University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ___4____

Date: ____Dec 1_____

Group Number: ____119_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Stuart Chen	48414957	r4v4j	stuartcc6@gmail.com
Allan Xing	28532901	p7i4r	allanx01@students.cs.ubc.ca
Eric Fu	57440844	y2c8w	ericfu55@student.ubc.ca

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

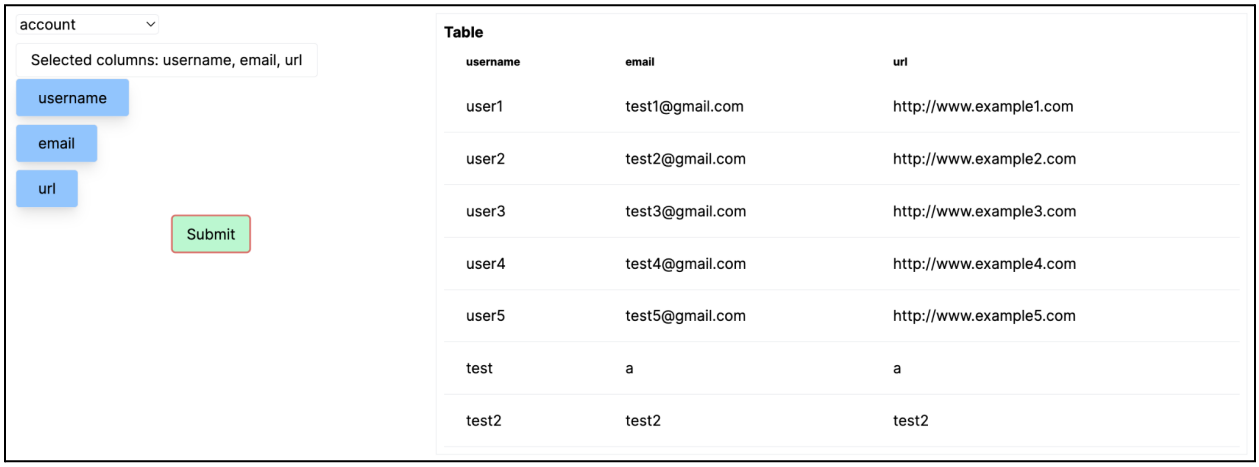
University of British Columbia, Vancouver
Department of Computer Science

Description:
We made a social media platform similar to applications like Instagram, Twitter/X, and Facebook, using PostgreSQL to store all of our information. An essential feature we created was allowing users to register an account with us and login with their credentials. They are able to communicate with other users in real time using a chat room feature, and they can choose to save and delete chat groups. Users can also create posts, or delete posts that they have already created. We have created the ability to interact with different kinds of posts through viewing and liking. Users are also able to follow and unfollow each other, and if a user is following someone then they can see that person's posts on the home page.

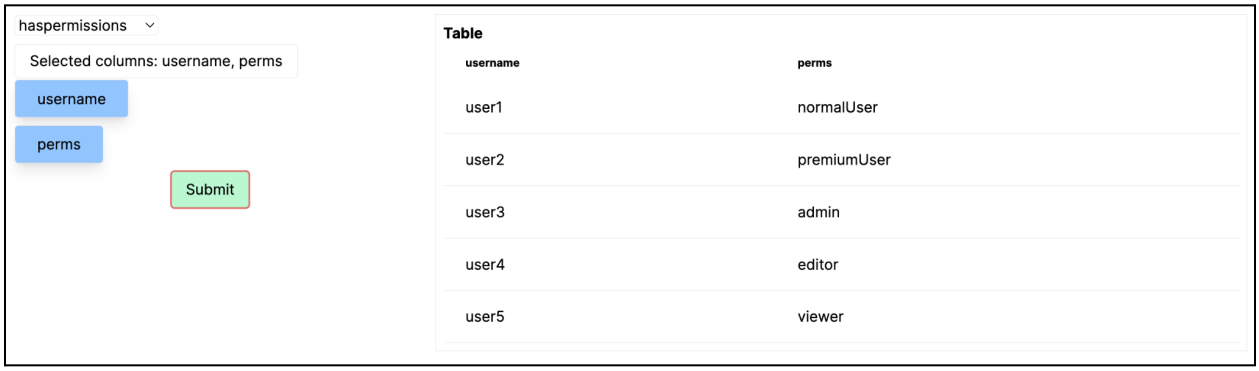
Difference between old and new schema:
There are some small differences between the old and new schema. In the new schema, we decided it would be easiest to turn the chatId into a SERIAL field because in PostgreSQL it would automatically increment and use a value, serving as the id for the chat. This allows us to automatically generate new ChatIds without any additional work. Also, we added a new field called directedTo in messages because one of the requirements for the project needed us to be able to update 2 non-primary key fields.

Copy of the Schema

- Account(<u>username: string</u>, email: string, URL: string)

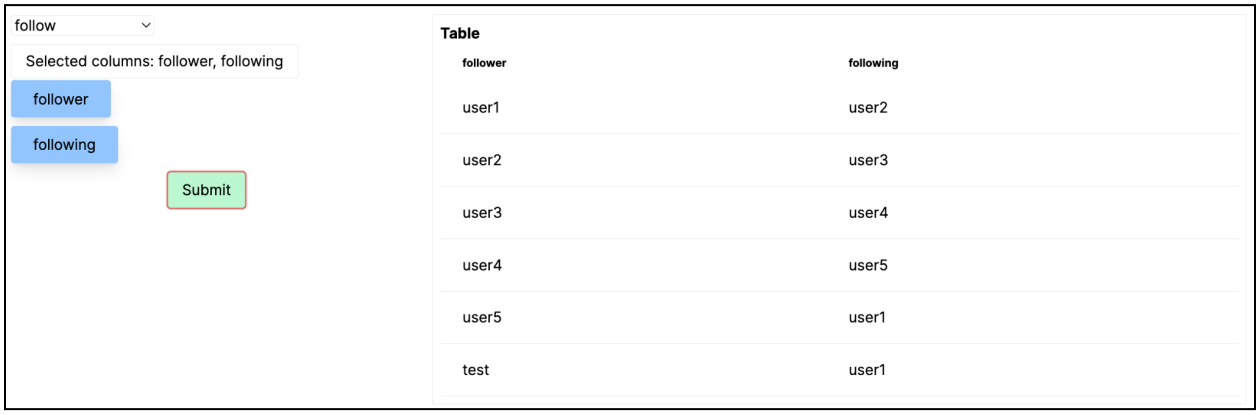


- <u>Relationship:</u> hasPermissions(<u>**user: string**</u>, **perms: string**)

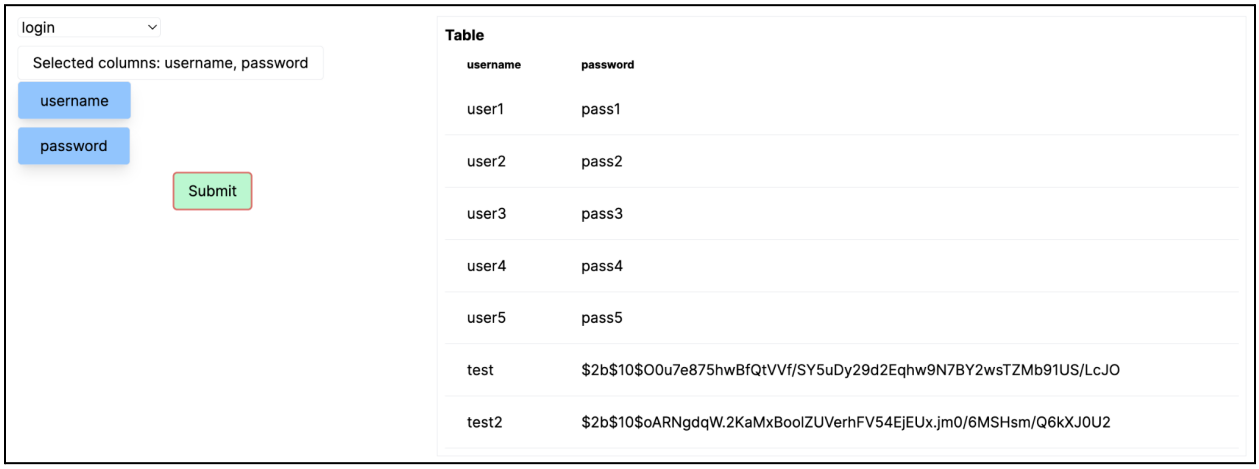


University of British Columbia, Vancouver
Department of Computer Science

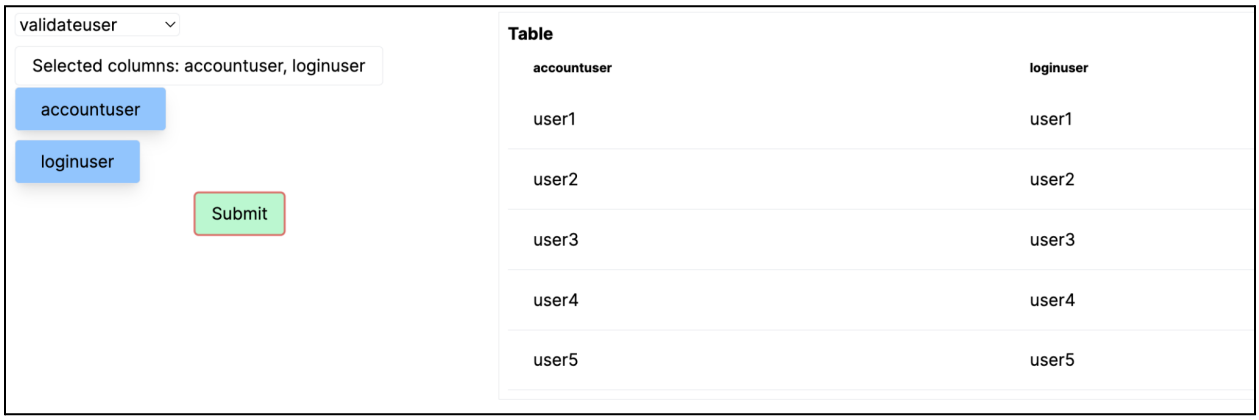
- Relationship: follow(**follower: string**, **following: string**)



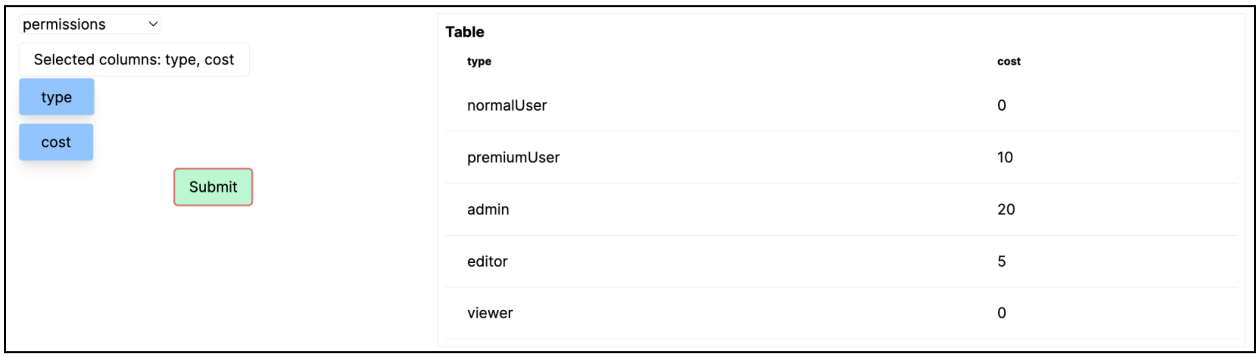
- Login(Username: string, Password: string)



