Design and Planning

Document Revision History

Rev. 1.0 2021-10-30 - initial version

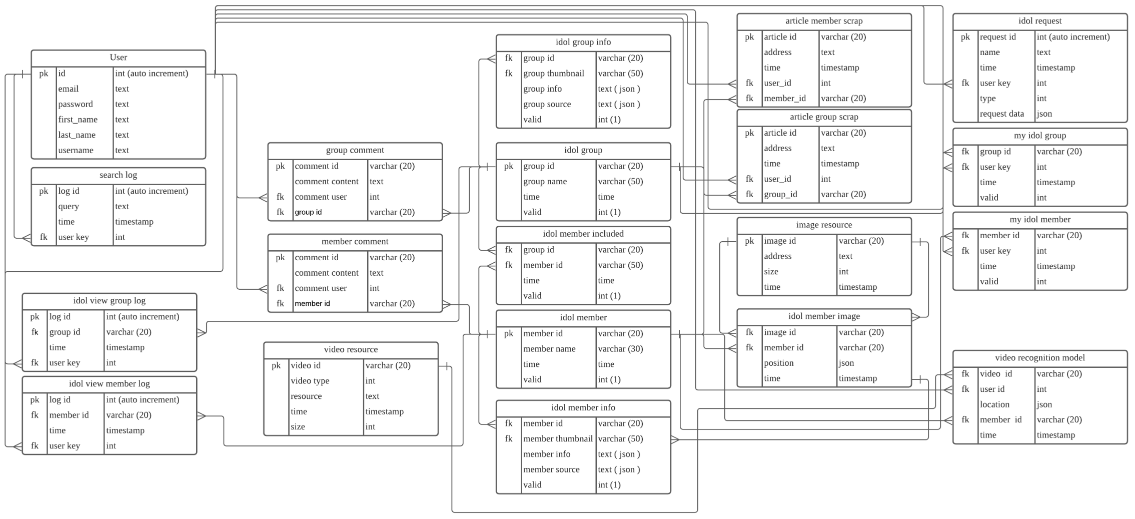
Member

Eunbin Kang, Sohyun Kim, Jiho Park, Youngchae Yoon

System Architecture

Model

Entity-Relationship diagram (E-R Diagram) of out model design is as follows:



User

pk	id	int (auto increment)
	email	text
	password	text
	first_name	text
	last_name	text
	username	text

search log

pk	log id	int (auto increment)
	query	text
	time	timestamp
fk	user key	int

idol view group log

pk	log id	int (auto increment)
fk	group id	varchar (20)
	time	timestamp
fk	user key	int

idol view member log

pk	log id	int (auto increment)
fk	member id	varchar (20)
	time	timestamp
fk	user key	int

group comment

pk	comment id	varchar (20)
	comment content	text
fk	comment user	int
fk	group id	varchar (20)

member comment

pk	comment id	varchar (20)
	comment content	text
fk	comment user	int
fk	member id	varchar (20)

video resource

pk	video id	varchar (20)
	video type	int
	resource	text
	time	timestamp
	size	int

idol group info

fk	group id	varchar (20)
fk	group thumbnail	varchar (50)
	group info	text (json)
	group source	text (json)
	valid	int (1)

idol group

pk	group id	varchar (20)
	group name	varchar (50)
	time	time
	valid	int (1)

idol member included

fk	group id	varchar (20)
fk	member id	varchar (50)
	time	time
	valid	int (1)

idol member

pk	member id	varchar (20)
	member name	varchar (30)
	time	time
	valid	int (1)

idol member info

fk	member id	varchar (20)
fk	member thumbnail	varchar (50)
	member info	text (json)
	member source	text (json)
	valid	int (1)

article member scrap

pk	article id	varchar (20)
	address	text
	time	timestamp
fk	user_id	int
fk	member_id	varchar (20)

article group scrap

pk	article id	varchar (20)
	address	text
	time	timestamp
fk	user_id	int
fk	group_id	varchar (20)

image resource

pk	image id	varchar (20)
	address	text
	size	int
	time	timestamp

idol member image

fk	image id	varchar (20)
fk	member id	varchar (20)
	position	json
	time	timestamp

idol request

pk	request id	int (auto increment)
	name	text
	time	timestamp
fk	user key	int
	type	int
	request data	json

