Seenit

by
Chaz Chang
Viet Dinh
My Nguyen

TEAM 24 - CS157A

Project Requirements

Project Description

Overview:

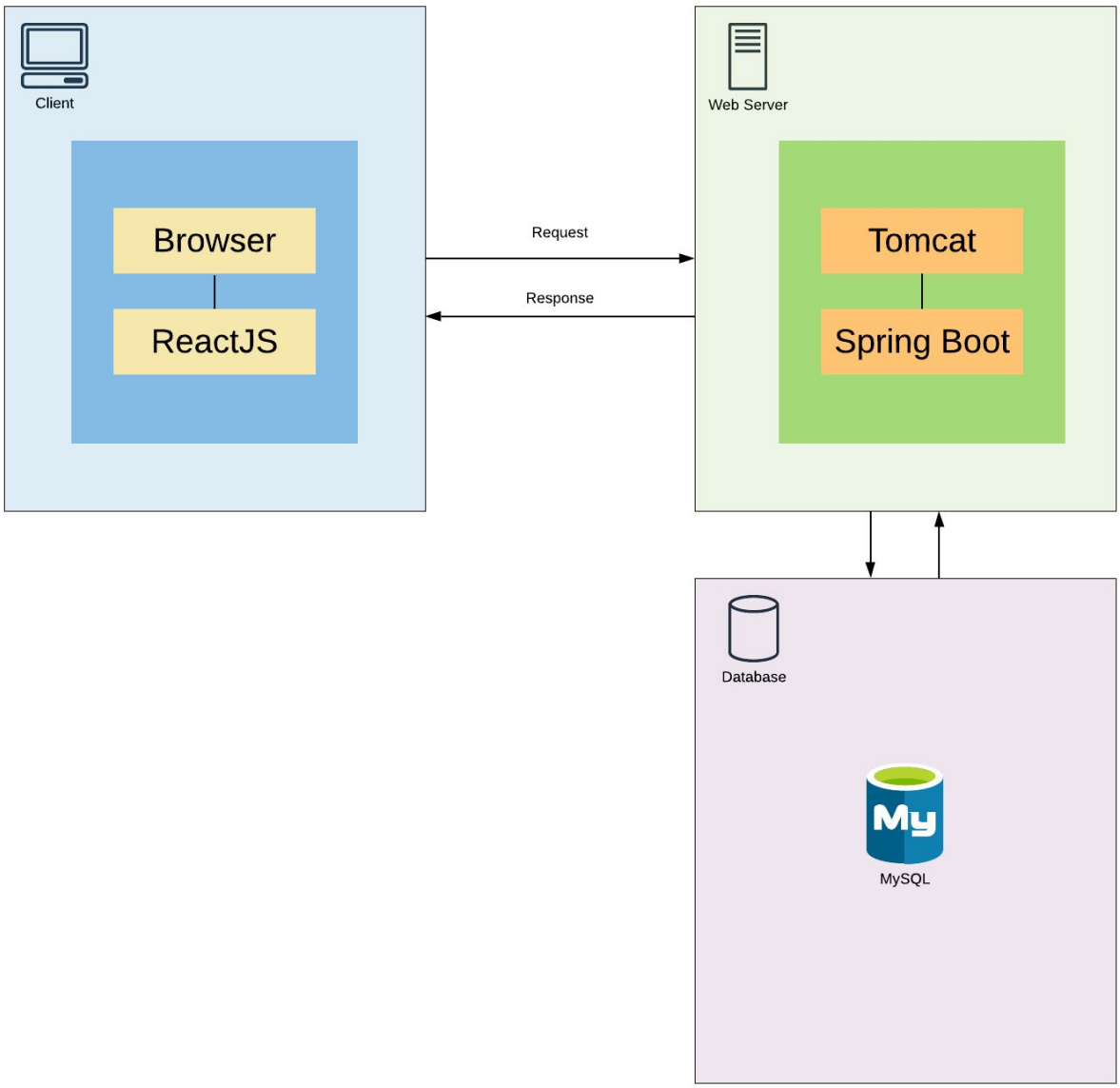
The goal of this application is to create a rating site and discussion website. Users can be able to search and sort through posts. Also, users can create posts, comment on them, upvote, downvote, and save posts. In addition, users can join a channel and create channels that will function as communities. Users can view posts, view channels detail, view all channel, search and sort posts without logging in. However, to create posts, comment on them, vote and save posts, users are required to login. Moreover, users have to log in to view profile dashboard, delete account and update account. This application will allow people to easily voice their opinion or share a story. People can easily ask questions and others can answer them. It will also provide a user-friendly interface and fast response time while maintaining high throughput.

Who would be interested in this application?

Users could be anyone who is interested in forming a community online, and interact with people with similar interests. Users can keep up to date with the daily life of others in their community. Companies and organizations can use Seenit to communicate with people and market products.

System Environment

Structure of the system (graph based on 3-tiered architecture):



HW/SW used:

- ReactJS
- Spring Boot
- Windows/macOS
- MySQL Workbench
- Visual Studio Code
- Slack
- Github

RDBMS Used:

- MySQL Version 5.6

Application languages:

- Java (with Spring Boot framework)
- SQL (with MySQL)
- JavaScript (with ReactJS)
- HTML/CSS

Functional Requirements

How users will interact with the system:

The application provides the same functionality for all the users. Internet connection is required for any activity on the system. A user shall be able to access the system for read and write. A user can access the system home page freely. They shall be able to read daily news and posts from various websites without logging in. Users can also be able to make a search based on their particular interests. However, users must log-in into the system and have their credentials checked for other activities such as make a new post, comment on a post, share a post, and join another channel. First-time users must register for an account to perform the activities listed under log-in.

Describe each individual function, functional process and I/O:

I. WITHOUT LOGGING IN, USERS CAN PERFORM THESE FUNCTIONS

Search

- The users shall be able to search for posts by post's title or its content. All the posts will be displayed when its title or content contained the words in the search box.
- If there is no matching results, nothing will be displayed.

Sort

- Using to sort the posts by Top, New or None
- Sort by Top will sort the posts by the upvotes they have.
- Sort by New will sort the posts by date in descending order.
- Sort by None means the result is random based on the data return from the endpoint.
- User can active the function simply by clicking the sort button and selecting from the drop down menu

View a post

- By clicking on any post, users shall be able to see its contents and all the comments associated with it.

View a channel details

- By clicking on a channel under Top channel, users shall be able to see all the posts on this channel. Especially, users can see the moderator and channel details which include the name of the channel and the number of members.

II. SIGN IN and SIGN UP FUNCTIONS

Log-in

- Returning users must sign-in to perform the following activities:
    - view profile dashboard
    - Create a post
    - Join a channel
    - comment on a post
    - save a post
    - vote on a post
- To log in, users must enter their username or email and their password.
- The system shall check the users' credentials. If the entered information is not matching with the information stored in database, an error message will be displayed. Otherwise, the system gives users access when the information is matching.

Register for an account

- There will be only one type of user; no admin account in this application. First time user must sign up for an account to perform those activities listed in log-in function
- There will be a redirect link. User will be asked to fill out some information such as username, email address and password.
- The system shall check if entered username is already existed. The system shall keep asking the users to enter a username until the name is available. Then, the system shall add the users' personal information into database.
- The system shall check if entered email is already existed

III. LOG-IN IS REQUIRED FOR THESE FUNCTIONS

View profile dashboard

- Users can see all the posts and comments they have made.
- They can also check the upvoted and downvoted posts as well as their saved posts.
- Any change in the database, for instance, comments added to a post and upvoted or downvoted are made, will be updated in the profile dashboard.

Join a channel

- Without logging in, users will get a message saying that they need to log in to join a channel.

- The users shall be able to join a channel that they are interested in.
- This function is important for creating a post.
- The system shall add the changes to database to keep track of the channels that a user joined.

Create a post

- The user must join at least one channel to be able to create a post.
- Title and content of the post are required before users hit the submit button. The created posts will be visible on the chosen channel.
- The system shall save all the contents in the database after posting is finished.

Comment on a post

- Without logging in, users will get a message saying that they need to log in first to comment on a post.
- The users shall be able to leave multiple comments on someone else's posts. Type what they want to say in the reply box, and hit post comment.
- Once a comment is submitted , the data will be saved in database. Users can check the comments by viewing that post. They can also check all comments that they have made on the profile dashboard under Comments.

Save a post

- Without logging in, users will get a message saying that they need to log in first to save any post.
- Otherwise, they can be able to save that post if they are interested in a particular post. The saved posts will be automatically added to their profile.
- By saving a post, users can always go to their profile to continue reading or comment on the posts.

Upvote

- On a post, users can click on the up arrow to upvote the post.
- This will increment the posts's points
- Then the post will be visible in their profile dashboard

Downvote

- On a post, users can click on the down arrow to downvote the post
- This will decrement the posts's points
- Then the post will be visible in their profile dashboard

Non-functional Issues

Graphical User Interface (GUI):

There are many design principles when it comes to web design. For our website, we will use seven most popular principles, which are Visual Hierarchy, Divine Proportions,

Hick's Law, Fitt's Law, Rule of Thirds, Gestalt Design Laws, and White Space and Clean Design.

- Visual Hierarchy: Certain parts of our website will be more important than others. We want to make those parts easily been seen and noticed by users. For example the account button, the scrolling posts, the filters, and the search box.
- Divine Proportions: The layout, the size of each components should follow the golden ratio which is 1.618. For example, if the layout width is 1200px, the width of the content area should be 742px.
- Hick's Law: "Hick's Law says that with every additional choice increases the time required to take a decision." So, we plan to minimize the options for dropdown menu buttons. This will encourage new users trying new functions.
- Fitt't Law: Button's size needs to follow a set of rules. The size of the button is proportion to its using-frequency.
- Rule of Thirds: Since our website will allow users to upload pictures. The size of a picture needs to follow the rule of thirds to make it more interesting.
- Gestalt Design Laws: Filter buttons, sorting buttons will be grouped together. Buttons will have consistent sizes.
- White Space and Clean Design: Website without white/blank space is hard to navigate. So, we will use white space to divide the components, boxes that have different functions.

Security

- User accounts need to be highly protected. We don't want their personal information leaked or hacked.
- Passwords are not stored in blank text. During user registration, passwords are hashed and salted using BCrypt before storing in the database. During login, the hashed passwords are compared
- Sessions are handled with JWT (JSON Web Tokens). When users log in, they are given a JWT to send for future requests. The backend uses the JWT Signature to see if the JWT is valid. The JWT Signature makes it more difficult for a malicious user to pretend to be some other user.

Access Control

- Anyone with internet can access the website.
- A user will be able to view posts/channels and search without logging in
- A user must login to create posts, comments, or channels.
- A user cannot edit posts or comments that belong to a different user

Performance

- Fast response time while maintaining high throughput

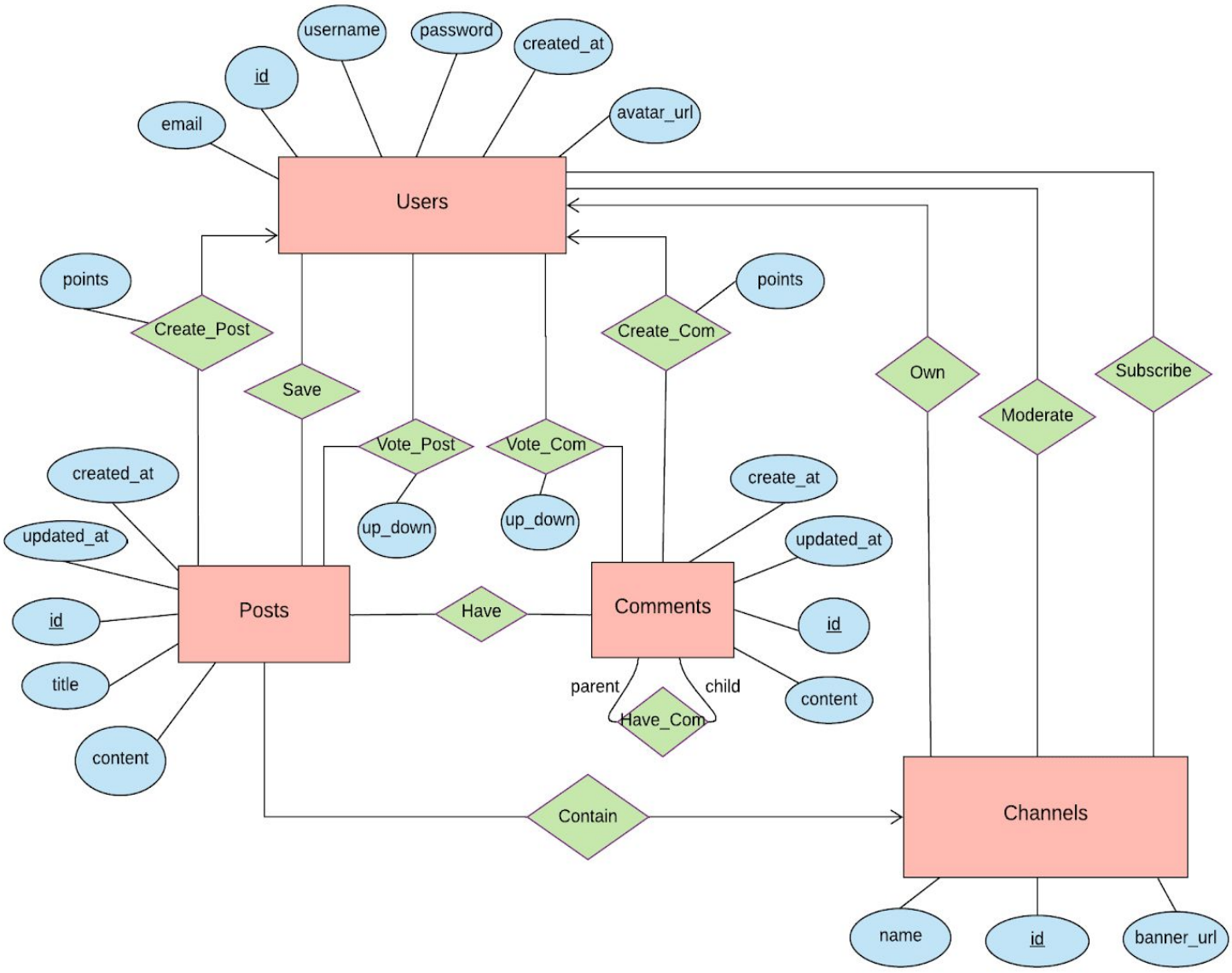
- The MySQL database will be optimized so queries don't take too long. The right data types and efficient SQL queries will make the database accesses faster and the database size smaller.
- ReactJS will be used to build the User Interface. ReactJS will allow the user to navigate through the application quickly by dynamically changing the current page instead of loading a whole new page from the server.

Scalability

- Able to add new functions and features while developing the app.

Project Design

Updated ERD Model



List all completely non-trivial FDs that apply to your design.

Users: id → username, password, email, created_at, avatar_url
username → id, password, email, created_at, avatar_url
email → id, username, password, created_at, avatar_url
Posts: id → title, content, created_at, updated_at
Comments: id → content, created_at, updated_at
Channels: id → name, banner_url
Create_Post: post_id → user_id, points
Create_Com: com_id → user_id, points
Contain: post_id → channel_id
Own: channel_id → user_id

Convert ER to schemas

Entity sets:

1. Users(id, email, username, password, created_at, avartar_url)
2. Posts(id, title, content, created_at, updated_at)
3. Comments(id, content, created_at, updated_at)
4. Channels(id, name, banner_url)

Relationships:

1. Create_Post(user_id, post_id, points)
2. Create_Com(user_id, com_id,points)
3. Have(post_id, com_id)
4. Have_Com(parent_id, child_id)
5. Save(user_id, post_id)
6. Vote_Post(user_id, post_id, up_down)
7. Vote_Com(user_id, com_id, up_down)
8. Contain(post_id, channel_id)
9. Own(user_id, channel_id)
10. Moderate(user_id, channel_id)
11. Subscribe(user_id, channel_id)

Explanation for each entity set and relationship, write a short description in plain English of what it represents or models

Entity sets:

1. Users: has ID number (PK), email, username, password, points, created_at, and avatar_url (image url). Stores user data
2. Posts: has ID number (unique), title, content, created_at, and updated_at. Stores Post data

3. Comments: has ID (PK), content, created_at, and updated_at. Stores comment data in a Post.
4. Channels: has ID (PK), name, and banner_url(image banner). Channel is also represented by unique id. Stores channel data

Relationships:

1. Own: has a channel_id and user_id. Stores who owns which channels
2. Moderate: has a channel_id and user_id. Stores who moderates which channels
3. Subscribe: has a channel_id and user_id. Stores who subscribes to which channels
4. Contain: has a channel_id and post_id. Stores which channels contain which posts
5. Create_Post: has post_id, user_id, and points. Stores which posts are created by which users. Also stores the number of points a post has.
6. Create_Com: has comment_id, user_id, and points. Stores which comments are created by which users. Also stores the number of points a comment has.
7. Vote_Post: has post_id, user_id, and up_down. Stores which posts are voted up or down by which users
8. Vote_Com: has comment_id, user_id, and up_down. Stores which comments are voted up or down by which users
9. Save: has user_id and post_id. Users can save a post so that they can read the post again later.
10. Have: has post_id (parent) and comment_id (child). Stores which posts have which comment replies
11. Have_Com: has comment_id1 (parent) and comment_id2 (child). Each comment might have some other comments. This table stores which comments have which comment replies.

Perform the normalization process, and perfect the relational database schemas to BCNF

The schemas are already in BCNF. For all relations, the left-hand side of all projected functional dependencies are superkeys

Format: <relation>: <projected functional dependencies>

Entity sets:

1. Users(id, email, username, password, created_at, avartar_url):
   id → username, password, email, created_at, avatar_url;
   username → id, password, email, created_at, avatar_url;
   email → id, username, password, created_at, avatar_url

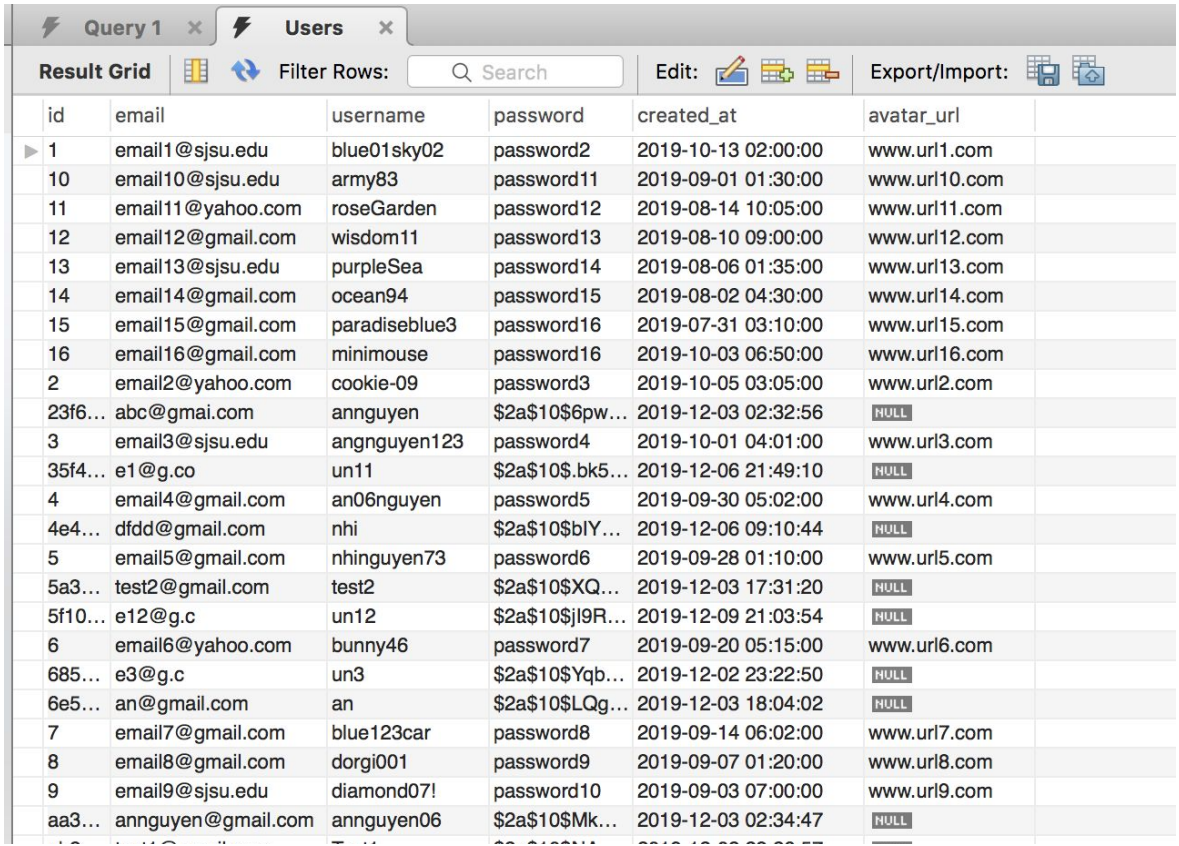
2. Posts(id, title, content, created_at, updated_at): id → title, content, created_at, updated_at
3. Comments(id, content, created_at, updated_at): id → content, created_at, updated_at
4. Channels(id, name, banner_url): id → name, banner_url

Relationships:

1. Create_Post(user_id, post_id, points): post_id → user_id, points
2. Create_Com(user_id, com_id,points): com_id → user_id, points
3. Have(post_id, com_id): none
4. Have_Com(parent_id, child_id): none
5. Save(user_id, post_id): none
6. Vote_Post(user_id, post_id, up_down): none
7. Vote_Com(user_id, com_id, up_down): none
8. Contain(post_id, channel_id): post_id → channel_id
9. Own(user_id, channel_id): channel_id → user_id
10. Moderate(user_id, channel_id): none
11. Subscribe(user_id, channel_id): none

Create and show at least 10 tables according to schemas and model the data stored in the database (Each table must contain at least 15 tuple instances.)

