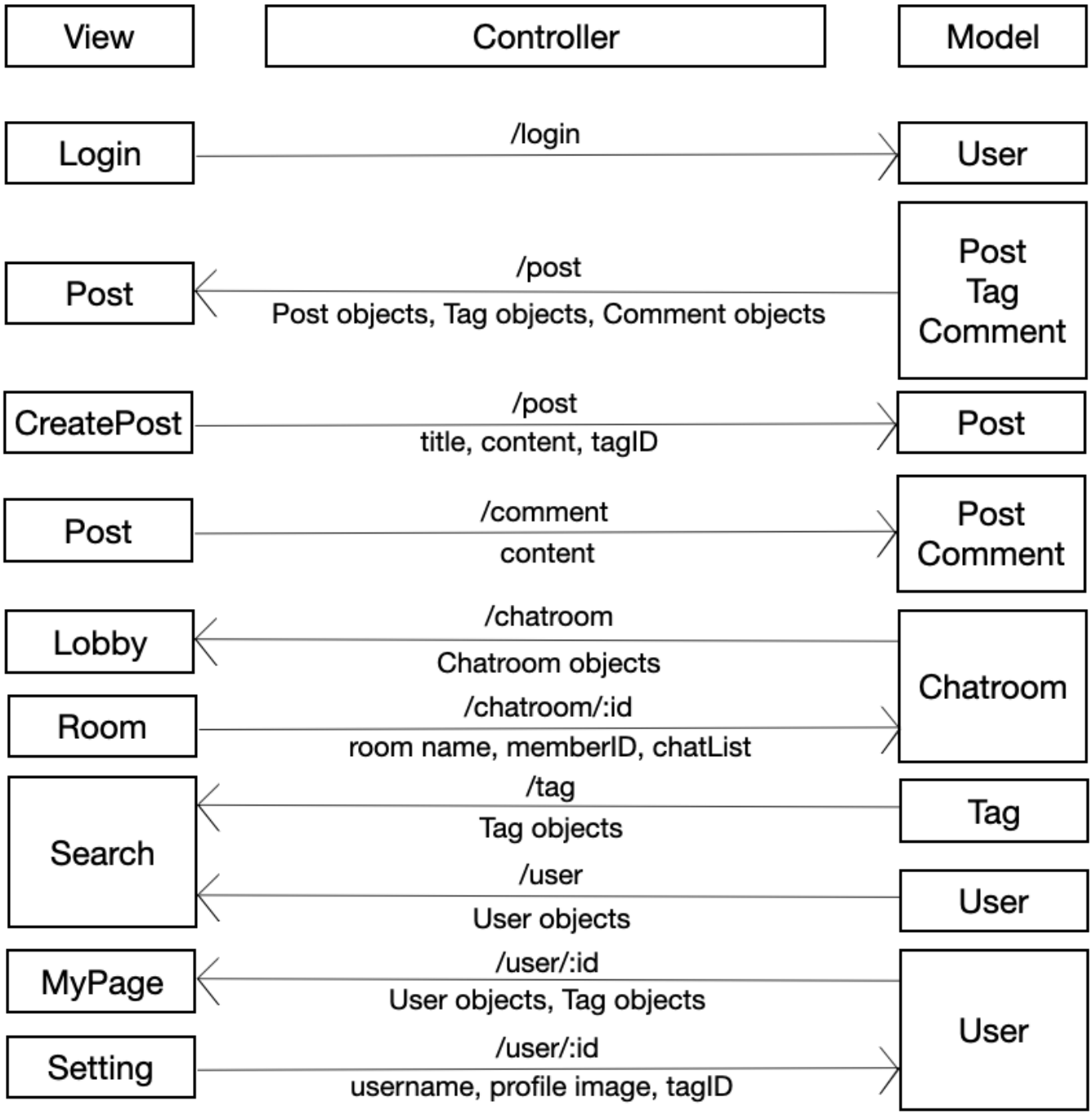
12. CreatePost Page (‘/user/:id/create’)

- Get game tag, image, and text as user inputs.
- If the user clicks ‘submit’ button, new post is created.