my idol group

fk	group id	varchar (20)
fk	user key	int
	time	timestamp
	valid	int

my idol member

fk	member id	varchar (20)
fk	user key	int
	time	timestamp
	valid	int

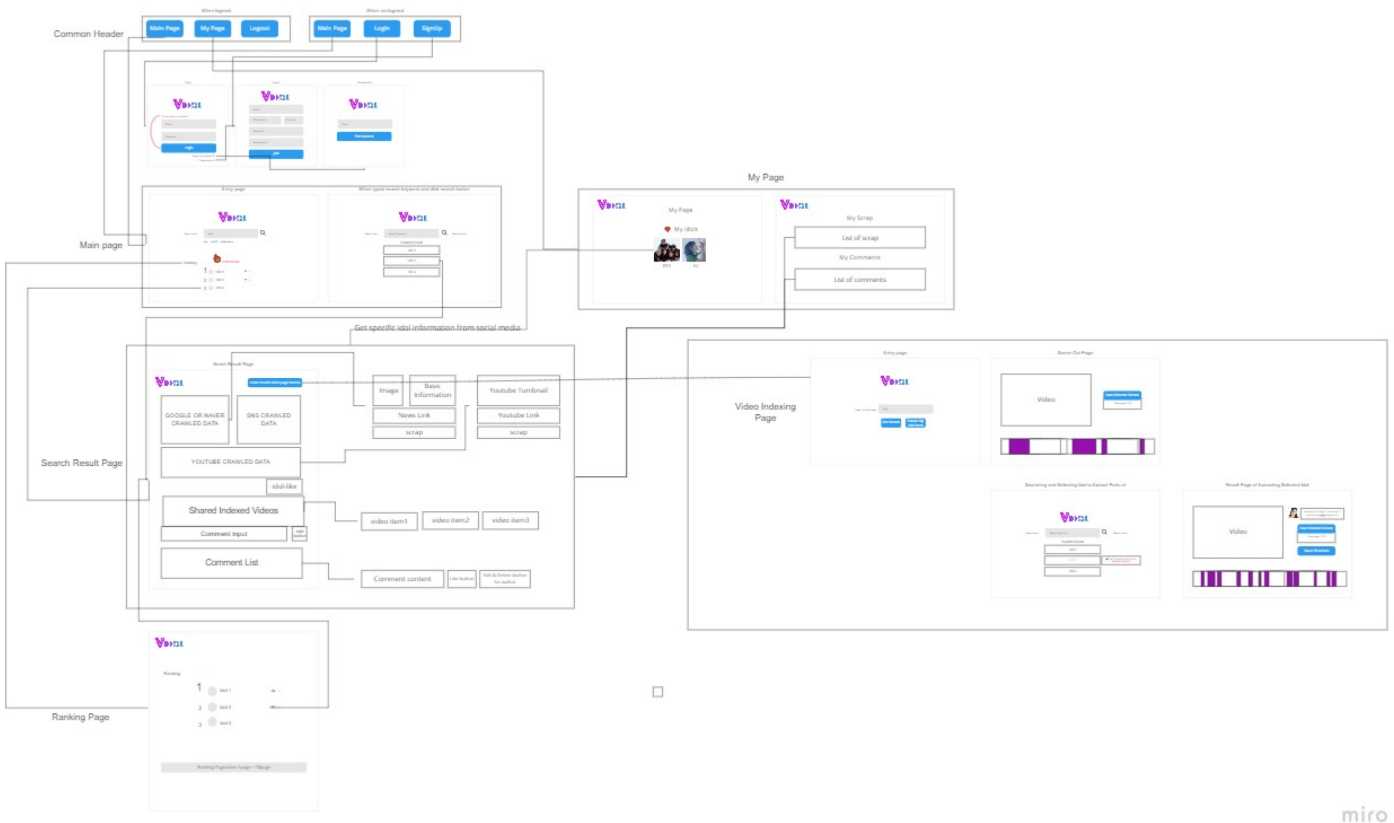
video recognition model

fk	video id	varchar (20)
fk	user id	int
	location	json
fk	member id	varchar (20)
	time	timestamp

Each square means Entity and Entities are connected by a line, which means Relationship. This means that when a line enters an entity in multiple branches, several entities may correspond.

View

User interface for our view design is as follows:



The functionality and requirement for each page are as follows:

1. Login

1-1. Signin Page ('/sign/login')

Field name	Type
signin-email-input	email (regex)
signin-password-input	password (string)
signin-submit-button	Button (form submit)
signin-account-create-button	Button
signin-find-account-button	Button

- User can log in the VIDOL service
- User can type email in `signin-email-input`
- User can type password in `signin-password-input`
- User can submit the form by click `signin-submit-button`
- If user's account information is wrong, remain on the `signin page` and show the error message to the user.
- If user wanted to create new account, click `signin-account-create-button` and redirect to `signup page` .
- If user forgot owned account, click `signin-find-account-button` and redirect to `find account page` .

1-2. Signup Page ('/sign/join')

Field name	Type
signup-email-input	email (regex)
signup-surname-input	name (regex)
signup-given-name-input	name (regex)
signup-password-input	password (string, regex)
signup-re-password-input	password (string, regex)
signup-submit-button	Button (form submit)
signup-login-button	Button
signup-find-account-button	Button

- User can create an account by his/her information
- User can type email in `signup-email-input`
- User can type password in `signup-password-input` and `signup-re-password-input`
- A value of `signup-password-input` and `signup-re-password-input` 's must be same.
- User can type his/her name in `signup-surname-input` and `signup-given-name-input`
- If user forgot owned account, click `signup-find-account-button` and redirect to `find account page` .
- If user have had an account already, click `signup-login-button` and redirect to `signin page` .
- After user click `signup-submit-button` , the form must check the validation of values of inputs.
- If the value of `signup-email-input` is conflict with exist accounts list from database, remain on the `signup page` and show the error message to user.
- If there is no problem in all fields, create the account by the values in the form and redirect to `signin page` .

1-3. Find Account Page ('/sign/findAccount')

Field name	Type
find-account-email-input	email (regex)
find-account-submit-button	Button (form submit)
find-account-login-button	Button
find-account-account-create-button	Button

- User can find his/her account.
- User can type email in `find-account-email-input`
- If the value of email in `find-account-email-input` is invalid through regex format, show the error message to the user.
- User can submit the form by click `find-account-submit-button`
- After click `find-account-submit-button` , check the email value exists in database.
- If exists, send a verification mail to the email address that a value of `find-account-email-input` .
- If not exist, show the error message to the user.
- If user have had an account already, click `find-account-login-button` and redirect to `signin page` .
- If user wanted to create new account, click `find-account-create-button` and redirect to `signup page` .

