SNU Moyeo

Design and Planning

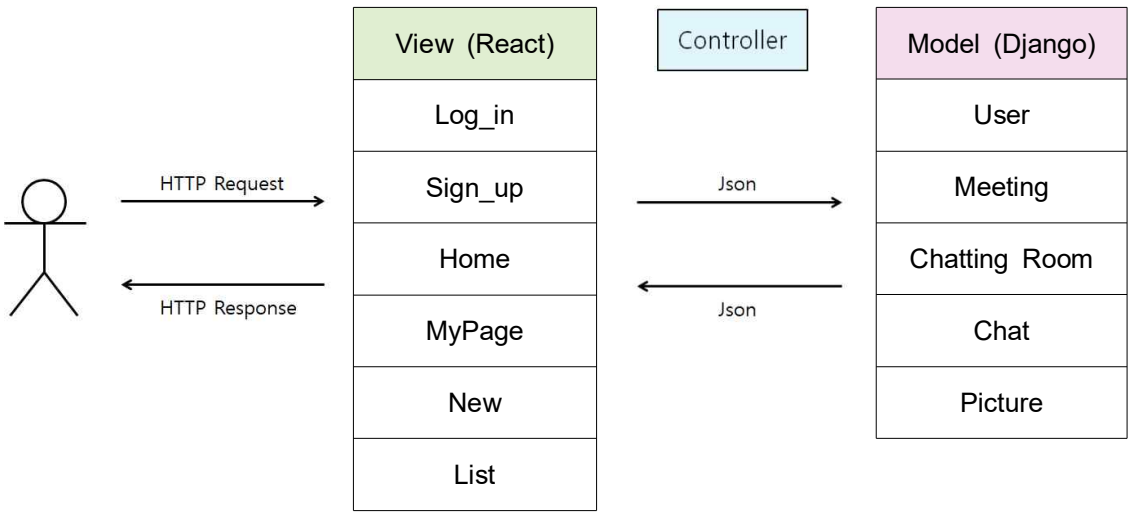
Members of group 2

최덕경 (컴퓨터공학부 2016-10399), 김동욱 (컴퓨터공학부 2016-16712),
이종민 (컴퓨터공학부 2016-15160), 정재훈 (컴퓨터공학부 2016-11464)

System Architecture

MVC

Here is MVC of our project, which has 6 Views and 5 Models.
Controller will be de described further in 'Controller' part.