1. User table

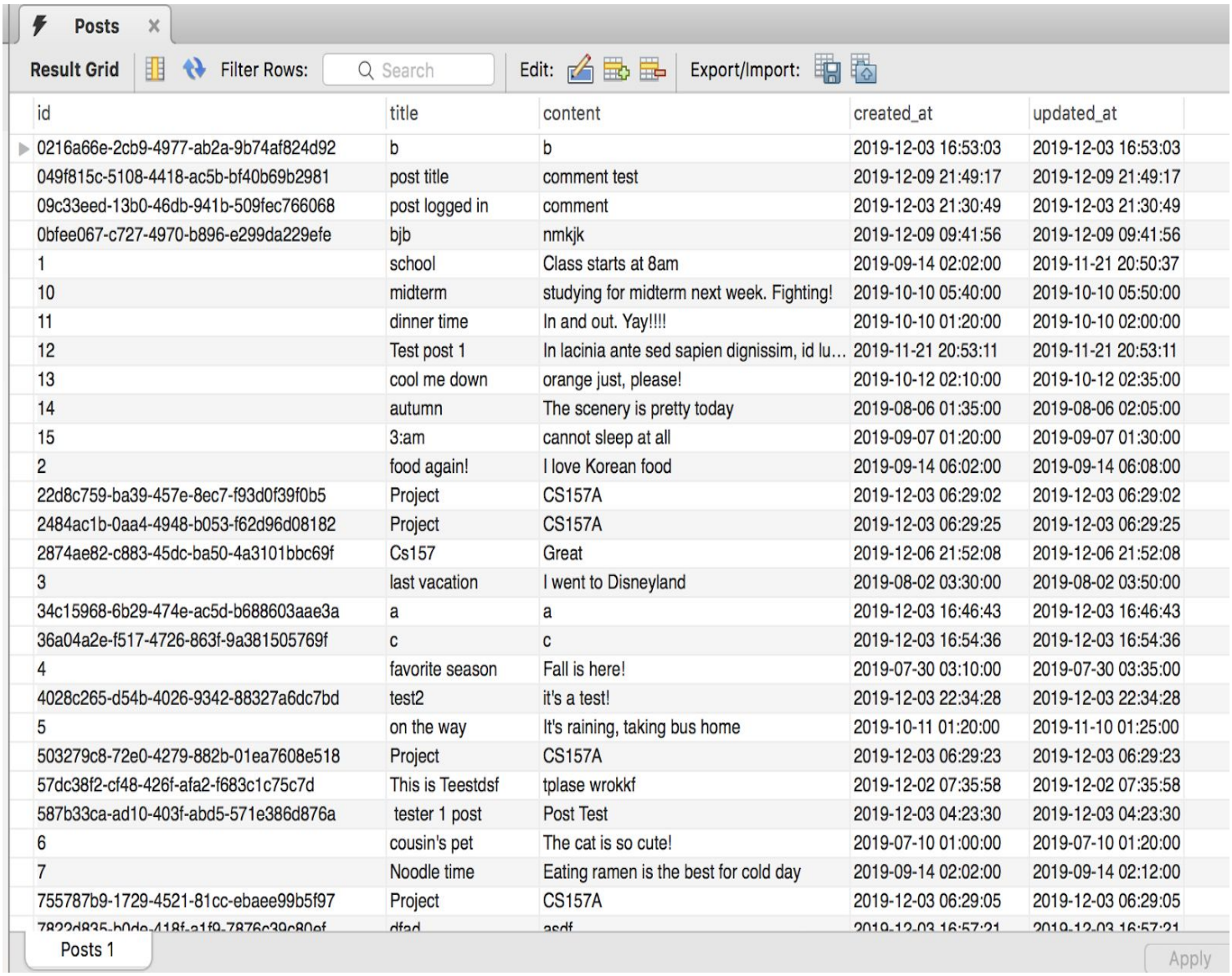


Query 1 | Users

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	email	username	password	created_at	avatar_url
1	email1@sjsu.edu	blue01sky02	password2	2019-10-13 02:00:00	www.url1.com
10	email10@sjsu.edu	army83	password11	2019-09-01 01:30:00	www.url10.com
11	email11@yahoo.com	roseGarden	password12	2019-08-14 10:05:00	www.url11.com
12	email12@gmail.com	wisdom11	password13	2019-08-10 09:00:00	www.url12.com
13	email13@sjsu.edu	purpleSea	password14	2019-08-06 01:35:00	www.url13.com
14	email14@gmail.com	ocean94	password15	2019-08-02 04:30:00	www.url14.com
15	email15@gmail.com	paradiseblue3	password16	2019-07-31 03:10:00	www.url15.com
16	email16@gmail.com	minimouse	password16	2019-10-03 06:50:00	www.url16.com
2	email2@yahoo.com	cookie-09	password3	2019-10-05 03:05:00	www.url2.com
23f6…	abc@gmai.com	annguyen	$2a$10$6pw…	2019-12-03 02:32:56	NULL
3	email3@sjsu.edu	angnguyen123	password4	2019-10-01 04:01:00	www.url3.com
35f4…	e1@g.co	un11	$2a$10$.bk5…	2019-12-06 21:49:10	NULL
4	email4@gmail.com	an06nguyen	password5	2019-09-30 05:02:00	www.url4.com
4e4…	dfdd@gmail.com	nhi	$2a$10$blY…	2019-12-06 09:10:44	NULL
5	email5@gmail.com	nhinguyen73	password6	2019-09-28 01:10:00	www.url5.com
5a3…	test2@gmail.com	test2	$2a$10$XQ…	2019-12-03 17:31:20	NULL
5f10…	e12@g.c	un12	$2a$10$jl9R…	2019-12-09 21:03:54	NULL
6	email6@yahoo.com	bunny46	password7	2019-09-20 05:15:00	www.url6.com
685…	e3@g.c	un3	$2a$10$Yqb…	2019-12-02 23:22:50	NULL
6e5…	an@gmail.com	an	$2a$10$LQg…	2019-12-03 18:04:02	NULL
7	email7@gmail.com	blue123car	password8	2019-09-14 06:02:00	www.url7.com
8	email8@gmail.com	dorgi001	password9	2019-09-07 01:20:00	www.url8.com
9	email9@sjsu.edu	diamond07!	password10	2019-09-03 07:00:00	www.url9.com
aa3…	annguyen@gmail.com	annguyen06	$2a$10$Mk…	2019-12-03 02:34:47	NULL

2. Posts table



Posts

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	title	content	created_at	updated_at
0216a66e-2cb9-4977-ab2a-9b74af824d92	b	b	2019-12-03 16:53:03	2019-12-03 16:53:03
049f815c-5108-4418-ac5b-bf40b69b2981	post title	comment test	2019-12-09 21:49:17	2019-12-09 21:49:17
09c33eed-13b0-46db-941b-509fec766068	post logged in	comment	2019-12-03 21:30:49	2019-12-03 21:30:49
0bfee067-c727-4970-b896-e299da229efe	bjb	nmkjk	2019-12-09 09:41:56	2019-12-09 09:41:56
1	school	Class starts at 8am	2019-09-14 02:02:00	2019-11-21 20:50:37
10	midterm	studying for midterm next week. Fighting!	2019-10-10 05:40:00	2019-10-10 05:50:00
11	dinner time	In and out. Yay!!!!	2019-10-10 01:20:00	2019-10-10 02:00:00
12	Test post 1	In lacinia ante sed sapien dignissim, id lu...	2019-11-21 20:53:11	2019-11-21 20:53:11
13	cool me down	orange just, please!	2019-10-12 02:10:00	2019-10-12 02:35:00
14	autumn	The scenery is pretty today	2019-08-06 01:35:00	2019-08-06 02:05:00
15	3:am	cannot sleep at all	2019-09-07 01:20:00	2019-09-07 01:30:00
2	food again!	I love Korean food	2019-09-14 06:02:00	2019-09-14 06:08:00
22d8c759-ba39-457e-8ec7-f93d0f39f0b5	Project	CS157A	2019-12-03 06:29:02	2019-12-03 06:29:02
2484ac1b-0aa4-4948-b053-f62d96d08182	Project	CS157A	2019-12-03 06:29:25	2019-12-03 06:29:25
2874ae82-c883-45dc-ba50-4a3101bbc69f	Cs157	Great	2019-12-06 21:52:08	2019-12-06 21:52:08
3	last vacation	I went to Disneyland	2019-08-02 03:30:00	2019-08-02 03:50:00
34c15968-6b29-474e-ac5d-b688603aae3a	a	a	2019-12-03 16:46:43	2019-12-03 16:46:43
36a04a2e-f517-4726-863f-9a381505769f	c	c	2019-12-03 16:54:36	2019-12-03 16:54:36
4	favorite season	Fall is here!	2019-07-30 03:10:00	2019-07-30 03:35:00
4028c265-d54b-4026-9342-88327a6dc7bd	test2	it's a test!	2019-12-03 22:34:28	2019-12-03 22:34:28
5	on the way	It's raining, taking bus home	2019-10-11 01:20:00	2019-11-10 01:25:00
503279c8-72e0-4279-882b-01ea7608e518	Project	CS157A	2019-12-03 06:29:23	2019-12-03 06:29:23
57dc38f2-cf48-426f-afa2-f683c1c75c7d	This is Teestdsf	tplase wrokkf	2019-12-02 07:35:58	2019-12-02 07:35:58
587b33ca-ad10-403f-abd5-571e386d876a	tester 1 post	Post Test	2019-12-03 04:23:30	2019-12-03 04:23:30
6	cousin's pet	The cat is so cute!	2019-07-10 01:00:00	2019-07-10 01:20:00
7	Noodle time	Eating ramen is the best for cold day	2019-09-14 02:02:00	2019-09-14 02:12:00
755787b9-1729-4521-81cc-ebaee99b5f97	Project	CS157A	2019-12-03 06:29:05	2019-12-03 06:29:05
7822d835-b0de-418f-a1f9-7876c39c80ef	dfad	asdf	2019-12-03 16:57:21	2019-12-03 16:57:21

Posts 1 | Apply

3. Comments table

Comments

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	content	created_at	updated_at
089ae706-4a0a-47ca-ae38-707357691ea3	be careful!	2019-12-06 11:01:03	2019-12-06 11:01:03
09b321f8-9759-443d-adf1-932a86794f98	comment reply	2019-12-03 21:54:40	2019-12-03 21:54:40
0fc3dca1-3b2d-451b-bfad-b0c3bc74167b	comment logged in	2019-12-03 21:29:29	2019-12-03 21:29:29
1	same here	2019-10-12 02:10:00	2019-10-12 02:35:00
10	I want some	2019-10-05 03:05:00	2019-10-05 03:25:00
11	Good Luck!!!	2019-08-02 04:30:00	2019-08-02 04:40:00
12	Yup, it's the best	2019-09-14 02:02:00	2019-09-14 02:12:00
13	My kids are waiting for that!	2019-08-02 03:30:00	2019-08-02 03:50:00
14	I also have two midterms	2019-10-10 05:40:00	2019-10-10 05:50:00
15	I just got out from there	2019-09-20 07:05:00	2019-09-20 07:20:00
16	I am completly unknown to su…	2019-10-15 02:29:36	2019-10-15 02:29:36
17	It really covers all aspects of c…	2019-10-15 05:29:36	2019-10-15 05:29:36
18	Please do	2019-10-15 08:29:36	2019-10-15 08:29:36
19	TABLE OF CONTENTS: ...	2019-10-15 09:29:36	2019-10-15 09:29:36
2	Why???	2019-10-10 01:20:00	2019-10-10 02:00:00
20	Thanks !	2019-10-15 09:45:36	2019-10-15 09:45:36
21	The head jerk gets me everyti…	2019-10-14 05:29:36	2019-10-14 05:29:36
22	Back... and to the left	2019-10-14 06:29:36	2019-10-14 06:29:36
23	“Never mind, I figured it out”	2019-10-10 08:29:36	2019-10-10 08:29:36
24	How many SO users does it t…	2019-10-10 08:35:36	2019-10-10 08:35:36
25	Lol	2019-10-10 09:45:36	2019-10-10 09:45:36
26	Artificial Intelligence	2019-10-15 09:46:36	2019-10-15 09:46:36
27	Networks	2019-10-15 09:47:36	2019-10-15 09:47:36
28	Database	2019-10-15 09:48:36	2019-10-15 09:48:36
29	Comp Architecture	2019-10-15 09:49:36	2019-10-15 09:49:36
3	Congratulation!	2019-08-14 05:05:00	2019-08-14 05:10:00
30	Formal Languages	2019-10-15 09:50:36	2019-10-15 09:50:36
31	Data Structures and Algorithms	2019-08-06 01:35:00	2019-08-06 01:35:00

Comments 1

4. Channels table

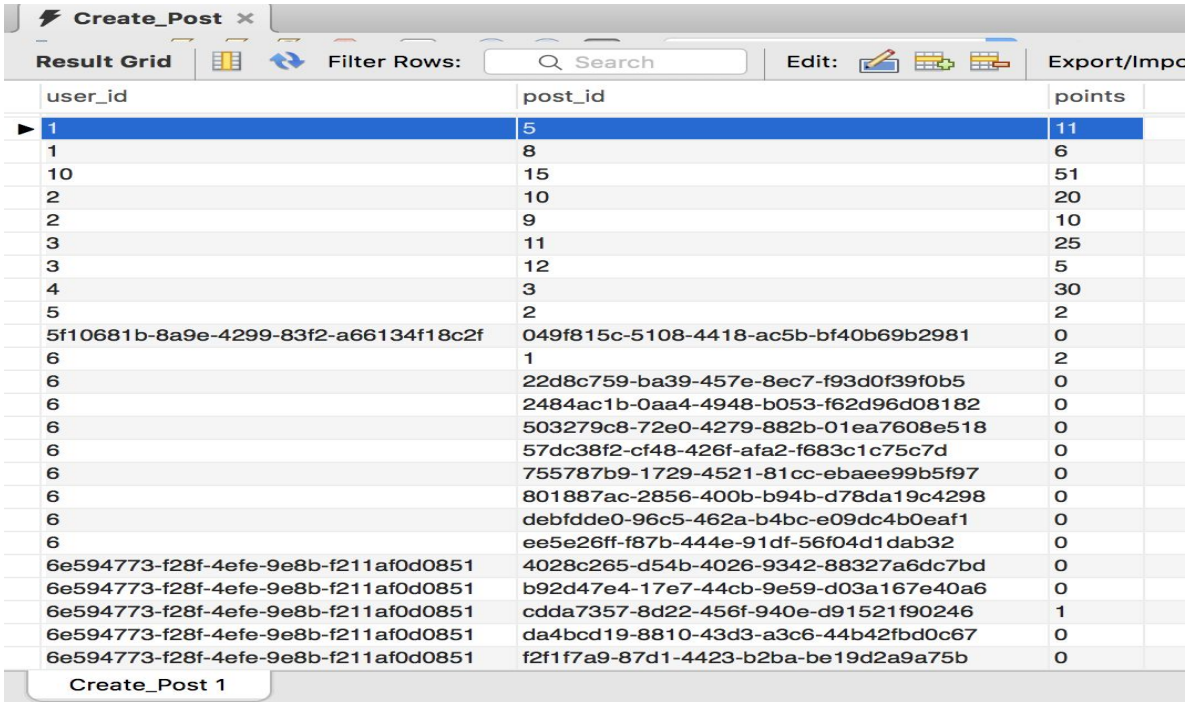
Channels

Result Grid | Filter Rows: Search | Edit:

id	name	banner_url
1	programming	www.banner1.com
10	myChannel10	www.banner10.com
11	myChannel11	www.banner11.com
12	myChannel12	www.banner12.com
13	myChannel13	www.banner13.com
14	myChannel14	www.banner14.com
15	myChannel15	www.banner15.com
16	myChannel16	www.banner16.com
2	science	www.banner2.com
3	cs157A	www.banner3.com
4	cs157B	www.banner4.com
5	cs157C	www.banner5.com
6	myChannel6	www.banner6.com
6ca294f4-2c66-4524-afde-4a30aed2f254	cs9001	www.bannertest.com
7	myChannel7	www.banner7.com
8	myChannel8	www.banner8.com
9	myChannel9	www.banner9.com

Channels 1

5. Create_post table



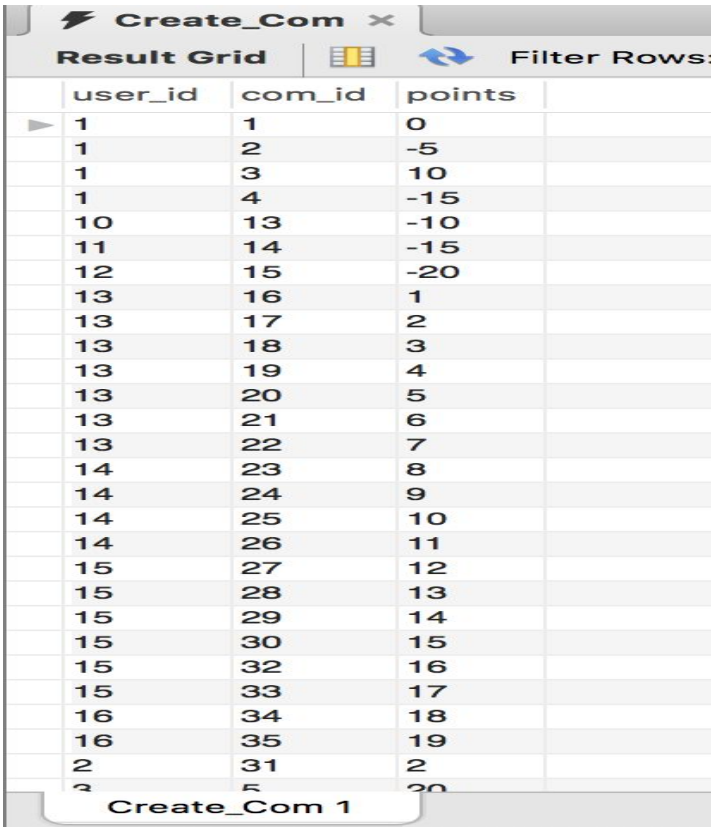
Create_Post

Result Grid | Filter Rows: Search | Edit: | Export/Impo

user_id	post_id	points
1	5	11
1	8	6
10	15	51
2	10	20
2	9	10
3	11	25
3	12	5
4	3	30
5	2	2
5f10681b-8a9e-4299-83f2-a66134f18c2f	049f815c-5108-4418-ac5b-bf40b69b2981	0
6	1	2
6	22d8c759-ba39-457e-8ec7-f93d0f39f0b5	0
6	2484ac1b-0aa4-4948-b053-f62d96d08182	0
6	503279c8-72e0-4279-882b-01ea7608e518	0
6	57dc38f2-cf48-426f-afa2-f683c1c75c7d	0
6	755787b9-1729-4521-81cc-ebaee99b5f97	0
6	801887ac-2856-400b-b94b-d78da19c4298	0
6	debfdde0-96c5-462a-b4bc-e09dc4b0eaf1	0
6	ee5e26ff-f87b-444e-91df-56f04d1dab32	0
6e594773-f28f-4efe-9e8b-f211af0d0851	4028c265-d54b-4026-9342-88327a6dc7bd	0
6e594773-f28f-4efe-9e8b-f211af0d0851	b92d47e4-17e7-44cb-9e59-d03a167e40a6	0
6e594773-f28f-4efe-9e8b-f211af0d0851	cdda7357-8d22-456f-940e-d91521f90246	1
6e594773-f28f-4efe-9e8b-f211af0d0851	da4bcd19-8810-43d3-a3c6-44b42fbd0c67	0
6e594773-f28f-4efe-9e8b-f211af0d0851	f2f1f7a9-87d1-4423-b2ba-be19d2a9a75b	0

Create_Post 1

6. Create_Com table



Create_Com

Result Grid | Filter Rows:

user_id	com_id	points
1	1	0
1	2	-5
1	3	10
1	4	-15
10	13	-10
11	14	-15
12	15	-20
13	16	1
13	17	2
13	18	3
13	19	4
13	20	5
13	21	6
13	22	7
14	23	8
14	24	9
14	25	10
14	26	11
15	27	12
15	28	13
15	29	14
15	30	15
15	32	16
15	33	17
16	34	18
16	35	19
2	31	2

Create_Com 1

7. Have table

Have

Limit to 1000 rows

Result Grid | Filter Rows: Search | Edit: | Export

post_id	com_id
09c33eed-13b0-46db-941b-509fec766068	ff55b16c-a454-4873-8d17-a1022dcbf7cd
1	1
1	2
1	3
2	4
2	5
2	6
22d8c759-ba39-457e-8ec7-f93d0f39f0b5	b1af5b67-457b-4bd4-b706-0543acf87ec5
2874ae82-c883-45dc-ba50-4a3101bbc69f	3e98ed07-923a-4cb2-827a-c247852ade4a
3	7
3	8
3	9
4	10
4	11
4	12
5	089ae706-4a0a-47ca-ae38-707357691ea3
5	0fc3dca1-3b2d-451b-bfad-b0c3bc74167b
5	4f58585b-77e9-462f-ab1f-7ac53bba2d9b
587b33ca-ad10-403f-abd5-571e386d876a	bbbe7519-ba1f-49b7-8c0e-2c0a6d3e8802
6	13
6	14
6	15
8	6557931a-1065-4f42-b5e7-e105e83a2c3b
8	8cbfedda-263a-452d-b424-795c0855bb51
88241fb9-35c8-4113-9f65-b5f300ef3486	09b321f8-9759-443d-adf1-932a86794f98

Have 1

8. Have_Com table

Have_Com

Result Grid | Fil

parent_id	child_id
16	17
17	18
18	19
21	22
23	24
23	25
26	27
26	28
29	30
10	26
11	29
1	31
2	32
3	33
4	34

Have_Com 1

9. Save table

Query 19 | Save

Result Grid | Filter Rows: Searc

user_id	post_id
1	1
1	2
1	3
2	1
2	2
2	3
3	10
3	11
3	12
4	14
4	8
4	9
5	1
5	2
5	3
6e594773-f28f-4efe-9e8b-f211af0d0851	3
da7e8692-839f-4001-87a3-c4676fe5110b	8
f26a12e1-5c58-4a69-922e-480a132835d2	2

Save 1

10. Vote_Com table

Vote_Com

Result Grid | Filter

user_id	com_id	up_down
1	1	1
1	2	0
1	3	1
2	1	0
2	2	1
2	3	0
3	10	1
3	11	0
3	12	1
4	14	0
4	7	0
4	9	0
5	1	1
5	2	1
5	3	1

Vote_Com 1

11. Own table

Own

Result Grid | Filter Rows: | Sear

user_id	channel_id
5	1
7	10
8	11
10	12
11	13
12	14
14	15
2	2
4	3
2	4
3	5
1	6
1	6ca294f4-2c66-4524-afde-4a30aed2f254
6	7
6	8
7	9
NULL	NULL

Own 1

12. Contain table

Contain

Result Grid | Filter Rows: | Search

channel_id	post_id
1	2
1	4
1	5
2	3
2	6
4	1
5	7
7	8
8	9
10	10
11	11
12	12
7	13
13	14
7	15
3	57dc38f2-cf48-426f-afa2-f683c1c75c7d
1	587b33ca-ad10-403f-abd5-571e386d876a
2	a6f93f27-28e7-40b6-a87d-6730e1265ef3
2	22d8c759-ba39-457e-8ec7-f93d0f39f0b5
2	801887ac-2856-400b-b94b-d78da19c4298
2	755787b9-1729-4521-81cc-ebaee99b5f97
2	ee5e26ff-f87b-444e-91df-56f04d1dab32
2	debfdde0-96c5-462a-b4bc-e09dc4b0eaf1
1	503279c8-72e0-4279-882b-01ea7608e518
1	2484ac1b-0aa4-4948-b053-f62d96d08182
7	850d15f0-45e9-4c82-b7b0-4b95f8af3458
7	b075066c-6644-441c-8968-e4fccb97b040
7	a2db7f6f-b67c-47ba-89e2-ad4dafff134f

Contain 1

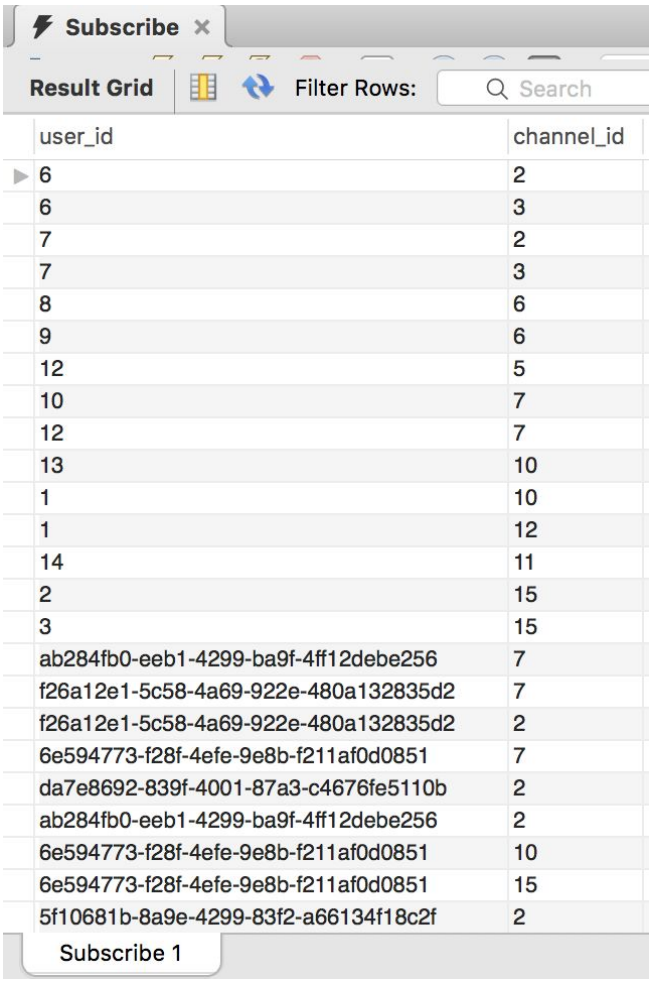
13. Moderate table

Moderate

Result Grid

user_id	channel_id
1	2
1	3
2	2
2	3
3	6
4	6
7	5
5	7
7	7
8	10
11	10
11	12
9	11
14	15
15	15