2. Search & Ranking

2-1. Main Page ('/')

Field name	Type
go-rank-button	Button
search-input	Input
search-button	Button

- User can check the top 10 idol search rankings in `HotRankingList`
- User can click one of the idol in the `HotRankingList` . When user clicks one of the idol in the `HotRankingList` , user is redirected to `Search Result Page('/search/:id')`
- User can click `go-rank-button` button. When user clicks `go-rank-button` button, user is redirected to `Ranking Page('/rank')`
- User can type idol search keyword in `search-input` input and click `search-button` button
- After searching, user can check the search result list
- User can click one of the search result list. When user clicks one one of the search result list, user is redirected to `Search Result Page('/search/:id')`

2-2. Search Result Page ('/search/:id')

Field name	Type
idol-like	Button
scrap	Button

Field name	Type
comment-input	Input
comment-create	Button
comment-edit	Button
comment-delete	Button
go-video-indexing	Button

- User can see crawled information from Internet, SNS, Youtube and shared indexed video
- User can click `idol-like` button
- After clicking `idol-like` button, current idol member or group is saved in My Page
- User can click `scrap` button
- After clicking `scrap` button, corresponding article address is saved in My Page
- User can see comments for the corresponding idol
- If user types content in `comment-input` input and clicks `comment-create` button, a new comment written by the user is posted to the current page
- If user already wrote a comment, the user can click `comment-edit` button
- After clicking `comment-edit` button, corresponding comment becomes editable and user can change comment content.
- After changing comment content, user can click `comment-edit` button and the corresponding comment becomes uneditable
- If user already wrote a comment, the user can click `comment-delete` button
- After clicking `comment-delete` button, delete-comment-confirm pops up
- After clicking "confirm" in delete-comment-confirm, corresponding comment is deleted
- User can click `go-video-indexing` button. When user clicks `go-video-indexing` button, user is redirected to `Video Indexing Page('/video')`

2-3. Ranking Page ('/rank') - Users can check all the idol search rankings - User can click one of the idol in the page. When user clicks one of the idol, user is redirected to `Search Result Page('/search/:id')`

3. Video Indexing

3-1. Entry Page ('/video')

Field name	Type
Video link	YouTube video link (regex)
Cut Scenes Button	Button
Extract my idol parts button	Button

- User can type the link of the video to get indexed.
- If typing nothing or link with wrong format, the buttons are disabled.
- User can move to /video/result by clicking Cut Scenes Button.
- User can move to /video/search by clicking Extract my idol parts button.
- Server starts to get the video from the link and index it.
- If the link is invalid, which means that the format is appropriate but the video is deleted or is private, alert appears and the page does not change.

3-2. Search Idol Page ('/video/search/')

Field name	Type
Search input	text
Search button	Button
Search result button	Button
Request support button	Button

- User can type the name of the idol trying to get parts of.
- If user types nothing, search button is disabled.
- If user clicks search button with input, search result appears under the input box.
- Buttons of idols that we have pretrained model are activated.
- Buttons of idols with no pretrained model are disabled.
- When user clicks idol button activated, the user gets redirected to /video/result.
- Next to idol buttons which are disabled, there is a 'Request support' button.
- When clicking 'Request support' button, request gets sent to the server.
- 'Request support' button gets diasbled when once clicked.

3-3. Scene Cut Page ('/video/result')

Field name	Type
Save Selected Scenes Button	Button
Scene	Button

- Users can see the timeline of the video indexed with scene changes.
- Users can click each scene to include or exclude it in the final video.
- When clicking not selected scene, it gets colored.
- When clicking already selected scene, its background gets removed.
- If no scene is selected, Save Selected Scenes button is disabled.
- When clicking Save Selected Scenes button, the video with selected scenes gets downloaded. User can stay at the page and keep editing.

3-4. Extracting Selected Idol Page ('/video/result')

Field name	Type
Save Selected Scenes Button	Button
Share Timelines Button	Button
Scene	Button

- Users can see the timeline of the video indexed with the selected idol.
- Users can click each scene to include or exclude it in the final video.
- When clicking not selected scene, it gets colored.
- When clicking already selected scene, its background gets removed.
- If no scene is selected, Save Selected Scenes button is disabled.
- When clicking Save Selected Scenes button, the video with selected scenes gets downloaded. User can stay at the page and keep editing.

- When clicking Share Timelines button, the server saves the timeline and matches it to the seleted idol.
- When sharing process is done, confirm button suggesting moving to the search result page of the idol to check the shared timelines appears.