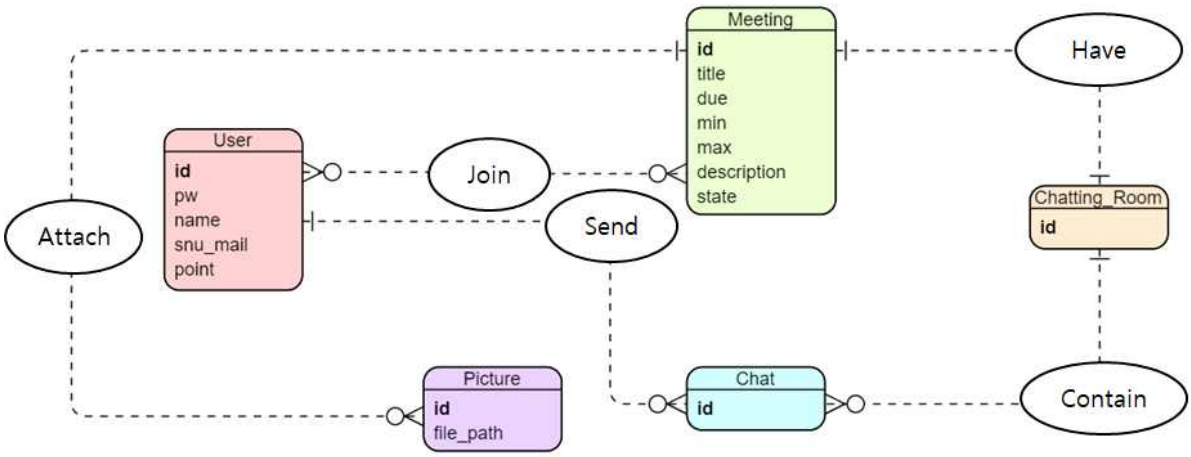


Model

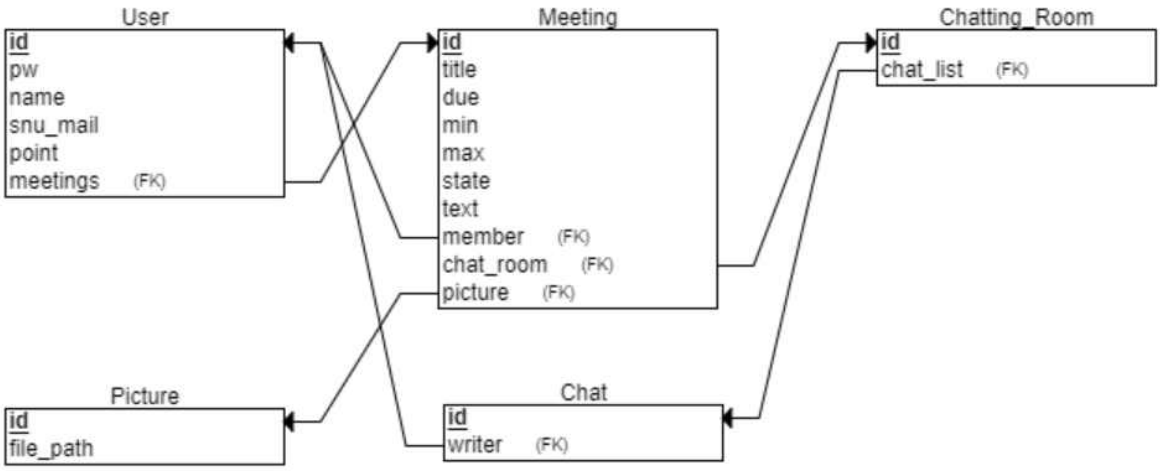
Here is **E-R (Entity Relationship) Diagram** for model design.

Rectangle stands for entity type. Oval stands for relation type, which represents a relation between multiple entity types. Each entity's attributes are listed inside the corresponding rectangle, and each underlined attribute is the primary key of the corresponding entity type.

① A user can **join** multiple meetings, and a meeting has multiple members(users).
② A user can **send** multiple chats(messages) to the corresponding chatting room.
③ A meeting **has** a chatting room.
④ A chatting room **contains** multiple chats(messages).
⑤ Multiple pictures can be **attached** to a meeting.