14. Subscribe



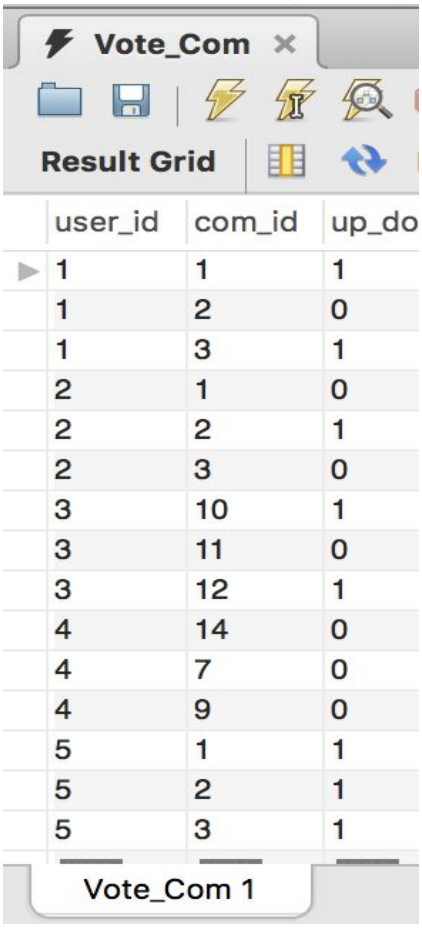
Subscribe

Result Grid | Filter Rows: Search

user_id	channel_id
6	2
6	3
7	2
7	3
8	6
9	6
12	5
10	7
12	7
13	10
1	10
1	12
14	11
2	15
3	15
ab284fb0-eeb1-4299-ba9f-4ff12debe256	7
f26a12e1-5c58-4a69-922e-480a132835d2	7
f26a12e1-5c58-4a69-922e-480a132835d2	2
6e594773-f28f-4efe-9e8b-f211af0d0851	7
da7e8692-839f-4001-87a3-c4676fe5110b	2
ab284fb0-eeb1-4299-ba9f-4ff12debe256	2
6e594773-f28f-4efe-9e8b-f211af0d0851	10
6e594773-f28f-4efe-9e8b-f211af0d0851	15
5f10681b-8a9e-4299-83f2-a66134f18c2f	2

Subscribe 1

15. Vote_Com



Vote_Com

Result Grid

user_id	com_id	up_do
1	1	1
1	2	0
1	3	1
2	1	0
2	2	1
2	3	0
3	10	1
3	11	0
3	12	1
4	14	0
4	7	0
4	9	0
5	1	1
5	2	1
5	3	1

Vote_Com 1

Implementation

Detail explanations of how your DB application system was implemented.

We have the front-end handled by React framework. With React, we just need to know the basics of HTML and CSS. There are a few css libraries that we use such as Material-UI and styled-components. These libraries help us build a better dynamic and responsive UI. At the front-end, we also use Redux for state management. One of the hardest parts in building the front end is state management. There is a lot of overhead when using Redux, but without Redux, when the app getting bigger, it will be super hard to be maintained.

To communicate with the backend, the front-end will use fetch() function, which supports all the HTTP methods. We use Spring Boot to build our Rest API, which will handle all the logics and communicate to MySQL database. To connect with MySQL, we just need to add the connector dependency to the Pom file. Working with MySQL using Object mapping created a big overhead. There are not many tutorials on mapping Java class to tables. I spent a few weeks just to do research on mapping. This part is the hardest part when building our backend.

The end result is great because it's super simple to query and modify the database. We can treat tables like classes. All the mapped classes are in the "model" folder. We don't need to create a model for a relationship that doesn't have extra attributes. For database manipulation, we use JPA Data framework, which support both native SQL query and JPQL query. To do that we just need to create an interface that extended JpaRepository interface and inject a @Repotory Bean to the interface. If we want to get all posts from Post table using JPQL, we can do:

```
@Query("SELECT p FROM Post")
List<Post> findAllPostCustom();
```

The syntax is really similar with native SQL. To write out endpoints, we inject @RestController Bean to our classes. To write a GET request, we inject @GetMapping(<url name>) and to write Post request, we do @PostMapping(<url name>). The syntaxes will be similar for DELETE, and PUT. With Spring Boot, creating endpoints is really simple and quick. We also add security to our backend using spring-boot-starter-security library.

Keep tracks of implementations from design

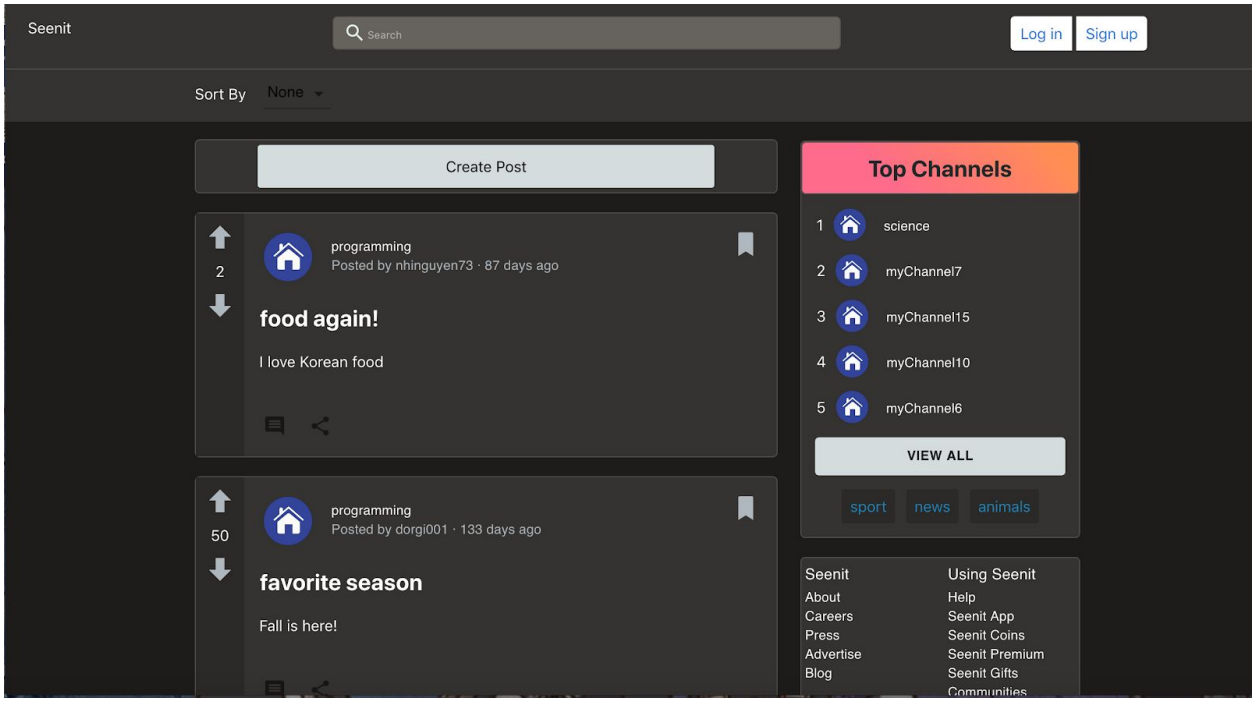
Identify the entities, attributes, dependencies, relationships, constraints, etc. (show screenshots of corresponding tables, GUI, execution results, and so on.)

Show functions/features associated with query, insertion, updating, and deletion operations. (Screenshots)

Note: **The queries are written in server/src/main/java/com/seenit/server/repository, the completed functions are written in server/src/main/java/com/seenit/server/controller.**

Home page

- A user can always go back to the home page by clicking on "Seenit" in the header



Top Channel

Query:

```
@Query("SELECT c.id as id, c.name as name, COUNT(c.id) AS members FROM Channel c JOIN c.subscribers GROUP BY c.id")
Page<TopChannels> findTopChannels(Pageable pageable);
```

LOG-IN not REQUIRED FUNCTIONS

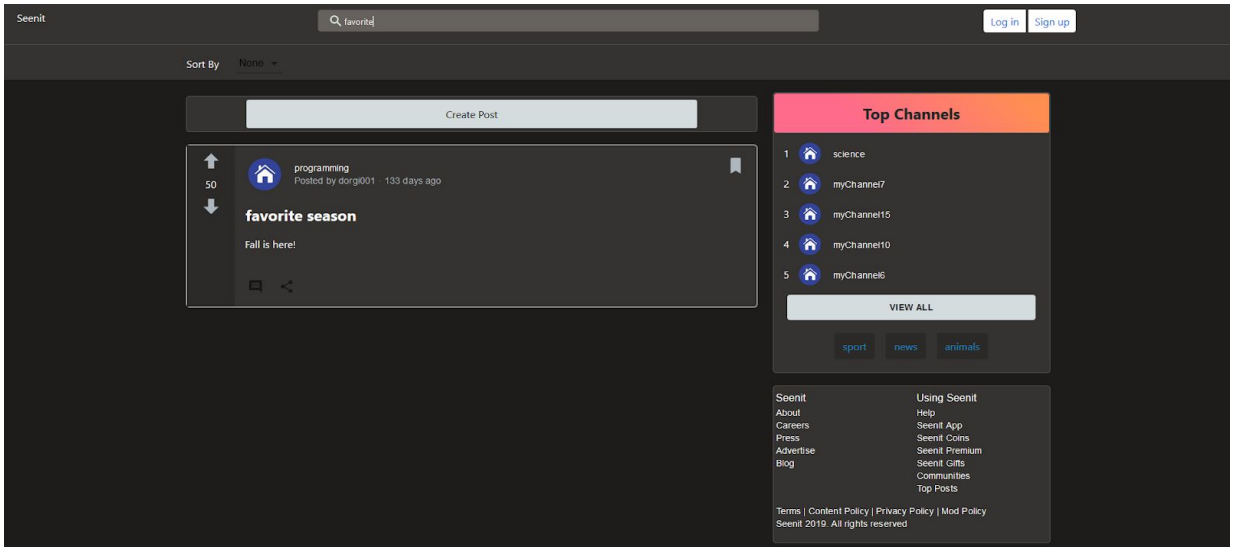
Search: Type in search bar at the top and press enter

Query:

```
@Query("SELECT p, ch, ca.points, ca.user " +
    "FROM Post p JOIN p.createdBy ca JOIN p.channel ch " +
    "WHERE p.title LIKE %?1% OR p.content LIKE %?1%")
List<Object[]> findBySearch(String search);
```

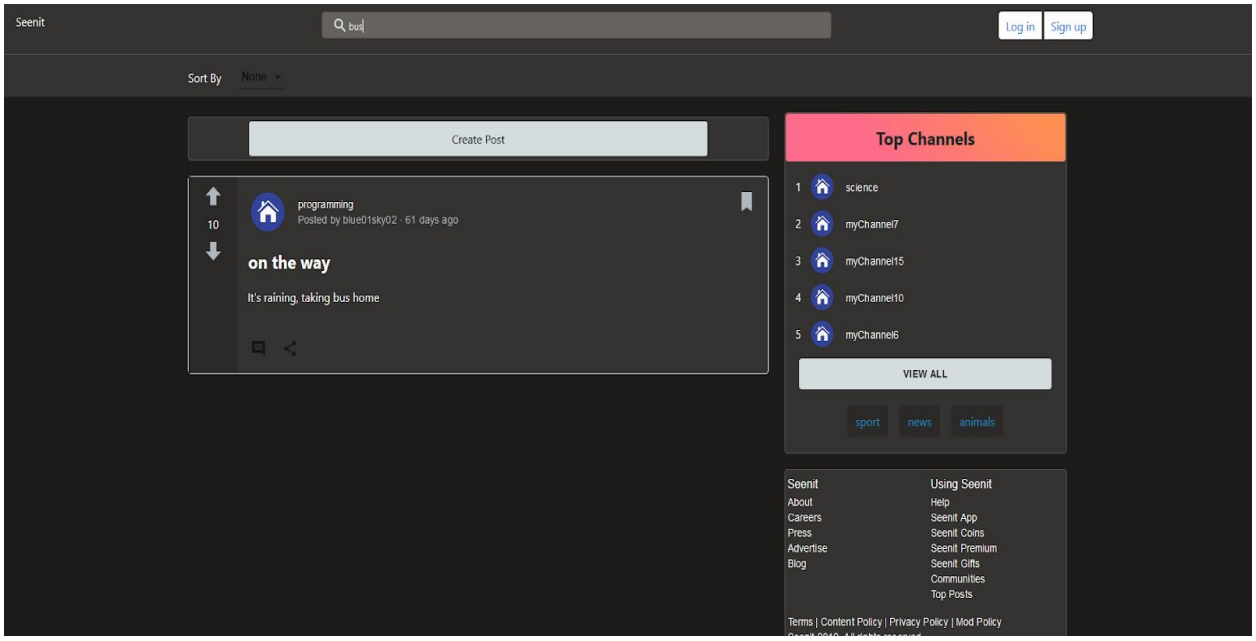
1. Search by post title:

- For instance, enter ‘favorite’ in the search bar



2. Search by post content:

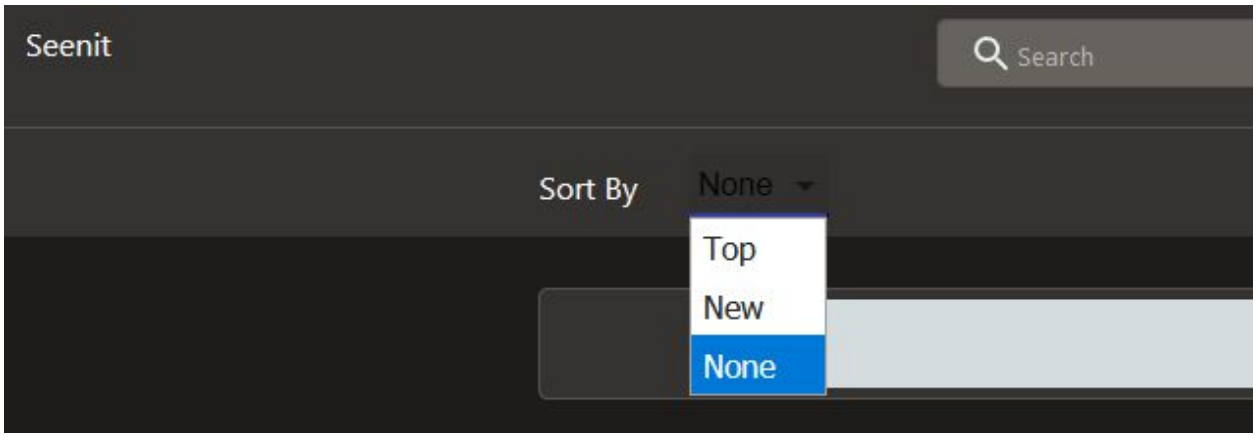
- For instance, enter ‘bus’ in the search bar



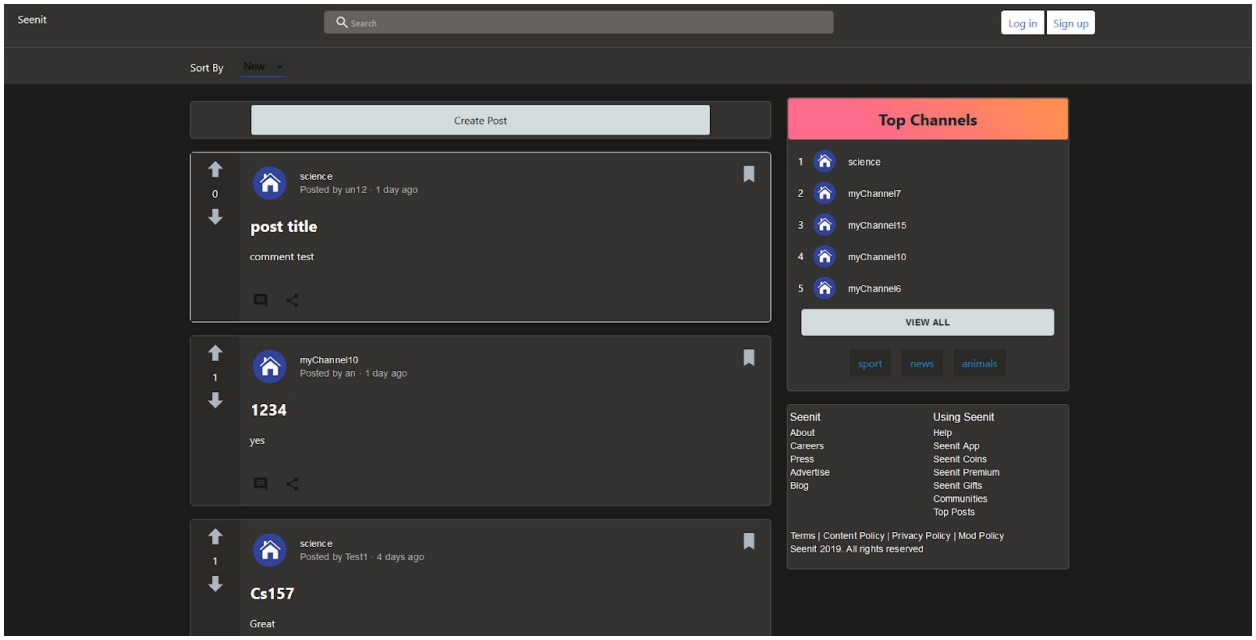
Sort: Next to “Sort By”, click on the drop down to sort by Top, New, or None

Query:

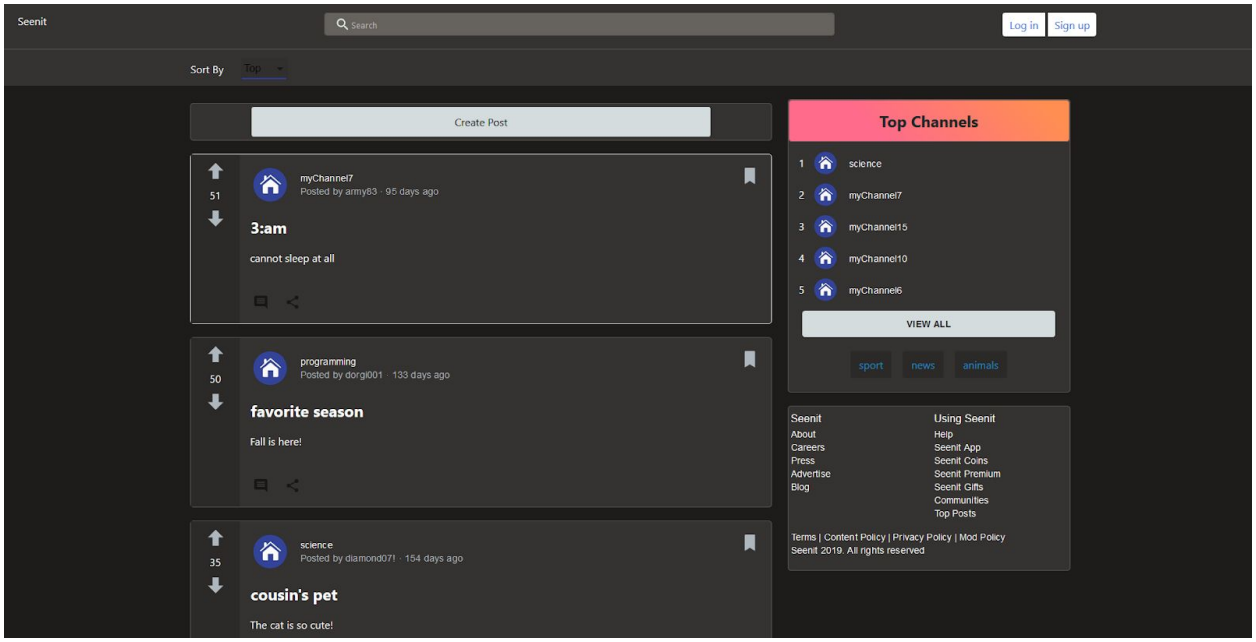
```
@Query("SELECT p, ch, ca.points, ca.user FROM Post p JOIN p.createdBy ca JOIN p.channel ch")
Page<Object[]> findAllObjectSort(Pageable pageable);
```



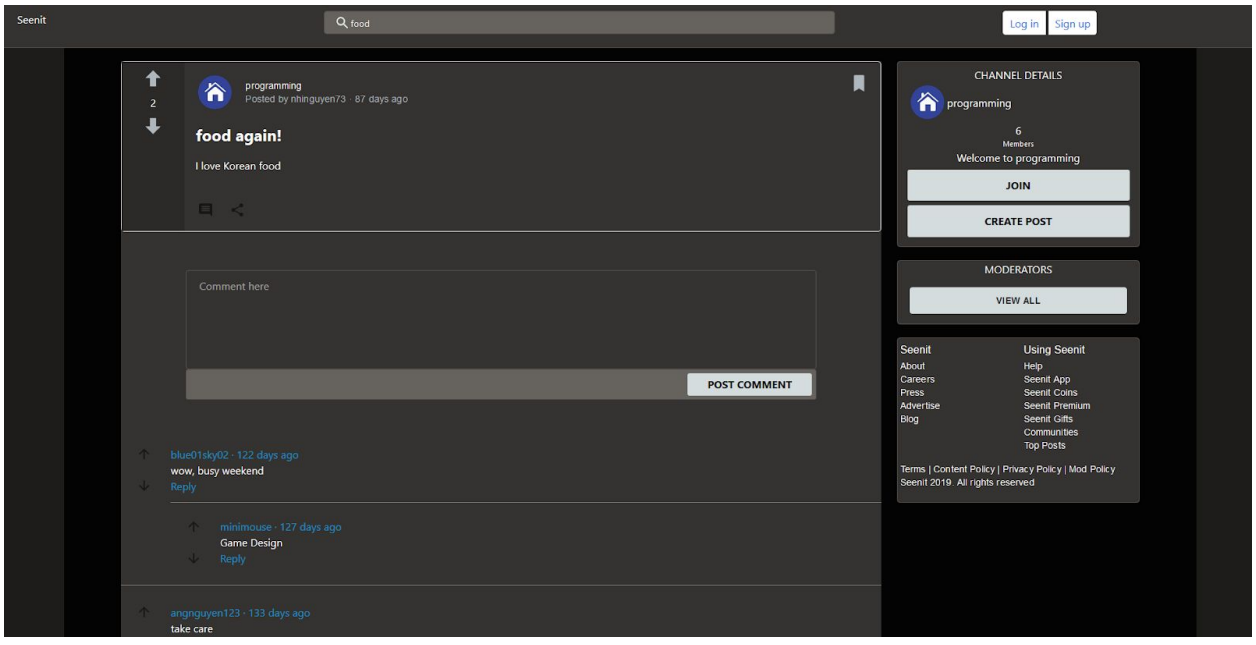
1. Sort by New:



2. Sort by Top (highest points):



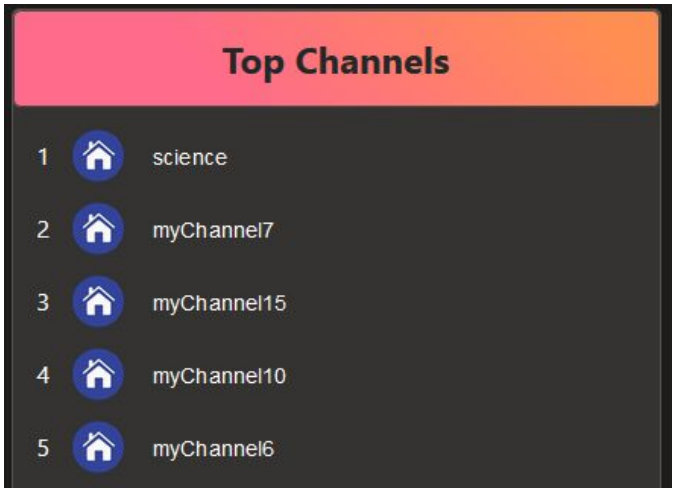
View a Post: Click on a post to view it



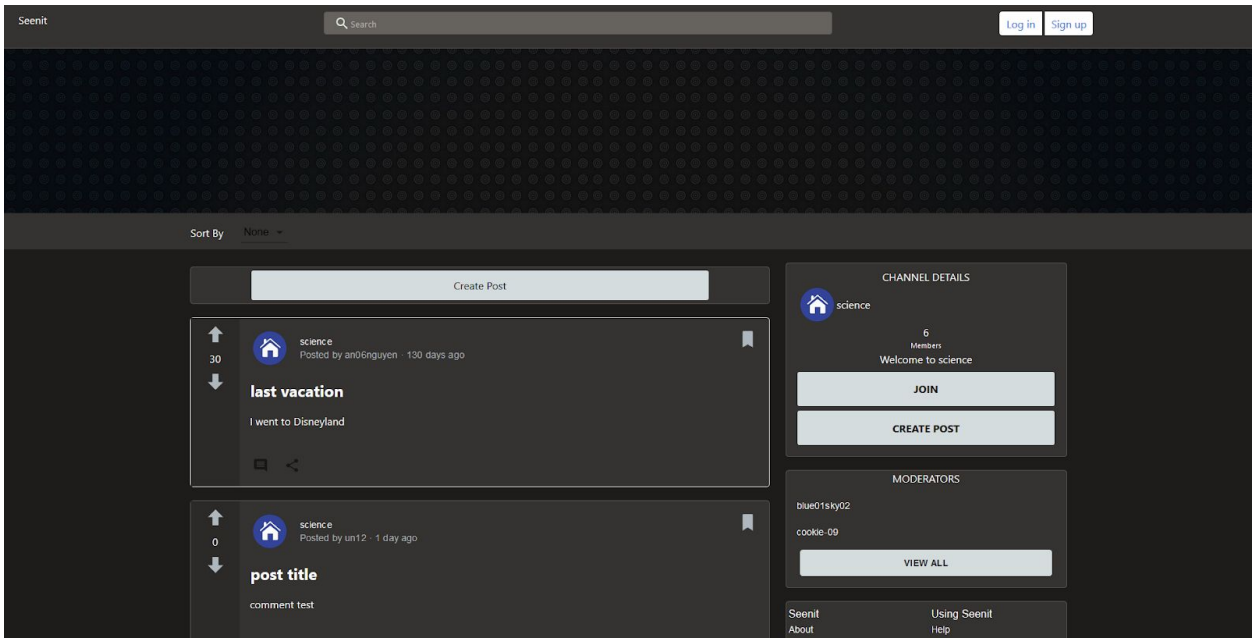
Query:

```
@Query("SELECT c FROM Comment c JOIN c.postCom post JOIN c.createdBy cb where post.id = ?1")
List<CommentDetails> findComByPostId(String postId);
```

View a channel details: click on a channel to view details



- Channel detail page will display channel's name, number of joined members, list of moderators, and all the posts are made on the channel.



Queries:

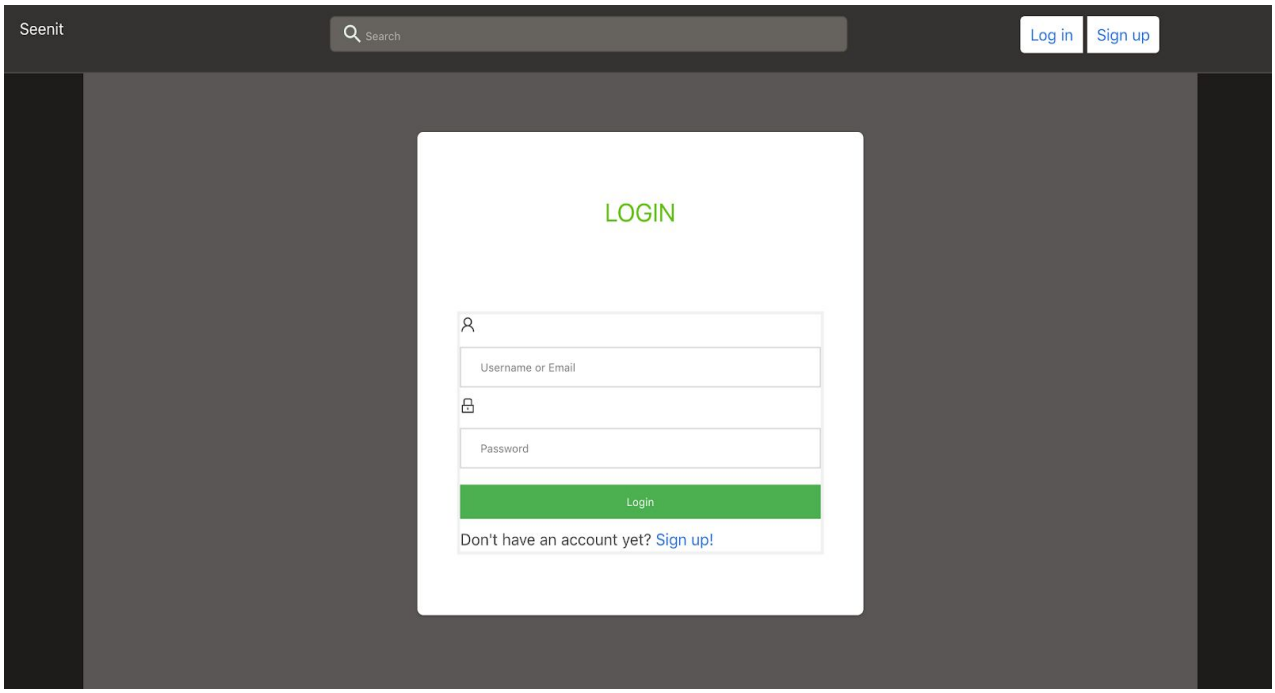
```
@Query("SELECT c, COUNT(c.id) FROM Channel c JOIN c.subscribers WHERE c.id = ?1 GROUP BY c.id")
List<Object[]> findNumberOfMembersByChannelId(String id);
```

```
@Query("Select u FROM User u JOIN u.moderatedChannels mods WHERE mods.id = ?1")
List<UserIdName> findModeratorsByChannelId(String id);
```

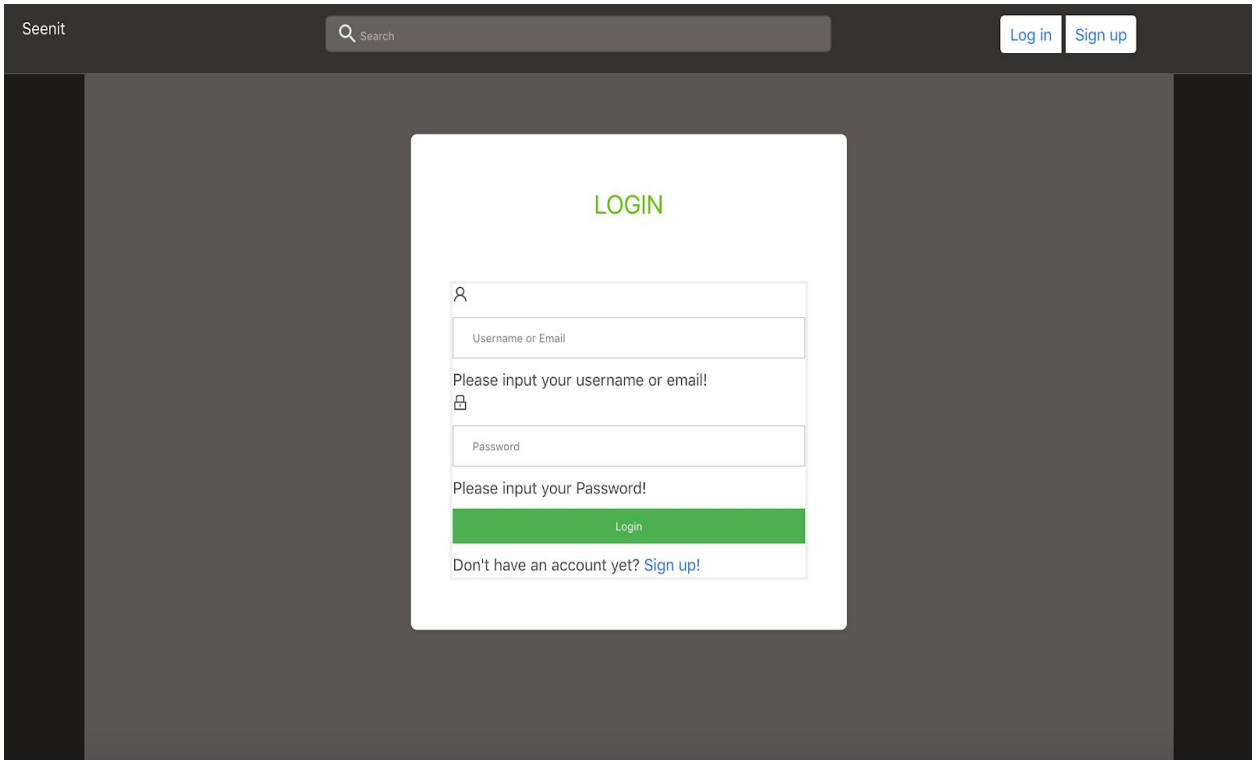
```
@Query("SELECT p, ch, ca.points, ca.user FROM Post p JOIN p.createdBy ca JOIN p.channel ch WHERE ch.id = ?1")
Page<Object[]> findAllPostsByChannel(String channelId, Pageable pageable);
```

Log-in function (click on the Log-in button on the header)

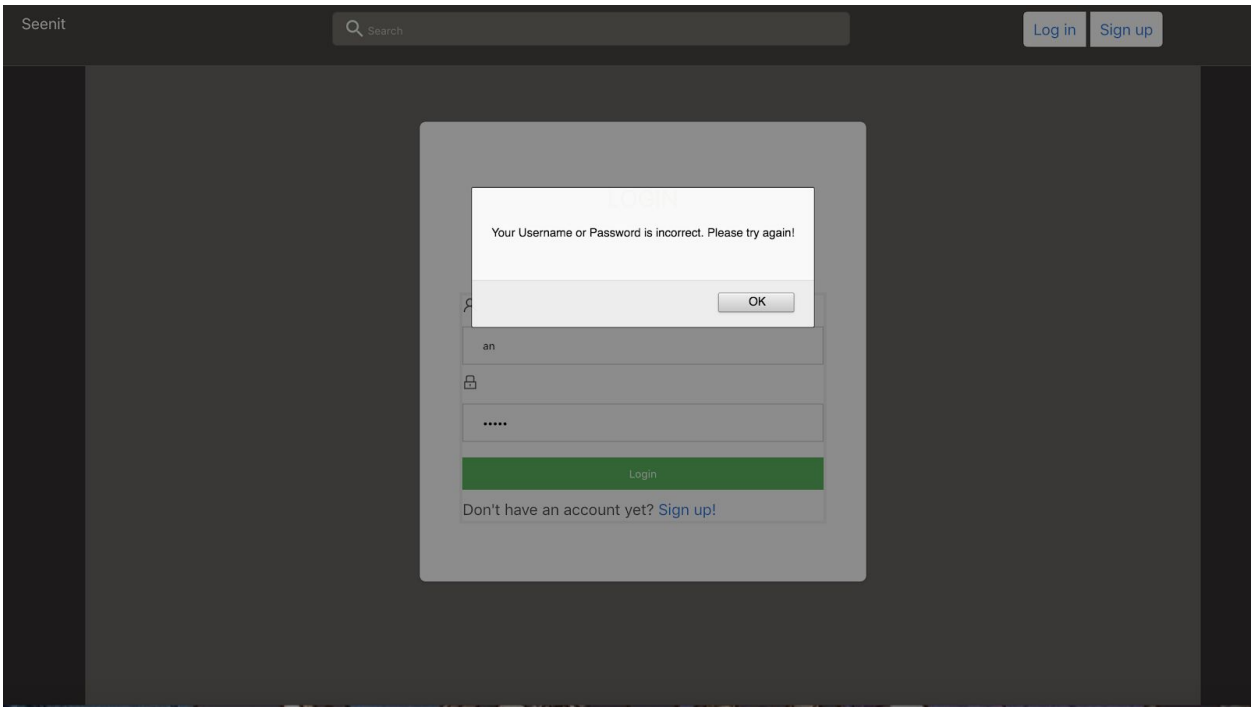
1. Log-in form



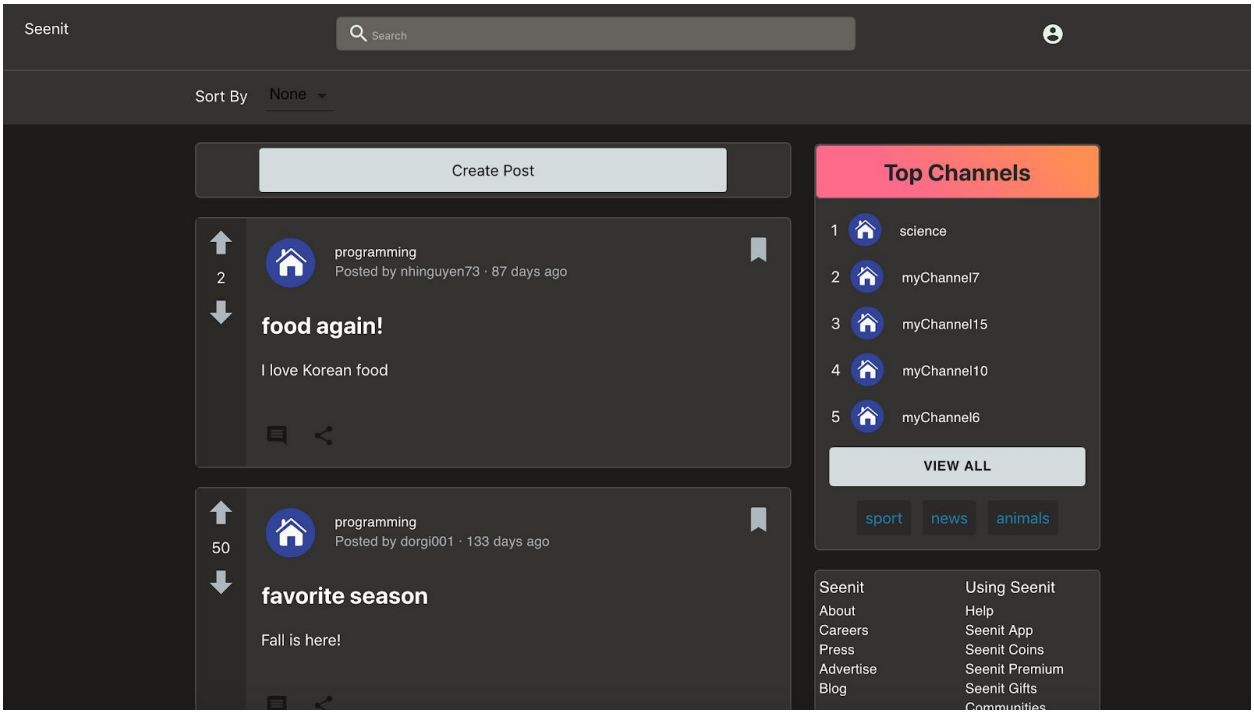
2. Must fill all the fields in log-in form



3. Log-in validation
    - Display an error message when username or password does not match.

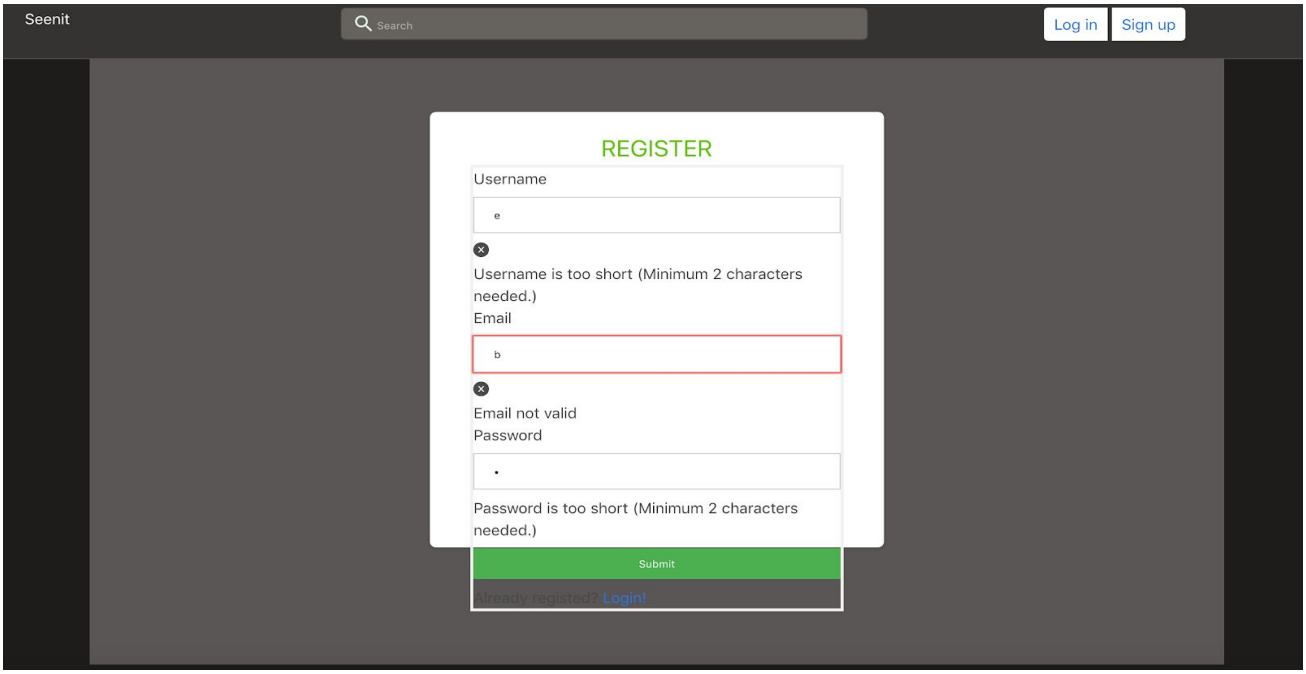


- Successfully log-in

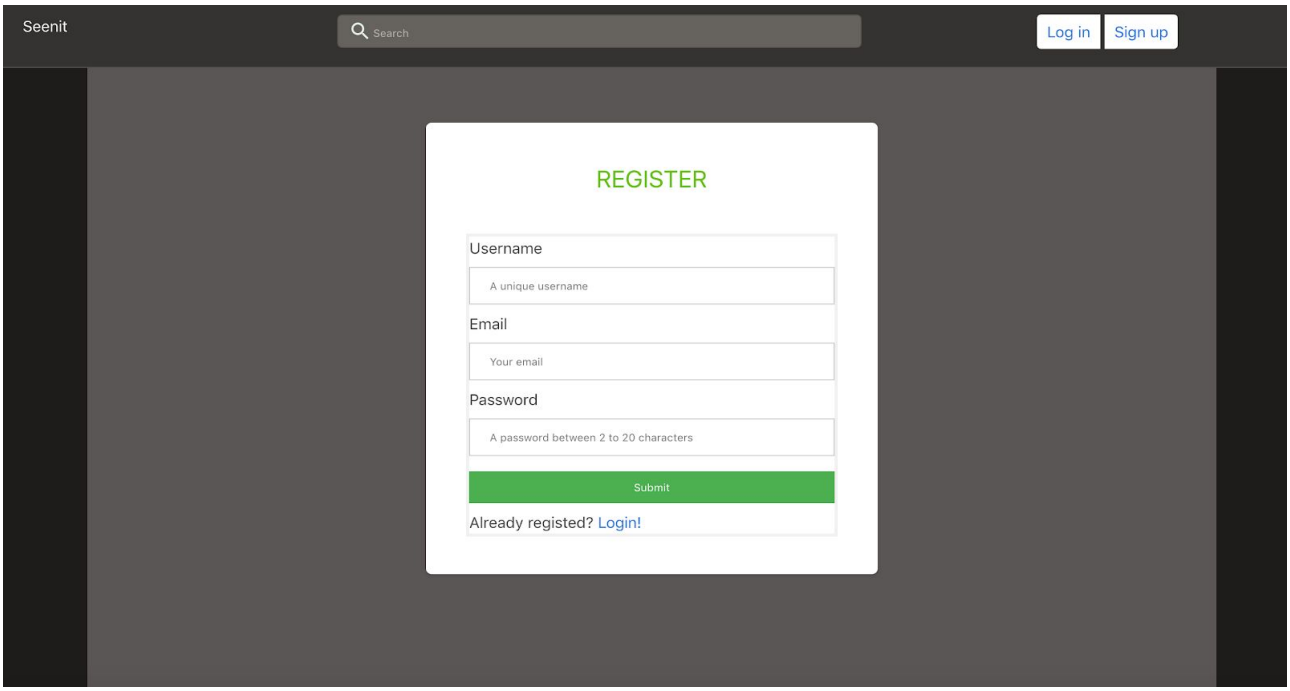


Register function (click on the Sign-up button on the header)