13. Setting Page (‘/user/:id/setting’)

- Information of the user(‘avatar’, ’username’, ‘tag’s) are shown as placeholders.
- If the user clicks ‘Save’ button, update user info by getting user inputs.
- If the user unset ‘Receive shallWe’ toggle, the user does not receive shallWe’s.

Controller



Left side is the view part(frontend) and right side is model part(backend). Left-to-right arrow describes API and JSON requests with user inputs from the view, and right-to-left arrow describes API and JSON responses with data from the model.

Design Details

Frontend Design

Here is frontend containers and components.
The attributes and the methods of each container and component are listed in each box.
Conatiners are based on 'Page' unit of our service, where the components are composed of sub-components, main-components(which consists sub-components), independent components(which has no relationship with other components)

Frontend Components

Container

Login
- logIn: button
- + onClickLogInButton

MyPage
- myProfile: Profile
- myPost: button
- myTag: button
- createPost: button
- + onClickPost
- + onClickCreatePost
- + onClickTag

Setting
- profileImage: image
- username: input
- tags: Tag
- save: button
- back: button
- + onClickSave
- + onClickBack

Post
- tags: Tag
- togglePostType: toggle
- post: Post
- + onTogglePostType

Search
- searchInput: input
- search: button
- addTag: button
- deleteTag: button
- followUser: button
- unfollowUser: button
- + onClickSearch
- + onClickAddTag
- + onClickDeleteTag
- + onClickFollowUser
- + onClickUnfollowUser

Lobby
- createNewRoom: button
- tag: Tag
- join: button
- sure: button
- sorry: button
- + onClickCreateNewRoom
- + onToggleTag
- + onClickJoin
- + onClickSure
- + onClickSorry

CreatePost
- title: input
- content: input
- image: image
- tag: Tag
- create: button
- back: button
- + onClickCreatePost
- + onClickBack

UserPage
- followUser: button
- unfollowUser: button
- userPost: button
- back: button
- + onClickFollowUser
- + onClickUnfollowUser
- + onClickUserPost
- + onClickBack

Chatroom
- toggleGlobal: toggle
- invite: button
- chat: input
- send: button
- roomInfo: button
- + onToggleGlobal
- + onClickInvite
- + onClickSendButton
- + onClickRoomInfo

Component

PostInUserPage
- userInfo: User
- tag: text
- image: image
- content: text
- edit: button
- delete: button
- comment: toggle
- + onClickEdit
- + onClickDelete
- + onClickComment
- + onClickOutside

MainPost
- userInfo: User
- tag: Tag
- shallWe: button
- like: button
- comment: toggle
- + onSendShallWe
- + onClickLike
- + onClickComment

PopComment
- comment: Comment
- author: text
- content: input
- confirm: button
- + onClickConfirm
- + onClickOutside

InviteFriend
- Title: text
- user: User
- invite: button
- + onClickInvite
- + onClickOutside

Profile
- profileImage: image
- username: text
- friendList: User
- setting: button
- + onClickSetting

MenuBar
- post: button
- lobby: button
- search: button
- myPage: button
- + onClickPostButton
- + onClickLobbyButton
- + onClickSearchButton
- + onClickMyPageButton