- Relationship: validateUser(**accountUser: string**, **loginUser: string**)

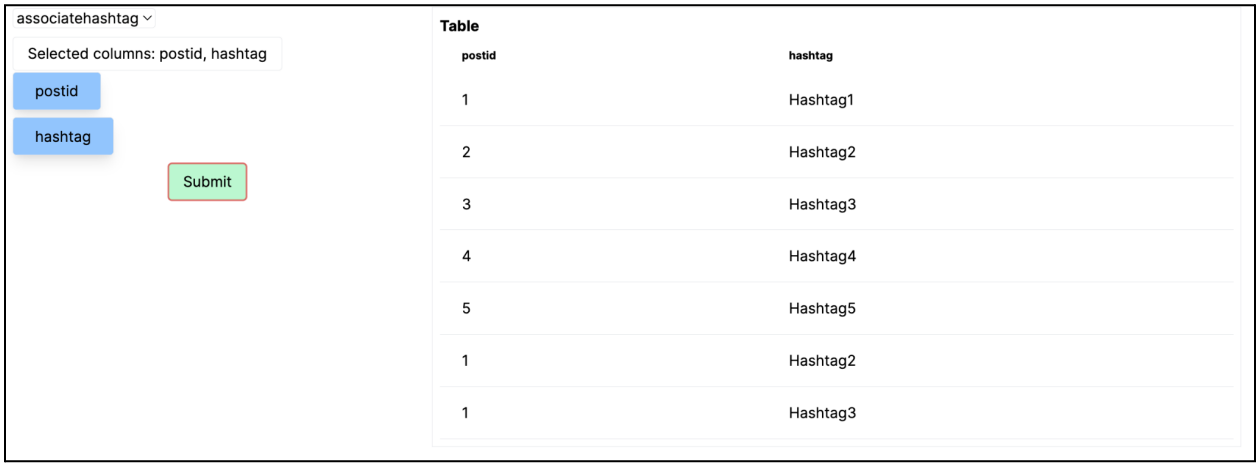


- Permissions(type: string, cost: number)

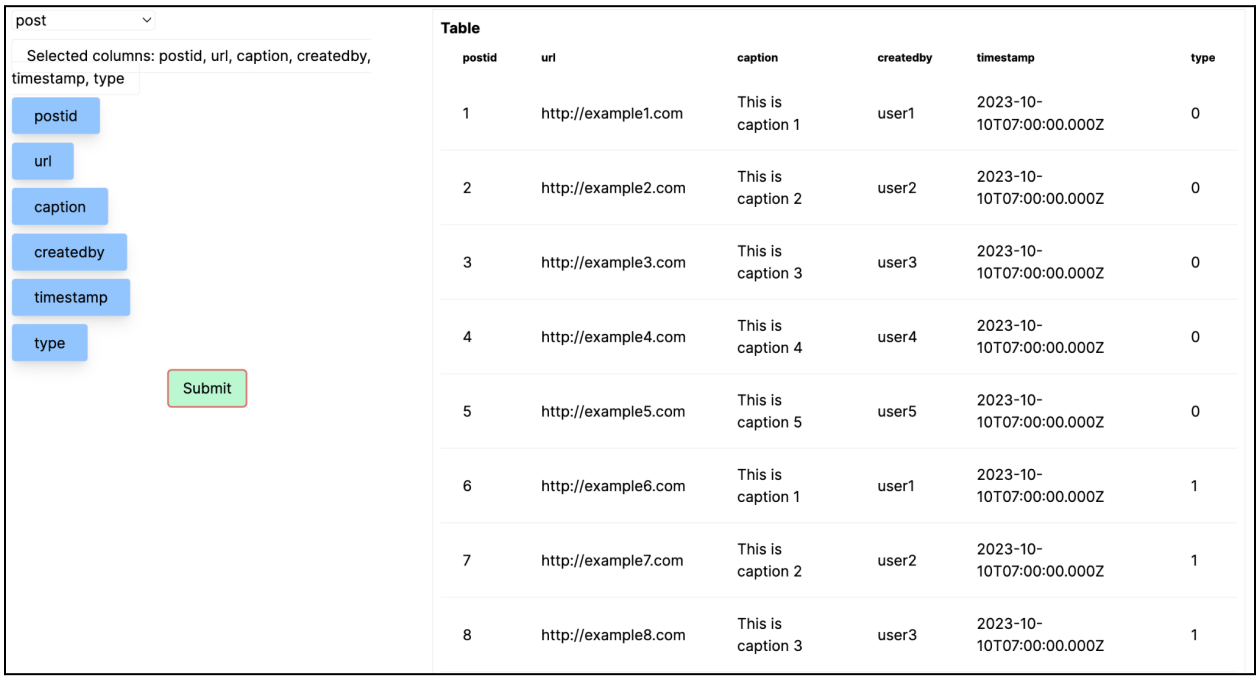


- Hashtag(text: string, color: string)

- Relationship: associateHashtag(**postID: number**, hashTag: string)



- Post(<u>postID: number,</u> URL: string, caption: string, advertisement?: number, **createdBy: string**, type: number)



postid	url	caption	createdby	timestamp	type
1	http://example1.com	This is caption 1	user1	2023-10-10T07:00:00.000Z	0
2	http://example2.com	This is caption 2	user2	2023-10-10T07:00:00.000Z	0
3	http://example3.com	This is caption 3	user3	2023-10-10T07:00:00.000Z	0
4	http://example4.com	This is caption 4	user4	2023-10-10T07:00:00.000Z	0
5	http://example5.com	This is caption 5	user5	2023-10-10T07:00:00.000Z	0
6	http://example6.com	This is caption 1	user1	2023-10-10T07:00:00.000Z	1
7	http://example7.com	This is caption 2	user2	2023-10-10T07:00:00.000Z	1
8	http://example8.com	This is caption 3	user3	2023-10-10T07:00:00.000Z	1
9	http://example9.com	This is caption 4	user4	2023-10-10T07:00:00.000Z	1
10	http://example10.com	This is caption 5	user5	2023-10-10T07:00:00.000Z	1
11	http://example11.com	This is caption 1	user1	2023-10-10T07:00:00.000Z	2
12	http://example12.com	This is caption 2	user2	2023-10-10T07:00:00.000Z	2
13	http://example13.com	This is caption 3	user3	2023-10-10T07:00:00.000Z	2
14	http://example14.com	This is caption 4	user4	2023-10-10T07:00:00.000Z	2
15	http://example15.com	This is caption 5	user5	2023-10-10T07:00:00.000Z	2
16	http://example1.com	This is caption 16	user1	2023-10-10T07:00:00.000Z	0
17	http://example1.com	This is caption 17	user1	2023-10-10T07:00:00.000Z	0
18	http://example1.com	This is caption 18	user1	2023-10-10T07:00:00.000Z	0

- Relationship: likePost(**postID: string, acc: string**)

likepost

Selected columns: postid, acc

postid

acc

Submit

Table

postid	acc
1	user1
2	user2
3	user3
4	user4
5	user5

- Caption(caption: string, advertisement: boolean, **postID: number**)

caption

Selected columns: caption, postid, advertisement

caption

postid

advertisement

Submit

Table

caption	postid	advertisement
This is caption 1	1	true
This is caption 2	2	false
This is caption 3	3	true
This is caption 4	4	false
This is caption 5	5	true

- IsA: TextPost(**postID: number**, content: string)

textpost

Selected columns: postid, content

postid

content

Submit

Table

postid	content
1	This is caption 1
2	This is caption 2
3	This is caption 3
4	This is caption 4
5	This is caption 5

- IsA: VideoPost(**postID: number**, content: BLOB)

videopost

Selected columns: postid, content

postid

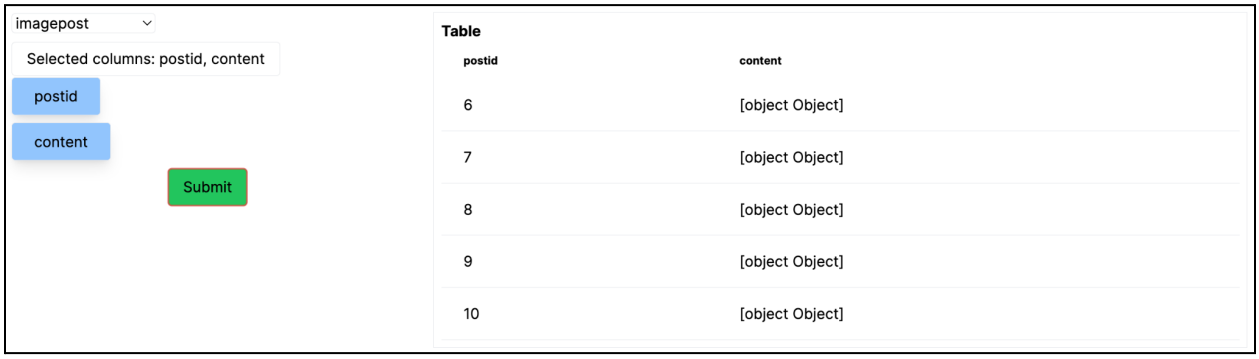
content

Submit

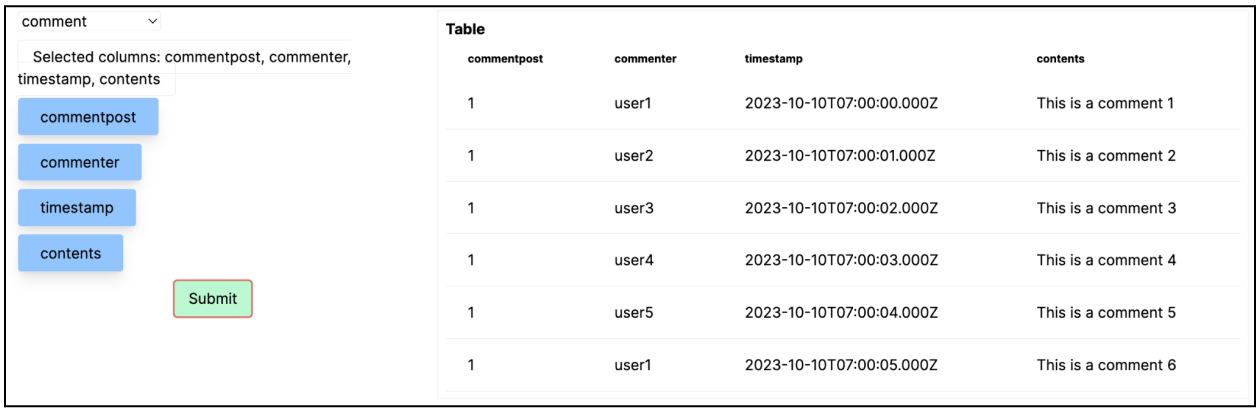
Table

postid	content
11	[object Object]
12	[object Object]
13	[object Object]
14	[object Object]
15	[object Object]