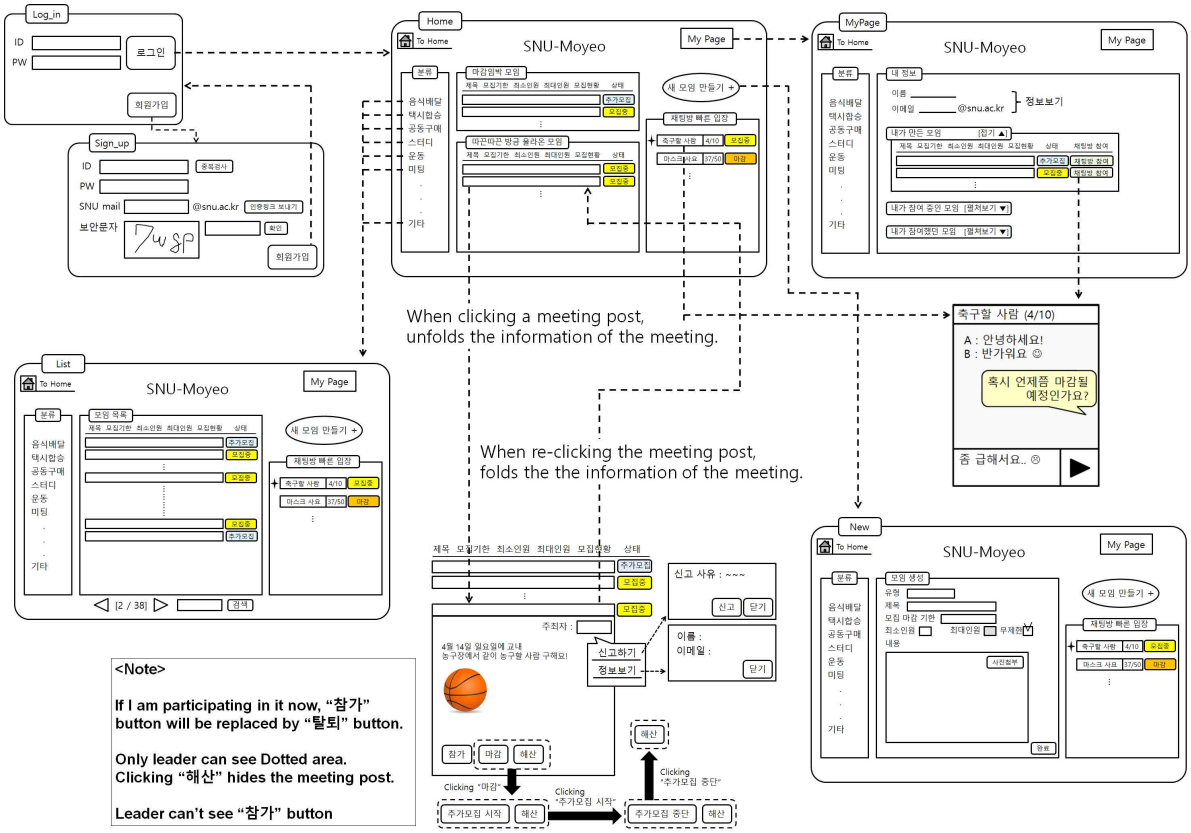


Here is **Relation Schema Diagram** based on E-R diagram. This is also the structure of our django models in back-end. (FK) stands for foreign key, and each arrow indicates which model the corresponding foreign key refers to. The methods that each model has are omitted, which will be discussed later.



View

Here is **user interface (UI)** for view design.



Note 1 : The arrows from "To Home" buttons are omitted. (It means "Go to <Home> page")

Note 2 : Duplicated arrows are omitted.
For example, although "My Page" buttons are located in several pages, only one arrow is drawn here. That's because all "My Page" buttons have the same meaning: "Go to <MyPage> page"

The functionality and requirement for each page are described as below. Duplicated functionalities are described only once. For example, the functionality that creates a new meeting by clicking the [새 모임 만들기 +] button is provided by both the <Home> page, <List> page, and <New> page, the explanation is described only once, not three times.

1. <Log_in> page ('/log_in')

- Sign in
- If the user clicks the [회원가입] button, navigate to the <Sign_up> page

2. <Sign_up> page ('/sign_up')

- Sign up a new user
- Get 'ID' and 'PW' as user inputs
- Check if the user is SNU member by 'SNU mail'

- Check if the user is robot by 보안문자
- If the user clicks [회원가입] button, navigate to the <Log_in> page

3. <Home> page (‘/ ’)