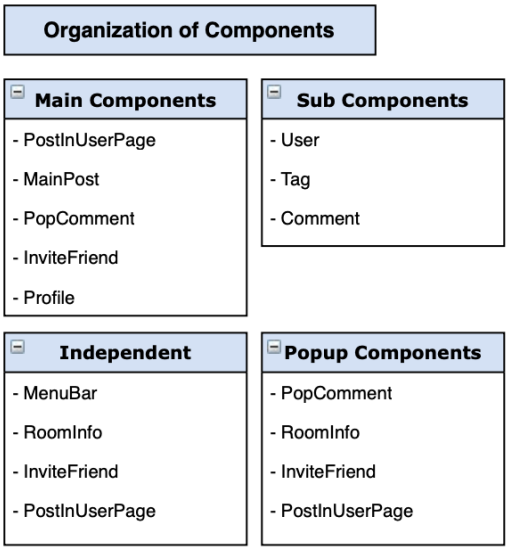
RoomInfo
- roomname: input
- tag: dropdown
- maxUserNum: dropdown
- create: button
- + onChangeTag
- + onChangeMaxUserNum
- + onClickCreateNewRoom
- + onClickOutside

PostInGrid
- image: button
- + onClickImage

User
- profileImage: image, button
- username: text
- recent: icon
- + onClickProfileImage

Tag
- name: text
- clickTag: toggle
- + onToggleTag

Comment
- author: text
- content: text
- edit: prompt
- delete: button
- + onClickEdit
- + onClickDelete



Frontend Algorithms

[Containers]

1. Login

- onClickLogInButton(): Redirect to 'Discord' and authenticate user

2. MyPage

- onClickPost() : Show all the details of selected post in pop-up form.→ Call backend api
- onClickCreatePost() : Redirect to CreatePost page
- onClickTag() : Toggle the selected tag (display posts filtered by selected tags)

3. Post

- onTogglePostType() : Toggle whether to watch recommended global posts or only friend's posts

4. Search

- onClickSearch(keyword : string) : Search and show the result of users and tags objects with the keyword in their name
- onClickAddTag() : Add selected tag to taglist of user. Added tags are shown as toggle type buttons on My Page, User Page, Post Page and Lobby.
- onClickDeleteTag() : Delete selected tag from taglist of user.
- onClickFollowUser() : Add selected user to userlist of user.
- onClickUnfollowUser() : Delete selected user from userlist of user.

5. Lobby

- onClickCreateNewRoom() : Create new chatroom and redirect to it with user in it.
- onToggleTag() : Toggle the clicked tag. Chatrooms with selected tags are shown to the user.
- onClickJoin() : Redirect to selected chatroom.
- onClickSure() : By answering yes to shallWe, user redirects to selected chatroom.
- onClickSorry() : By rejecting the shallWe, the chatroom disappears from the chatroom list.

6. CreatePost

- onClickCreatePost(image: image, content: content, tags: int) : Create new post and redirects to MyPage.

7. UserPage

- onClickFollowUser() : Add selected user to userlist of user.
- onClickUnfollowUser() : Delete selected user from userlist of user.
- onClickUserPost() : Pop up the selected post with details.

8. Chatroom

- onToggleGlobal(): If toggled as Global, recent chatroom will be visible to everyone.
- onClickSendButton(): If message input is not blank, the content will be sent to Chat log.

[Components]

1. MenuBar

- onClickPostButton(): Redirect to Post page.
- onClickLobbyButton(): Redirect to Lobby page.
- onClickSearchButton(): Redirect to Search page.
- onClickMyPageButton(): Redirect to Mypage page.

2. MainPost

- onSendShallWe(): Send 'ShallWe' to author of the post.
- onClickLike(): increase likes of the post.
- onClickComment(content: string): Make new comment. Disabled if content is empty.

3. User

- onClickAvatar(): Redirect to Userpage page of the selected user.

4. PostInUserPage

- onClickDelete(): Delete selected post.
- onClickComment(): Pop up comments list.

5. PostInGrid

- onClickImage(): Pop up selected post.

6. Tag

- onToggleTag(): Toggle state of selected tag.