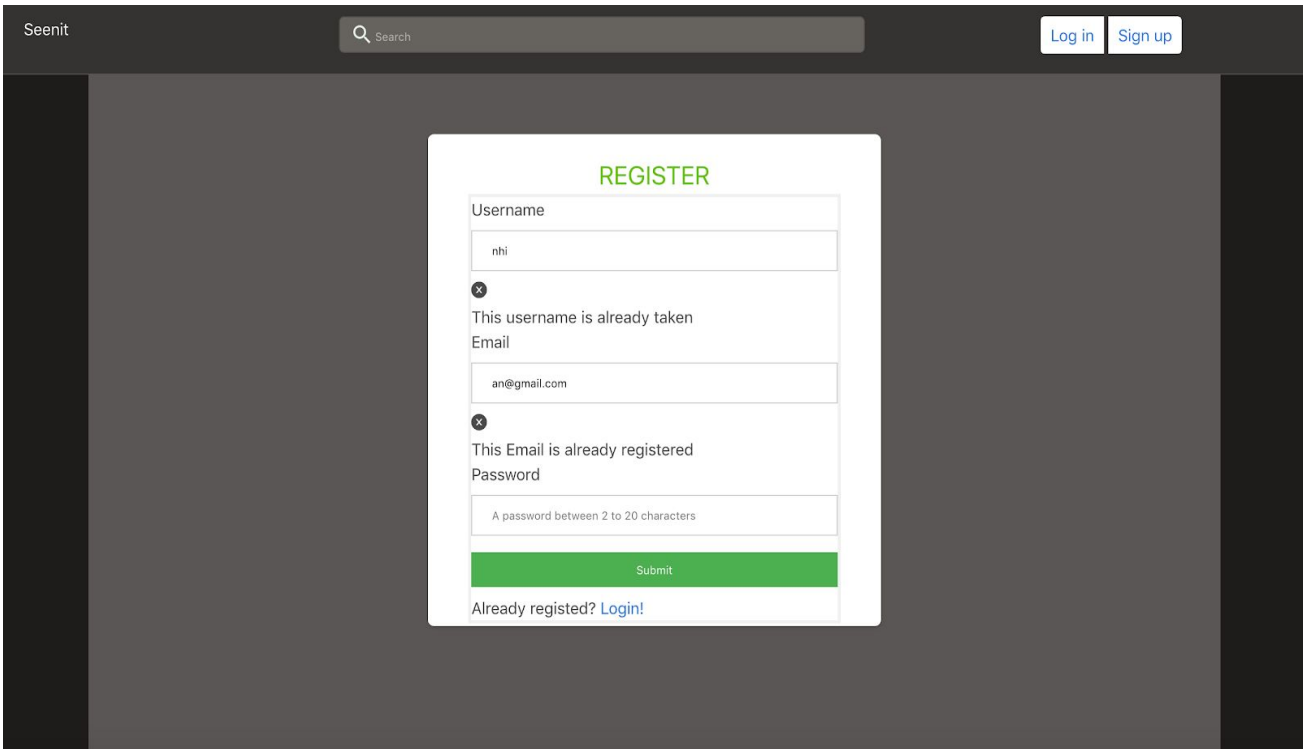
1. Sign-up form



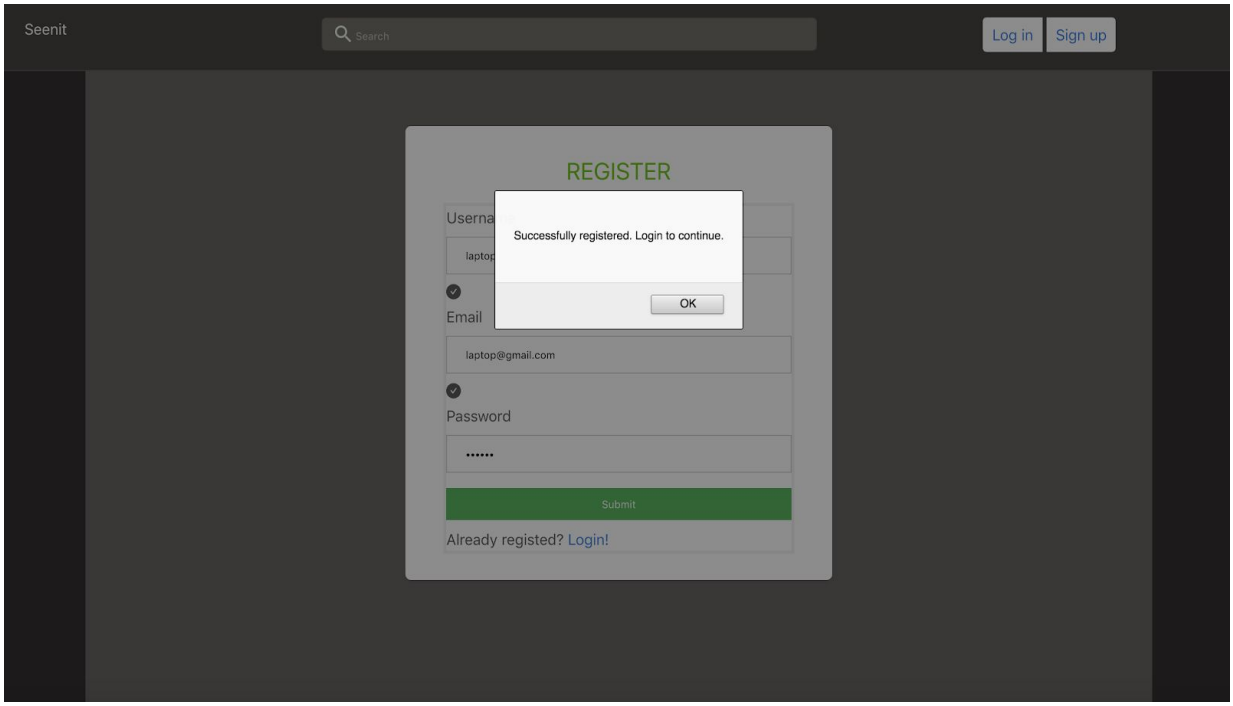
2. Does not meet the requirements



3. Sign-up validation
    - Username or Email is already registered



- Users will get a popup message when they successfully registered for an account

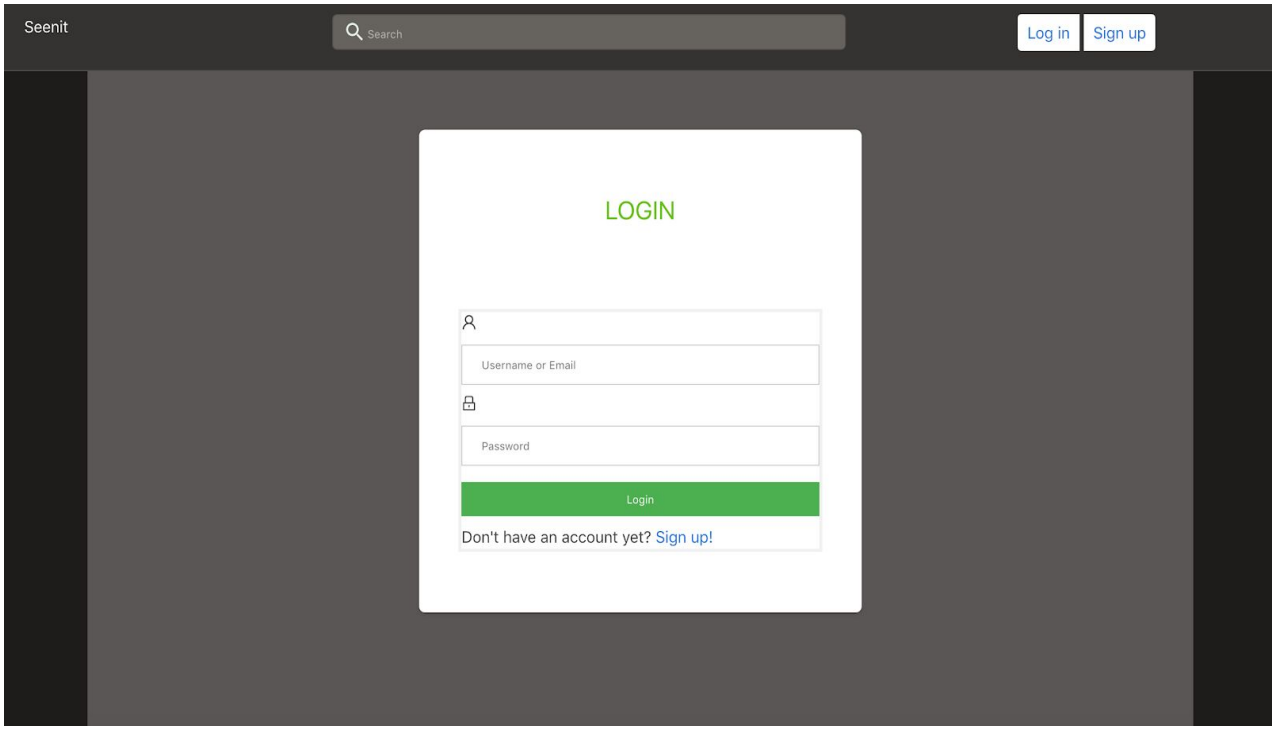


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	email	username	password	created_at	avatar_url
4981ef94-1032-4667-b8b3-da39d07098cc	hello@gmail.com	hello	$2a$10$jv21n.LQxNlzoE.qkP.UY.wlOSpd3rMeR...	2019-12-11 05:07:28	NULL
2090d5b1-92f4-42d9-9f92-d8e1a8be6c5c	mom@gmail.com	mom	$2a$10$fNGv7AmOgMEuxt6yjKfd8eBTci0euk.q...	2019-12-11 04:31:49	NULL
c7e6f1ab-b468-4eef-ae5b-ed60b6388527	testerror@gmail.com	testerror	$2a$10$uHFmi5tM0JIyQ4D0cHaApu6cIqykwSy...	2019-12-11 04:00:40	NULL
5f3c5959-9887-4291-9255-d08381815120	e21@g.c	un21	$2a$10$rqHleKdAxQj0lNWbOsMB/OQvhD77g26...	2019-12-11 03:49:29	NULL
a0d5c83c-d930-4e74-a21d-ea498e7651c4	test3@gmail.com	test3	$2a$10$uUKck2k1uCOcRggRWNlys.Tcy4qNL4Q...	2019-12-11 03:49:15	NULL
51a3c554-4be9-4af5-9185-4bdb88095968	car@gmail.com	car	$2a$10$F7Ll8/shR79/sSWFeXK5OuIhdbi3o4Sp...	2019-12-11 03:36:51	NULL
247d997e-3081-4ed5-85e3-efdd61172657	gdsf@gmail.com	newTester	$2a$10$O6MZ1LxgLSIwehWlJXFl3Ots6EyuwtD...	2019-12-11 03:35:58	NULL
35193b63-17a0-4207-bbc9-cfbace6bfa57	core@gmail.com	core	$2a$10$piGGaDh9NuLqow8QLYuDi.AtqXrUrXp2...	2019-12-11 03:13:08	NULL
90715c8b-cb5a-4a2c-84c1-62d66bfaf491	button@gmail.com	button	$2a$10$DzNELfuduY4j4BEzrRo86O.J7X1ZIh/4D...	2019-12-11 03:12:14	NULL
f5579f28-37ff-4513-a9f9-508fff5a8aa3	phone@gmail.com	phone	$2a$10$JARmX3ULMGpAMVgH9zlQDeBm9KPFN...	2019-12-11 03:11:26	NULL
6c7ee6fa-8f82-4fbf-830b-fcc848af2f56	laptop@gmail.com	laptop	$2a$10$H1rh5IkYDVX73iW8qyiAmOYGcme2CN...	2019-12-11 02:56:38	NULL
5f10681b-8a9e-4299-83f2-a66134f18c2f	e12@g.c	un12	$2a$10$jI9RX5Fpole9TwsfVaAvBOcJ3qnrRznH...	2019-12-09 21:03:54	NULL
35f4957b-e381-4689-bd21-18a6cc7c60df	e1@g.co	un11	$2a$10$.bk5qHBlQB4vEiSOo2q9n.9X.wdsvbTY...	2019-12-06 21:49:10	NULL

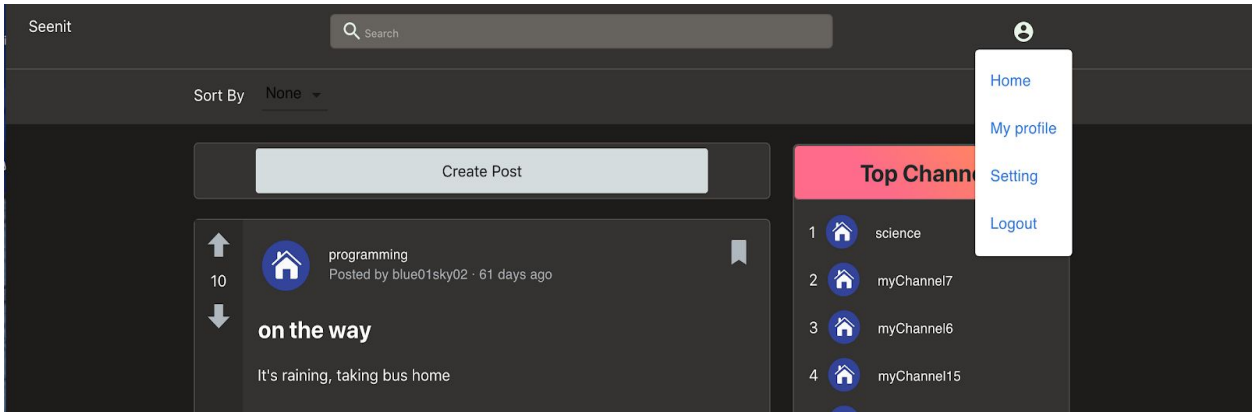
Users 1

- After that, users will be redirect to log-in page

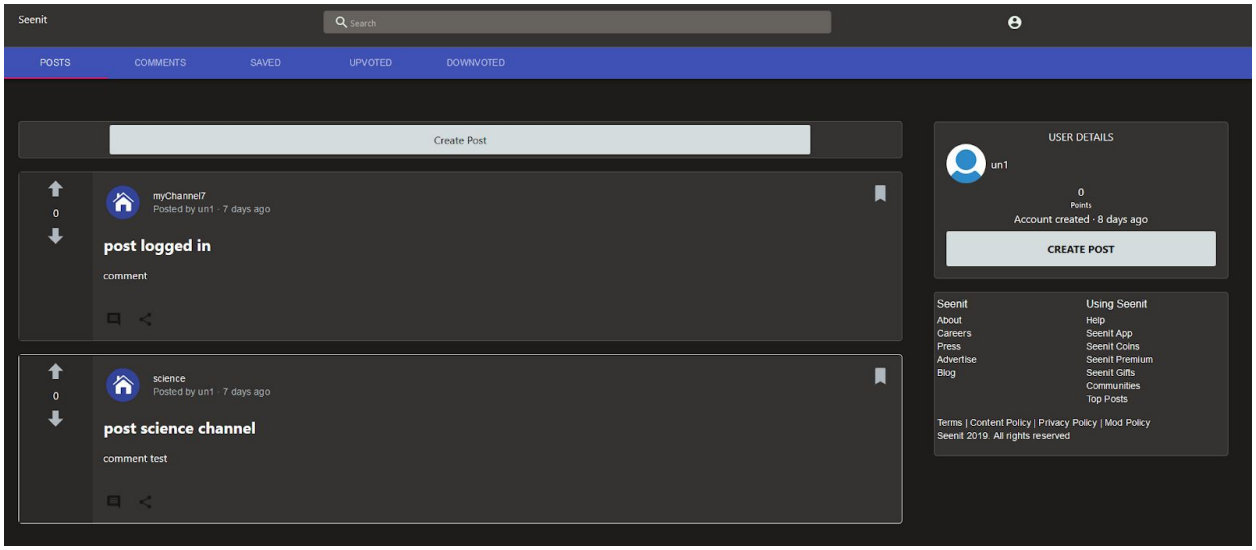


LOG-IN REQUIRED FUNCTIONS

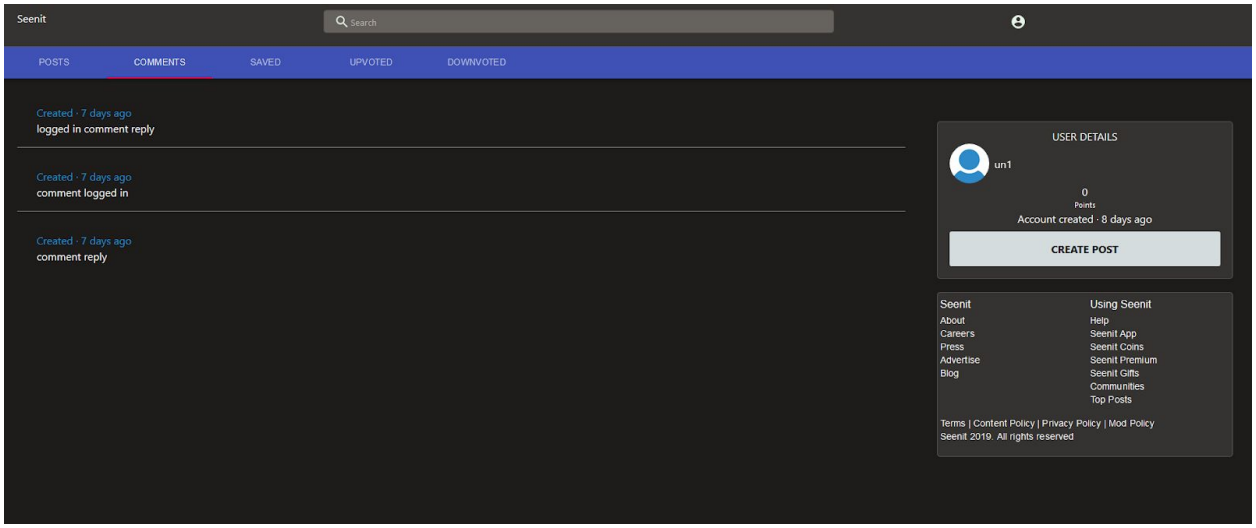
View profile dashboard: To view the personal dashboard click on the user icon located on the right side of the header and choose ‘My profile’.



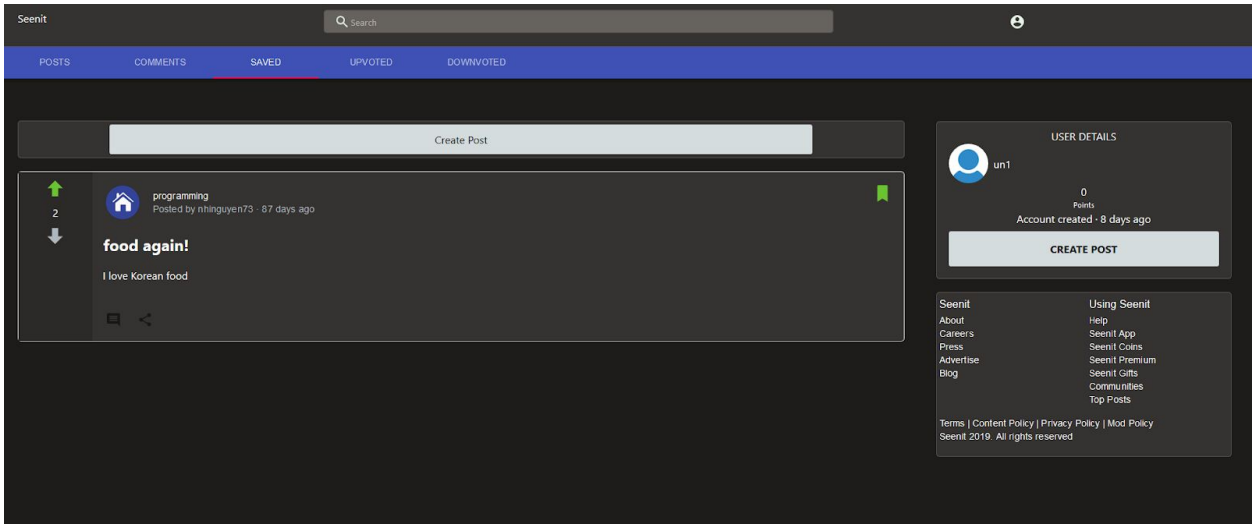
1. Dashboard: Click on “Posts”
    - All the posts that the user made on a particular channel, will be displayed



2. Dashboard: Click on “Comments”
    - All the comments that the user made will be displayed.



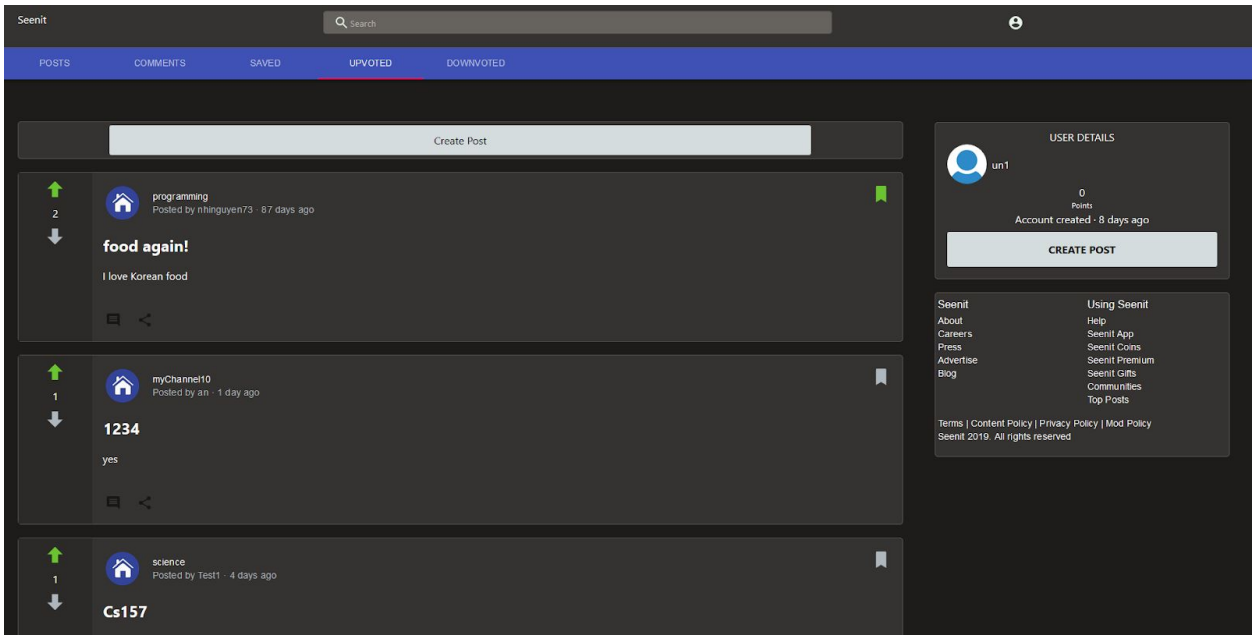
3. Dashboard: Click on “Saved”
    - All the post that the user saved will be displayed.