- If the user clicks the [My page] button, navigate to the <MyPage> page
- If the user clicks the [새 모임 만들기 +] button, navigate to the <New> page
- In the left side bar [분류], show the meeting types
- If the user clicks a meeting type on the left side bar [분류], navigate to the <List> page
- In the right side bar [채팅방 빠른 입장], show the meetings that I am participating in
- If the user clicks a meeting of [채팅방 빠른 입장], pop up the chatting room of the meeting
- In the middle area, show two meeting lists
  ① In the upper area [마감임박 모임], show the list of the meetings of which deadline is coming
  ② In the lower area [따끈따끈 방금 올라온 모임], show the list of the meetings created just now
- If the user clicks a meeting post, unfold the content of the meeting
  ① Case 1 : The user is not leader of the meeting
    ⑴ If the user has already belonged to the meeting, he can see [참가] button
    ⑵ If the user doesn't belong to the meeting, he can see [탈퇴] button
  ② Case 2 : The user is the leader of the meeting
    ⑴ If the state of the meeting is [모집 중], he can see [마감], [해산] button
    ⑵ If the state of the meeting is [모집 마감], he can see [추가모집 시작], [해산] button
    ⑶ If the state of the meeting is [추가 모집 중], he can see [추가모집 중단], [해산] button
    ⑷ If the state of the meeting is [추가모집 마감], he can see [해산] button
- If the user clicks the [참가] button, he can enter the meeting
- If the user clicks the [탈퇴] button, he can withdraw from the meeting
- If the user(who is the leader) clicks the [마감] button, he can stop recruiting of the meeting
- If the user(who is the leader) clicks the [해산] button, he can break up the meeting
- If the user(who is the leader) clicks the [추가모집 시작] button, he can make an additional recruitment of the meeting
- If the user(who is the leader) clicks the [추가모집 중단] button, he can stop the additional recruitment of the meeting

4. <New> page (‘/new’)

- Create a new meeting
- The user types the meeting information(type, title, deadline, minimum number of member, maximum number of member, description) and completes creating a new meeting by clicking the [완료] button

5. <List> page (‘/list/:meeting_type’)

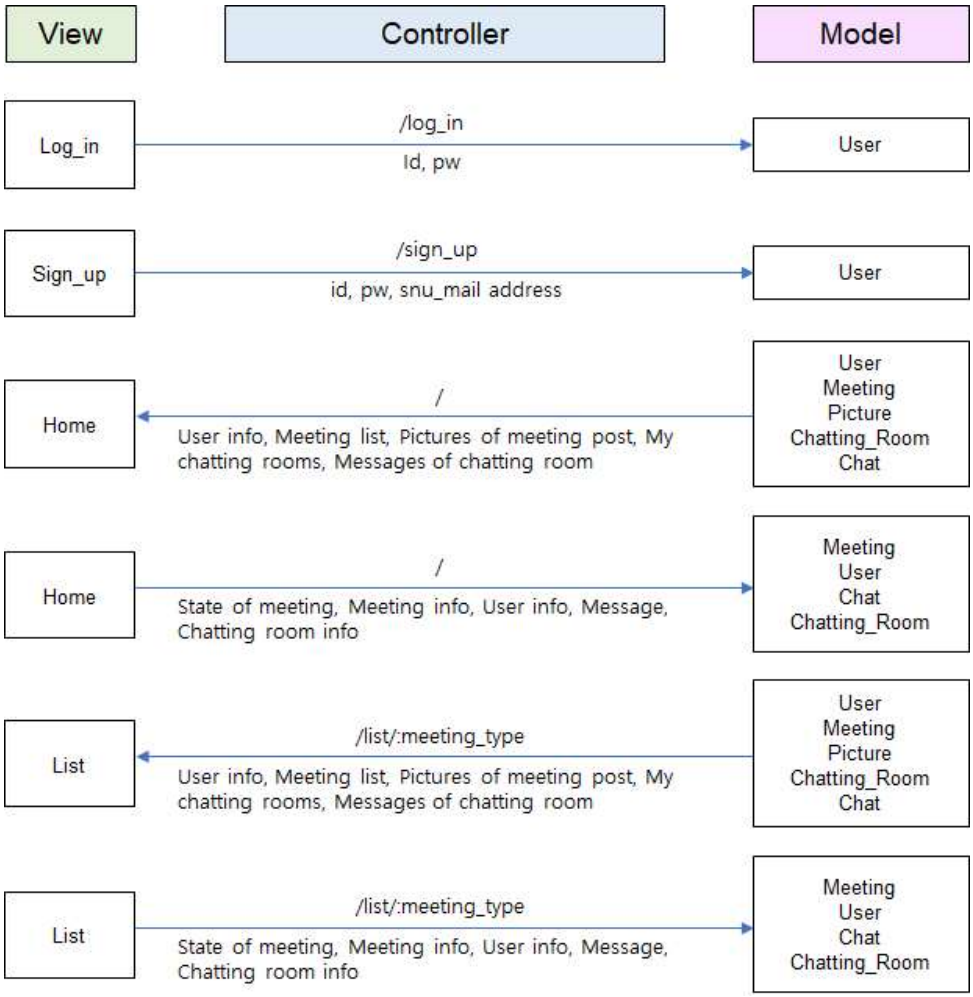
- Show the selected type of meeting list
- If the user types a search keyword and clicks the [검색] button, the corresponding meeting posts are searched and shown