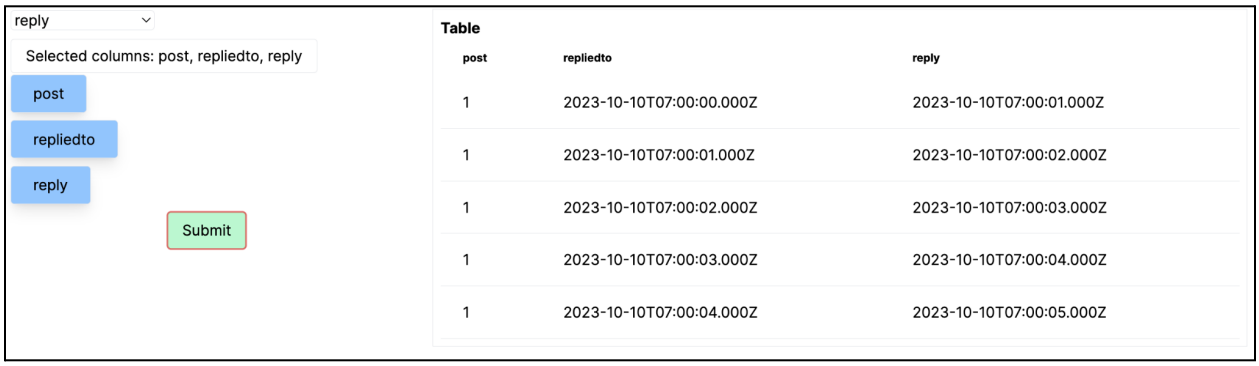
- IsA: ImagePost(**postID: number**, content: BLOB)



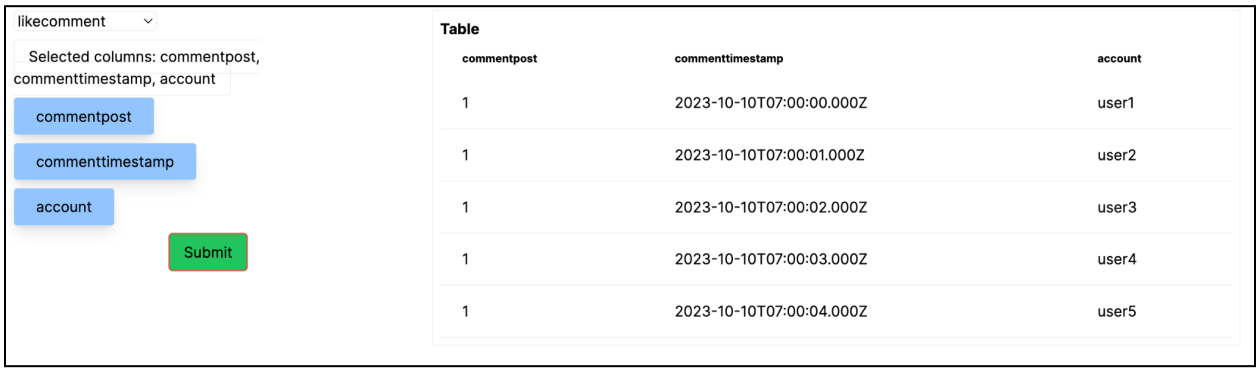
- Weak entity: Comment(**commentsOn: number**, **commenter: string**, timeStamp: timestamp, contents: string)



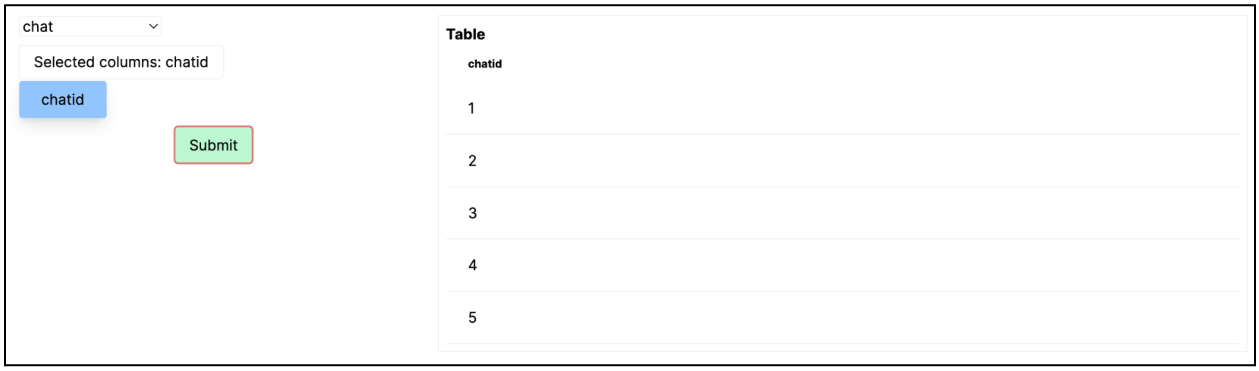
- Relationship: reply(**post: number, repliedTo: string, reply: string**)



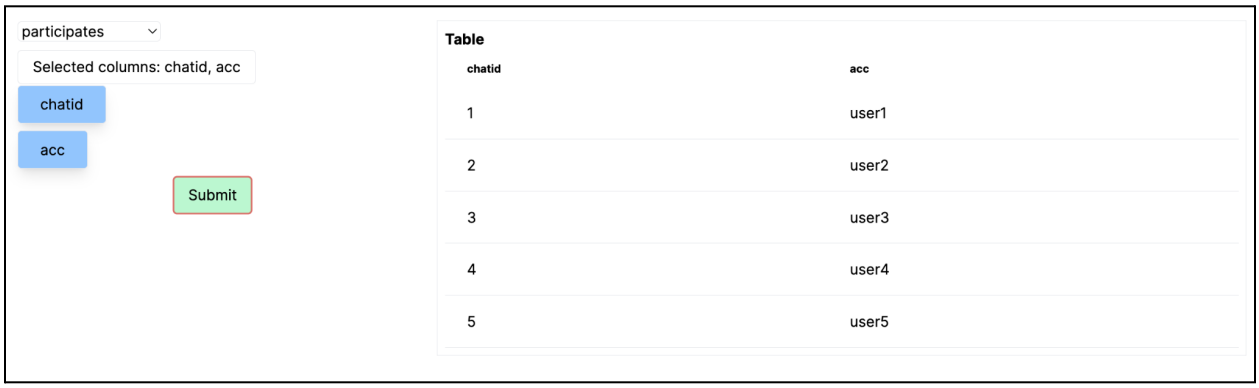
- Relationship: likeComment(**commentPost:number** , **commentTimeStamp: timestamp**, **acc: string**)



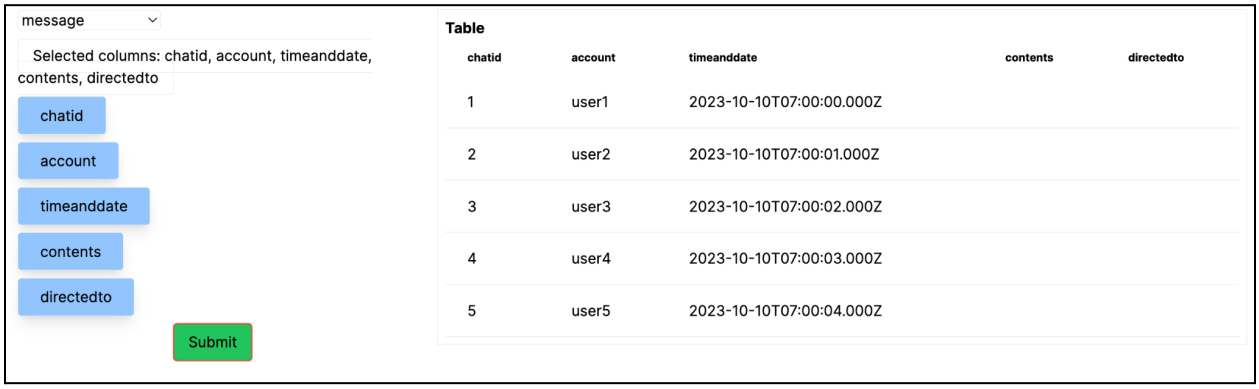
- Chat(chatID: number)



- Relationship: participates(**chatID: number**, **acc: string**)



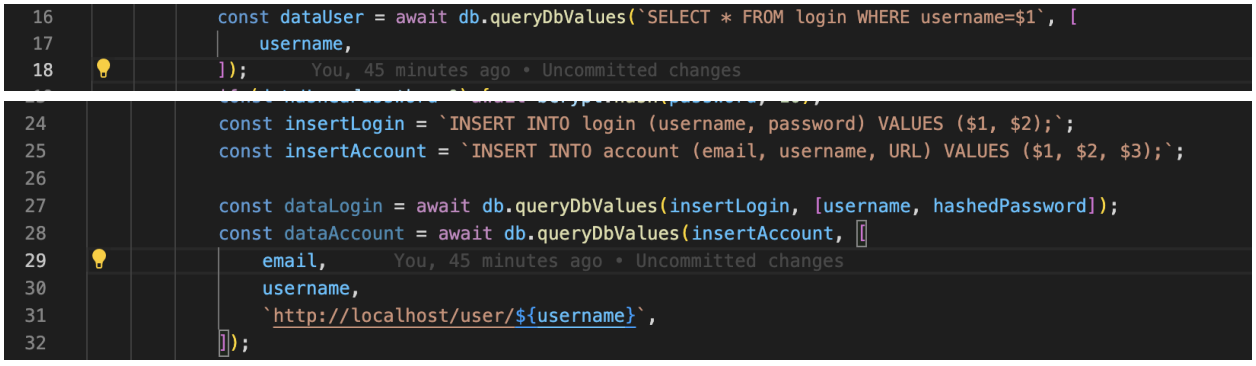
- Weak entity: Message(**chatID: number**, **account: string**, timeAndDate: string, contents: string, directedTo: string)



List of all SQL queries used

all the following files in: backend/src/controller

Account.js



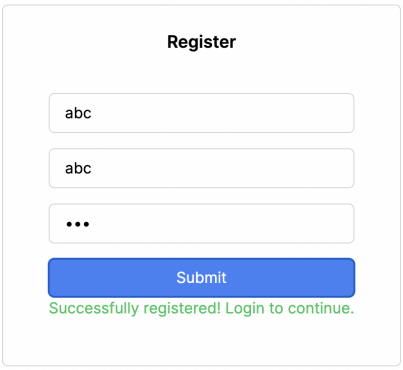
Above 2 screenshots before and after (Create account and login):
Before:

University of British Columbia, Vancouver
Department of Computer Science

Table

username	password
user1	pass1
user2	pass2
user3	pass3
user4	pass4
user5	pass5
test	$2b$10$O0u7e875hwBfQtVVf/SY5uDy29d2Eqhw9N7BY2wsTZMb91US/LcJO
test2	$2b$10$oARNgdqW.2KaMxBoolZUVerhFV54EjEUx.jm0/6MSHsm/Q6kXJ0U2
new	$2b$10$.uBAykN2aUFo.bJdTaYtf.pYJCNWHGm.nHVPP.yHRVXjxchf1VEDC
asdf	$2b$10$5HDQZUqgDrqVo7xMvsI/qO0FzuIFA.m3OReYQeZS9e7212SHXYYUO
fdsa	$2b$10$iNPN3iwtQe3HSjbe2/ezd./.WqfXF61gixwO0WEE6HFQTcp1Owryy

DURING:



AFTER:

University of British Columbia, Vancouver
Department of Computer Science

Table

username	password
user1	pass1
user2	pass2
user3	pass3
user4	pass4
user5	pass5
test	$2b$10$O0u7e875hwBfQtVVf/SY5uDy29d2Eqhw9N7BY2wsTZMb91US/LcJO
test2	$2b$10$oARNgdqW.2KaMxBoolZUVerhFV54EjEUx.jm0/6MSHsm/Q6kXJ0U2
new	$2b$10$.uBAykN2aUFo.bJdTaYtf.pYJCNWHGm.nHVPP.yHRVXjxchf1VEDC
asdf	$2b$10$5HDQZUqgDrqVo7xMvsl/qO0FzuIFA.m3OReYQeZS9e7212SHXYYUO
fdsa	$2b$10$iNPN3iwtQe3HSjbe2/ezd./.WqfXF61gixwO0WEE6HFQTcp1Owryy
abc	$2b$10$8V0MAsvd8LfDaI52bd04tOQ7vE4kcH/lxHezliT8NcvdhPcwlmlWa

```
47          const sql = `SELECT * FROM account WHERE username=$1;`;
48          const data = await db.queryDbValues(sql, values);
```