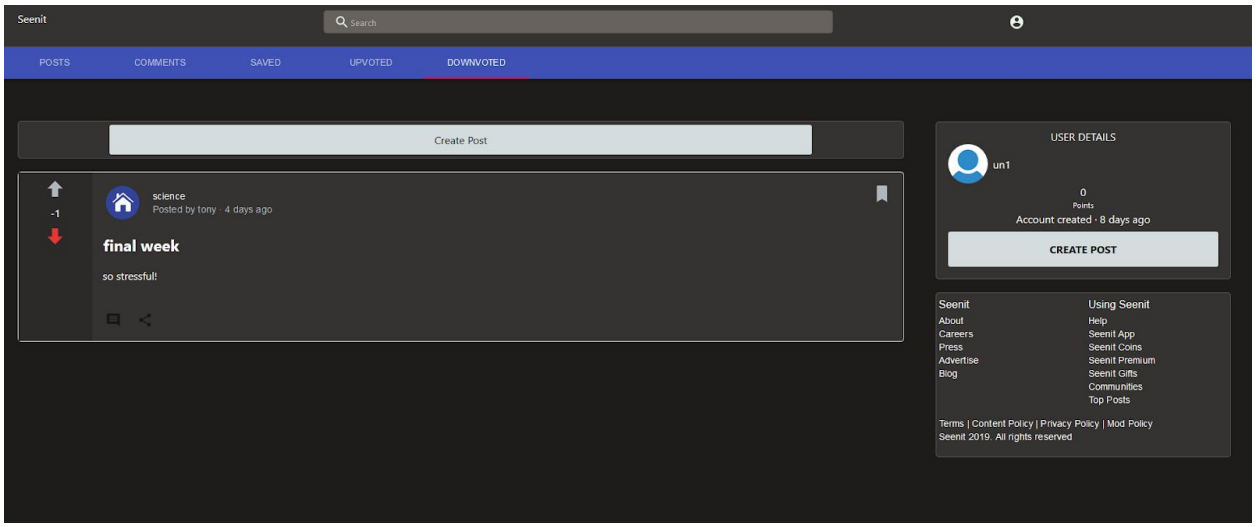
Query:

```
@Query("SELECT p, ch, ca.points, ca.user FROM Post p JOIN p.createdBy ca JOIN p.usersSavePost usp JOIN p.channel ch WHERE usp.id = ?1")
List<Object[]> findAllSavedPostsByUserId(String userId);
```

4. Dashboard: Click on “Upvoted”
    - All the post that the user likes will be displayed.



5. Dashboard: Click on “Downvoted”
    - All the post that the user dislike will be displayed.

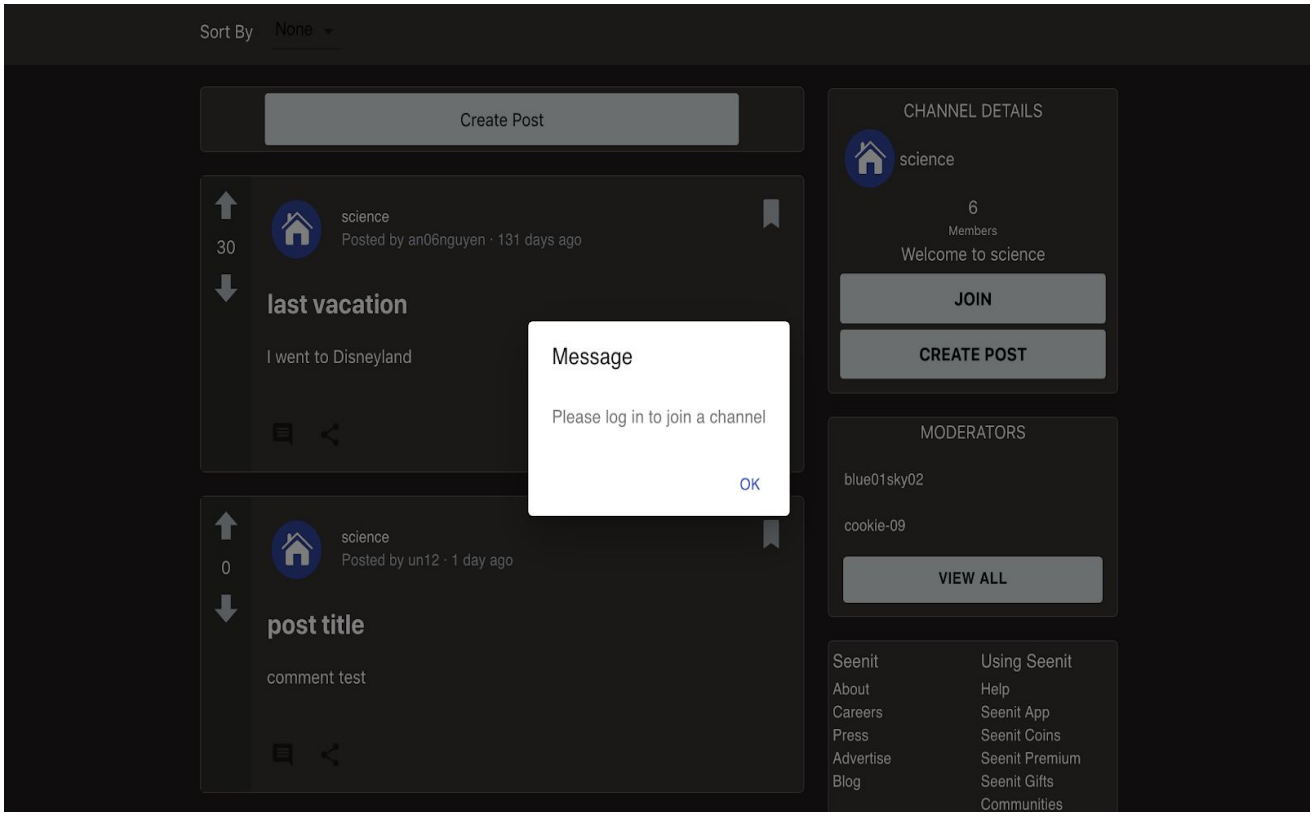


Query:

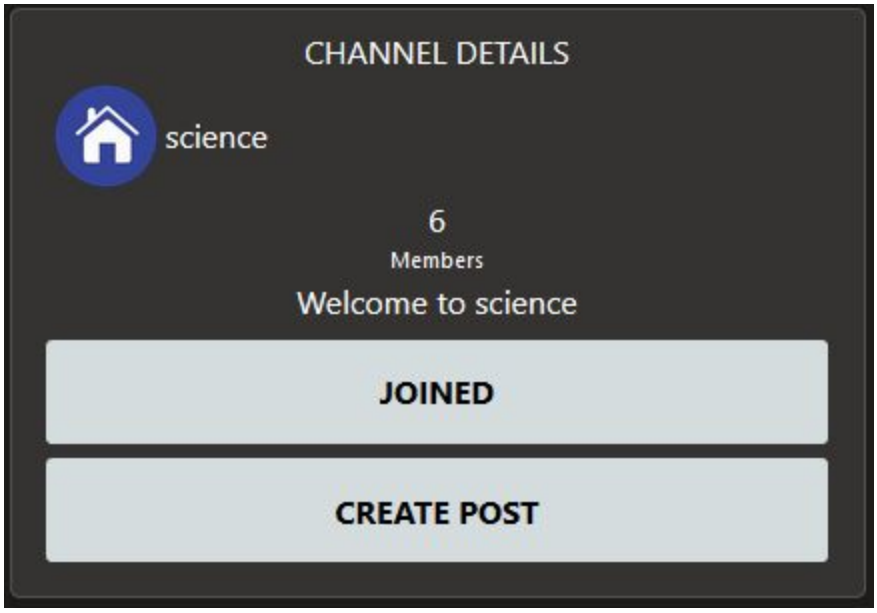
```
@Query("SELECT p, ch, ca.points, ca.user " +
    "FROM Post p JOIN p.createdBy ca JOIN p.channel ch JOIN p.usersVoted uv " +
    "WHERE uv.userVote.id = ?1 AND uv.isUp = ?2")
List<Object[]> findAllObjectByVoter(String userId, int up1OrDownNeg1);
```

Join a channel: to join a channel, choose a channel under Top Channel. It’s very important that user must join at least one channel to create a post. Click on “Join” to join a channel

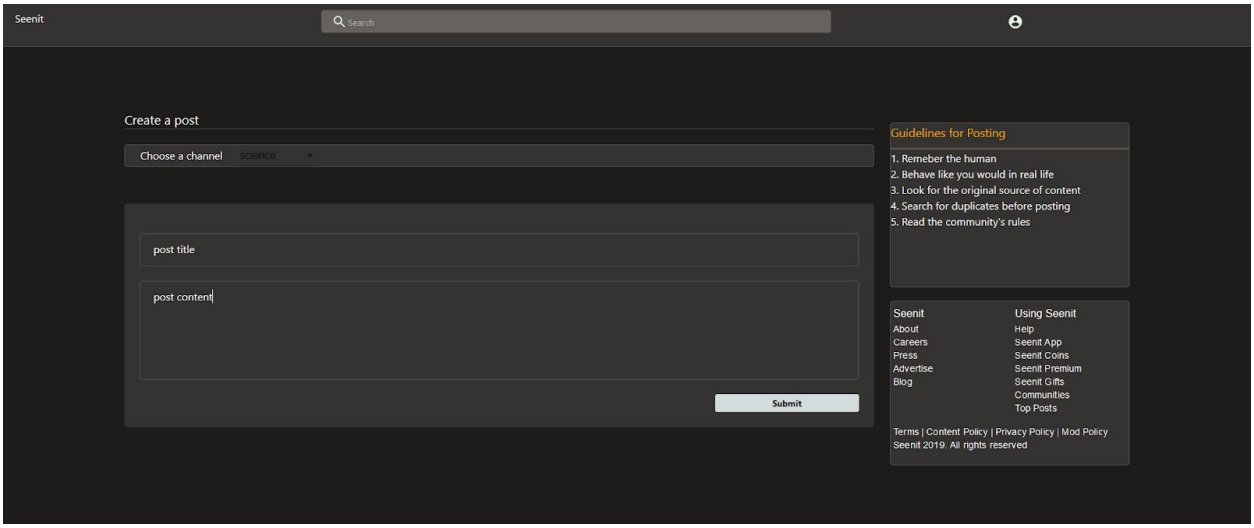
1. If users are not log-in, they will get a message that asks them to log-in



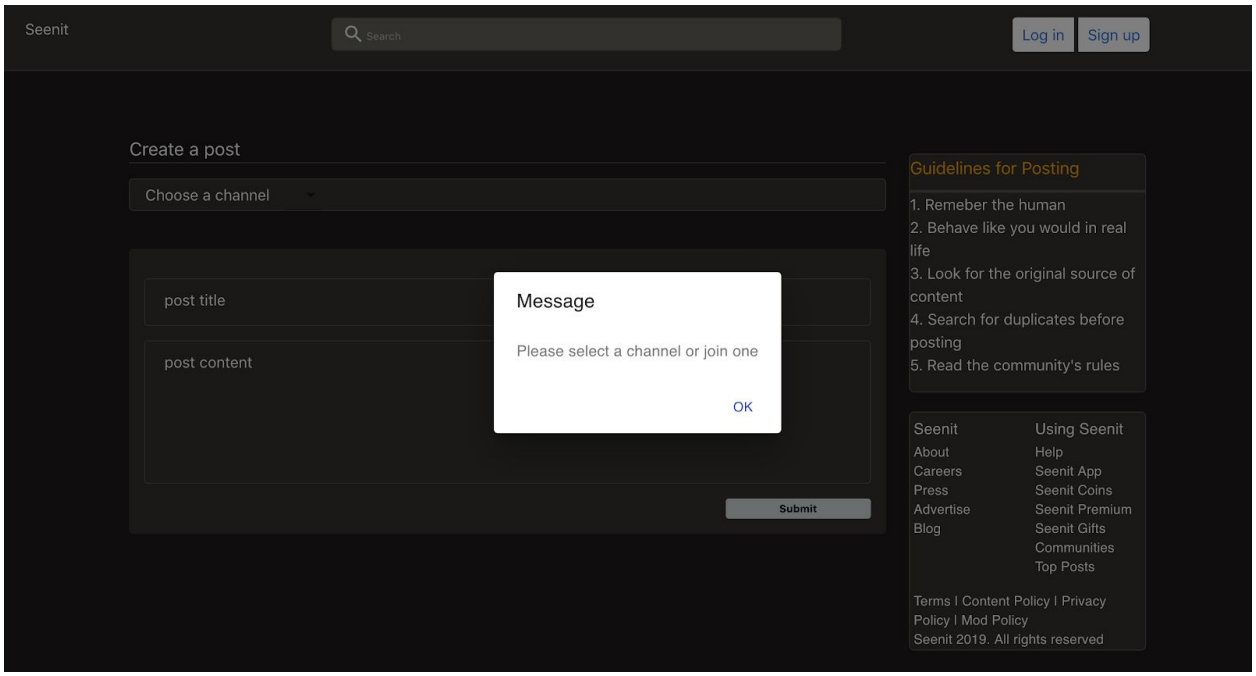
2. Otherwise, the number of members will be increased by one. If users want to leave channel, click on “Joined”.



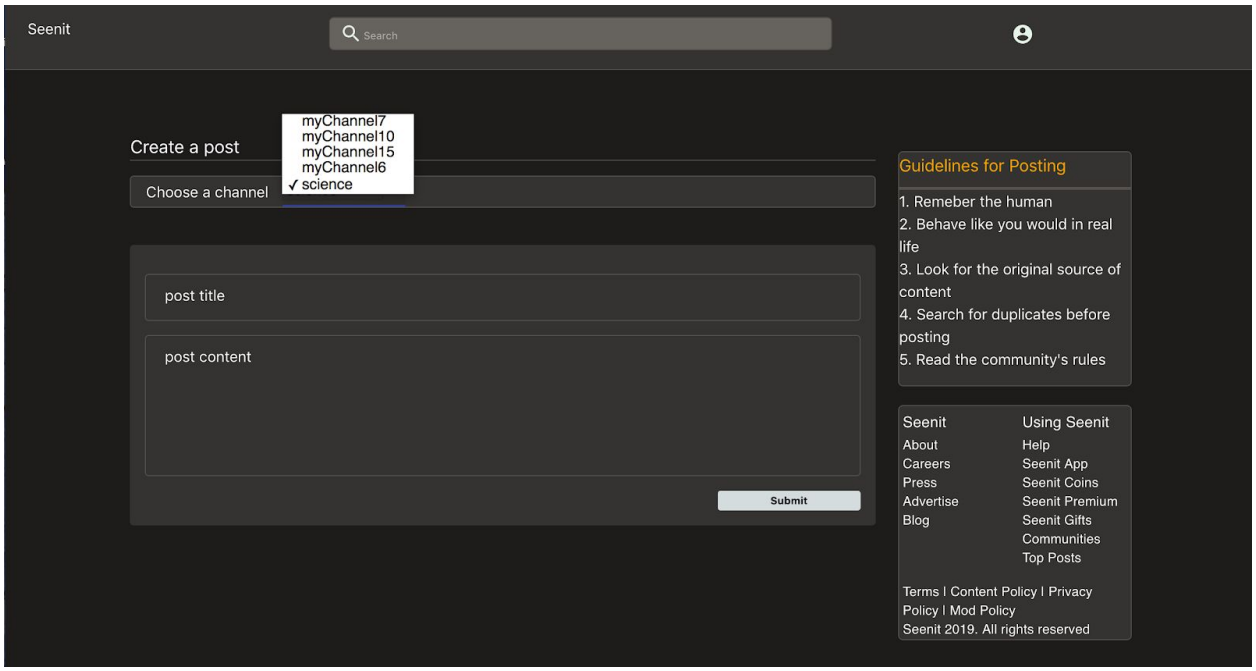
Create a post: click on “Create a Post”. Fill out channel, title, content, and click submit



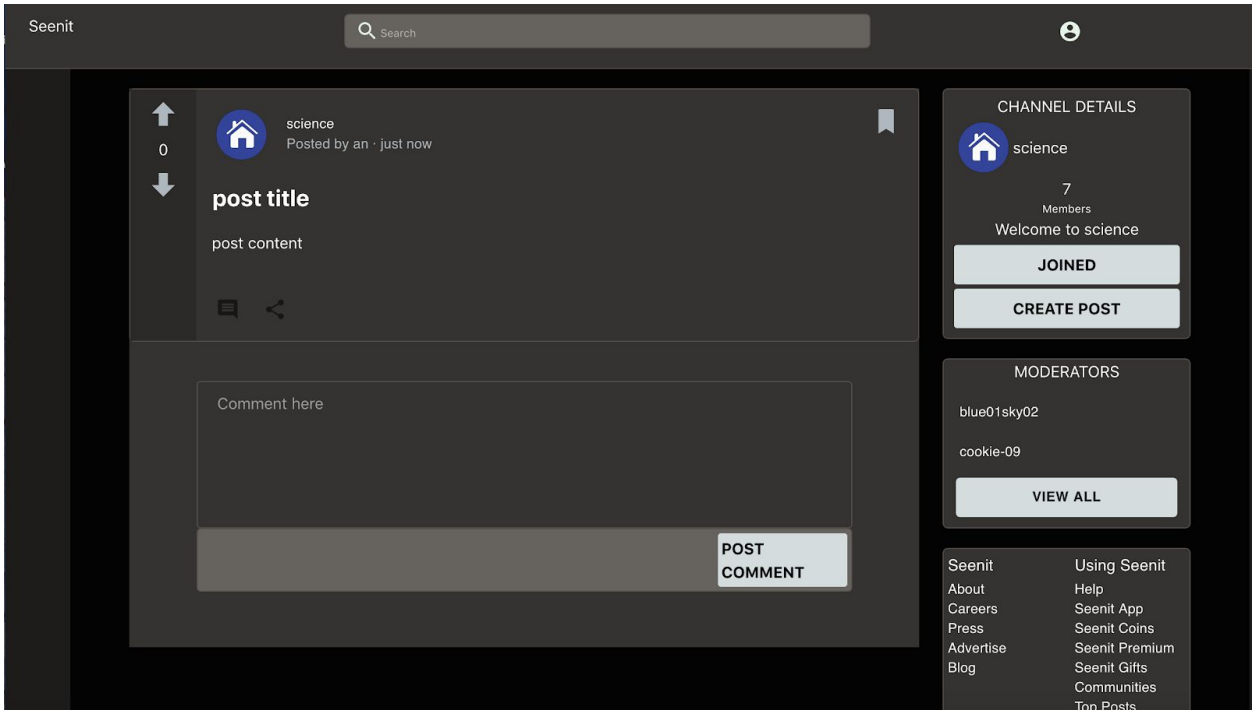
1. If users are not log-in, they’re not connected to any channels that they already joined. When they hit submit, a message is displayed as “Please select a channel or join one”. In other words, they have to log-in first.



2. If users are log-in, they should be connected to all channels. Select one to create a post.



- Successfully create a post. This post will be added to the channel associated with it.

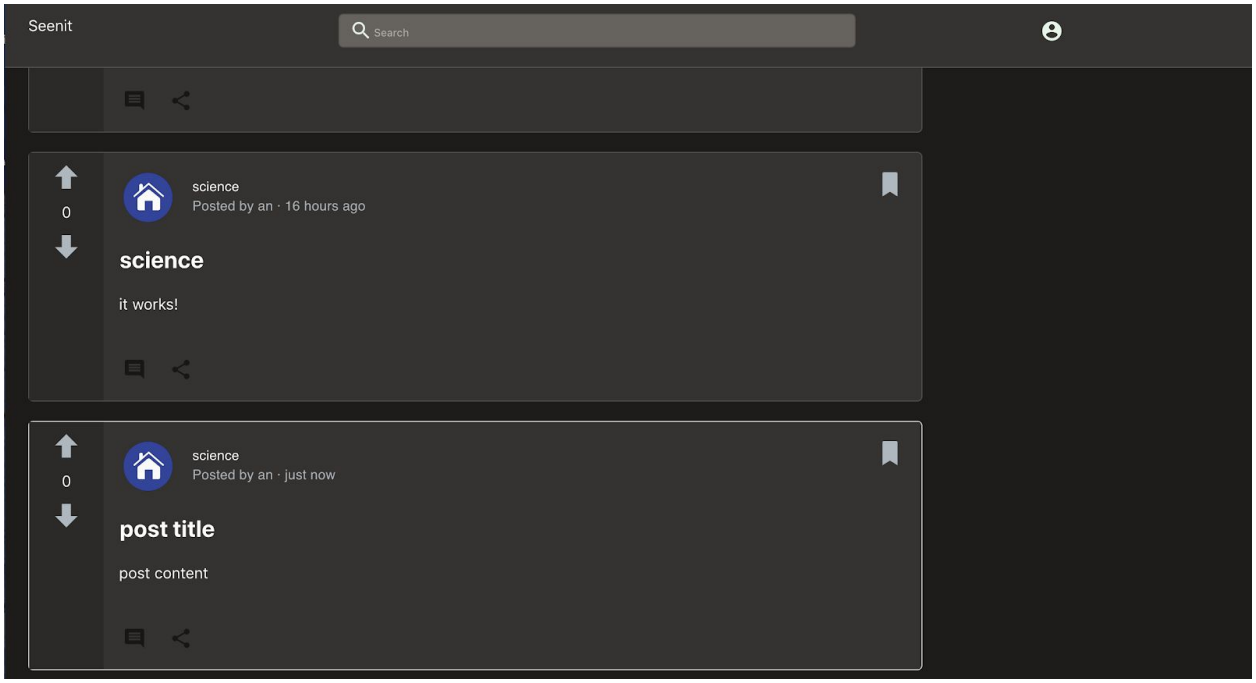


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	title	content	created_at	updated_at
01c7962d-6d4c-484f-80e8-d17788869b52	post title	post content	2019-12-11 20:00:38	2019-12-11 20:00:38
f2f1f7a9-87d1-4423-b2ba-be19d2a9a75b	science	it works!	2019-12-11 04:00:33	2019-12-11 04:00:33
049f815c-5108-4418-ac5b-bf40b69b2981	post title	comment test	2019-12-09 21:49:17	2019-12-09 21:49:17
0bfee067-c727-4970-b896-e299da229efe	bjb	nmkjk	2019-12-09 09:41:56	2019-12-09 09:41:56
8ee4806d-65f8-44a4-967a-61b0ff7d254a	bjb	nmkjk	2019-12-09 09:41:56	2019-12-09 09:41:56
cdda7357-8d22-456f-940e-d91521f90246	1234	yes	2019-12-09 09:41:10	2019-12-09 09:41:10
2874ae82-c883-45dc-ba50-4a3101bbc69f	Cs157	Great	2019-12-06 21:52:08	2019-12-06 21:52:08
9e5816ad-0cb0-4f52-af90-96ca5992449a	final week	so stressful!	2019-12-06 11:02:49	2019-12-06 11:02:49
b92d47e4-17e7-44cb-9e59-d03a167e40a6	test2	just a test!	2019-12-05 10:20:14	2019-12-05 10:20:14
da4bcd19-8810-43d3-a3c6-44b42fbd0c67	in class	presentation!	2019-12-03 22:39:37	2019-12-03 22:39:37
4028c265-d54b-4026-9342-88327a6dc7bd	test2	it's a test!	2019-12-03 22:34:28	2019-12-03 22:34:28
88241fb9-35c8-4113-9f65-b5f300ef3486	post science ...	comment test	2019-12-03 21:54:28	2019-12-03 21:54:28
09c33eed-13b0-46db-941b-509fec766068	post logged in	comment	2019-12-03 21:30:49	2019-12-03 21:30:49

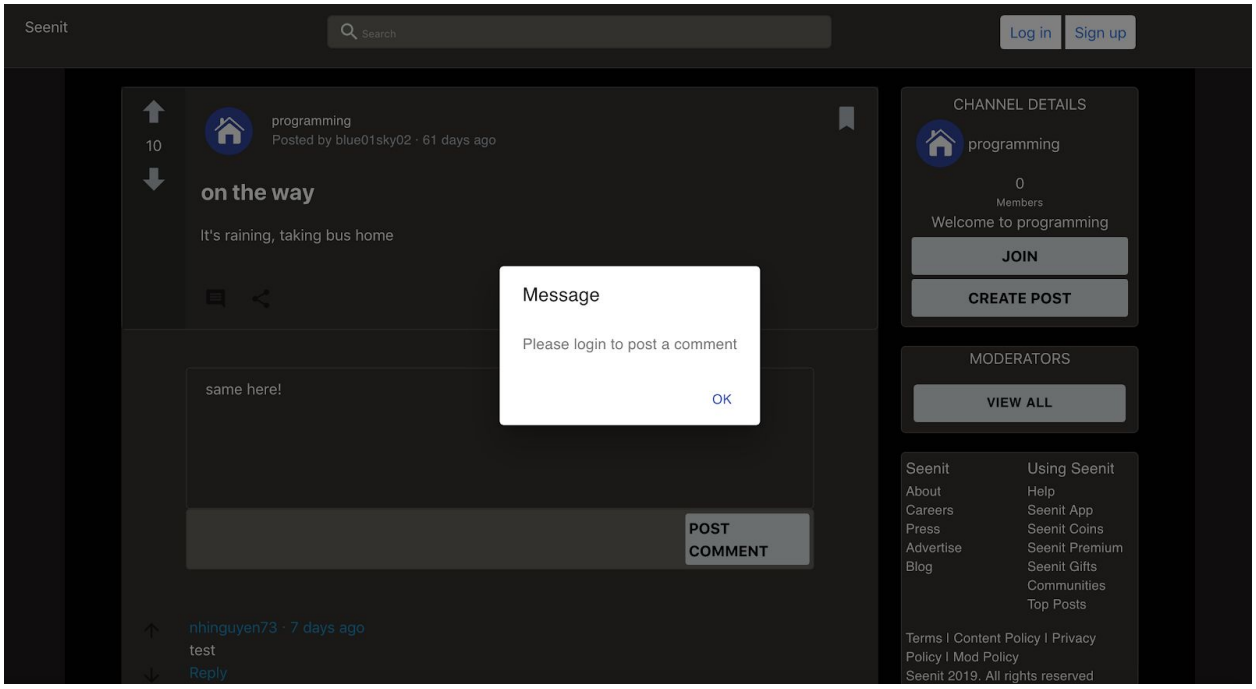
Posts 1

- The post should be displayed in the profile dashboard under “POSTS” as well.