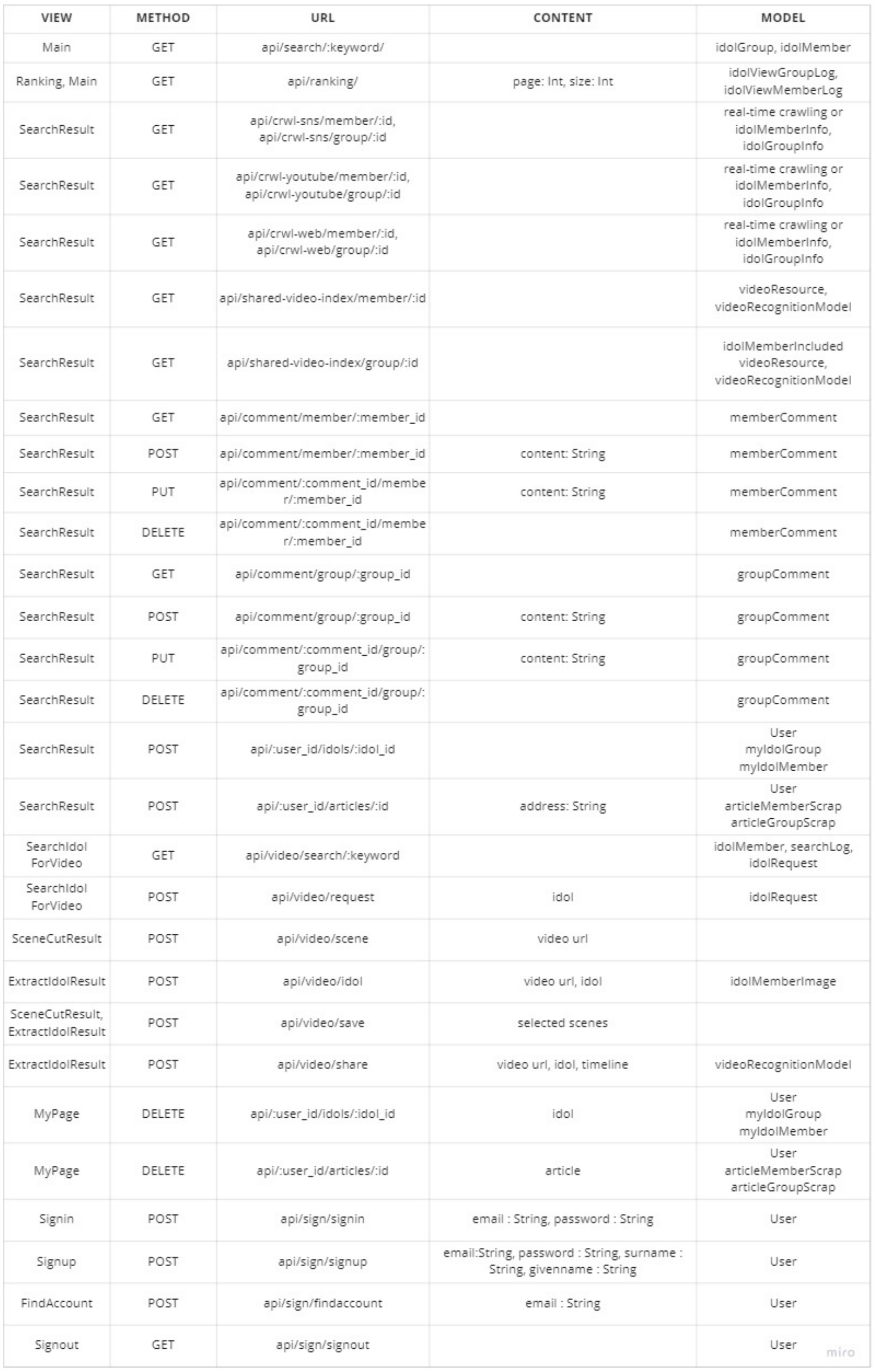
4. My Page

4-1. My Page('mypage/:id')

Field name	Type
go-back	Button
cancel-like	Button
delete-article	Button

- Users can check my activities in `My Page('/mypage/:id')`
- User can go to `Main page('/')` by clicking `go-back`
- Users can see their favorite idols list in `List of my idols`
- When user clicks one of the idol in `List of my idols` , user is redirected to `Search Result Page('/search/:id')`
- When user clicks `cancel-like` button next to idol's name, that idol is removed from `List of my idols` and user redirects to updated page.
- Users can see their scraped articles list in `Scraped articles`
- When user clicks one of the articles in `Scraped articles` , user is redirected to `Search Result Page('/search/:id')` where that article exists
- When user clicks `delete-article` button next to article, that article is removed from `Scraped articles` and user redirects to updated page
- Users can see their comments in `My Comments`
- When user clicks comment's content, user is redirected to `Search Result Page('/search/:id')`