(REQUIRED AGGREGATION WITH HAVING QUERY)

```
63              const sql = `SELECT createdBy
64          FROM Post
65          GROUP BY createdBy
66          HAVING COUNT(postID) >= $1;`;
67              const data = await db.queryDbValues(sql, values);
```

BEFORE

Aggregation with HAVING, users having number of posts SELECT createdBy FROM Post GROUP BY createdBy HAVING COUNT(postID) > 5;
Input One enter natural number:4
Query

AFTER

Aggregation with HAVING, users having number of posts SELECT createdBy FROM Post GROUP BY createdBy HAVING COUNT(postID) > 5;
Input One enter natural number:4
Query
user1

Caption.js

```
8           const text = "INSERT INTO caption (caption, postID, advertisement) VALUES($1, $2, $3)"
```

```
19          const data = await db.queryDbValues("SELECT caption, postID FROM Caption;")
```

University of British Columbia, Vancouver
Department of Computer Science

Chat.js

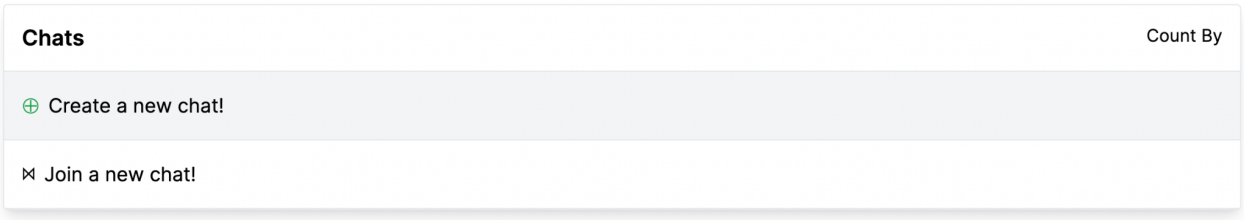
```
7       const createChat = `INSERT INTO chat DEFAULT VALUES RETURNING chatId;`;
8       const data = await db.queryDb(createChat);
9       const chatId = data[0].chatid;
```

For ABOVE 1 screenshot:
Before:

Table: chat

chatid
1
3
4
5
6

DURING:



AFTER:

Table: chat

chatid
1
3
4
5
6
7

University of British Columbia, Vancouver
Department of Computer Science

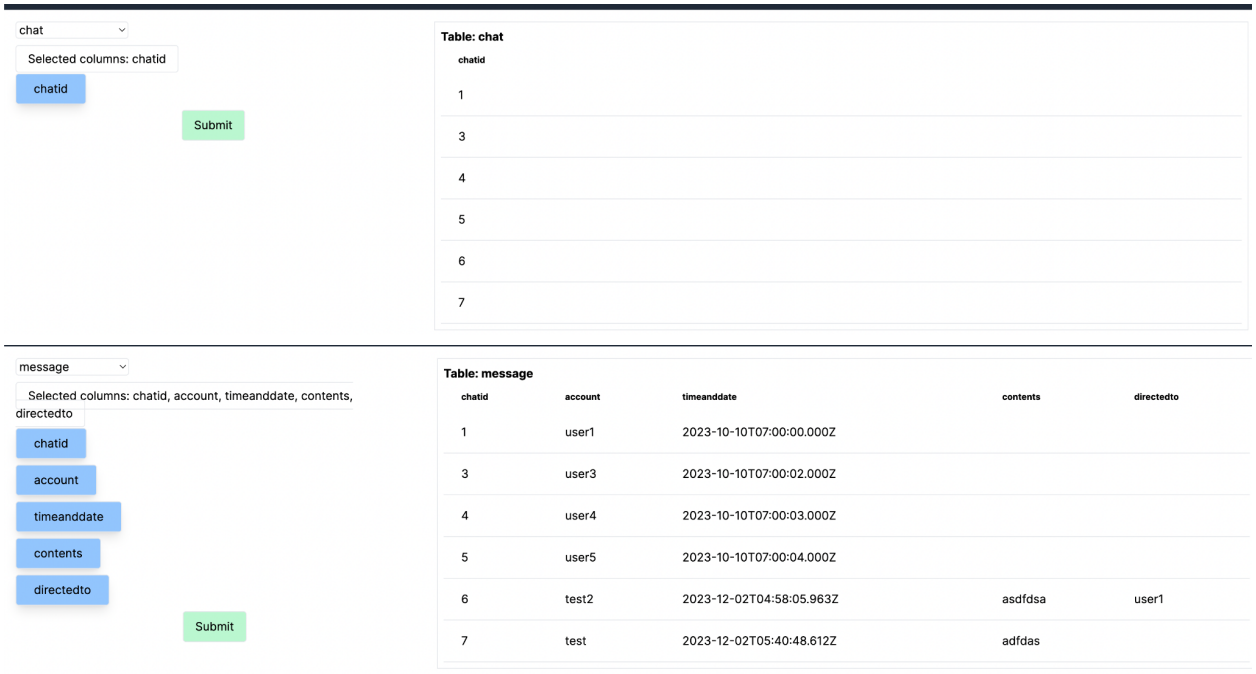
```
11          const joinChat = `INSERT INTO participates (chatId, acc) VALUES ($1, $2);`;
12          await db.queryDbValues(joinChat, [chatId, username]);
```

```
24      // no need to sanitize this, already checked in auth
25      const getChats = `SELECT * FROM chat WHERE chatId IN (SELECT chatId FROM participates WHERE acc='${username}');`;
26      const chats = await db.queryDb(getChats);
```

```
30          const getParticipants = `SELECT acc FROM participates p, chat c WHERE c.chatId=$1 AND p.chatId=$2;`;
31          const participants = await db.queryDbValues(getParticipants, [
32              chat.chatid,
33              chat.chatid,
34          ]);
```

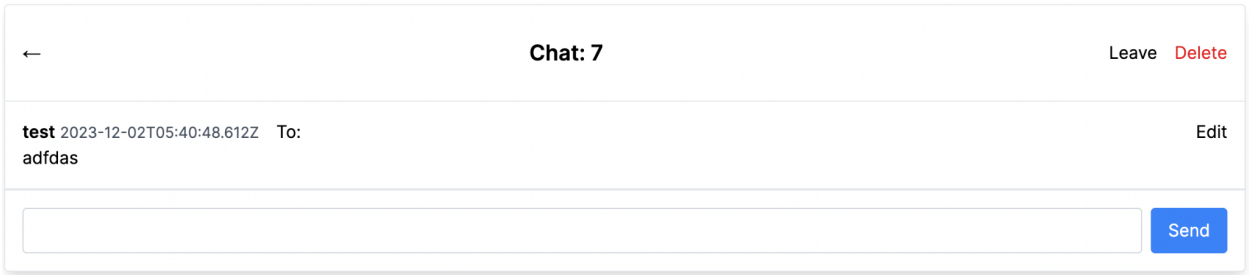
```
54              const deleteChat = `DELETE FROM chat WHERE chatId = $1;`;
55              await db.queryDbValues(deleteChat, [chatId]);
```

ABOVE 1 Screenshot (REQUIRED DELETE STATEMENT)
BEFORE:

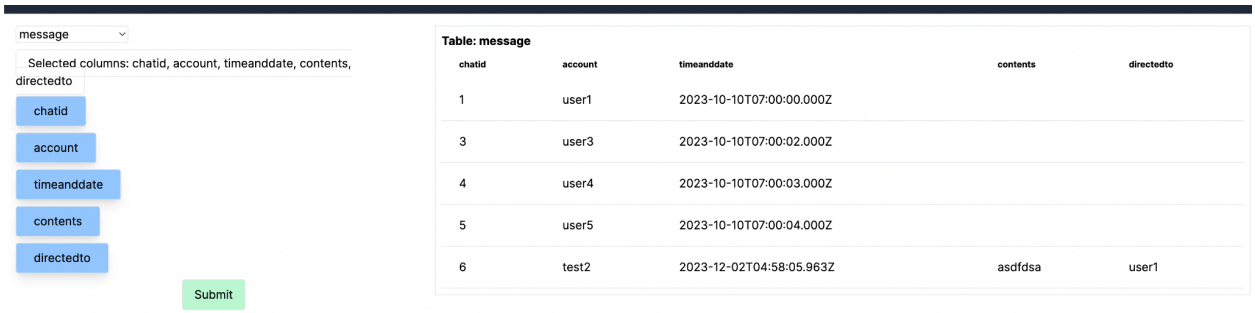


DURING (we delete the chat, so it should cascade and delete the messages)

University of British Columbia, Vancouver
Department of Computer Science



AFTER:




```
70  const getMessages = `SELECT * FROM message WHERE chatId=$1 ORDER BY timeAndDate ASC;`;
71  const messages = await db.queryDbValues(getMessages, [chatId]);
```

```
93  const insertParticipates = `INSERT INTO participates (chatId, acc) VALUES ($1, $2);`;
94  await db.queryDbValues(insertParticipates, [chatId, username]);
```

```
104 const doesChatExist = `SELECT * FROM chat WHERE chatId=$1;`;
105 const rows = await db.queryDbValues(doesChatExist, [chatId]);
```

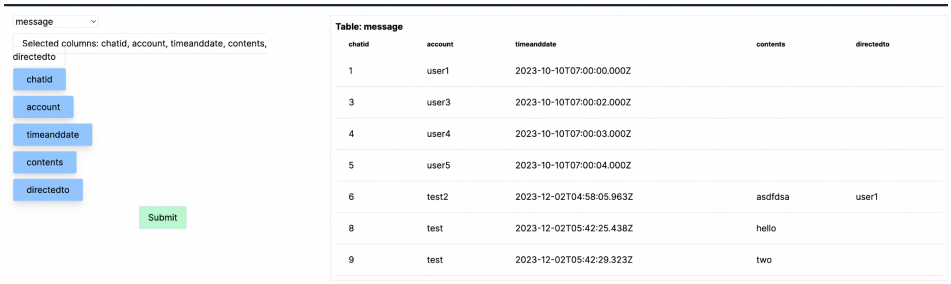
```
120 const deleteParticipates = `DELETE FROM participates WHERE chatId = $1 AND acc = $2;`;
121 await db.queryDbValues(deleteParticipates, [chatId, username]);
```

```
136 const createMessage = `INSERT INTO message (chatId, account, timeAndDate, contents) VALUES ($1, $2, TO_TIMESTAMP($3 / 1000.0), $4) RETURNING timeAndDate;`;
137 const data = await db.queryDbValues(createMessage, [
138     chatId,
139     username,
140     timeNow,
141     message,
142 ]);
```