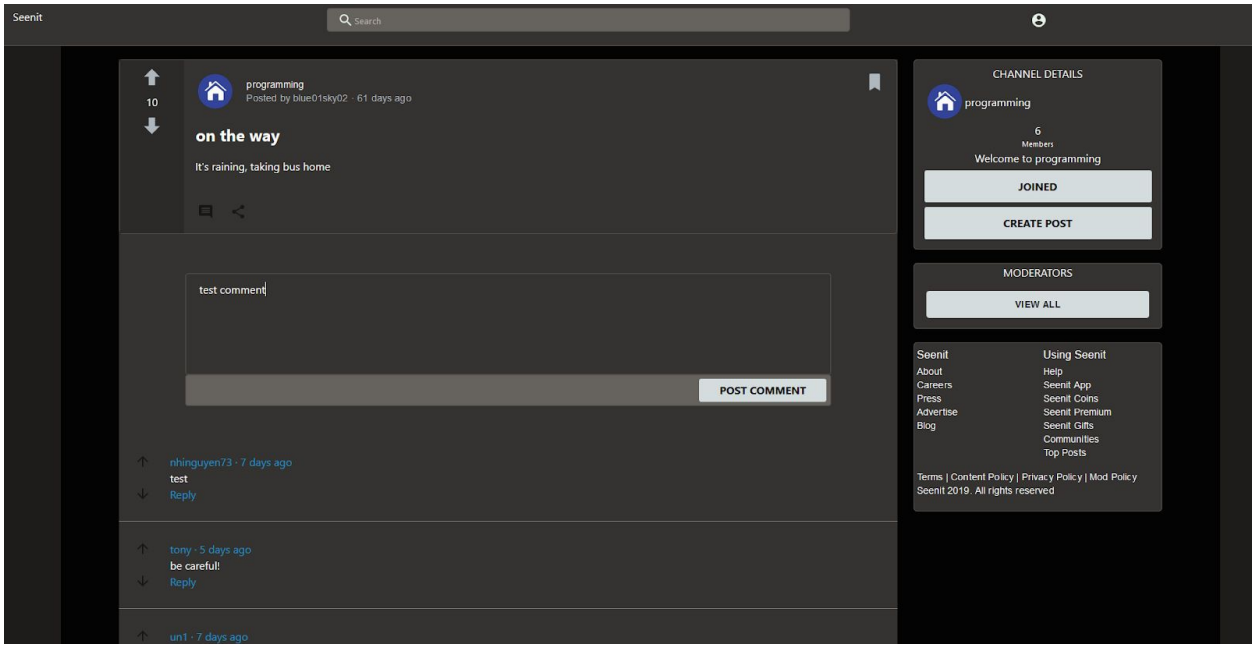


Comment on a post: Select a post. Fill out the comment field and click on “Post Comment”

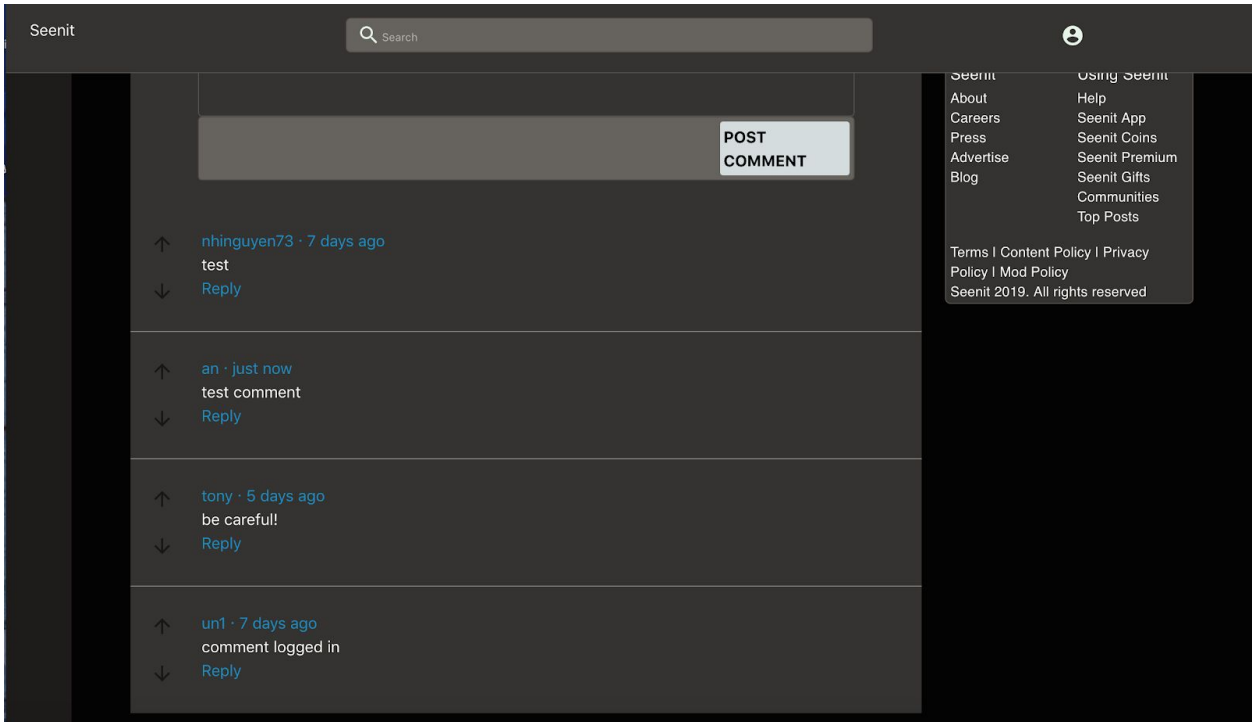
1. If users are not log-in, they will get an error message when they try to post a comment.



2. If users are log-in, they shall be able to post a comment under that post.



- For instance, “test comment” is added under the post

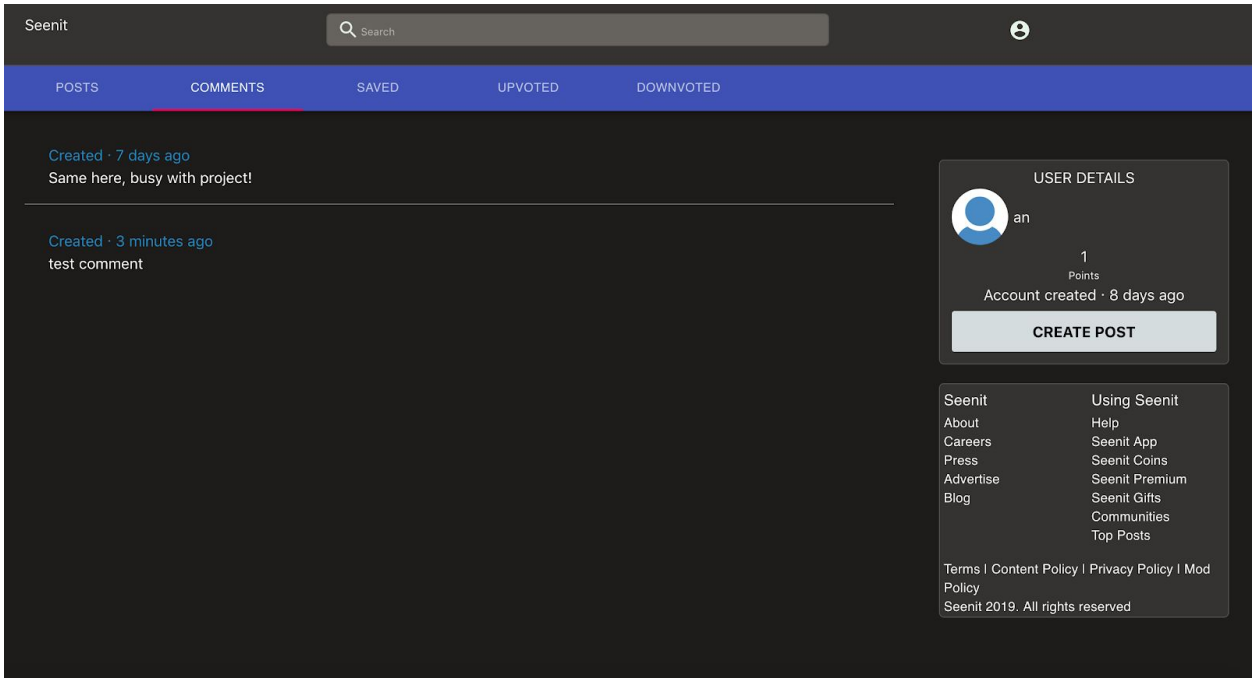


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	content	created_at	updated_at
803cbe84-40af-4eac-bc59-a4e5a3747d52	test comment	2019-12-11 19:44:12	2019-12-11 19:44:12
3e98ed07-923a-4cb2-827a-c247852ade4a	hi	2019-12-06 21:53:36	2019-12-06 21:53:36
089ae706-4a0a-47ca-ae38-707357691ea3	be careful!	2019-12-06 11:01:03	2019-12-06 11:01:03
bbbe7519-ba1f-49b7-8c0e-2c0a6d3e8802	Test comment	2019-12-05 04:48:34	2019-12-05 04:48:34
b1af5b67-457b-4bd4-b706-0543acf87ec5	Same here, busy with proje...	2019-12-03 22:36:24	2019-12-03 22:36:24
09b321f8-9759-443d-adf1-932a86794f98	comment reply	2019-12-03 21:54:40	2019-12-03 21:54:40
ff55b16c-a454-4873-8d17-a1022dcbf7cd	logged in comment reply	2019-12-03 21:38:24	2019-12-03 21:38:24
0fc3dca1-3b2d-451b-bfad-b0c3bc74167b	comment logged in	2019-12-03 21:29:29	2019-12-03 21:29:29
6557931a-1065-4f42-b5e7-e105e83a2c3b	Another one	2019-12-03 21:03:48	2019-12-03 21:03:48
8cbfedda-263a-452d-b424-795c0855bb51	test add comment	2019-12-03 20:59:37	2019-12-03 20:59:37
4f58585b-77e9-462f-ab1f-7ac53bba2d9b	test	2019-12-03 20:58:25	2019-12-03 20:58:25
30	Formal Languages	2019-10-15 09:50:36	2019-10-15 09:50:36
29	Comp Architecture	2019-10-15 09:49:36	2019-10-15 09:49:36

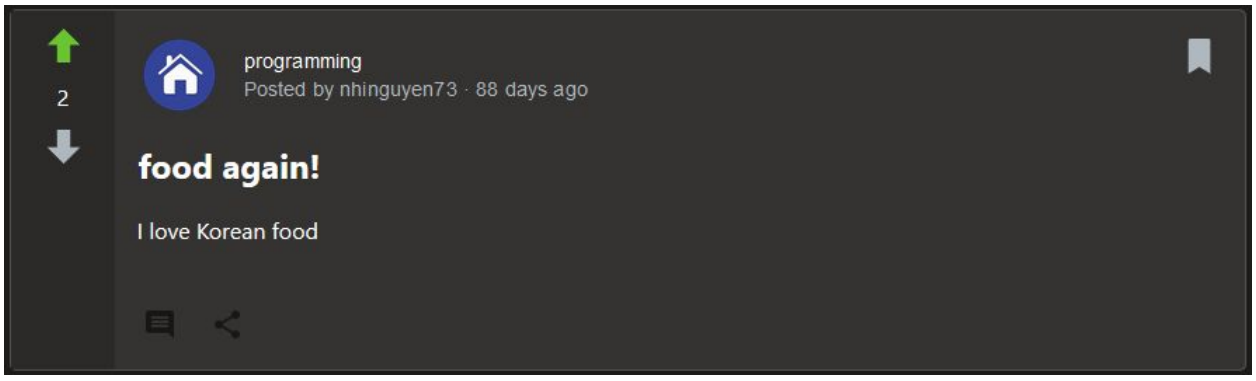
Comments 1

- This comment will be also displayed in profile dashboard under “COMMENTS”



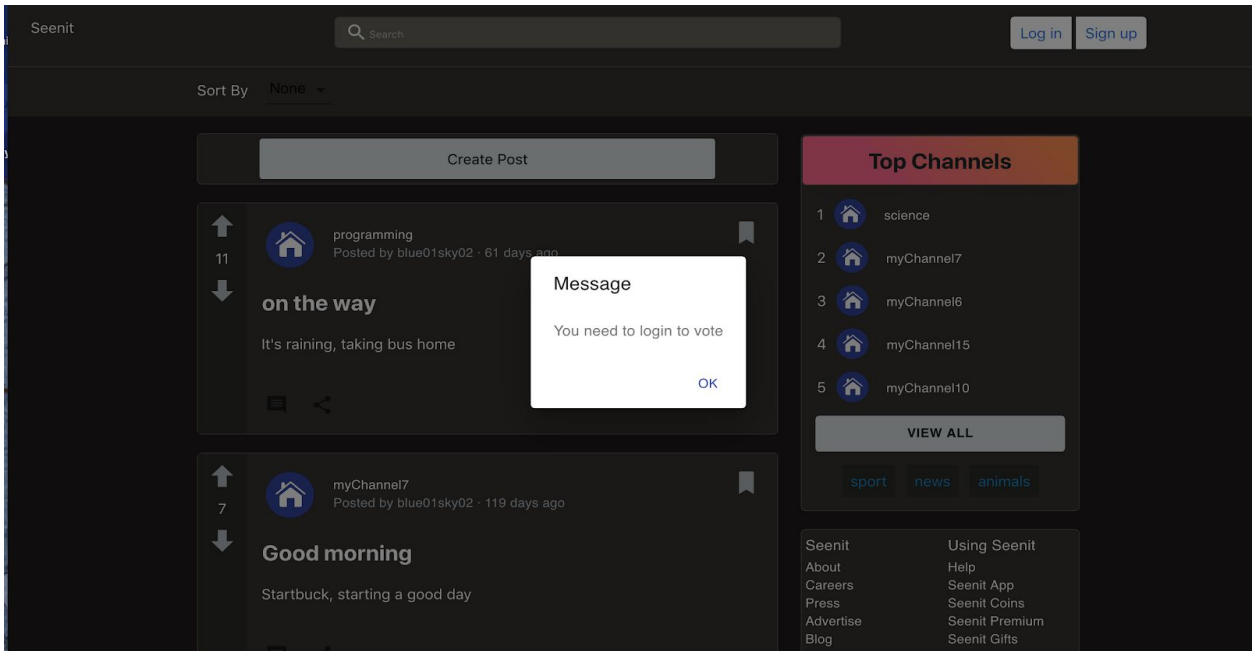
Save a post: click on the bookmark icon, to save a post

Click on the bookmark icon again to unsave it

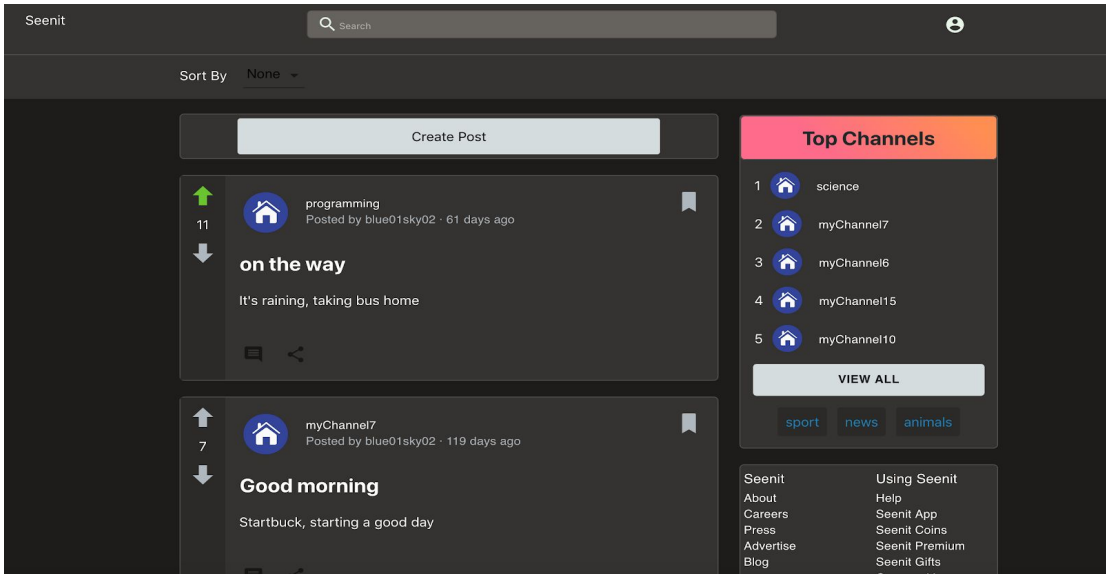


Upvote: Click on the upward arrow on the left side of each post.

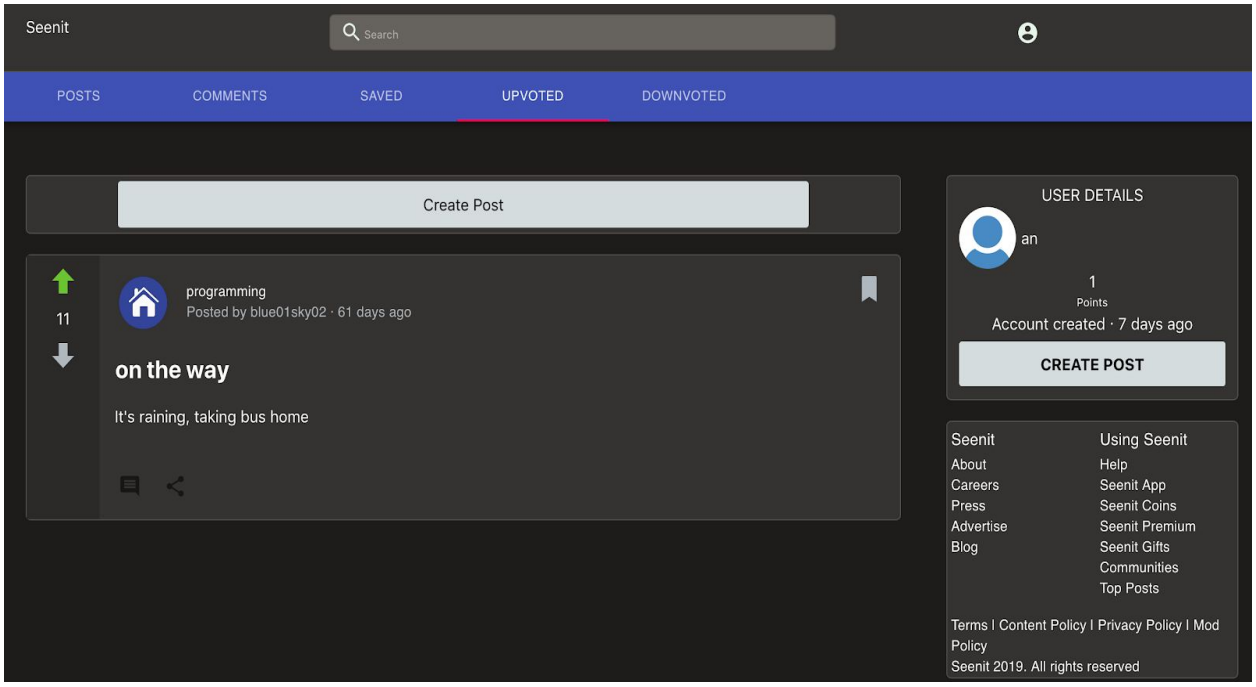
1. If users are not log-in, they will get an error message.



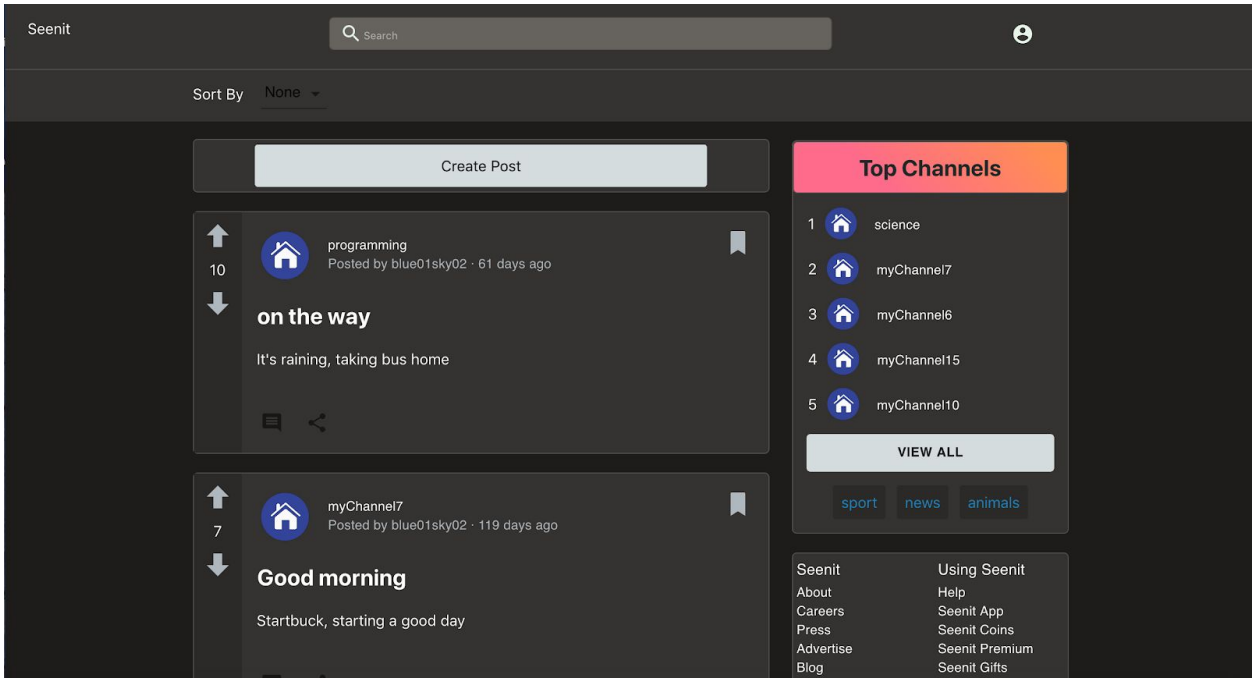
2. If the users are log-in, they will see a green upward arrow. The number increases from 10 to 11.