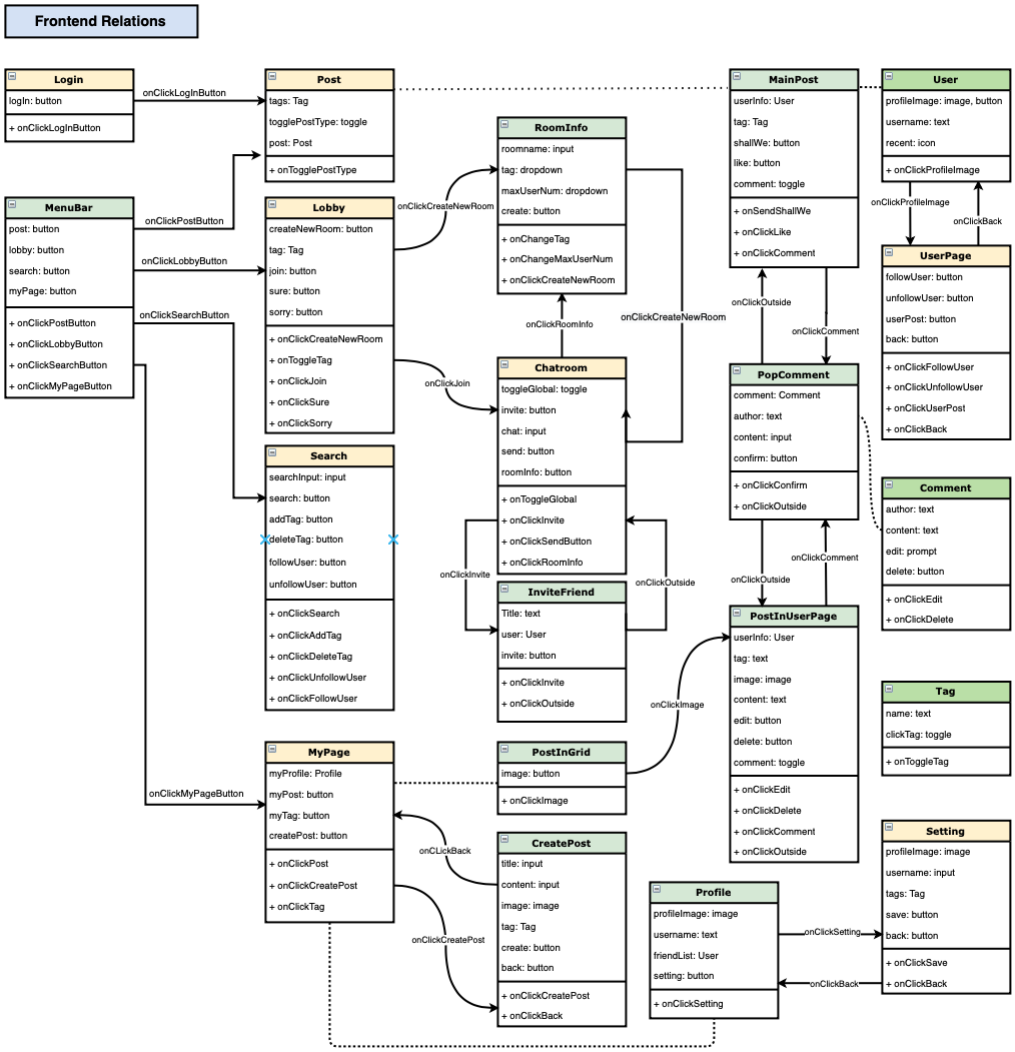
7. RoomInfo

- onChangeTag(): Select tag of the room.
- onClickCreateNewRoom(): Make new Chatroom and redirect to the Chatroom page.

8. Comment

- onClickDelete(): Delete selected comment.