Controller

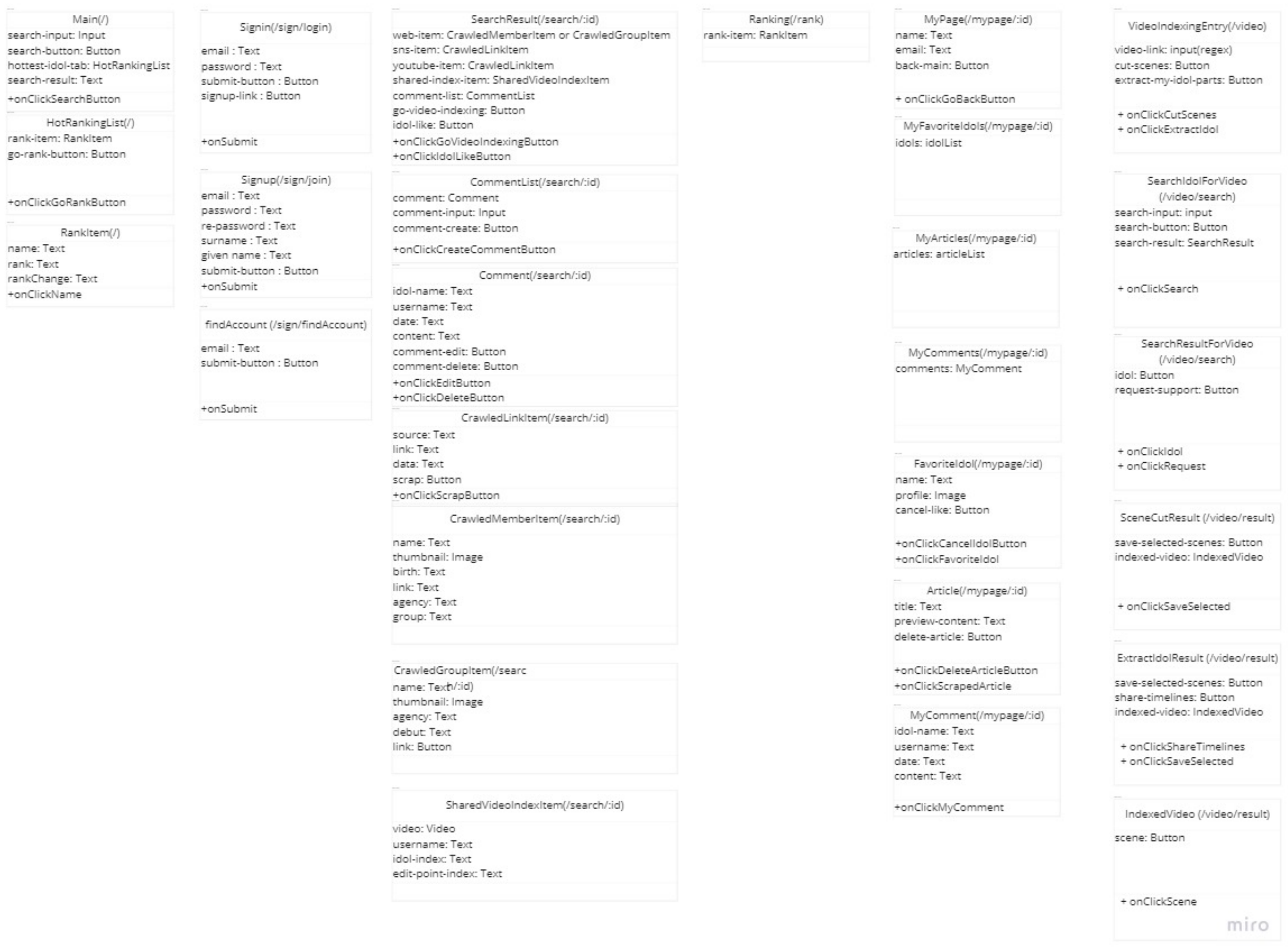


VIEW	METHOD	URL	CONTENT	MODEL
Main	GET	api/search/:keyword/		idolGroup, idolMember
Ranking, Main	GET	api/ranking/	page: Int, size: Int	idolViewGroupLog, idolViewMemberLog
SearchResult	GET	api/crwl-sns/member/:id, api/crwl-sns/group/:id		real-time crawling or idolMemberInfo, idolGroupInfo
SearchResult	GET	api/crwl-youtube/member/:id, api/crwl-youtube/group/:id		real-time crawling or idolMemberInfo, idolGroupInfo
SearchResult	GET	api/crwl-web/member/:id, api/crwl-web/group/:id		real-time crawling or idolMemberInfo, idolGroupInfo
SearchResult	GET	api/shared-video-index/member/:id		videoResource, videoRecognitionModel
SearchResult	GET	api/shared-video-index/group/:id		idolMemberIncluded videoResource, videoRecognitionModel
SearchResult	GET	api/comment/member/:member_id		memberComment
SearchResult	POST	api/comment/member/:member_id	content: String	memberComment
SearchResult	PUT	api/comment/:comment_id/member/:member_id	content: String	memberComment
SearchResult	DELETE	api/comment/:comment_id/member/:member_id		memberComment
SearchResult	GET	api/comment/group/:group_id		groupComment
SearchResult	POST	api/comment/group/:group_id	content: String	groupComment
SearchResult	PUT	api/comment/:comment_id/group/:group_id	content: String	groupComment
SearchResult	DELETE	api/comment/:comment_id/group/:group_id		groupComment
SearchResult	POST	api/:user_id/idols/:idol_id		User myIdolGroup myIdolMember
SearchResult	POST	api/:user_id/articles/:id	address: String	User articleMemberScrap articleGroupScrap
SearchIdol ForVideo	GET	api/video/search/:keyword		idolMember, searchLog, idolRequest
SearchIdol ForVideo	POST	api/video/request	idol	idolRequest
SceneCutResult	POST	api/video/scene	video url	
ExtractIdolResult	POST	api/video/idol	video url, idol	idolMemberImage
SceneCutResult, ExtractIdolResult	POST	api/video/save	selected scenes	
ExtractIdolResult	POST	api/video/share	video url, idol, timeline	videoRecognitionModel
MyPage	DELETE	api/:user_id/idols/:idol_id	idol	User myIdolGroup myIdolMember
MyPage	DELETE	api/:user_id/articles/:id	article	User articleMemberScrap articleGroupScrap
Signin	POST	api/sign/signin	email : String, password : String	User
Signup	POST	api/sign/signup	email:String, password : String, surname : String, givenname : String	User
FindAccount	POST	api/sign/findaccount	email : String	User
Signout	GET	api/sign/signout		User

