CPSC 304 Project Cover Page

Milestone #: 4

Date: 05/04/24

Group Number: 60

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Sangita Dutta	29933983	sdutta06	sangita7dutta@gmail.com
Delsther James Edralin	26279729	dedralin	dedralin@student.ubc.ca
Aman Johal	40801152	amanj21	ajohal21@gmail.com

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

2. Single SQL script used to create all tables and data

Options:

1) Clone the repo, set up your db in PostgreSQL according to README.
   npm start should automatically run script to initialize database with tables and data

Alternatively if cloning and npm does not work.
Please download initiating sql file
db.sql

1) Using PostgreSQL (via terminal):
   /i db.sql

2) Using PostgreSQL via pgAdmin4
   Paste contents of file db.sql in the query editor and run.

3. PDF file containing:

a. A short description of the final project, and what it accomplished

The project is a social platform that manages outdoor, dog walking activities. The domain of this project focuses on "pet wellbeing", which relates to pet care, recreation, and health. This mainly caters for people that are dog owners and want to guarantee their dogs to receive consistent stimulation by regular walks.

b. A description of how your final schema differed from the schema you turned in

Changed meetups to total participation with walk, meaning all meetups are walks. And total participation for a post to a walk, meaning you can only make a post from a completed walk (existing walkid).

c. A copy of the schema and screenshots that show what data is present in each relation after the SQL script from item #2 is run.

friendpost(**notificationid**, postlink, friendname)

	notificationid [PK] integer	postlink character varying (255)	friendname character varying (255)
1	1	post/2/2	Jack Bean
2	2	post/3/3	One Three
3	3	post/1/1	John Doe
4	4	post/3/3	One Three
5	5	post/1/1	John Doe
6	6	post/2/2	Jack Bean
7	7	post/4/4	Nul Ly
8	8	post/3/3	One Three

friendship(**ownerid1, ownerid2**, dateoffriendship)

	ownerid1 [PK] integer	ownerid2 [PK] integer	dateoffriendship date
1	1	2	2024-03-21
2	2	1	2024-03-21
3	1	3	2024-03-22
4	3	1	2024-03-22
5	2	3	2024-02-13
6	3	2	2024-02-13
7	3	4	2023-01-01
8	4	3	2023-01-01

logs(**notificationid, taskid**)

	notificationid [PK] integer	taskid [PK] integer
1	9	1
2	10	2
3	11	3
4	12	4
5	13	5
6	14	5

on_meetup(meetupid, **walkid**, time, location, date)

	meetupid [PK] integer	walkid integer	time time without time zone	location character varying (255)	date date
1	1	3	10:00:00	Vancouver Downtown	2024-02-28
2	2	7	08:00:00	Central Pong	2024-02-28
3	3	8	09:00:00	Metrotown	2024-02-20
4	4	9	17:00:00	UBC	[null]

organizes_walktask(taskid, **ownerid**, date, walkeventtype)

	taskid [PK] integer	ownerid integer	date date	walkeventtype event_type
1	1	1	2024-03-23	walk
2	2	2	2024-04-05	hike
3	3	3	2024-03-28	dog park
4	4	4	2024-04-01	run
5	5	5	2024-03-15	walk

owner(ownerid, email)

	ownerid [PK] integer	email character varying (255)
1	1	john@example.com
2	2	beanstalk@example.com
3	3	one@example.com
4	4	null@example.com
5	5	popolice@example.com

owner_contact(**email**, phonenumber)

	email [PK] character varying (255)	phonenumber character varying (255)
1	john@example.com	1234567890
2	beanstalk@example.com	0987654321
3	one@example.com	131313131
4	null@example.com	[null]
5	popolice@example.com	911

owner_name(**ownerid**, firstname, lastname)

	ownerid [PK] integer	firstname character varying (255)	lastname character varying (255)
1	1	John	Doe
2	2	Jack	Bean
3	3	One	Three
4	4	Null	Ly
5	5	Popo	Lice

owns_dog(dogid, **ownerid**, name, breed)

	dogid [PK] integer	ownerid integer	name character varying (255)	breed character varying (255)
1	1	1	Doggers	Labrador
2	2	2	Arfy	Bull Terrier
3	3	3	Blanky	[null]
4	4	4	Tory	[null]
5	5	5	K91	German Sheperd
6	6	5	K92	German Sheperd

owns_dog_birthday(**dogid, ownerid**, name, birthday)

	dogid [PK] integer	ownerid integer	name character varying (255)	birthday date
1	1	1	Doggers	2018-01-01
2	2	2	Arfy	2020-02-20
3	3	3	Blanky	2021-03-12
4	4	4	Tory	[null]
5	5	5	K91	[null]
6	6	5	K92	[null]

photo(**mediaid**, filter)

	mediaid [PK] integer	filter character varying (255)
1	1	Vintage
2	2	Normal
3	4	Sepia
4	6	[null]
5	7	[null]

post_media(**postid**, mediaid, url)

	postid [PK] integer	mediaid [PK] integer	url character varying (255)
1	1	1	photo (1).jpg
2	1	2	photo (1).webp
3	1	3	video (1).mp4
4	2	4	photo (10).jpg
5	2	5	video (6).mp4
6	3	6	photo (3).jpg
7	3	7	photo (9).jpg
8	3	8	video (2).mp4
9	5	9	video (4).mp4
10	6	10	video (5).mp4

post_media_date(**url**, datecreated)

	url [PK] character varying (255)	datecreated date
1	photo (1).jpg	2024-03-21
2	photo (1).webp	2024-03-21
3	video (1).mp4	2024-03-21
4	photo (10).jpg	2024-03-22
5	video (6).mp4	2024-03-24
6	photo (3).jpg	2024-03-23
7	photo (9).jpg	2024-03-23
8	video (2).mp4	2024-03-23
9	video (4).mp4	2024-03-24
10	video (5).mp4	2024-03-24

post_walk(postid, **walkid**)

	postid [PK] integer	walkid integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6

post_walk_content(**postid**, content)

	postid [PK] integer	content character varying (255)
1	1	Me and my doggers walking along.
2	2	GGRRR BARK BARK WOOF GGGGRRR GRR BARKNFKFLH FMSMANBARK WOOF WOOF
3	3	Met up with the gang today.
4	4	I ran 60 kilometers. My Tory is not surviving this.
5	5	Walking our K9s today.
6	6	Spotted this chihuahua doing a marathon.

post_walk_owner(**postid**, ownerid)

	postid [PK] integer	ownerid integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	5

post_walk_tag(**postid**, tag)

	postid [PK] integer	tag [PK] character varying (255)
1	1	doggers
2	1	pond
3	2	inspirational
4	3	jackAndJohn
5	3	doggers
6	4	runToryRun
7	5	K9
8	5	911
9	6	spotted
10	6	runToryRun

receives_notifications(notificationid, **ownerid**, notifcontent)

	notificationid [PK] integer	ownerid integer	notifcontent character varying (255)
1	1	1	New Post from Jack Bean
2	2	1	New Post from One Three
3	3	2	New Post from John Doe
4	4	2	New Post from One Three
5	5	3	New Post from John Doe
6	6	3	New Post from Jack Bean
7	7	3	New Post from Nul Ly
8	8	4	New Post from One Three
9	9	1	Time to walk Doggers!
10	10	2	Time to walk Arfy!
11	11	3	Time to walk Blanky!
12	12	4	Time to walk Tory!
13	13	5	Time to walk K91!
14	14	5	Time to walk K92!