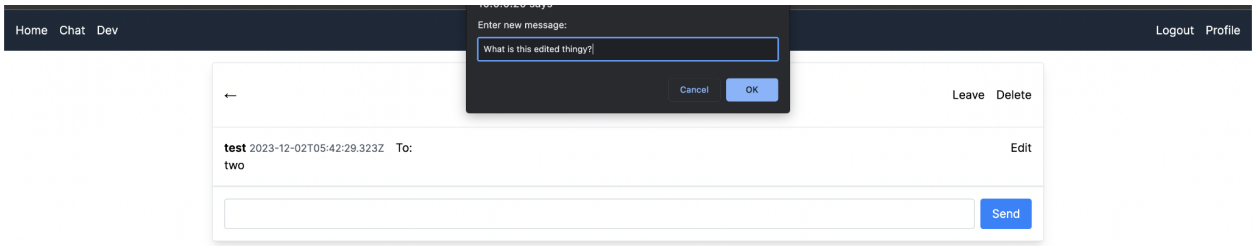
```
158 const editMessage = `UPDATE message SET contents=$1, directedto=$2 WHERE chatId=$3 AND account=$4 and timeAndDate=$5::timestamptz;`;
159 const data = await db.queryDbValues(editMessage, [
160     contents,
161     directedToStr,
162     chatId,
163     account,
164     timeanddate,
165 ]);
```

ABOVE 1 SCREENSHOT (REQUIRED UPDATE STATEMENT)
BEFORE:

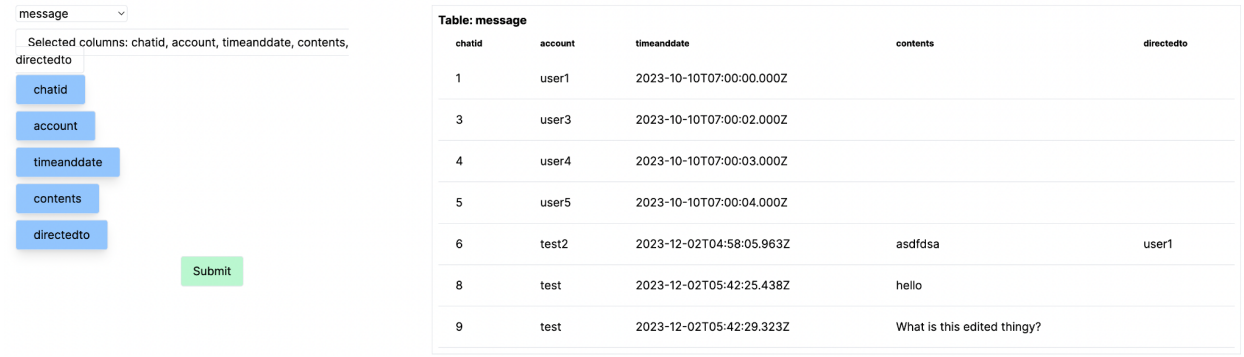
University of British Columbia, Vancouver
Department of Computer Science



DURING:



AFTER:




```
235     const participatesInChat = `SELECT * FROM participates WHERE acc=$1 AND chatId=$2;`;
236     const rows = await db.queryDbValues(participatesInChat, [username, chatId]);
```

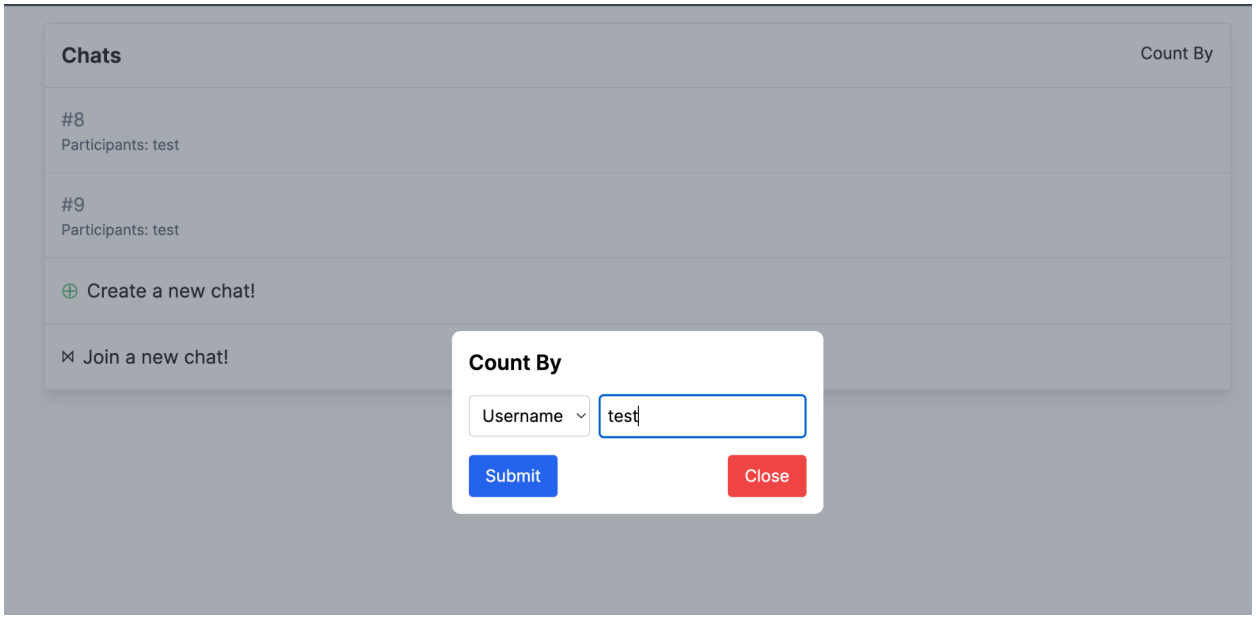
```
179         const getChats = `
180         SELECT COUNT(*) AS count
181         FROM chat c, participates p
182         WHERE c.chatId = p.chatId
183         AND c.chatId IN (
184             SELECT chatId FROM participates WHERE acc=$1
185         )
186         GROUP BY p.acc
187         HAVING p.acc=$2;`;
188
189         const chats = await db.queryDbValues(getChats, [countText, username]);
```

University of British Columbia, Vancouver
Department of Computer Science

FOR ABOVE 1 screenshot (REQUIRED AGGREGATION WITH GROUP BY STATEMENT)

BEFORE:

DURING:



AFTER: Same as before as this doesn't change anything.

Dev.js

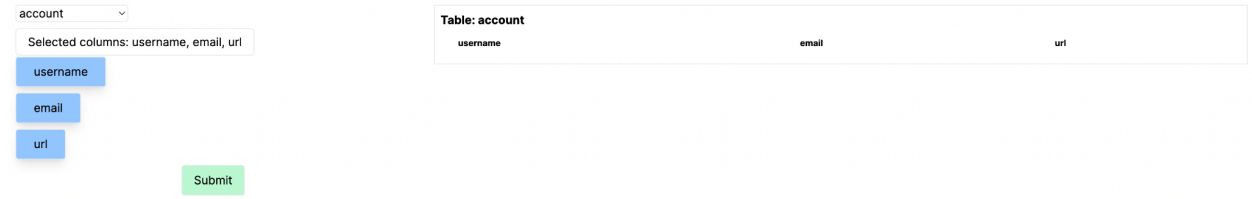
```
const getTableColumns = `SELECT column_name FROM information_schema.columns WHERE table_name = $1;`
const columns = await db.queryDbValues(getTableColumns, [tableName]);
```

```
const queryString = `SELECT ${columnStr} FROM ${tableName};`;
const rows = await db.queryDb(queryString);
```

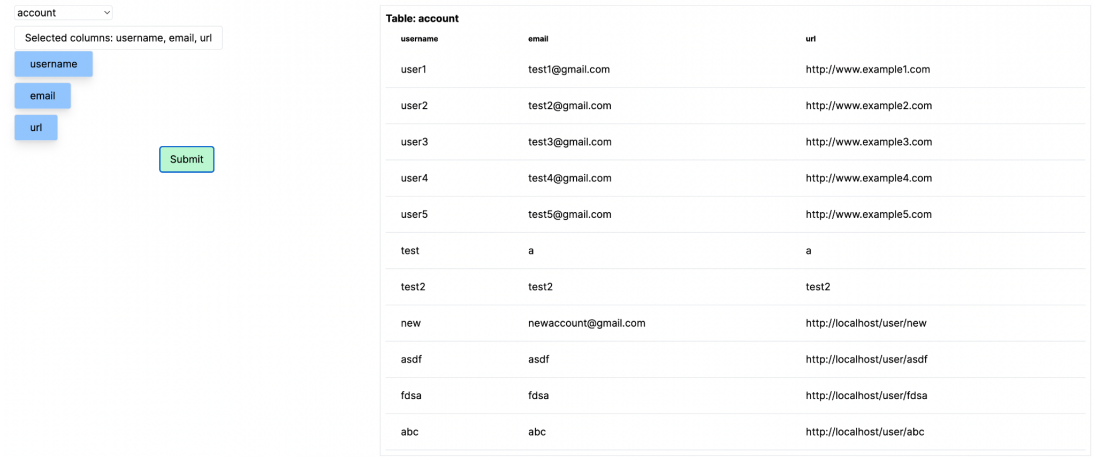
University of British Columbia, Vancouver
Department of Computer Science

```
47          const getTables = `SELECT tablename FROM pg_catalog.pg_tables WHERE schemaname='public';`;
48          const tables = await db.queryDb(getTables);
```

ABOVE 3 screenshots for the **projection criteria (REQUIRED PROJECTION STATEMENT)**
BEFORE:



AFTER / DURING:

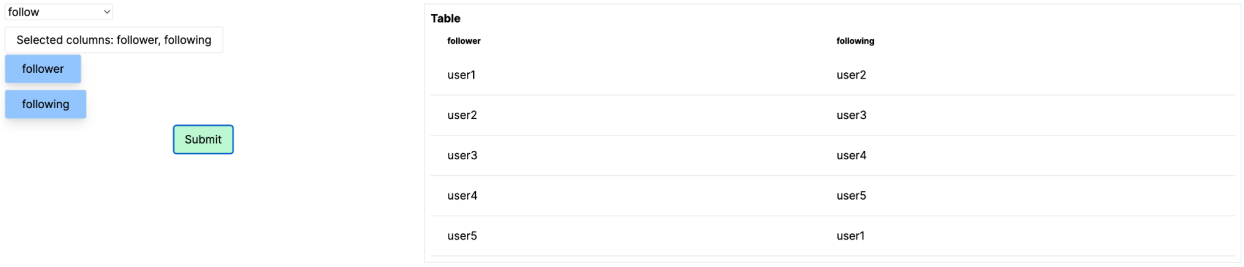


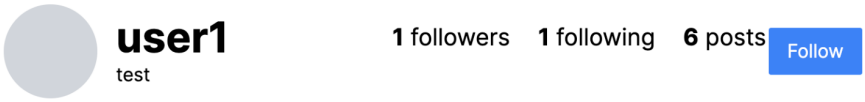
Follow.js

```
11          const data = await db.queryDbValues(
12              `INSERT INTO follow (follower, following) VALUES ($1, $2);`, [follower, following]
13          );
```

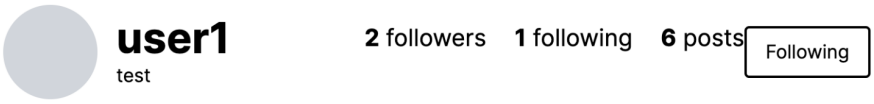
before:

University of British Columbia, Vancouver
Department of Computer Science

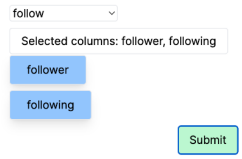


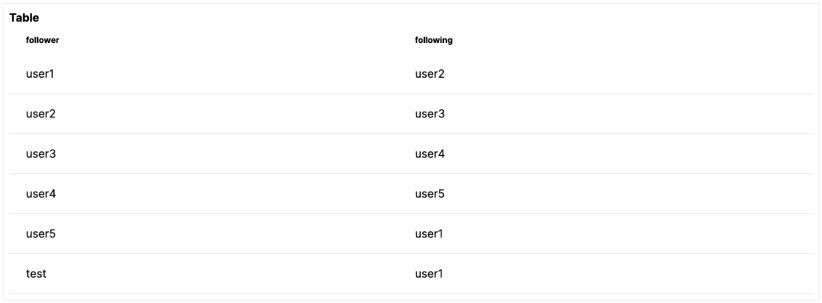


After



University of British Columbia, Vancouver
Department of Computer Science

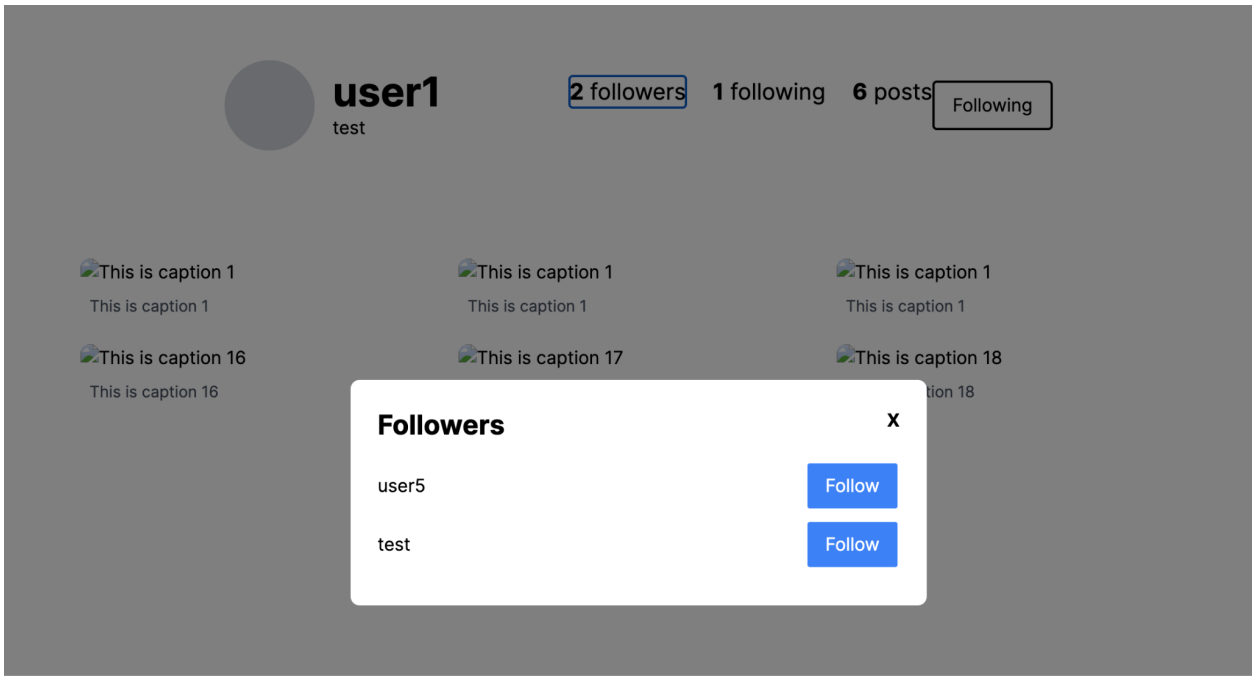





```
27          const data = await db.queryDbValues(
28              `DELETE FROM follow WHERE follower=$1 AND following=$2;`, [follower, following]
29              );
```

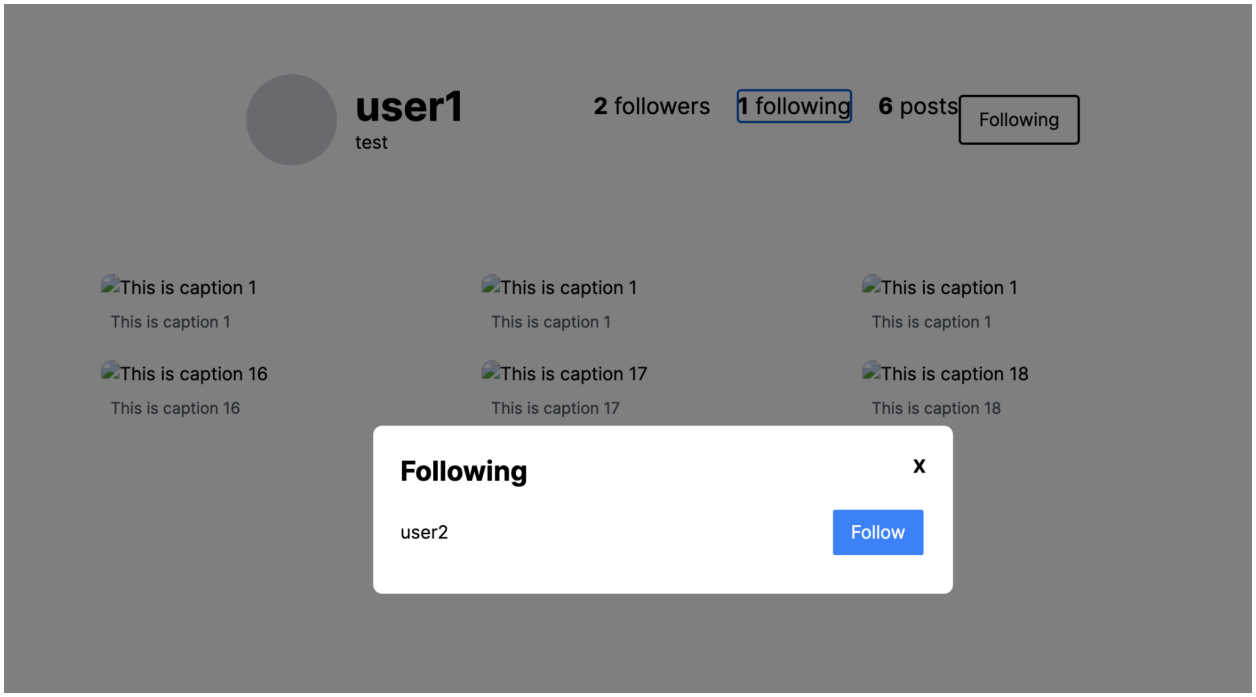
reverse of the above 2 images

```
43          const data = await db.queryDbValues(`SELECT follower FROM follow WHERE following=$1;`, [id]);
```




```
56          const data = await db.queryDbValues(`SELECT following FROM follow WHERE follower=$1;`, [id]);
```

University of British Columbia, Vancouver
Department of Computer Science

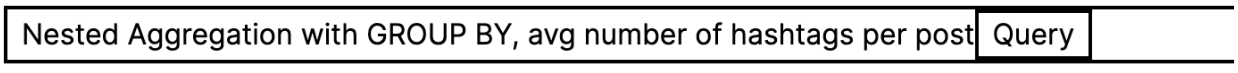


Hashtag.js (REQUIRED NESTED AGGREGATION WITH GROUP BY QUERY and division)

NESTED AGGREGATION WITH GROUP BY STATEMENT SCREENSHOTS

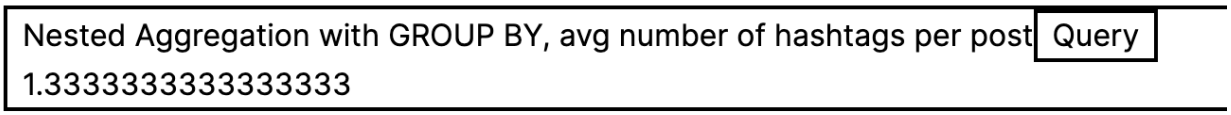
```
15                const sql = `SELECT AVG(hashtag_count) as avg_hashtags_per_post
16                    FROM (
17                        SELECT postID, COUNT(hashTag) as hashtag_count
18                        FROM associateHashtag
19                        GROUP BY postID
20                    ) as hashtag_counts;`;
21                const data = await db.queryDb(sql);
```

Before:



After:

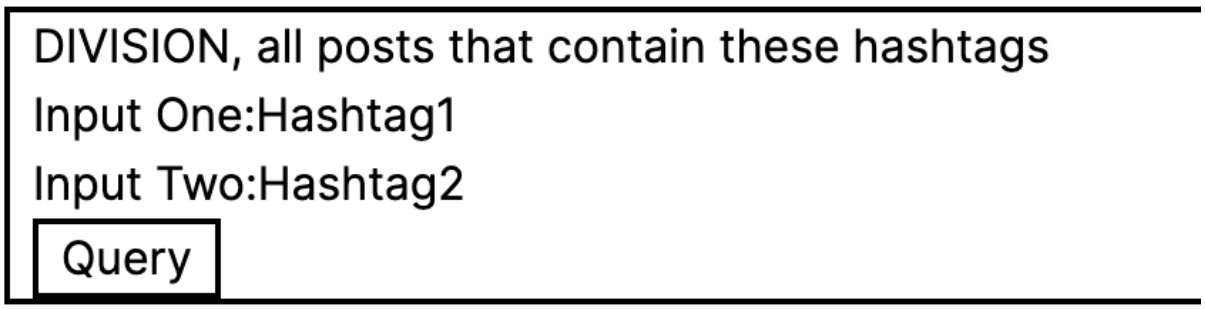
University of British Columbia, Vancouver
Department of Computer Science



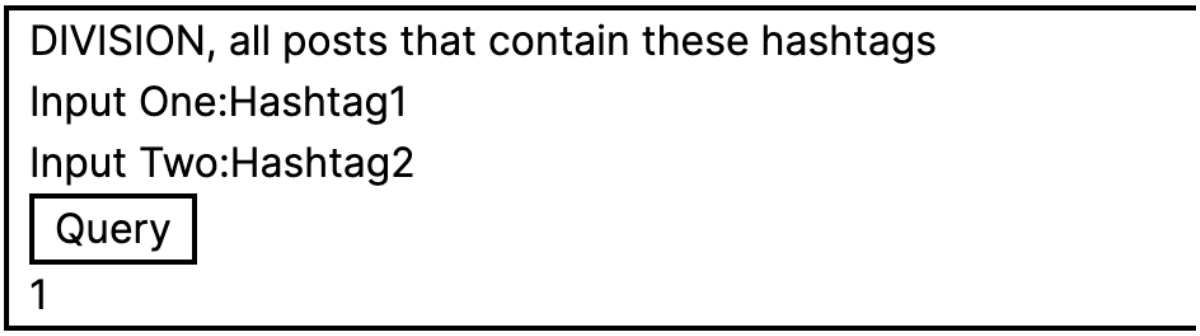
DIVISION QUERY SCREENSHOTS

```
31          const sql = `SELECT postID
32              FROM associateHashtag
33              WHERE hashTag IN ($1, $2)
34              GROUP BY postID
35              HAVING COUNT(DISTINCT hashTag) = 2;`;
36          const data = await db.queryDbValues(sql, values);
```

Before:



After:



ImagePost.js

```
8           const text = `INSERT INTO ImagePost(postID, content) VALUES($1, $2);`;
9           const values = [pid, file]
10          const data = await db.queryDbValues(text, values);
```