Design Details

Frontend Components

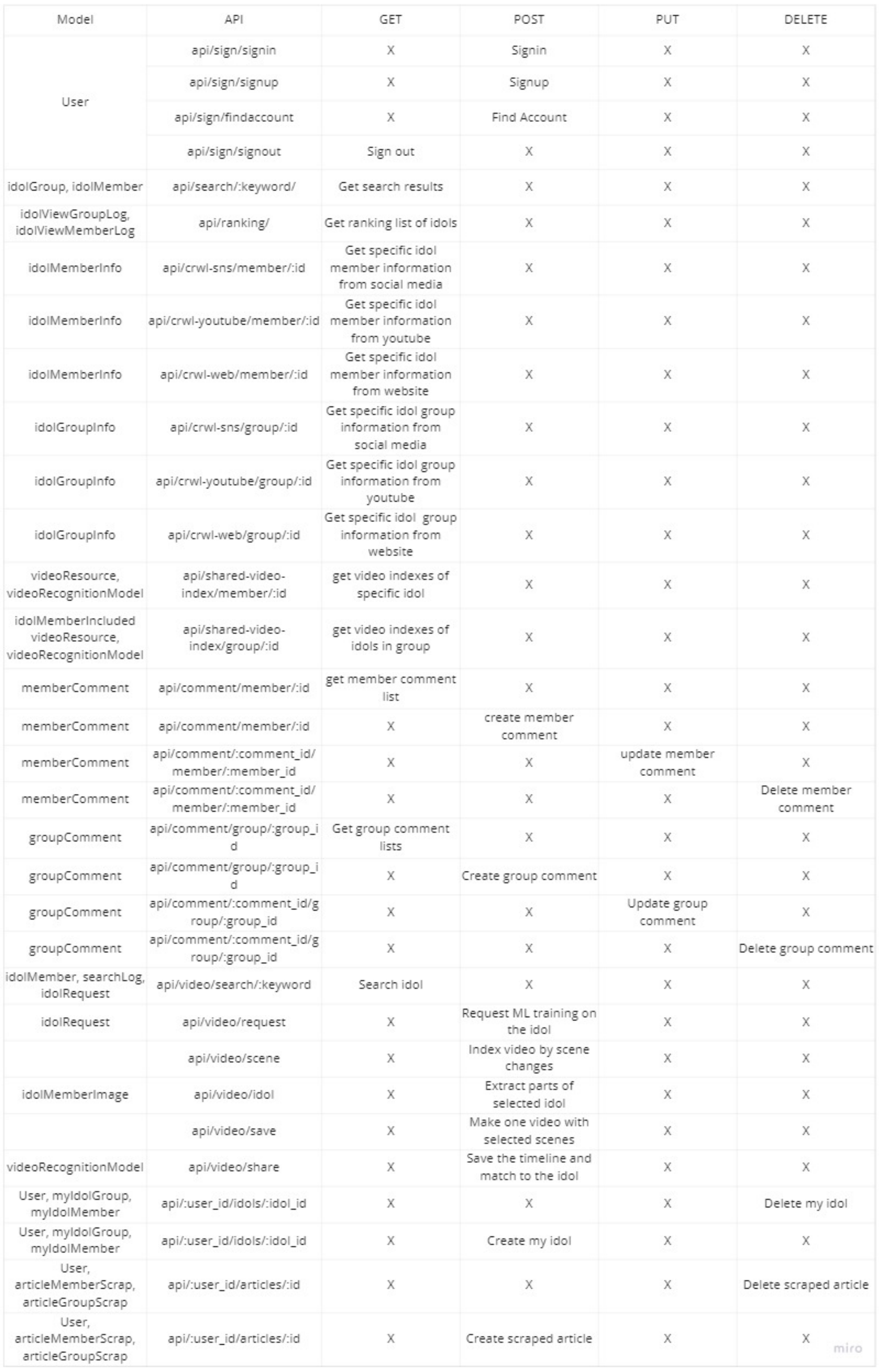
Tables below are the frontend components. The attributes and the methods of each component are listed in each box.