schedules(**meetupid, ownerid**)

	meetupid [PK] integer	ownerid [PK] integer
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
6	3	2
7	3	3
8	4	1
9	4	3

taggedin(**dogid, postid**)

	dogid [PK] integer	postid [PK] integer
1	1	3
2	4	6

video(**mediaid**, duration)

	mediaid [PK] integer	duration time without time zone
1	3	00:00:15
2	5	00:00:10
3	8	[null]
4	9	00:00:11
5	10	[null]

walk(walkid, location)

	walkid [PK] integer	location character varying (255)
1	1	Central Pond
2	2	Beyond the sea
3	3	Vancouver Downtown
4	4	Long Lane Park
5	5	Police Station Park
6	6	Long Lane Park
7	7	Central Pond
8	8	Metrotown
9	9	UBC
10	10	Police Station
11	11	Long Lane Park
12	12	Central Pond

walk_date(**walkid**, date)

	walkid [PK] integer	date date
1	1	2024-03-23
2	2	2024-02-20
3	3	2024-02-28
4	4	2024-01-14
5	5	2024-01-19
6	6	2024-01-14
7	7	2024-02-28
8	8	2024-02-20
9	9	[null]
10	10	2024-05-28
11	11	2024-01-01
12	12	2024-02-14

walk_dist(**walkid**, distance)

	walkid [PK] integer	distance double precision
1	1	2.5
2	2	[null]
3	3	[null]
4	4	60
5	5	[null]
6	6	[null]
7	7	[null]
8	8	[null]
9	9	[null]
10	10	[null]
11	11	50
12	12	[null]

walkalert(**notificationid**, dogname)

	notificationid [PK] integer	dogname character varying (255)
1	9	Doggers
2	10	Arfy
3	11	Blanky
4	12	Tory
5	13	K91
6	14	K92

wentfor(**dogid, walkid**, rating)

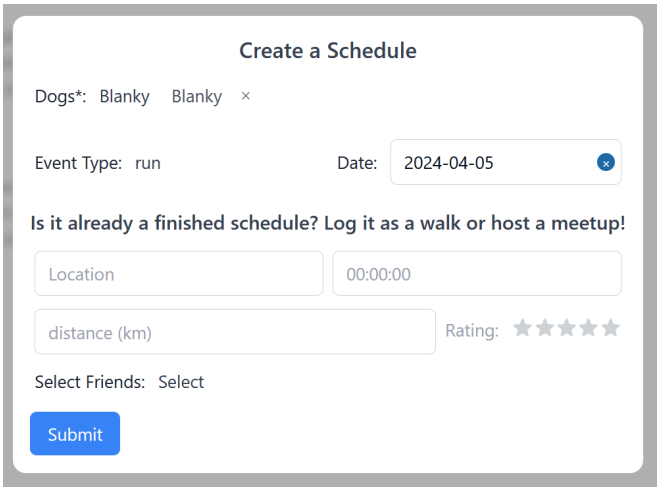
	dogid [PK] integer	walkid [PK] integer	rating rate_score
1	1	1	4
2	2	2	5
3	1	3	3
4	2	3	3
5	3	3	3
6	4	4	[null]
7	5	5	1
8	6	5	1
9	6	6	[null]
10	1	7	3
11	2	7	3
12	2	8	[null]
13	3	8	1
14	1	9	5
15	3	9	1
16	5	10	[null]
17	6	10	[null]
18	4	11	1
19	1	12	5

d. A list of all SQL queries used and where it can be found in the code (i.e., file name

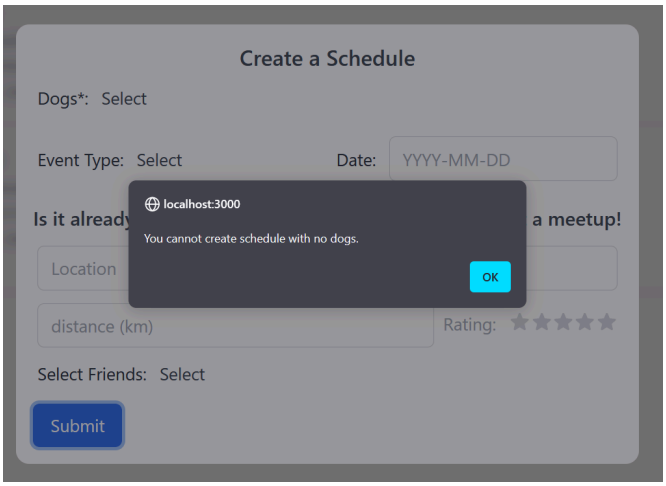
and line number(s))

INSERT – Schedule a task

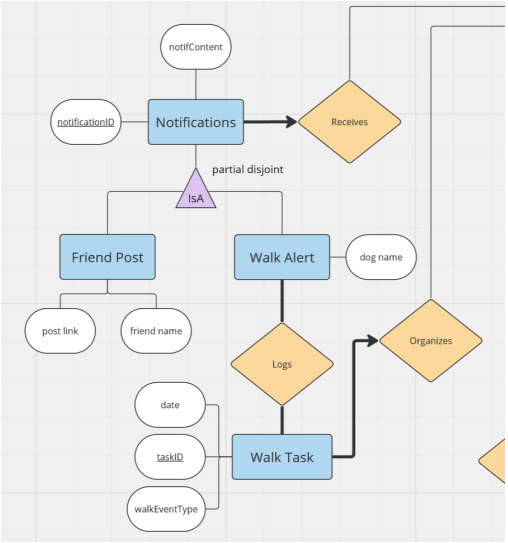
The user is able to pre-schedule a task in which they can turn into a finished walk. This is seen in the form when “Schedule a Task!” button is pressed at the sidebar.



The user is asked to input dogs, event type, and date. They are required to choose at least one dog from their own dog list, or dogs of their friends. Otherwise they get an alert. These values are based on the Owns_Dog tuples.



If there is no location inserted, this form will only affect tables to create a schedule. Affected tables are: Receives_Notifications (and WalkAlert from ISA), Organizes_WalkTask, and Logs, which are these parts of our diagram:



Filename: *src/backend/services/notificationService.js* (Line 30)

Results from the database when user inputs Dog as Blanky, Event as run, and date as 2024-04-05.