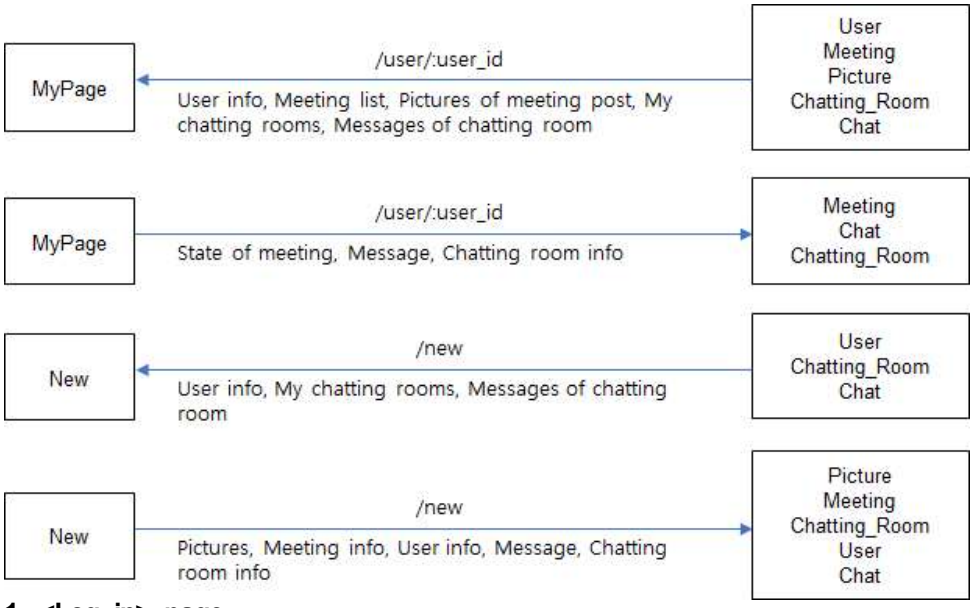
6. <MyPage> page ('/user/:user_id')

- Show my account's information (name and SNU mail address)
- In the [내가 만든 모임] menu, show the meetings that I created
- In the [내가 참여 중인 모임] menu, show the meetings that I am participating in
- In the [내가 참여했던 모임] menu, show the meetings that I participated in (in the past)
- If the user clicks the [채팅방 참여] button, pop up the corresponding chatting room

Controller

Here is controller design. Left side is view part (front-end) and right side is model part (back-end). Left-to-right arrow represents HTTP request with user inputs from view, which creates, modify, or delete specific models. Right-to-left arrow represents HTTP response with data from model, which means that some data of specific models are read. There is a corresponding URL(API) above each arrow, which the controller uses to transfer Json data between view side and model side. Those data are described below the arrow.





1. <Log_in> page

- When logging in, send my user id and password

2. <Sign_up> page

- When signing up, send new user id, password, and snu_mail address

3. <Home> and <List> page

- For identifying me, fetch user information
- For listing meeting posts, fetch meeting list
- For showing pictures when clicking a meeting post, fetch pictures of the meeting post
- For popping up a chatting room, fetch the chatting rooms that I am participating in
- For showing messages when popping up a chatting room, fetch the messages in the room
- When changing state of a meeting, send the new state information
- When joining a meeting, send the meeting information and my user information (ⓐ)
- When chatting, create and send the message and the chatting room info

4. <MyPage> page

- Except ⓐ, the same as <Home> and <List> page (Joining a meeting in this page is imposible.)