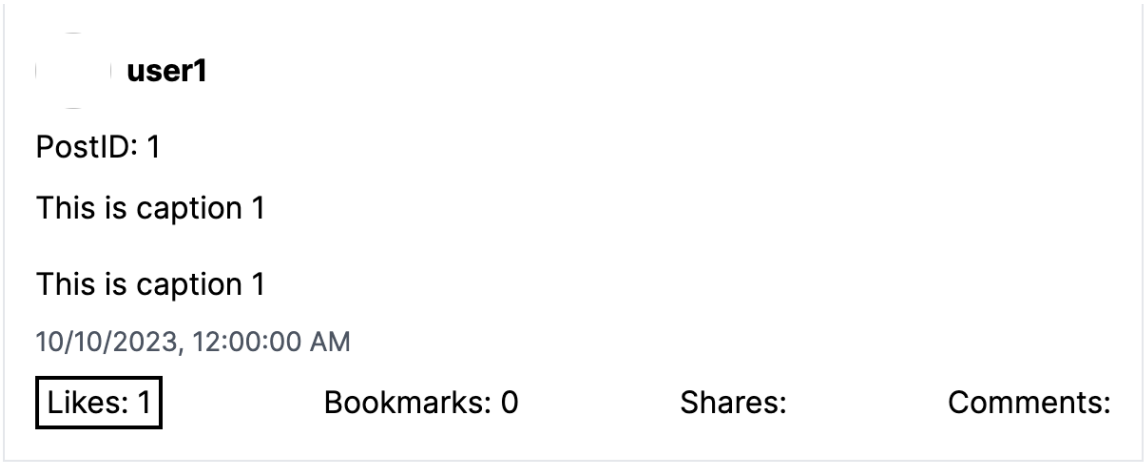
```
19          const data = await db.queryDb("SELECT postID, content FROM ImagePost;")
```

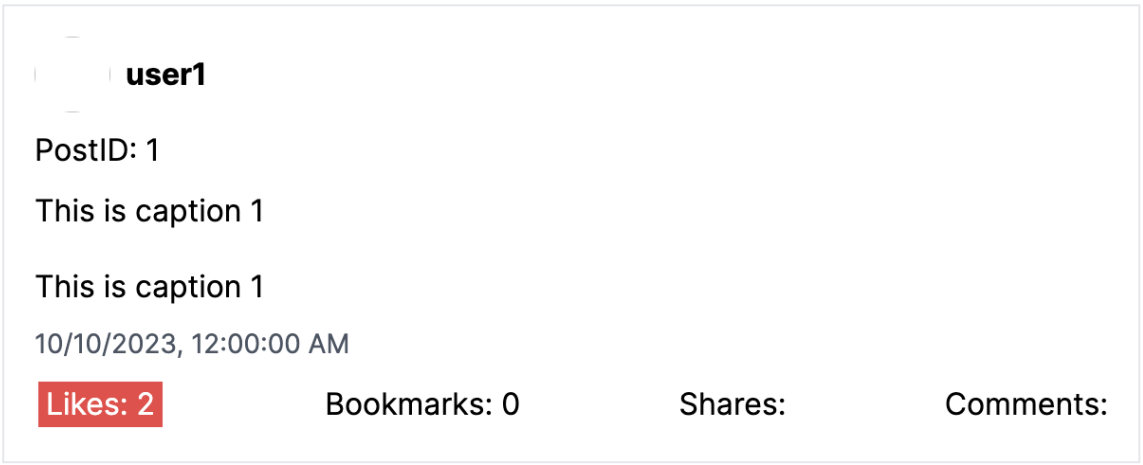
Likes.js

```
13      const data = await db.queryDbValues(`DELETE FROM likePost WHERE postID=$1 AND acc=$2;`, [post, user]);
```

```
25      const data = await db.queryDbValues(`INSERT INTO likePost(postID, acc) VALUES ( $1, $2);`, [post, user]);
```

```
35      const data = await db.queryDbValues(`SELECT * FROM likePost WHERE postID=$1;`, [post]);
```





Post.js (REQUIRED INSERT STATEMENT)

University of British Columbia, Vancouver
Department of Computer Science

```
18          const pidData = await db.queryDb(`SELECT MAX(postID) AS max FROM Post;`);
```

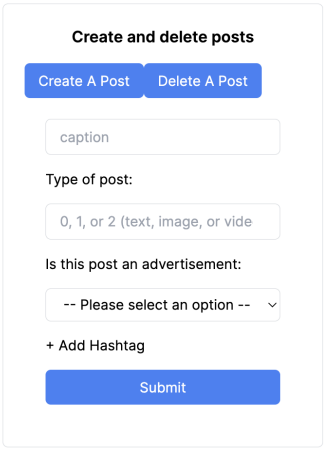
```
30          db.queryDbValues(`SELECT * FROM Hashtags WHERE text=$1`, [hashtag.text]).then((res) => {
```

```
36              db.queryDbValues(
37                  `INSERT INTO Hashtags (text, color) VALUES ($1, $2)`, [hashtag.text, hashtag.color.substring(1)]
38              );
```

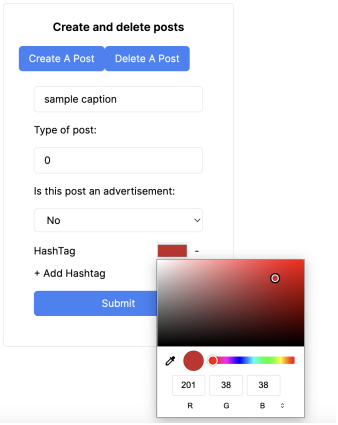
```
50      const text = `INSERT INTO post (postID, URL, caption, createdBy, timestamp, type) VALUES
51          ($1, $2, $3, $4, to_timestamp($5 / 1000.0), $6);`;
52      const values = [pid, `${POST_URL}/${pid}`, caption, username, time, typeInt];
53      const data = await db.queryDbValues(text, values);
```

```
81              db.queryDbValues(
82                  `INSERT INTO associateHashtag (postID, hashtag) VALUES ($1, $2)`, [pid, hashtag.text]
83              )
```

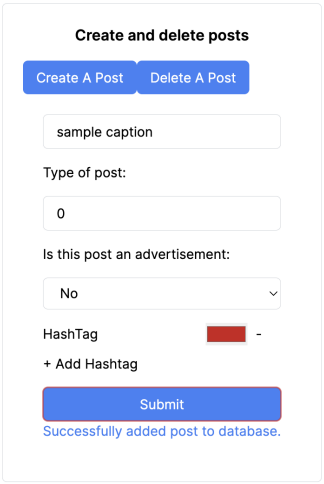
Before:



During:



After:



INSERT STATEMENT FOR POST: SCREENSHOTS

Delete a post:

```
99          const dataUser = await db.queryDbValues(
100             `SELECT username FROM account WHERE username=$1;`, [username]
101         );
```

```
106         const checkCreator = await db.queryDbValues(
107             `SELECT createdBy FROM Post WHERE postID = $1;`, [target]
108         );
```

```
117         const data = await db.queryDbValues(`DELETE FROM Post WHERE postID = $1`, [target]);
```

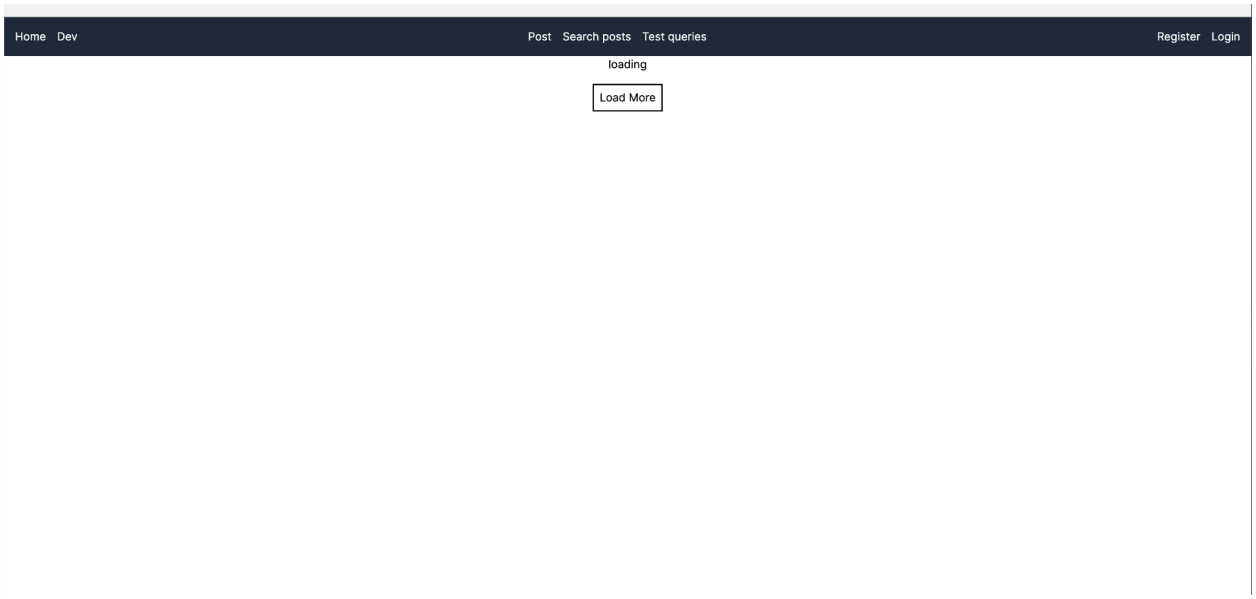
```
149     if (username) {
150         values.push(username)
151         query = `
152             SELECT Post.*,
153                 CASE
154                     WHEN Post.type = 0 THEN TextPost.content
155                     WHEN Post.type = 1 THEN encode(ImagePost.content, 'base64')
156                     WHEN Post.type = 2 THEN encode(VideoPost.content, 'base64')
157                 END AS content
158             FROM Post
159             LEFT JOIN TextPost ON Post.postID = TextPost.postID AND Post.type = 0
160             LEFT JOIN ImagePost ON Post.postID = ImagePost.postID AND Post.type = 1
161             LEFT JOIN VideoPost ON Post.postID = VideoPost.postID AND Post.type = 2
162             WHERE Post.createdBy=$1;
163         `;
164     } else if (postId) {
165         values.push(postId)
166         query = `
167             SELECT Post.*,
168                 CASE
169                     WHEN Post.type = 0 THEN TextPost.content
170                     WHEN Post.type = 1 THEN encode(ImagePost.content, 'base64')
171                     WHEN Post.type = 2 THEN encode(VideoPost.content, 'base64')
172                 END AS content
173             FROM Post
174             LEFT JOIN TextPost ON Post.postID = TextPost.postID AND Post.type = 0
175             LEFT JOIN ImagePost ON Post.postID = ImagePost.postID AND Post.type = 1
176             LEFT JOIN VideoPost ON Post.postID = VideoPost.postID AND Post.type = 2
177             WHERE Post.postID=$1;
178         `;
179     } else {
180         query = "SELECT * FROM Post;";      allanx01, 20 hours ago • home page feed d
181     }
182
183     data = await db.queryDbValues(query, values);
```

University of British Columbia, Vancouver
Department of Computer Science

(REQUIRED JOIN STATEMENT)