Before:

```
SELECT DISTINCT rn.notificationid, rn.notifcontent, wa.dogname,
                ow.taskid, ow.date, ow.walkeventtype
FROM receives_notifications rn
JOIN walkalert wa ON wa.notificationid = rn.notificationid
JOIN logs l ON l.notificationid = wa.notificationid
JOIN organizes_walktask ow ON l.taskid = ow.taskid
```

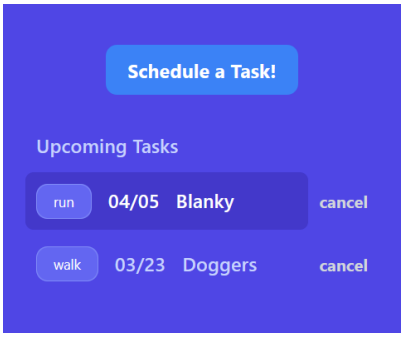
Data Output | Messages | Notifications

	notificationid integer	notifcontent character varying (255)	dogname character varying (255)	taskid integer	date date	walkeventtype event_type
1	12	Time to walk Tory!	Tory	4	2024-04-01	run
2	10	Time to walk Arfy!	Arfy	2	2024-04-05	hike
3	13	Time to walk K91!	K91	5	2024-03-15	walk
4	9	Time to walk Doggers!	Doggers	1	2024-03-23	walk
5	11	Time to walk Blanky!	Blanky	3	2024-03-28	dog park
6	14	Time to walk K92!	K92	5	2024-03-15	walk

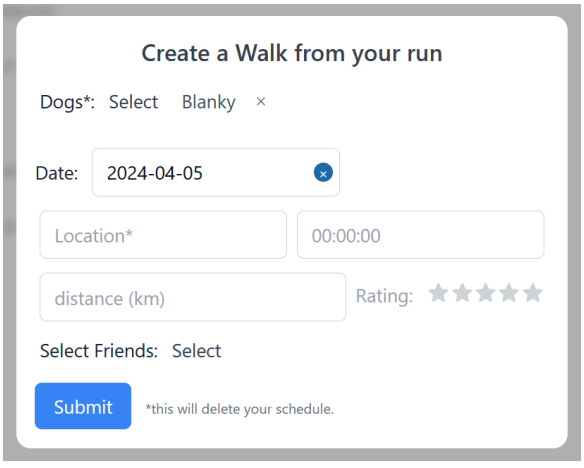
After:

	notificationid integer	notifcontent character varying (255)	dogname character varying (255)	taskid integer	date date	walkeventtype event_type
1	14	Time to walk K92!	K92	5	2024-03-15	walk
2	11	Time to walk Blanky!	Blanky	3	2024-03-28	dog park
3	13	Time to walk K91!	K91	5	2024-03-15	walk
4	15	Time to run with Blanky!	Blanky	6	2024-04-05	run
5	9	Time to walk Doggers!	Doggers	1	2024-03-23	walk
6	10	Time to walk Arfy!	Arfy	2	2024-04-05	hike
7	12	Time to walk Tory!	Tory	4	2024-04-01	run

These attributes are inserted to different tables. We can also view this right after we hit the submit button in our app.



We can create a walk by completing the form from the “Schedule a Task” button, but we can also use our current data from our upcoming tasks. If we click on one upcoming task, we receive this form:



If we submit Blanky, 2024-04-05, and a new location called Downtown we see these results:

Before:

```
1  SELECT DISTINCT w.walkid, wf.dogid, od.name, w.location, wd.date, wdi.distance
2  FROM walk w JOIN walk_date wd ON w.walkid = wd.walkid
3  JOIN walk_dist wdi ON w.walkid = wdi.walkid
4  LEFT JOIN wentfor wf ON w.walkid = wf.walkid
5  LEFT JOIN owns_dog od ON wf.dogid = od.dogid
```

Data Output | Messages | Notifications

	walkid integer	dogid integer	name character varying (255)	location character varying (255)	date date	distance double precision
1	1	1	Doggers	Central Pond	2024-03-23	2.5
2	2	2	Arfy	Beyond the sea	2024-02-20	[null]
3	3	1	Doggers	Vancouver Downtown	2024-02-28	[null]
4	3	2	Arfy	Vancouver Downtown	2024-02-28	[null]
5	3	3	Blanky	Vancouver Downtown	2024-02-28	[null]
6	4	4	Tory	Long Lane Park	2024-01-14	60
7	5	5	K91	Police Station Park	2024-01-19	[null]
8	5	6	K92	Police Station Park	2024-01-19	[null]
9	6	6	K92	Long Lane Park	2024-01-14	[null]
10	7	1	Doggers	Central Pond	2024-02-28	[null]
11	7	2	Arfy	Central Pond	2024-02-28	[null]
12	8	2	Arfy	Metrotown	2024-02-20	[null]
13	8	3	Blanky	Metrotown	2024-02-20	[null]
14	9	1	Doggers	UBC	[null]	[null]
15	9	3	Blanky	UBC	[null]	[null]
16	10	5	K91	Police Station	2024-05-28	[null]
17	10	6	K92	Police Station	2024-05-28	[null]
18	11	4	Tory	Long Lane Park	2024-01-01	50
19	12	1	Doggers	Central Pond	2024-02-14	[null]

After: (cut off the details from row 1 to 15)

16	10	5	K91	Police Station	2024-05-28	[null]
17	10	6	K92	Police Station	2024-05-28	[null]
18	11	4	Tory	Long Lane Park	2024-01-01	50
19	12	1	Doggers	Central Pond	2024-02-14	[null]
20	13	3	Blanky	Downtown	2024-04-05	[null]

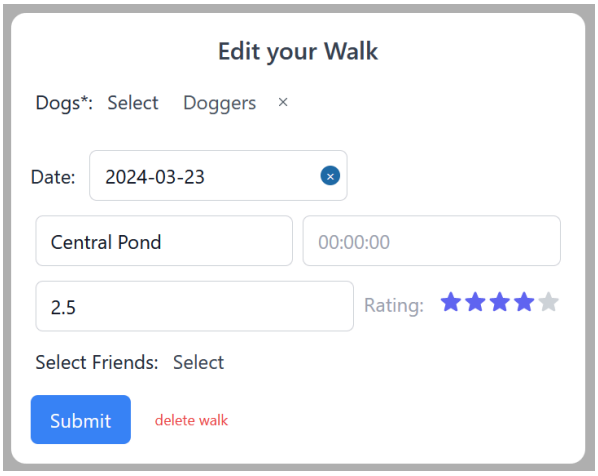
We also had the scheduled walk based on this deleted.

	notificationid integer	notifcontent character varying (255)	dogname character varying (255)	taskid integer	date date	walkeventtype event_type
1	14	Time to walk K92!	K92	5	2024-03-15	walk
2	11	Time to walk Blanky!	Blanky	3	2024-03-28	dog park
3	13	Time to walk K91!	K91	5	2024-03-15	walk
4	9	Time to walk Doggers!	Doggers	1	2024-03-23	walk
5	10	Time to walk Arfy!	Arfy	2	2024-04-05	hike
6	12	Time to walk Tory!	Tory	4	2024-04-01	run

DELETE – Delete a walk (and subsequent post details)

File name: *src/backend/services/walkService.js* (Line 155)

Our main deletions are deleting walks and posts that are made by the owner. For the walks, these can be done by the user through clicking one of their previous walks from the sidebar, which then shows an edit/delete form:



This compound tuple is seen in the database like this:

```
SELECT DISTINCT w.walkid, wf.dogid, od.name, pw.postid, w.location, wd.date,
                wdi.distance, wf.rating
FROM walk w JOIN walk_date wd ON w.walkid = wd.walkid
JOIN walk_dist wdi ON w.walkid = wdi.walkid
LEFT JOIN wentfor wf ON w.walkid = wf.walkid
LEFT JOIN owns_dog od ON wf.dogid = od.dogid
LEFT JOIN post_walk pw ON pw.walkid = w.walkid
LEFT JOIN post_media pm ON pw.postid = pm.postid
```

Data Output | Messages | Notifications

	walkid integer	dogid integer	name character varyir	postid integer	location character varying (255)	date date	distance double precis	rating rate_score
1	1	1	Doggers	1	Central Pond	2024-03-23	2.5	4
2	2	2	Arfy	2	Beyond the sea	2024-02-20	[null]	5
3	3	1	Doggers	3	Vancouver Downtown	2024-02-28	[null]	3
4	3	2	Arfy	3	Vancouver Downtown	2024-02-28	[null]	3
5	3	3	Blanky	3	Vancouver Downtown	2024-02-28	[null]	3
6	4	4	Tory	4	Long Lane Park	2024-01-14	60	[null]
7	5	5	K91	5	Police Station Park	2024-01-19	[null]	1
8	5	6	K92	5	Police Station Park	2024-01-19	[null]	1
9	6	6	K92	6	Long Lane Park	2024-01-14	[null]	[null]
10	7	1	Doggers	[null]	Central Pond	2024-02-28	[null]	3

Once “delete walk” is clicked, the database will remove this.