Frontend Algorithms

Algorithms required for implementation are written below, based on their component. 1. Main - `onClickSearchButton()` : Call backend API(GET /api/search/:keyword) 2. HotRankingList - `onClickGoRankButton()` : Redirect to `Ranking Page ('/rank')` 3. RankItem - `onClickName()` : Redirect to `Search Result Page ('/search/:id')` 4. SearchResult - `onClickGoVideoIndexingButton()` : Redirect to `Video Indexing Page ('/video')` - `onClickIdolLikeButton()` : Call backend API(POST /:user_id/idols/:idol_id) 5. CommentList - `onClickCreateCommentButton()` : Call backend API(POST /api/comment/member/:member_id or POST /api/comment/group/:group_id) 6. Comment - `onClickEditButton()` : If comment is not editable, make comment editable. If comment is editable, call backend API(PUT /api/comment/:comment_id/member/:member_id or PUT /api/comment/:comment_id/group/:group_id) - `onClickDeleteButton()` : Pops up delete-comment-confirm and if user clicks "confirm", call backend API(DELETE /api/comment/:comment_id/member/:member_id or DELETE /api/comment/:comment_id/group/:group_id) 7. CrawledLinkItem - `onClickScrapButton()` : Call backend API(POST /:user_id/articles/:article_id) 8. MyPage - `onClickBackButton()` : Redirect to `Main Page('/')` 9. FavoriteIdol - `onClickCancelIdolButton()` : Call backend API(DELETE /:user_id/idols/:idol_id) and redirect to `My Page(/mypage/:id)` - `onClickFavoriteIdol()` : Redirect to `Search Result Page('/search/:id')` of selected Idol. 10. Article - `onClickDeleteArticleButton()` : Call backend API(DELETE /:user_id/articles/:article_id) and redirect to `My Page(/mypage/:id)` - `onClickScrapedArticle()` : Redirect to `Search Result Page('/search/:id')` of selected article. 11. MyComment - `onClickMyComment()` : Redirect to `Search Result Page('/search/:id')` where my comment at. 12. VideoIndexingEntry - `onClickCutScenes` : Call backend API (POST /video/scene) and redirect to Scene Cut Result page (/video/result) when getting response. - `onClickExtractIdol` : Redirect to Search Idol page for video indexing (/video/search) 13. SearchIdolForVideo - `onClickSearch` : Call backend API (GET /video/search/:keyword) and update search result. 14. SearchResultForVideo - `onClickIdol` : Call backend API (POST /video/idol) and redirect to Extract Idol Result page (/video/result) when getting response. - `onClickRequest` : Call backend API (POST /video/request). 15. SceneCutResult - `onClickSaveSelected` : Call backend API (POST /video/save). 16. ExtractIdolResult - `onClickShareTimelines` : Call backend API (POST /video/share). - `onClickSaveSelected` : Call backend API (POST /video/save). 17. IndexedVideo: - `onClickScene` : Add the scene to selected scenes if the scene is not selected. Remove the scene from selected scenes if it is already seleted. In both cases, play the scene.