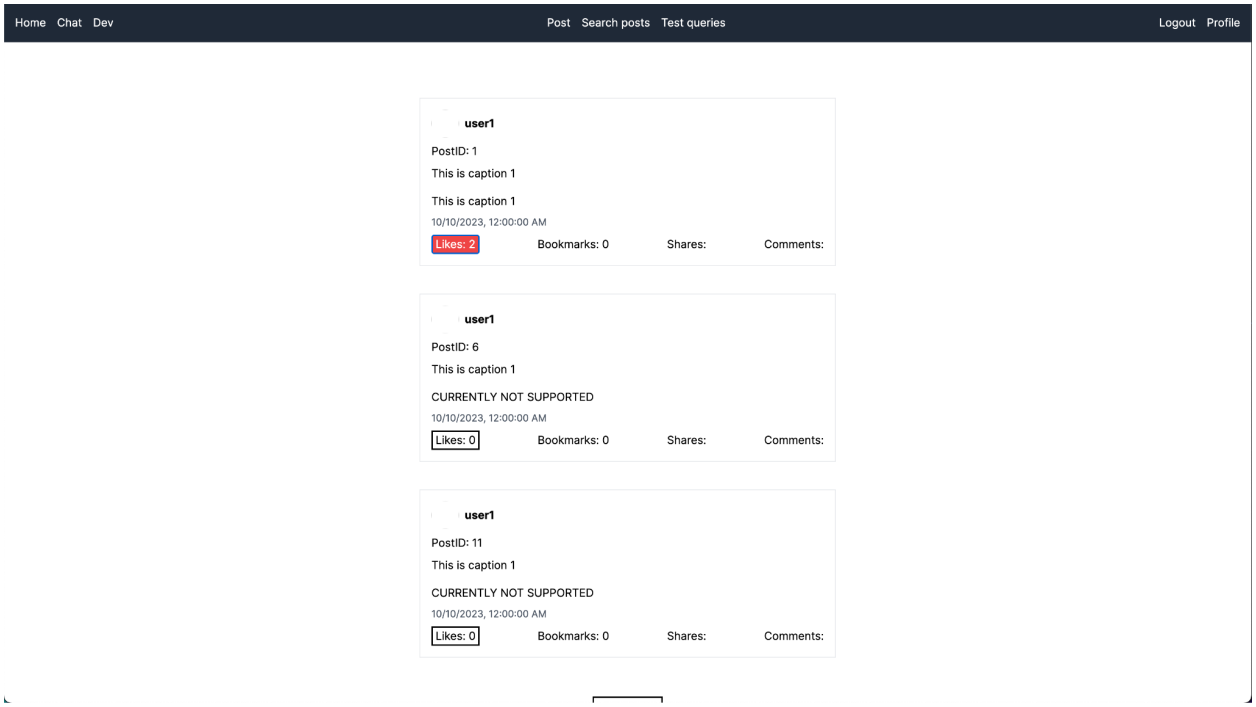
```
200             const sql = `SELECT Post.*,
201                             CASE
202                                 WHEN Post.type = 0 THEN TextPost.content
203                                 WHEN Post.type = 1 THEN encode(ImagePost.content, 'base64')
204                                 WHEN Post.type = 2 THEN encode(VideoPost.content, 'base64')
205                             END AS content
206                         FROM
207                             Post
208                             LEFT JOIN TextPost ON Post.postID = TextPost.postID AND Post.type = 0
209                             LEFT JOIN ImagePost ON Post.postID = ImagePost.postID AND Post.type = 1
210                             LEFT JOIN VideoPost ON Post.postID = VideoPost.postID AND Post.type = 2
211                         INNER JOIN
212                             follow ON Post.createdBy = follow.following
213                         WHERE
214                             follow.follower = $1
215                         ORDER BY
216                             Post.timestamp DESC
217                         LIMIT $2
218                         OFFSET $3;`;
219             const data = await db.queryDbValues(sql, [user, limit, offset]);
```

BEFORE



AFTER

University of British Columbia, Vancouver
Department of Computer Science

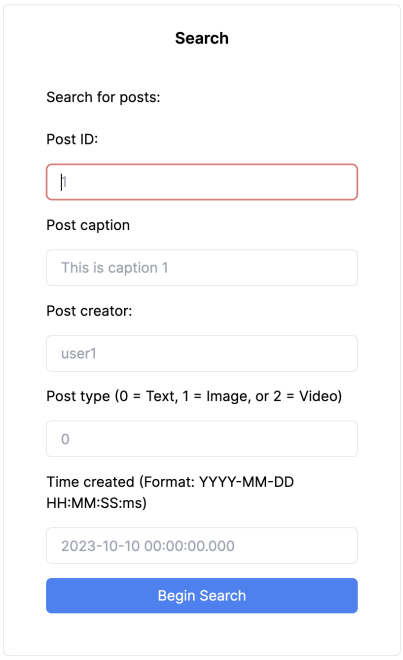


Search.js (REQUIRED SELECTION STATEMENT)

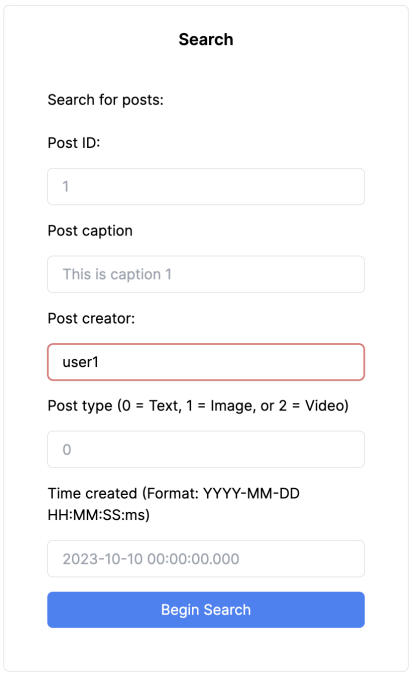
```
10      const { postId, caption, createdBy, type, postTime } = req.query;
11      var query = `SELECT * FROM Post WHERE `;
12      var values = [];
13      var i = 1;
14      if (postId !== undefined) {
15          const pid = parseInt(postId);
16          query += `postID=$${i} AND `;
17          values.push(pid);
18          i++;
19      }
20      if (caption !== undefined) {
21          query += `caption=$${i} AND `;
22          values.push(caption);
23          i++;
24      }
25      if (type !== undefined) {
26          const typeInt = parseInt(type);
27          query += `type=$${i} AND `;
28          values.push(typeInt);
29          i++;
30      }
31      if (createdBy !== undefined) {
32          query += `createdBy=$${i} AND `;      You, 34 minutes ago • Uncommitted changes
33          values.push(createdBy);
34          i++;
35      }
36      if (postTime !== undefined) {
37          query += `timestamp=$${i} AND `;
38          values.push(postTime);
39          i++;
40      }
41      let finalQuery = query.substring(0, query.length - 5) + ";";
42      const data = await db.queryDbValues(finalQuery, values);
```

University of British Columbia, Vancouver
Department of Computer Science

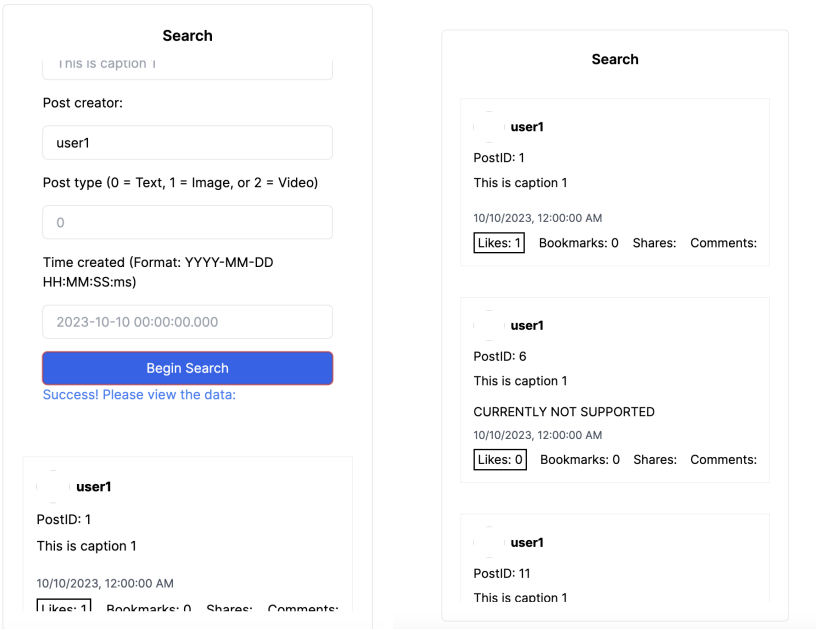
Before:



During:



After:



SELECT STATEMENT FOR SEARCH: SCREENSHOTS

```
62        const data = await db.queryDb(`SELECT * FROM Account;`);
```

```
95                  query = `
96                      SELECT Post.*,
97                          CASE
98                              WHEN Post.type = 0 THEN TextPost.content
99                              WHEN Post.type = 1 THEN encode(ImagePost.content, 'base64')
100                             WHEN Post.type = 2 THEN encode(VideoPost.content, 'base64')
101                         END AS content
102                     FROM Post
103                     LEFT JOIN TextPost ON Post.postID = TextPost.postID AND Post.type = 0
104                     LEFT JOIN ImagePost ON Post.postID = ImagePost.postID AND Post.type = 1
105                     LEFT JOIN VideoPost ON Post.postID = VideoPost.postID AND Post.type = 2
106                     WHERE Post.createdBy=$1;
107                 `;
```

```
110                 query = `
111                     SELECT Post.*,       You, 1 hour ago • uncomment get
112                         CASE
113                             WHEN Post.type = 0 THEN TextPost.content
114                             WHEN Post.type = 1 THEN encode(ImagePost.content, 'base64')
115                             WHEN Post.type = 2 THEN encode(VideoPost.content, 'base64')
116                         END AS content
117                     FROM Post
118                     LEFT JOIN TextPost ON Post.postID = TextPost.postID AND Post.type = 0
119                     LEFT JOIN ImagePost ON Post.postID = ImagePost.postID AND Post.type = 1
120                     LEFT JOIN VideoPost ON Post.postID = VideoPost.postID AND Post.type = 2
121                     WHERE Post.postID=$1;
122                 `;
```

```
124                 query = "SELECT * FROM Post;";
```

```
144             const sql = `SELECT Post.*,
145                             CASE
146                                 WHEN Post.type = 0 THEN TextPost.content
147                                 WHEN Post.type = 1 THEN encode(ImagePost.content, 'base64')
148                                 WHEN Post.type = 2 THEN encode(VideoPost.content, 'base64')
149                             END AS content
150                         FROM
151                             Post
152                             LEFT JOIN TextPost ON Post.postID = TextPost.postID AND Post.type = 0
153                             LEFT JOIN ImagePost ON Post.postID = ImagePost.postID AND Post.type = 1
154                             LEFT JOIN VideoPost ON Post.postID = VideoPost.postID AND Post.type = 2
155                         INNER JOIN
156                             follow ON Post.createdBy = follow.following
157                         WHERE
158                             follow.follower = $1
159                         ORDER BY
160                             Post.timestamp DESC
161                         LIMIT $2
162                         OFFSET $3;`;
163             const data = await db.queryDbValues(sql, [user, limit, offset]);
```

TextPost.js

University of British Columbia, Vancouver
Department of Computer Science

```
 9        const text = `INSERT INTO textpost(postID, content) VALUES($1, $2);`;
10        const values = [pid, caption]
11        const data = await db.queryDbValues(text, values);
```

```
20    const data = await db.queryDbValues("SELECT postID, content FROM textpost;", [])
```

VideoPost.js

```
 8        const text = `INSERT INTO VideoPost(postID, content) VALUES($1, $2);`;
 9        const values = [pid, file]
10        const data = await db.queryDbValues(text, values);
```

```
31        const data = await db.queryDb("SELECT postID, caption FROM VideoPost;")
```

THIS FILE IS /backend/src/middleware/PassportConfig.js

```
 9    const query = `SELECT * FROM login WHERE username=$1`;
10
11    try {
12        db.queryDbValues(query, [username]).then((data) => {
```

```
45      const query = `SELECT * FROM login WHERE username=$1`;
46      const data = await db.queryDbValues(query, [id]);
```