Data Output | Messages | Notifications

	walkid integer	dogid integer	name character vary	postid integer	location character varying (255)	date date	distance double precis	rating rate_score
1	2	2	Arfy	2	Beyond the sea	2024-02-20	[null]	5
2	3	1	Doggers	3	Vancouver Downtown	2024-02-28	[null]	3
3	3	2	Arfy	3	Vancouver Downtown	2024-02-28	[null]	3
4	3	3	Blanky	3	Vancouver Downtown	2024-02-28	[null]	3
5	4	4	Tory	4	Long Lane Park	2024-01-14	60	[null]
6	5	5	K91	5	Police Station Park	2024-01-19	[null]	1
7	5	6	K92	5	Police Station Park	2024-01-19	[null]	1
8	6	6	K92	6	Long Lane Park	2024-01-14	[null]	[null]
9	7	1	Doggers	[null]	Central Pond	2024-02-28	[null]	3
10	7	2	Arfy	[null]	Central Pond	2024-02-28	[null]	3
11	8	2	Arfy	[null]	Metrotown	2024-02-20	[null]	[null]
12	8	3	Blanky	[null]	Metrotown	2024-02-20	[null]	[null]

We can also check the affected table, Post_Walk, has been cascaded and deleted in the database and the newsfeed of the application.

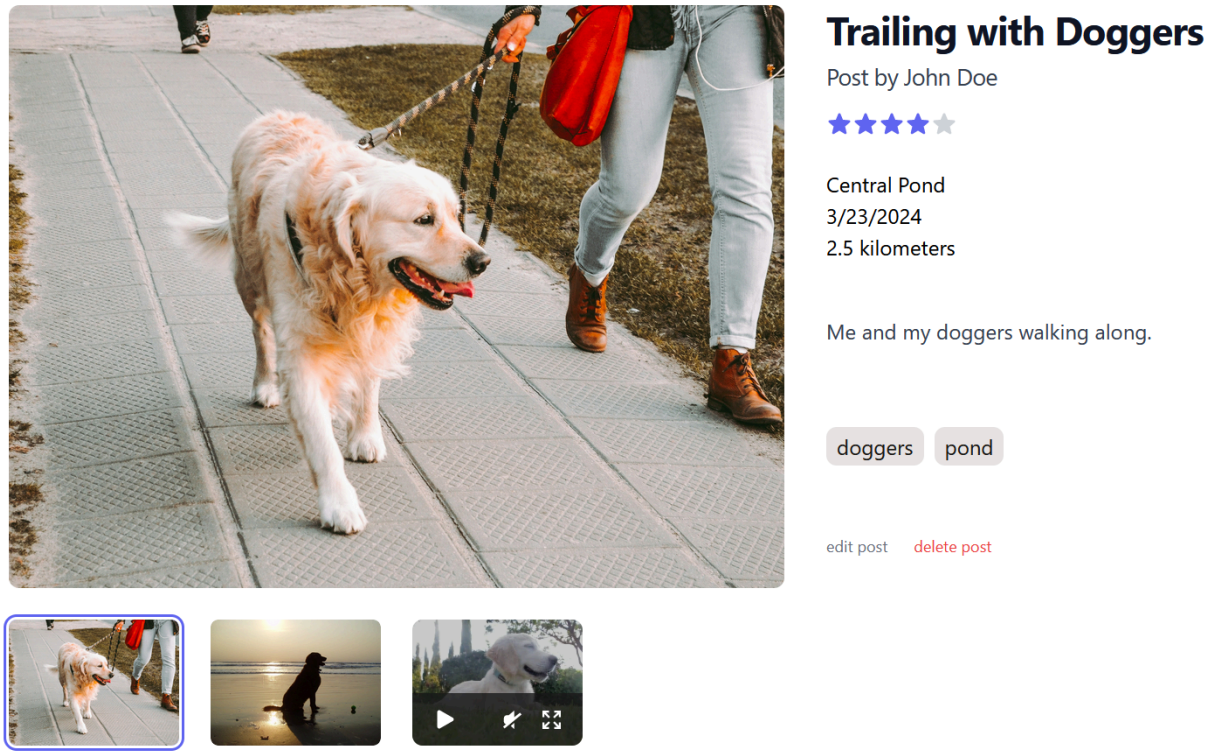
Query Query History

```
1  SELECT * FROM public.post_walk
2  ORDER BY postid ASC LIMIT 100
```

Data Output Messages Notifications

	postid [PK] integer	walkid integer
1	2	2
2	3	3
3	4	4
4	5	5
5	6	6

For deleting posts however, this would not delete the walk. As you can only post in the application if it was from a logged walk. But deleting the post would also delete the Post_Media tuples related to it that fetches the images/videos.



UPDATE - Update post information (rating, location, distance content, dogs)

Filenames and functionalities:

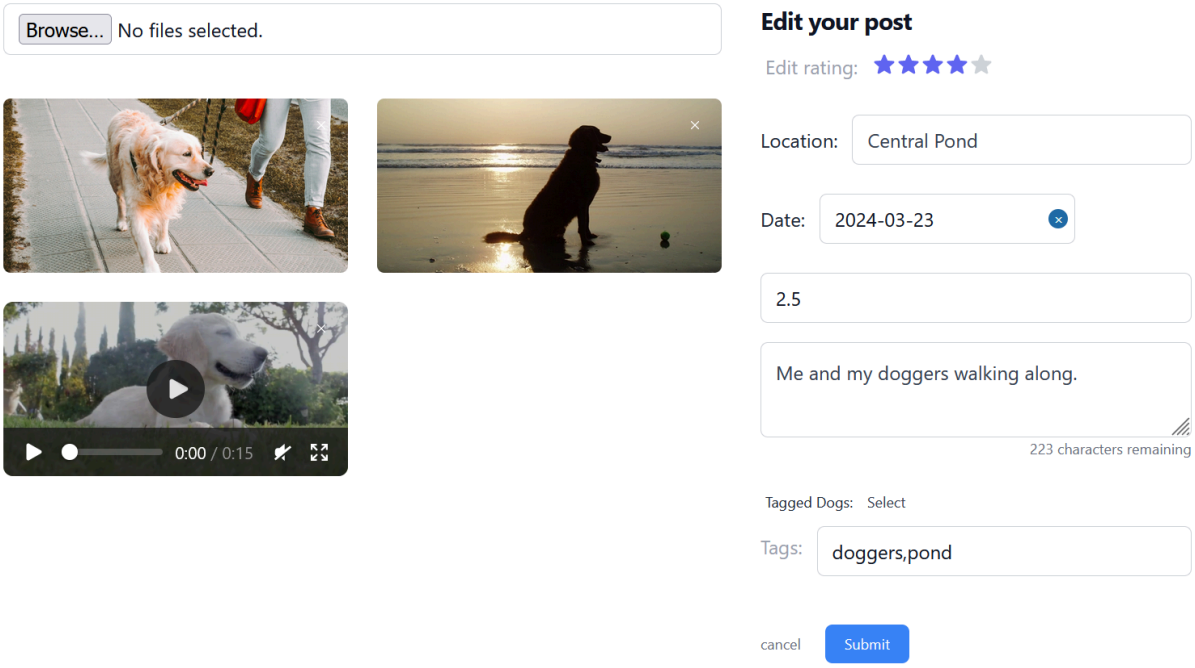
Changing location - *src/backend/services/walkService.js* (Line 190)