Backend Design

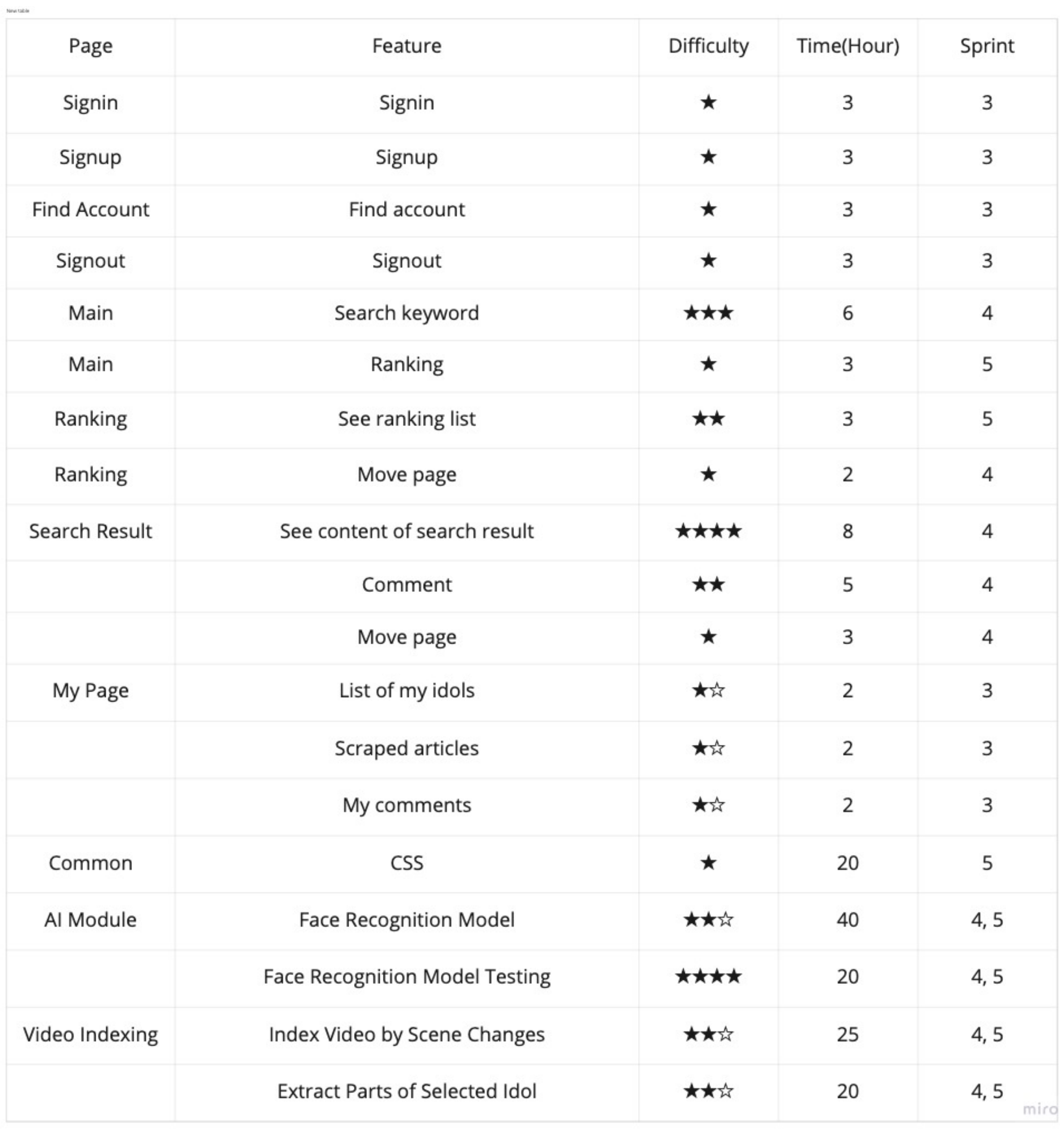


Model	API	GET	POST	PUT	DELETE
User	api/sign/signin	X	Signin	X	X
	api/sign/signup	X	Signup	X	X
	api/sign/findaccount	X	Find Account	X	X
	api/sign/signout	Sign out	X	X	X
idolGroup, idolMember	api/search/:keyword/	Get search results	X	X	X
idolViewGroupLog, idolViewMemberLog	api/ranking/	Get ranking list of idols	X	X	X
idolMemberInfo	api/crwl-sns/member/:id	Get specific idol member information from social media	X	X	X
idolMemberInfo	api/crwl-youtube/member/:id	Get specific idol member information from youtube	X	X	X
idolMemberInfo	api/crwl-web/member/:id	Get specific idol member information from website	X	X	X
idolGroupInfo	api/crwl-sns/group/:id	Get specific idol group information from social media	X	X	X
idolGroupInfo	api/crwl-youtube/group/:id	Get specific idol group information from youtube	X	X	X
idolGroupInfo	api/crwl-web/group/:id	Get specific idol group information from website	X	X	X
videoResource, videoRecognitionModel	api/shared-video-index/member/:id	get video indexes of specific idol	X	X	X
idolMemberIncluded videoResource, videoRecognitionModel	api/shared-video-index/group/:id	get video indexes of idols in group	X	X	X
memberComment	api/comment/member/:id	get member comment list	X	X	X
memberComment	api/comment/member/:id	X	create member comment	X	X
memberComment	api/comment/:comment_id/member/:member_id	X	X	update member comment	X
memberComment	api/comment/:comment_id/member/:member_id	X	X	X	Delete member comment
groupComment	api/comment/group/:group_id	Get group comment lists	X	X	X
groupComment	api/comment/group/:group_id	X	Create group comment	X	X
groupComment	api/comment/:comment_id/group/:group_id	X	X	Update group comment	X
groupComment	api/comment/:comment_id/group/:group_id	X	X	X	Delete group comment
idolMember, searchLog, idolRequest	api/video/search/:keyword	Search idol	X	X	X
idolRequest	api/video/request	X	Request ML training on the idol	X	X
	api/video/scene	X	Index video by scene changes	X	X
idolMemberImage	api/video/idol	X	Extract parts of selected idol	X	X
	api/video/save	X	Make one video with selected scenes	X	X
videoRecognitionModel	api/video/share	X	Save the timeline and match to the idol	X	X
User, myIdolGroup, myIdolMember	api/:user_id/idols/:idol_id	X	X	X	Delete my idol
User, myIdolGroup, myIdolMember	api/:user_id/idols/:idol_id	X	Create my idol	X	X
User, articleMemberScrap, articleGroupScrap	api/:user_id/articles/:id	X	X	X	Delete scraped article
User, articleMemberScrap, articleGroupScrap	api/:user_id/articles/:id	X	Create scraped article	X	X

Implementation Plan

We are going to make planned outcomes for each sprint. There are several features that we should implement. First, we are going to work on basic features. At sprint 2, we focused on designing and planning for our service such as Controllers, Models, and Backend designs. Moreover, we are going to set up skeleton code for our web application by react and Django. At sprint 3, we planned that implement about users' authentication function and view, and users' information page (my page). AI module that could be working face recognition and indexing video by scene change is hard and essential part of our project, so we are going to prepare data for training our AI models. At sprint 4, we implement Main page and Search result. At the end of sprint4, face recognition model and indexing video model will be implemented simply and begin testing. At sprint 5, idols' ranking algorithm will be applied our service. Finally inspecting for all code used in our project will be done.

Testing is an important step for our project, so unit tests and integration tests are required. Implementation plan is organized in table below, you can check difficulties of each part and when we implemented.


New table

Page	Feature	Difficulty	Time(Hour)	Sprint
Signin	Signin	★	3	3
Signup	Signup	★	3	3
Find Account	Find account	★	3	3
Signout	Signout	★	3	3
Main	Search keyword	★★★	6	4
Main	Ranking	★	3	5
Ranking	See ranking list	★★	3	5
Ranking	Move page	★	2	4
Search Result	See content of search result	★★★★	8	4
	Comment	★★	5	4
	Move page	★	3	4
My Page	List of my idols	★☆	2	3
	Scraped articles	★☆	2	3
	My comments	★☆	2	3
Common	CSS	★	20	5
AI Module	Face Recognition Model	★★☆	40	4, 5
	Face Recognition Model Testing	★★★★	20	4, 5
Video Indexing	Index Video by Scene Changes	★★☆	25	4, 5
	Extract Parts of Selected Idol	★★☆	20	4, 5

Testing Plan

All tests will be conducted automatically with the testing frameworks specified below. This strategy will ensure more robust and safer codes along the entire project sprints.

Unit Testing Every component and module needs to be unit tested before being commited. Code coverage will be kept over 90%. External components(database, JSON requests, etc.) will be mocked in the unit test.

Frontend (React): Jest, Enzyme

Backend (Django): Python Unit Test

Functional Testing Every API will be tested with Jest/Enzyme and Python Unit Test. Our models will be mocked for API tests.

Frontend (React): Jest, Enzyme

Backend (Django): Python Unit Test

Acceptance & Integration Testing We will use Travis CI for the continuous integration and SonarCloud for code quality static analysis.