5. <New> page

- For identifying me, fetch user information
- For popping up a chatting room, fetch the chatting rooms that I am participating in
- For showing messages when popping up a chatting room, fetch the messages in the room
- When attaching some pictures and complete creating a new meeting, create and send the pictures
- When creating a new meeting, create a meeting and a chatting room, send the meeting information and my user information
- When chatting, create and send the message and the chatting room info

Design Details

1. Front-end Design

Front-end Components

Here is front-end components of our web. The attributes and the functions of each component are described in the corresponding box.

Log_in
id: input pw: input log_in: button sign_up: button
+ onClickLogInButton + onClickSignUpButton

Home
meeting: button meeting_content: Meeting meeting_state: text my_page: button left_sidebar: Left_sidebar Right_sidebar: Right_sidebar
+ ngOnInit + onClickMeetingButton + onClickMyPageButton

List
meeting: button meeting_content: Meeting meeting_state: text my_page: button search_keyword: input search: button page: text prev: button next: button left_sidebar: Left_sidebar Right_sidebar: Right_sidebar
+ ngOnInit + onClickMeetingButton + onClickMyPageButton + onClickSearchButton + onClickPrevButton + onClickNextButton

Sign_up
id: input pw: input snu_mail: input captcha: g-captcha send_mail : button check_dup : button sign_up: button
+ ngOnInit + onClickSendMailButton + onClickCheckDupButton + onClickSignUpButton + onClickCheckCaptchaButton

MyPage
name: text snu_mail: text meeting: button meeting_content: Meeting meeting_state: text my_page: button meeting_self: button meeting_current: button meeting_history: button join_chatroom: button left_sidebar: Left_sidebar
+ ngOnInit + onClickMeetingButton + onClickMyPageButton + onClickMeetingSelfButton + onClickMeetingCurrentButton + onClickMeetingHistoryButton + onClickJoinChatroomButton

New
type: input title: input due: date-local-time min: input max: input infinity: checkbox description: input attach: button done: button my_page: button left_sidebar: Left_sidebar Right_sidebar: Right_sidebar
+ ngOnInit + onClickMyPageButton + onClickAttachButton + onClickDoneButton

Left_sidebar
to_home: button type: button
+ onClickToHomeButton + onClickTypeButton

Meeting
title: text leader: button description: text picture: image join: button withdraw: button break_up: button close: button start_extra_recruit: button stop_extra_recruit: button
+ onClickLeaderButton + onClickJoinButton + onClickWithdrawButton + onClickBreakUpButton + onClickCloseButton + onClickStartExtraRecruitButton + onClickStopExtraRecruitButton

Right_sidebar
make_new: button join_chatroom: button meeting_state: text
+ onClickMakeNewButton + onClickJoinChatroomButton

Front-end Algorithms

1. ngOnInit : Check the user is logged in by calling backend api, and redirect to login page if user is not signed in.

2. Log_in

- `onClickLogInButton(id: string, pw: string)`: Call back-end sign-in api. If sign-in is accepted, redirect to the <Home> page. If sign-in is rejected, alert it.
- `onClickSignUpButton()`: Redirect to the <Sign_up> page.

3. Sign_up

- `ngOnInit()`
- `onClickSendMailButton(snu_mail: string)`: Send mail for authentication to user's SNU mail.
- `onClickCheckDupButton(id: string)`: Check whether id is duplicated.
- `onClickSignUpButton(id: string, pw: string)`: Make my account. If sign-up is successful, redirect to the <Log_in> page. Check whether id duplication test has already done.

4. Home

- `ngOnInit()`
- `onClickMeetingButton()`: Unfold the content of the clicked meeting.
- `onClickMyPageButton()`: Redirect to the <MyPage> page.

5. List

- `ngOnInit()`
- `onClickMeetingButton()`: Unfold the content of the clicked meeting.
- `onClickMyPageButton()`: Redirect to the <MyPage> page.
- `onClickSearchButton(search_keyword: string)`: Find the meetings whose title has search keyword.
- `onClickPrevButton()`: Show the previous page of the meeting list.
- `onClickNextButton()`: Show the next page of the meeting list.