Adding distance - *src/backend/services/walkService.js* (Line 220)

Changing rating - *src/backend/services/wentForService.js* (Line 31)

Removing a dog (pk) - *src/backend/services/wentForService.js* (Line 46)

To update the post in the application, the user has to click the “edit post” button, which can only be seen if the post is created by the owner. This mostly can only update Post_Walks’, Walk’s, and WentFor’s attributes.



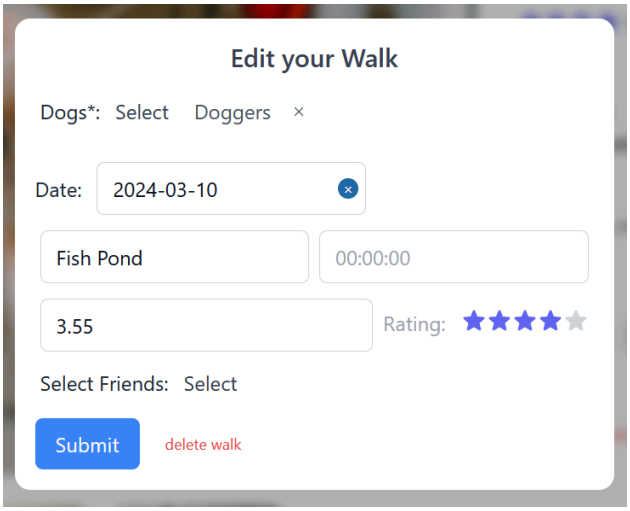
The values of this post can be seen in the database:

14	Doggers	Central Pond	2024-02-28	[null]	[null]	[null]	3
15	Doggers	Central Pond	2024-03-23	Me and my doggers walking along.	photo (1).jpg	2.5	4
16	Doggers	Central Pond	2024-03-23	Me and my doggers walking along.	photo (1).webp	2.5	4
17	Doggers	Central Pond	2024-03-23	Me and my doggers walking along.	video (1).mp4	2.5	4

If we change the values such as location, date, content, it will update these tuples:

13	Doggers	Central Pond	2024-02-14	[null]	[null]	[null]	5
14	Doggers	Central Pond	2024-02-28	[null]	[null]	[null]	3
15	Doggers	Fish Pond	2024-03-10	Swimming my dog	photo (1).jpg	3.55	4
16	Doggers	Fish Pond	2024-03-10	Swimming my dog	photo (1).webp	3.55	4
17	Doggers	Fish Pond	2024-03-10	Swimming my dog	video (1).mp4	3.55	4
18	Doggers	UBC	[null]	[null]	[null]	[null]	5

They can also update walk tuples by clicking their previous walks in the sidebar. But it cannot update Post_Walks' attributes.



SELECTION - Searching for Owners

We implemented a search bar that allows users to enter free form text, using the wildcard operator we would execute the query that would filter down all owners to the matching ones
Filename: *src/backend/services/ownerService.js* (Line 197)

Before:
We represent the before with a blank search bar, therefore no tuples returned.

During:
Add “J” to our search bar

After/During:
What the SQL query returns given a text input (in this case “J”)

```
SELECT array_agg(CONCAT(firstname, ' ', lastname))
FROM owner_name
WHERE firstname LIKE 'J%' OR lastname LIKE 'J%';
```

Data Output | Messages | Notifications

	array_agg text[]
1	{"John Doe","Jack Bean"}

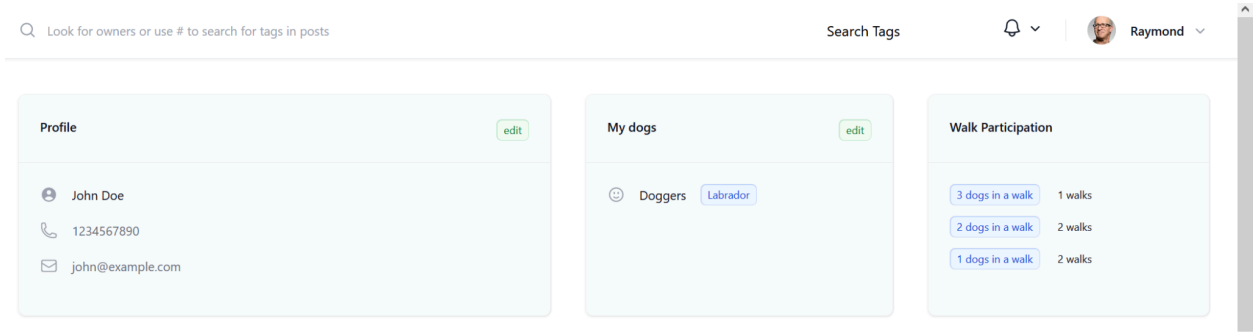
After: The user is able to select the name from the drop down and taken to that profile page

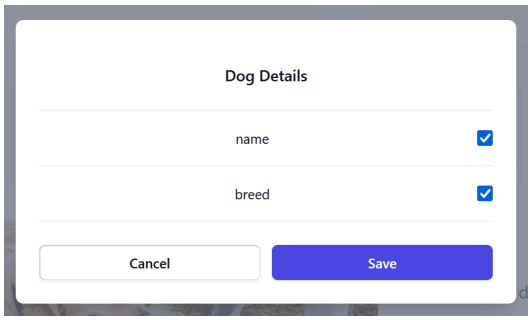
PROJECTION - Displaying dog data on a user’s profile

The user is able to select/deselect the data that they want to show for their dog (name, breed).

Filenames:
src/backend/services/ownsDogService.js (Line 101)
src/frontend/src/components/ProfilePage/myDogModal.js (Line 31)

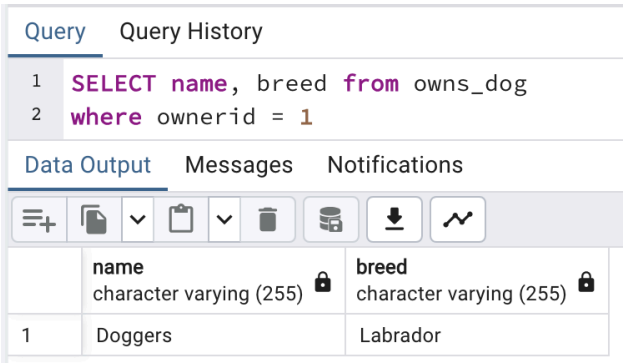
This can be seen in the Profile Page of the Owner:





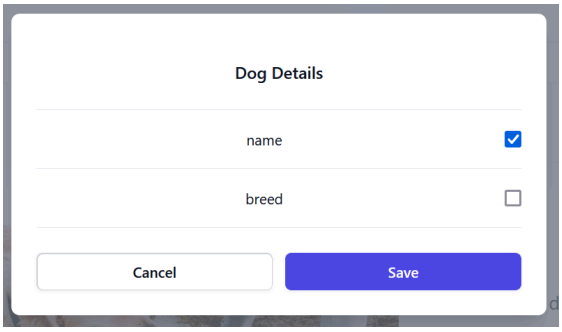
Before:

The default is to show all data from the dog table (for the given ownerid).