Frontend Relations



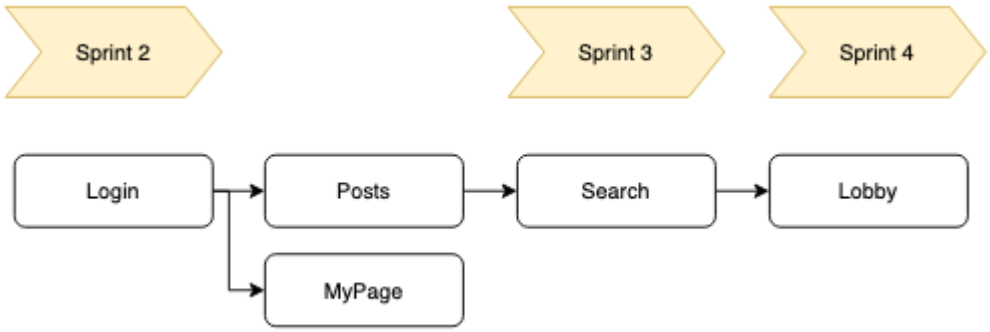
Backend Design

Here is detailed specifications of RESTful APIs for backend design.

Model	API	GET	POST	PUT	DELETE
User	/signup	X	Create new user	X	X
	/signin	X	Log in	X	X
	/signout	Log out	X	X	X
	/user	Get all users	X	X	X
	/user/:id	Get specific user	X	Change user info	X
Post	/post	Get all posts	Create new post	X	X
	/post/:id	Get specific post	X	Increase number of Like	Delete specific post
Chatroom	/chatroom	Get all chatrooms	Create new chatroom	X	X
	/chatroom/:id	Get specific chatroom	X	Change room name, memberID, chatList	Delete specific room
Tag	/tag	Get all tags	X	X	X
	/tag/:id	Get specific tag	X	X	X
Comment	/comment	Get all comments	Create new comment	X	X
	/comment/:id	Get specific comment	X	Change specific comment	Delete specific comment

Implementation Plan

The implementation plan gives the description of programming tasks divided by sprints and members. As all user stories are based on the corresponding pages, the plan is described based on pages as well. The pages are already divided considering dependencies in the UI: 'Login', 'Posts', 'Lobby', 'Search' and 'MyPage'. We will develop 'Login' part in sprint 2 which uses Discord API. Then in sprint 3,'Posts', 'Search', and 'MyPage' part, mainly functioning as SNS service, will be developed. Finally, in sprint 4, 'Lobby' part that is related to chatroom will be developed.



The implementation plan is specified by features as below. The estimated difficulty and time of plan are also indicated.

Page	Feature	Difficulty (1-5)	Time (min)	Sprints	Person	Challenge
Login	Login with Discord account	3	60	2	송영우, 곽혜선	Login with Discord API
Posts	Posts with certain restrictions	4	240	3	송영우	Filter posts with toggles
MyPage	Posts with certain restrictions, Zoom selected post	4	240	3	이동주, 정해선	Implement functions of Post component
MyPage	Profile and friend list	3	120	3	곽혜선	
MyPage	Create post	3	120	3	송영우	
Search	Search tags and users	2	60	3	이동주	
Search	User page	3	180	3	정해선	
Lobby	List of shallWe and rooms	4	240	4	송영우, 곽혜선	Create Discord links with Discord API
Lobby	Set room information	1	60	4	이동주	
Lobby	Chatroom	5	300	4	이동주, 정해선	Implement chatting section
Lobby	Invite friend with pop-up	2	120	4	정해선	

Testing Plan

Unit Testing

All of components and modules should be tested automatically. We would test them with following tools in each sprint. Expected value of the code coverage is 90%.

React : Jest
Django : Python unit test

Functional Testing

All APIs should be tested automatically. In sprint 3, we would test APIs about authentication and posting. In sprint 4, we would test APIs about room and 'shallWe'. We would test them with following tools.

React : Jest
Django : Python unit Test

Acceptance & Integration Testing

Acceptance and integration testing would be done after implementation. In sprint 5, we would use Cucumber for acceptance testing with user stories in sprint 1. Also, we would use Trevis CI for Integration testing.

Integration Testing : Trevis CI