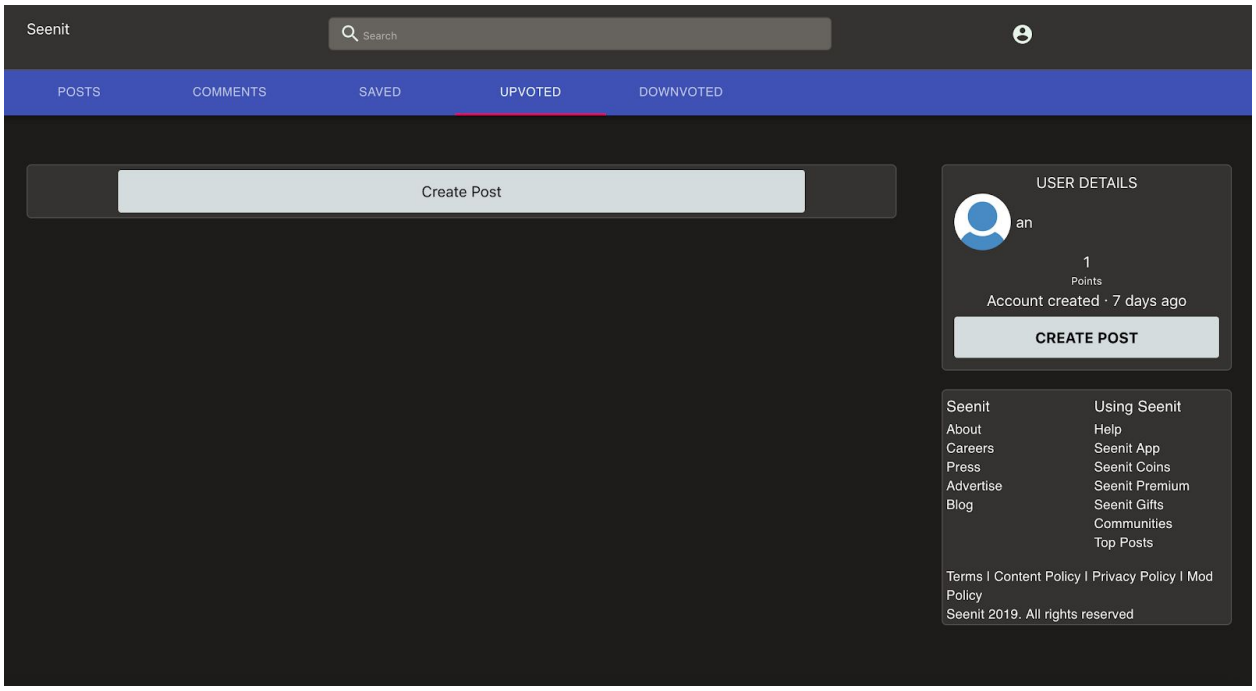
- The upvoted post will be saved in personal dashboard under “Upvoted”



- To unvote, click on the green arrow. The vote will be decreased from 11 to 10.

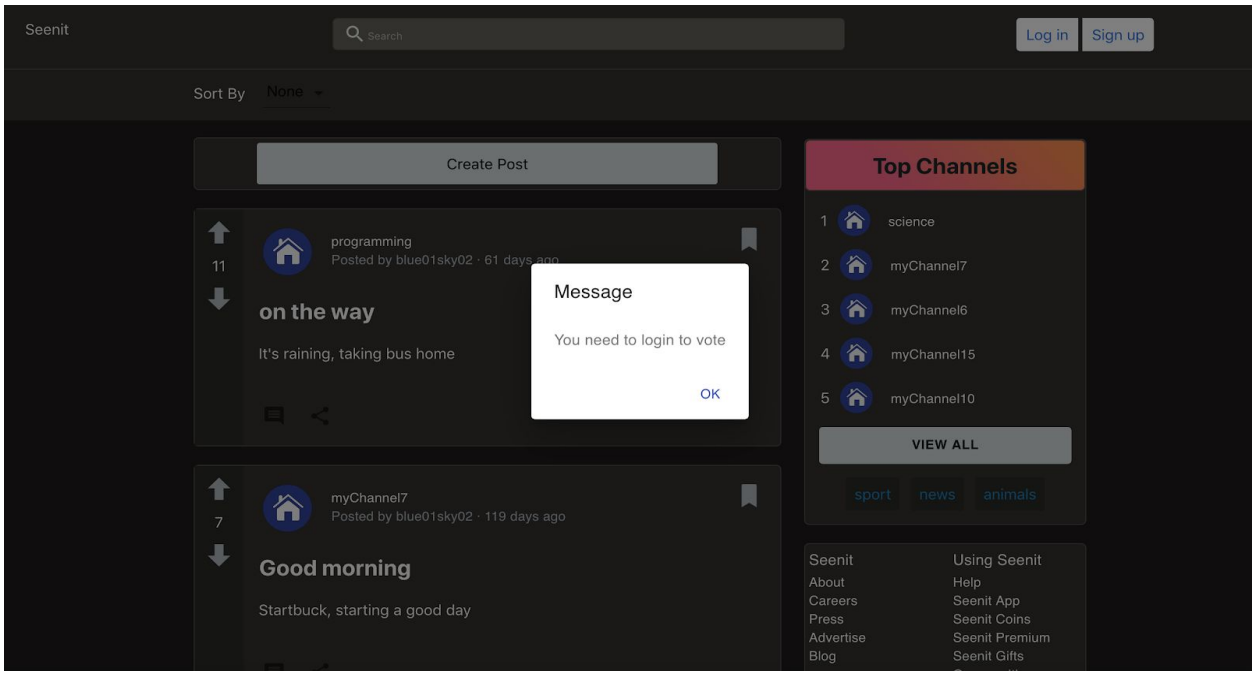


- The unvote post will be removed from the Downvoted in personal dashboard.

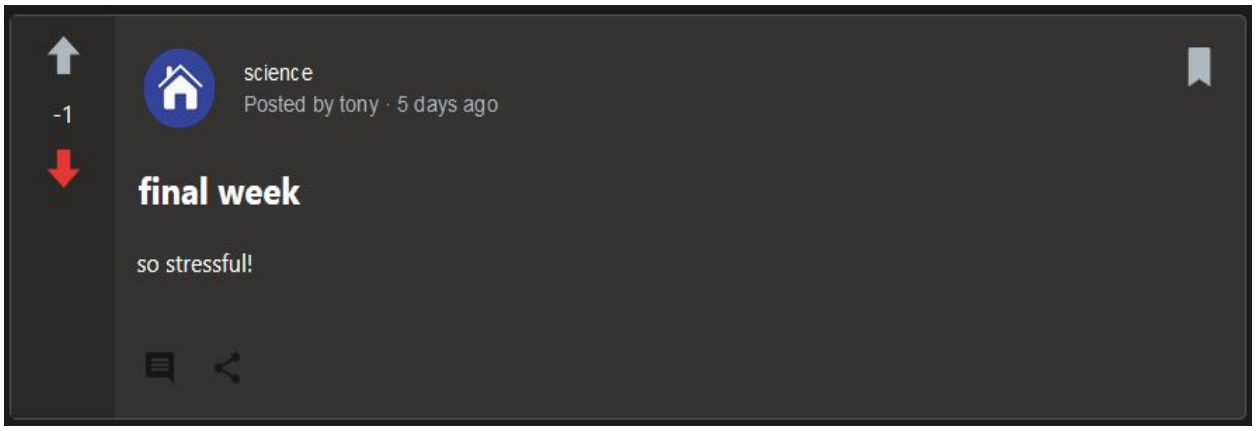


Downvote: Click on the downward arrow on the left side of each post.

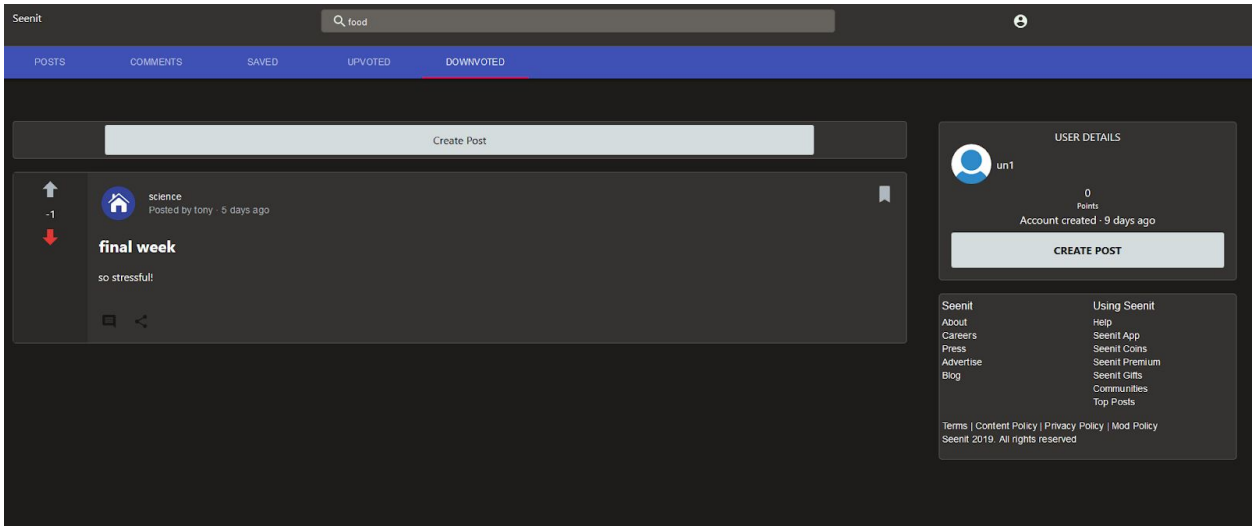
1. If users are not log-in, they will get an error message.



2. If the users are log-in, they will see a red downward arrow.



- The downvoted post will be saved in personal dashboard under “Downvoted”



Procedures (step by step) of how to set up and run your system

How To Try Locally

Requirements:

- Git
- Node.js (version >=10.0.0) (which comes with npm)
- JDK (version >=1.8)
- Maven (version >=3.2)
- Spring-2.1.8 RELEASE
- MySQL (version >=5.1.47)

If you use an appropriate IDE, you don't need to manually install Maven and Spring.

From your command line:

```
# Clone this repository
$ git clone https://github.com/CS157A-Team24/Seenit
$ cd Seenit
```

Client side

```
# Go into the client repository
$ cd client
# Install dependencies
$ npm install or yarn install
# Run the app
$ npm start or yarn start
```

In case the website doesn't automatically pop up, go to http://localhost:3000/

Server side

Simply use an IDE such as IntelliJ and NetBeans, import/open the project and hit "run" or if you refer command line:

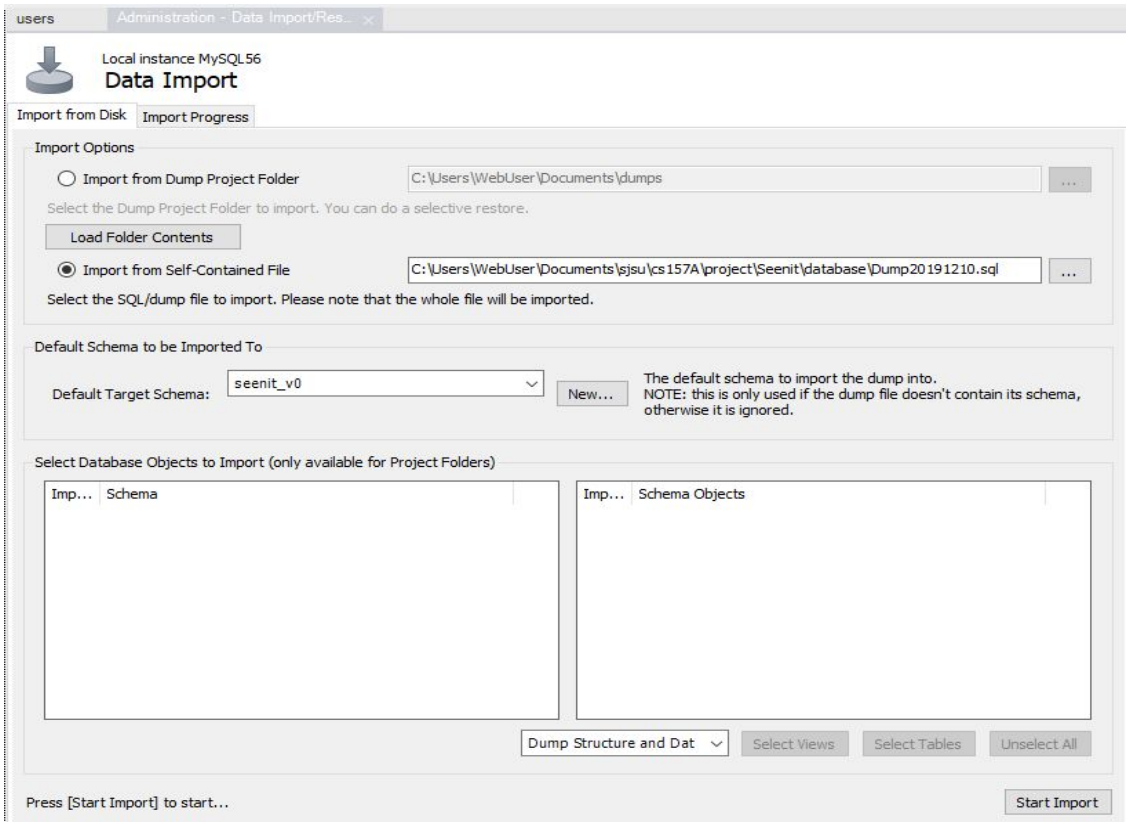
```
# Go into the server repository
$ cd server
# Compile and run
$ mvn spring-boot:run
```

Default port at http://localhost:8080/. Configure the application.properties file to match your MySQL server address if it isn't at localhost:3306. (Note: You can use our public database server which is already setup, so you don't need to modify anything)

Database

Setting up local Database:

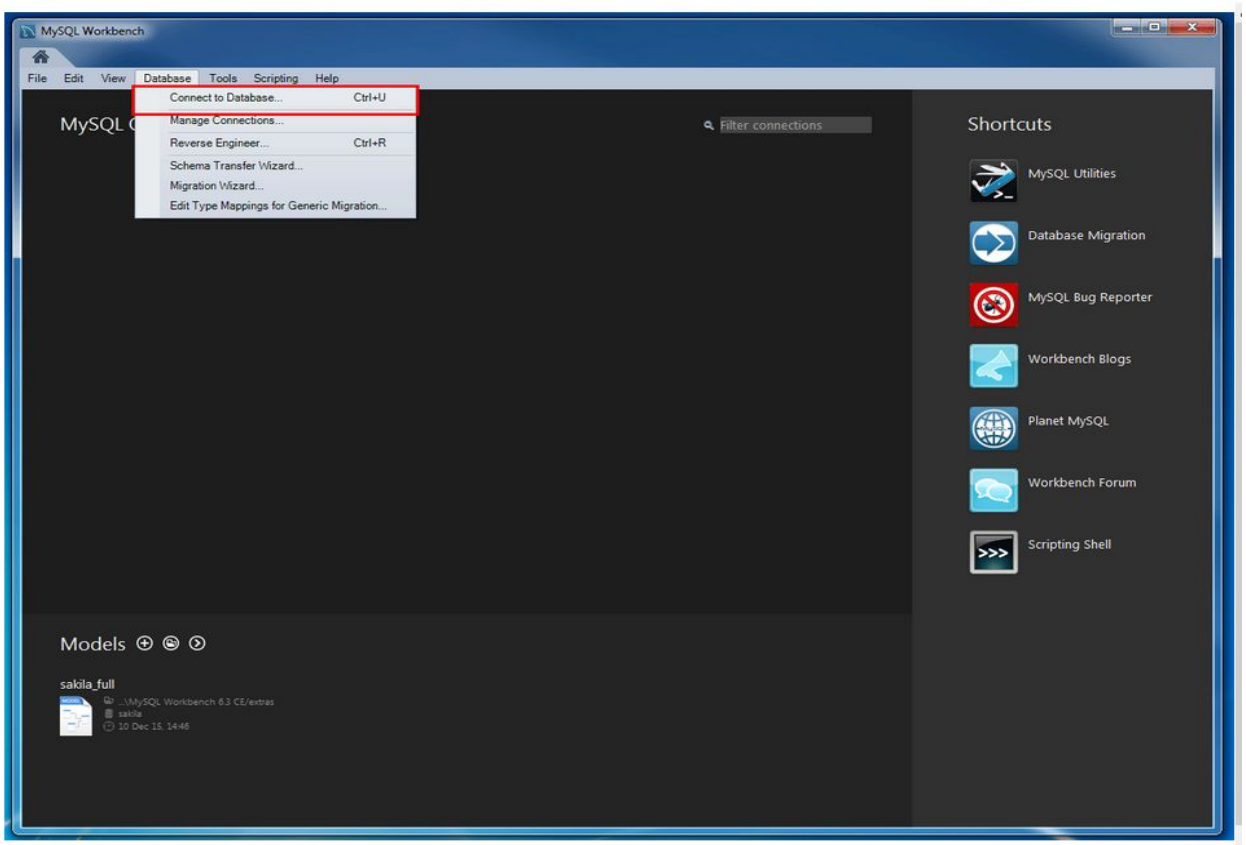
In MySQLWorkbench use a local server, go to Server -> Data Import, select Import from Self-Contained File, select the Dump20191210.sql in the database folder. In Default Target Schema create a new schema called "seenit_v0". Make sure "Dump Structure and Data" is selected. Click "Start Import".



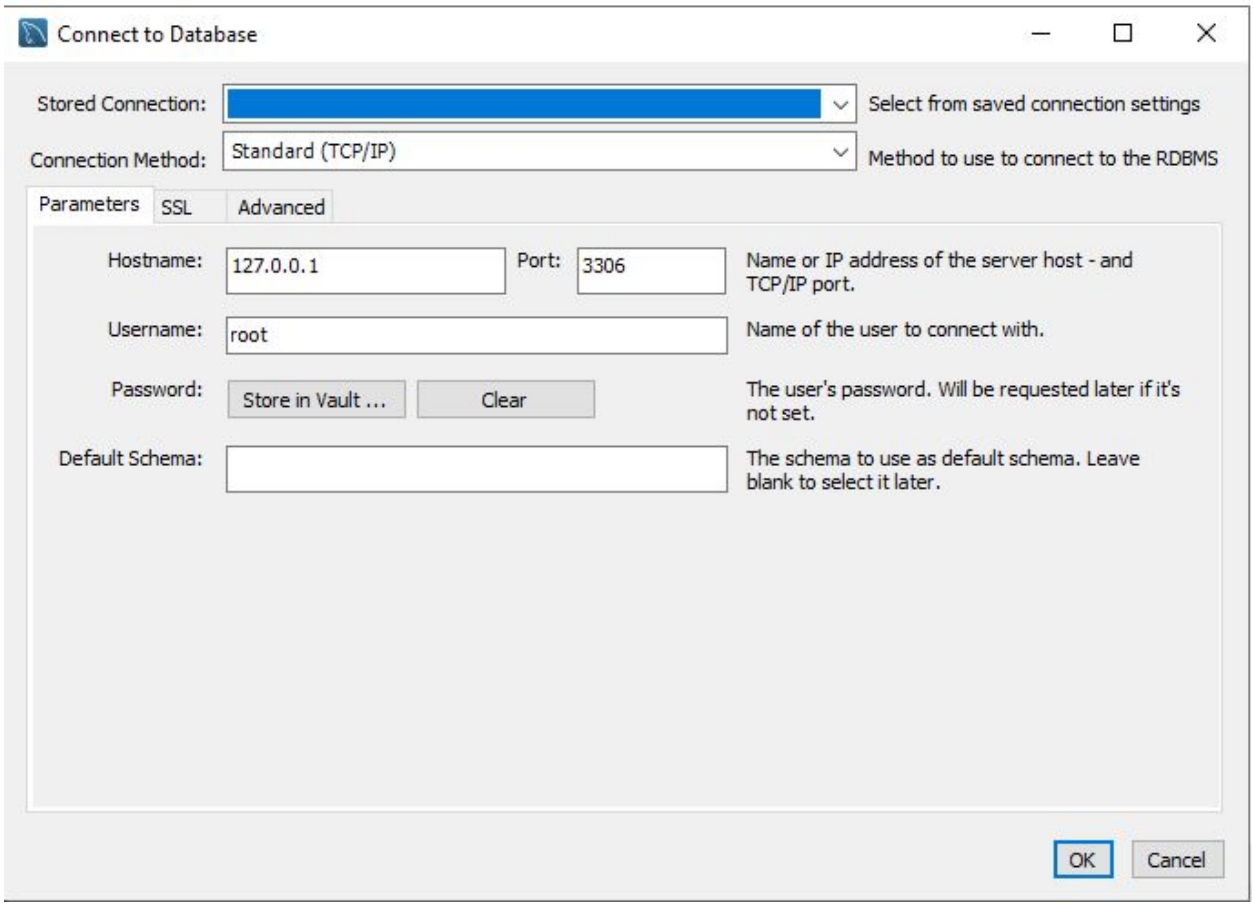
Edit server/src/main/resources/application.properties file, first 3 lines:

```
spring.datasource.url =
jdbc:mysql://localhost:3306/seenit_v0?useSSL=false
spring.datasource.username = <local MySQL server username>
spring.datasource.password = <local MySQL server password>
```

Note: If you use our hosted database server, you don't need to manually add data to your local host. Following the below instruction to connect the workbench to the cloud database



hostname: seenit-0.cxolnqt4sckz.us-west-1.rds.amazonaws.com
username: admin
password: seenit321



Project Conclusion

Statements from each team member about Lesson Learned from this DB project

Chaz Chang: Before this class I had no experience with building a 3-tier architecture, React, and Spring Boot. I only had a little experience in MySQL. I definitely learned a lot from this project. I learned about how 3-tier architectures handle security with sessions, JWT (this was used in our project), OAuth, etc. I learned how to use BCrypt to store encrypted passwords in the MySQL database. I learned how to use MVC pattern with Spring Boot to build a Rest API and handle requests from the front end. I learned how to write JPQL queries to retrieve information from MySQL. I learned how to use React to provide feedback while the user is filling out fields and not just when they submit the form.

Viet Dinh: I have built many web applications using React, but not Redux. I tried Redux before but had never built something the size of our project. After finished our project, I have more confidence to use Redux and I really like it. Debugging the state of React using Redux is amazing. This is my first time using Spring Boot, the reason why I chose

Spring Boot is because I did some research and found out it really simple to work with SQL databases. Although it has a lot of overhead to map classes to tables when there is a relationship that has extra attributes, it really simple to communicate with database from there. I learn to build a better Rest API using Spring Boot that can store data to a database server. The important lessons I learned from this class are how to use MySQL database, design and manage database.

My Nguyen: I have no experience in developing a web application before. We use ReactJS, HTML, CSS in the front end, but I don't know much about them. I learned how to use the basic CSS and HTML, and ReactJS to contribute to the frontend. Besides, I know nothing about Spring Boot, so it took time to learn and understand it fully. It's interesting to learn how to build a better Rest API using Spring Boot to store data to a database server. Since I mostly worked in the front end, I learned how to handle requests from the front end. I learned how the backend works as well based on the written code by other team members. Moreover, I learned how to design the project, identify the functional requirements, conduct ERD diagram, and convert them to schemas. By doing the project requirements and project design in the beginning, I have a more clear idea of our goal or what kind of application we want to develop.

Future improvement of your DB application

- The https protocol can be used for our web server to make sure the connection between users and server are encrypted
- NGINX can be used on the web server. NGINX can be used as a load balancer to distribute the work over many computers. Load balancing will help create high throughput and low response time by avoiding overloading a single computer. NGINX will allow more computing resources (CPU and memory) to be easily added to the existing system.
- Delete and create channel features
- Update and delete account features