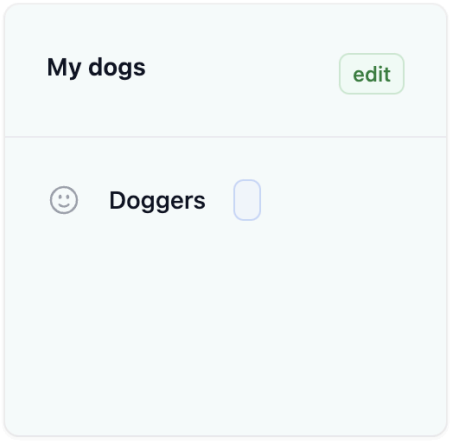


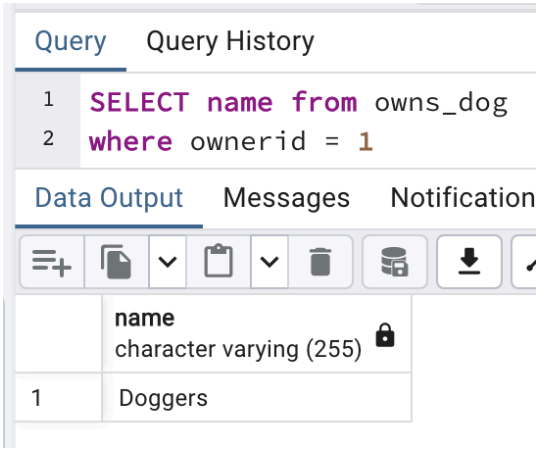
During:

The user deselects “breed”:



After:



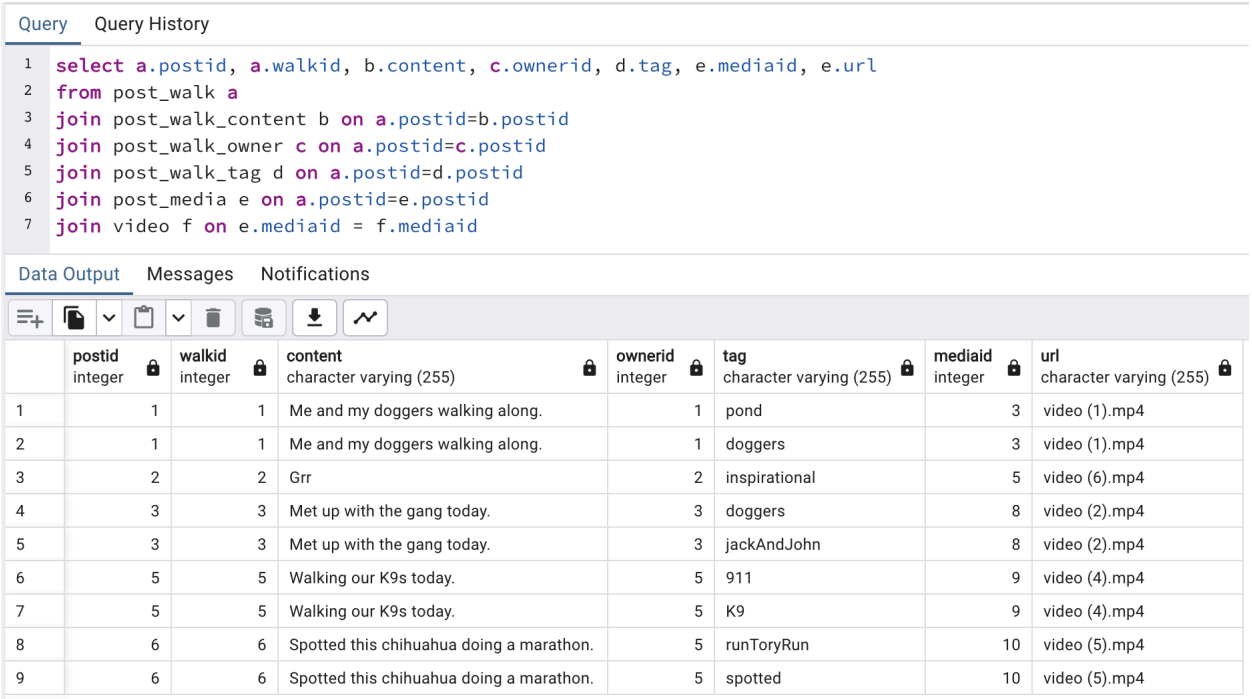


JOIN - The users ability to create and view posts with as much or as little info as they would like

While creating a post can be considered an INSERT statement, the user's view of the post is comprised of a JOIN of many tables (seen below)

Filename: *src/backend/services/postWalkService.js* (Line 26)

Before:



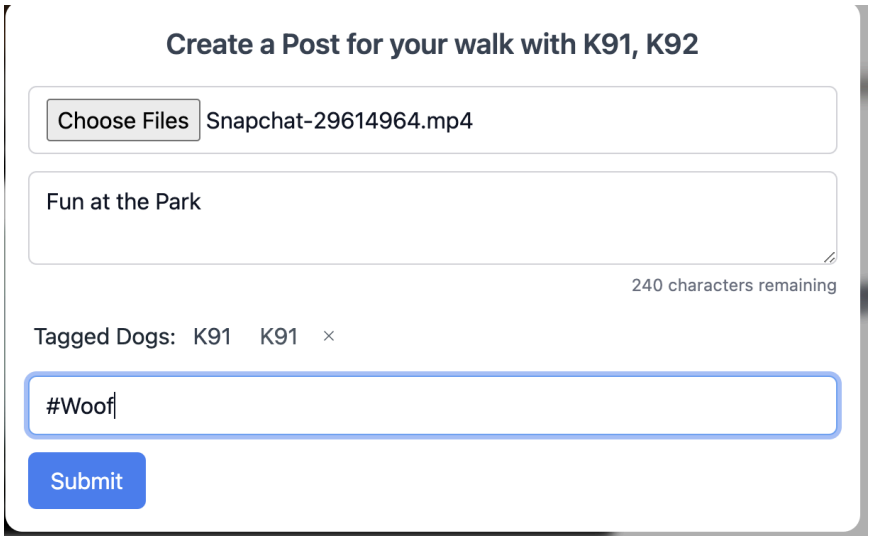
Query | Query History

```
1 select a.postid, a.walkid, b.content, c.ownerid, d.tag, e.mediaid, e.url
2 from post_walk a
3 join post_walk_content b on a.postid=b.postid
4 join post_walk_owner c on a.postid=c.postid
5 join post_walk_tag d on a.postid=d.postid
6 join post_media e on a.postid=e.postid
7 join video f on e.mediaid = f.mediaid
```