6. MyPage

- `ngOnInit()`
- `onClickMeetingButton()`: Unfold the content of the clicked meeting.
- `onClickMyPageButton()`: Redirect to the <MyPage> page.
- `onClickMeetingSelfButton()`: Unfold the list of meetings that I created.
- `onClickMeetingCurrentButton()`: Unfold the list of meetings that I am participating in.
- `onClickMeetingHistoryButton()`: Unfold the list of meetings that I participated in. (in the past)
- `onClickJoinChatroomButton(chatroom_id: number)`: Pop up the chatting room corresponding to chatroom_id.

7. New

- `ngOnInit()`
- `onClickMyPageButton()`: Redirect to <MyPage> page.
- `onClickAttachButton()`: Pop up the window for searching the path of the pictures to attach.
- `onClickDoneButton(title, due, min, max, description)`: Create a new meeting and redirect to the <Home> page.

8. Meeting

- `onClickLeaderButton(leader_id)`: Show the leader's information.
- `onClickJoinButton(meeting_id, user_id)`: Join the selected meeting.
- `onClickWithdrawButton(meeting_id, user_id)`: Withdraw from the selected meeting.
- `onClickBreakUpButton(meeting_id)`: Break up the selected meeting.
- `onClickCloseButton(meeting_id)`: Stop recruiting of the selected meeting.
- `onClickStartExtraRecruitButton(meeting_id)`: Start re-recruiting of the selected meeting.
- `onClickStopExtraRecruitButton(meeting_id)`: Stop re-recruiting of the selected meeting.