Data Output | Messages | Notifications

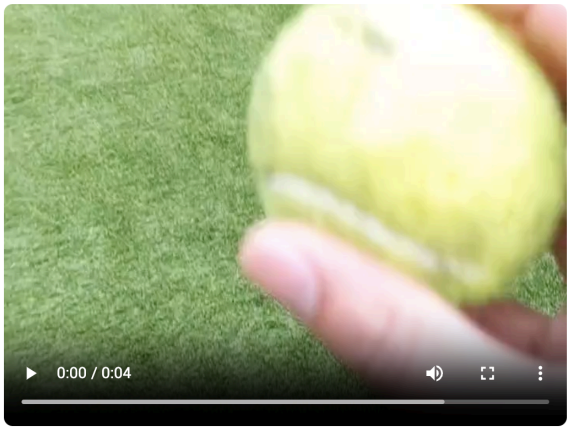
	postid integer	walkid integer	content character varying (255)	ownerid integer	tag character varying (255)	mediaid integer	url character varying (255)
1	1	1	Me and my doggers walking along.	1	pond	3	video (1).mp4
2	1	1	Me and my doggers walking along.	1	doggers	3	video (1).mp4
3	2	2	Grr	2	inspirational	5	video (6).mp4
4	3	3	Met up with the gang today.	3	doggers	8	video (2).mp4
5	3	3	Met up with the gang today.	3	jackAndJohn	8	video (2).mp4
6	5	5	Walking our K9s today.	5	911	9	video (4).mp4
7	5	5	Walking our K9s today.	5	K9	9	video (4).mp4
8	6	6	Spotted this chihuahua doing a marathon.	5	runToryRun	10	video (5).mp4
9	6	6	Spotted this chihuahua doing a marathon.	5	spotted	10	video (5).mp4

During:



After:

9	6	6	Spotted this chihuahua doing a marathon.	5	spotted	10	video (5).mp4
10	7	10	Fun at the Park!	1	woof	11	files - 1712343877402.mp4



Trailing with K91, K92

Post by John Doe

Police Station
5/28/2024

Fun at the Park!

woof

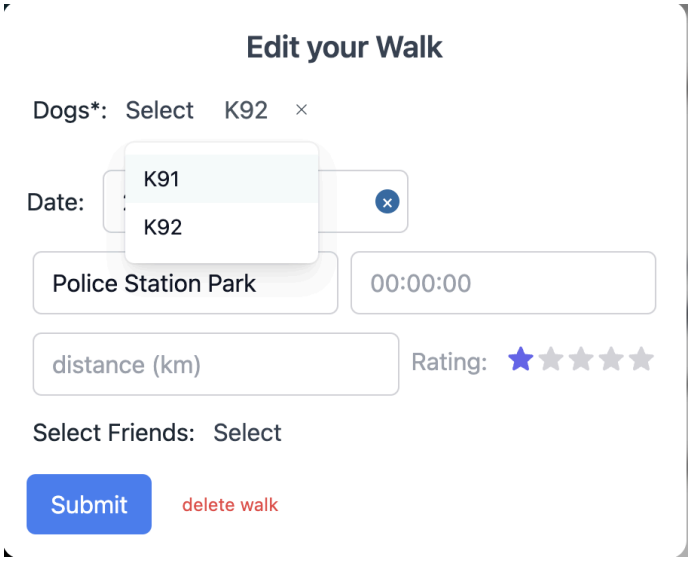
GROUP BY AGGREGATION - Number of dogs taken on a walk (count) is displayed after a walk is completed

The number of dogs taken on a walk is dynamically updated according to the users values
src/backend/services/walkService.js (Line 145)

Before:

	walkid integer	num_dogs bigint
1	10	2
2	5	1
3	6	1

During:



After:

	walkid integer	num_dogs bigint
1	10	2
2	5	2
3	6	1

HAVING - Select posts where their walks are either meetups, group dog walks, or none of them.

The user can filter their newsfeed to see only group dog walks or only meet ups (or all)

src/backend/services/postWalkService.js (Line 176)

src/frontend/src/components/HomePage/SidebarEdit.js (Line 24)

Before:

Having no conditions (aka all posts shown)

Query | Query History | Scratch Pad

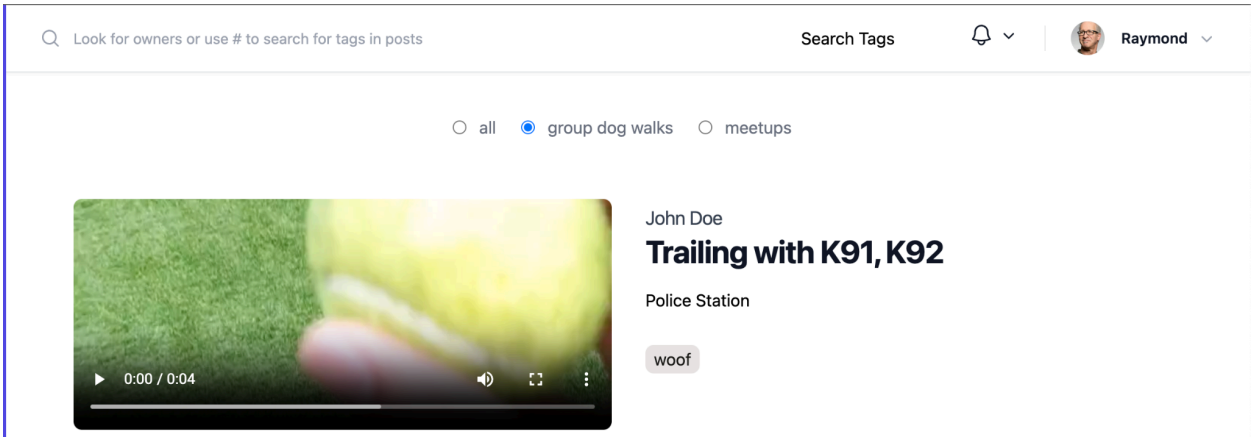
```
SELECT
          pw.postID,
          w.walkID,
          array_agg(DISTINCT od.name) AS dogs,
          own.ownerID,
          CONCAT(own.firstName, ' ', own.lastName) AS owner_name,
          w.location,
          array_agg(DISTINCT pm.url) AS urls,
          array_agg(DISTINCT CONCAT(own1.firstName, ' ', own1.lastName))
            FILTER (WHERE CONCAT(own1.firstName, ' ', own1.lastName) IS NOT NULL
            AND own1.ownerID <> own.ownerID)
              AS met_up_owners,
          array_agg(DISTINCT pwt.tag) AS tags
    FROM Post_Walk pw
    JOIN Walk w ON pw.walkID = w.walkID
```

Data Output | Messages | Notifications

	postid integer	walkid integer	dogs character varying[]	ownerid integer	owner_name text	location character varying (255)	urls character varying[]
1	7	10	{K91,K92}	1	John Doe	Police Station	{"files - 1712343877402.mp4"}
2	6	6	{K92}	5	Popo Lice	Long Lane Park	{"video (5).mp4"}
3	5	5	{K91,K92}	5	Popo Lice	Police Station Park	{"video (4).mp4"}
4	4	4	{Tory}	4	Null Ly	Long Lane Park	{NULL}
5	3	3	{Arfy,Blanky,Doggers}	3	One Three	Vancouver Downtown	{"photo (3).jpg","photo (9).jpg","video (2).mp4"}
6	2	2	{Arfy}	2	Jack Bean	Beyond the sea	{"photo (10).jpg","video (6).mp4"}
7	1	1	{Doggers}	1	John Doe	Central Pond	{"photo (1).jpg","photo (1).webp","video (1).mp4"}