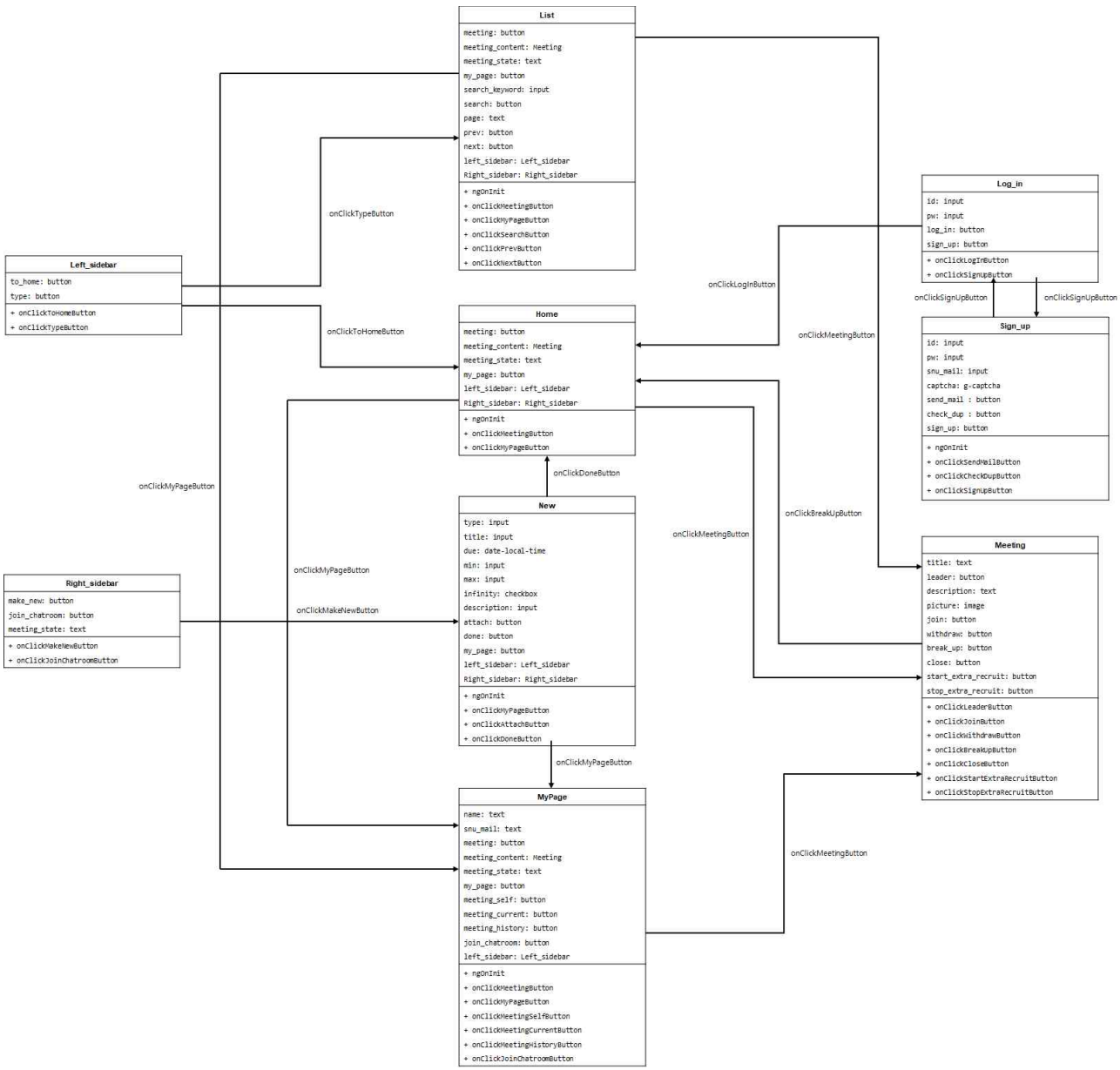
9. Left_sidebar

- `onClickToHomeButton()`: Redirect to the <Home> page.
- `onClickTypeButton(type: number)`: Redirect to the <List> page corresponding to the meeting type.

10. Right_sidebar

- `onClickMakeNewButton()`: Redirect to the <New> page.
- `onClickJoinChatroomButton(chatroom_id: number)`: Pop up the chatting room corresponding to chatroom_id.

Front-end Relations



2. Back-end Design

In the back-end design, we use models that was discussed in MVC section.

RESTful API

Detailed specifications of RESTful APIs are as following(refer to 'Controller' section) :

Model	API	GET	POST	PUT	DELETE
User	/sign_up	X	Create a new user(account)	X	X
	/log_in	X	Log in	X	X
	/user	X	Post user information	X	X
	/user/:id	Get user information	X	X	X
Chat	/chat/:chatroom_id	X	Post chat to specified chatting room	X	X
Chatting_Room	/chatroom	X	Post chatting room when creating a new meeting	X	X
	/chatroom/:id	Get chatting room	X	X	X
Picture	/picture/:meeting_id	X	Create a new picture	X	X
Meeting	/meeting	Get meeting list	Create a new meeting	X	X
	/meeting/:id	X	X	X	Delete specified meeting

Implementation Plan

Our implementation plan shows the division of works that will be done for each duration. Each user story will be implemented when programming for front-end design, and each model we designed for view will be implemented when programming for back-end design.

First, we will design <Log_in> and <Sign_up> pages and the basic skeleton of <Home> page in Sprint 2. After designing those pages, we will design 'Meeting' model in detail(member variables, member methods, etc.) in back-end first. That's because front-end programming depends on back-end design, and 'Meeting' model is connected to almost every pages except <Log_in> and <Sign_up> pages. After designing 'Meeting' model in detail, we will implement several functionalities related to 'Meeting' model. For example, creating a new meeting, closing a meeting, withdrawing from a meeting, start re-recruiting, stop re-recruiting, and breaking up a meeting, etc. After this we will implement <MyPage> page since it needs a lot of information about 'Meeting' model. Then Sprint 3 ends. In Sprint 4, we will implement chatting functionality in person, or by using some APIs. And lastly in Sprint 5, all other detailed tasks about <Home> page and 'Meeting' model will be dealt with. Also, we will add penalty policy. Other functions will be considered if necessary.