During:

The User selects Group dog walks



After:
The condition is inserted as a HAVING query and the posts are filtered

```
28  HAVING COUNT(DISTINCT od.dogid) >= 2
29  ORDER BY pw.postID DESC, wd.date DESC;
```

Data Output | Messages | Notifications

	postid integer	walkid integer	dogs character varying[]	ownerid integer	owner_name text	location character varying (255)	urls character varying[]
1	7	10	{K91,K92}	1	John Doe	Police Station	{"files - 1712343877402.mp4"}
2	5	5	{K91,K92}	5	Popo Lice	Police Station Park	{"video (4).mp4"}
3	3	3	{Arfy,Blanky,Doggers}	3	One Three	Vancouver Downtown	{"photo (3).jpg","photo (9).jpg","video (2).mp4"}

<u>NESTED AGGREGATION - The user can view walks according to number of dogs participating</u>
Located on the profile page there is a component called Walk Participation, this is dynamically updated to show the number of walks with 1, 2, or 3 dogs

Filename: *src/backend/services/postWalkService.js* (Line 108)

Before:

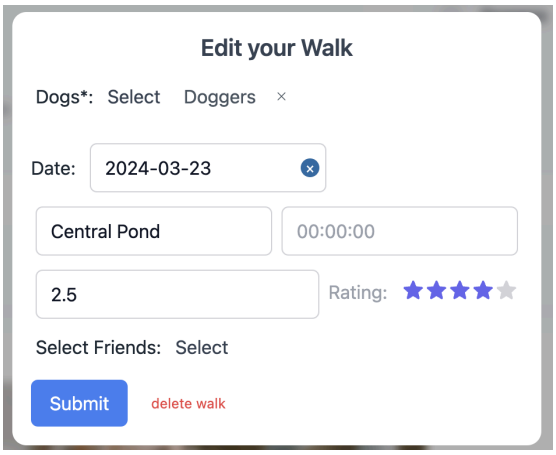
Query | Query History

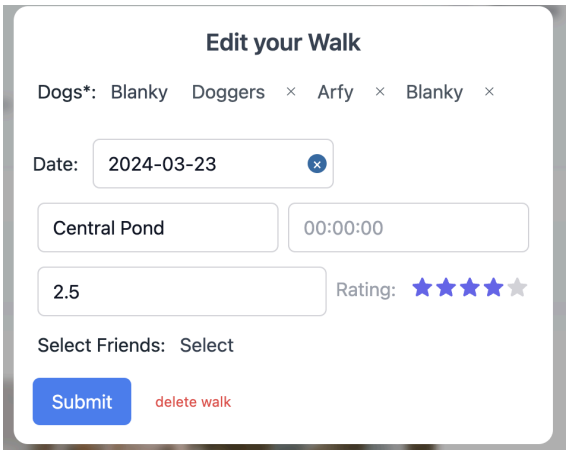
```
1   SELECT num_dogs, COUNT(*) AS num_walks
2   FROM (
3         SELECT w.walkid, COUNT(wf.dogid) AS num_dogs
4         FROM Walk w
5         JOIN WentFor wf ON w.walkid = wf.walkid
6         WHERE w.walkid IN (
7                   SELECT wf.walkid
8                   FROM WentFor wf
9                   JOIN owns_dog od ON wf.dogid = od.dogid
10                  WHERE od.ownerid = 1
11        )
12        GROUP BY w.walkid
13  ) AS walk_dog_count
14  GROUP BY num_dogs
15  ORDER BY num_dogs DESC
```

Data Output | Messages | Notifications

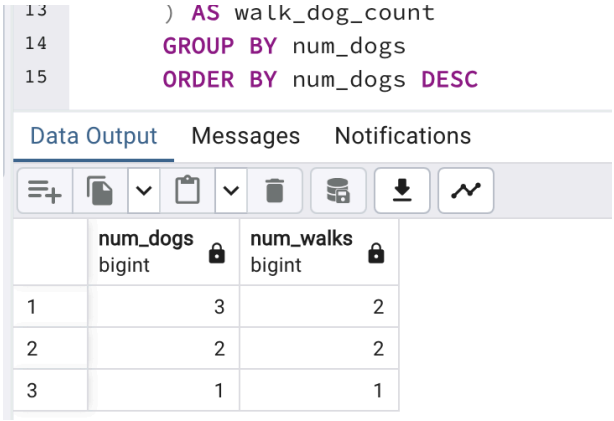
	num_dogs bigint	num_walks bigint
1	3	1
2	2	2
3	1	2

During:
The user will update a completed walk to add two additional dogs.





After:
The number of 3 dog walks is incremented by 1.



DIVISION - Find posts that have ALL the selected tags

findAllTags – src/backend/services/taggedInService.js (Line 59)

Before:

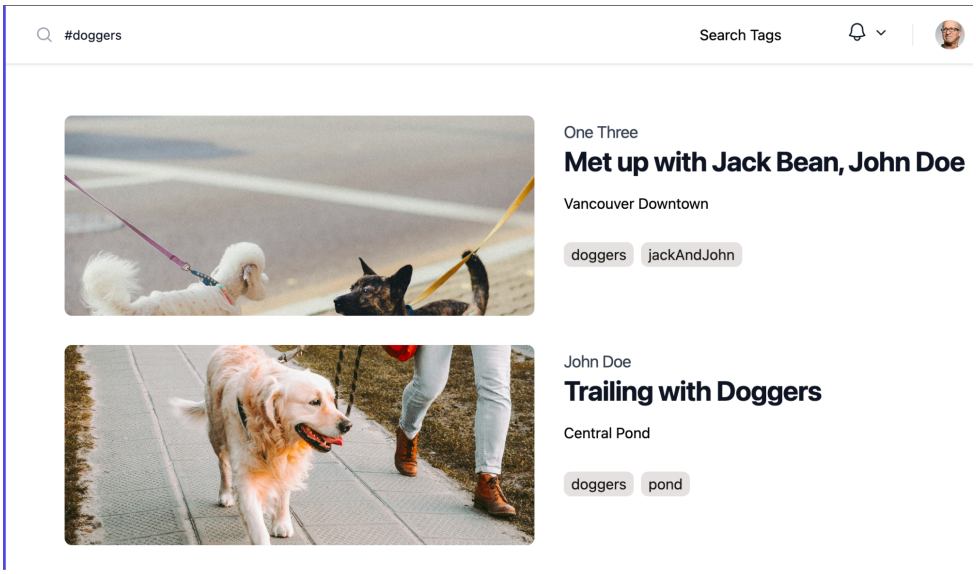
This query creates a table of the requested tags from the user and returns the post(s) that contain all the selected tags. We show the tag table to display walks with multiple tags:

	postid [PK] integer	tag [PK] character varying (255)
1	1	doggers
2	1	pond
3	2	inspirational
4	3	doggers
5	3	jackAndJohn
6	4	runToryRun
7	5	911
8	5	K9
9	6	runToryRun
10	6	spotted

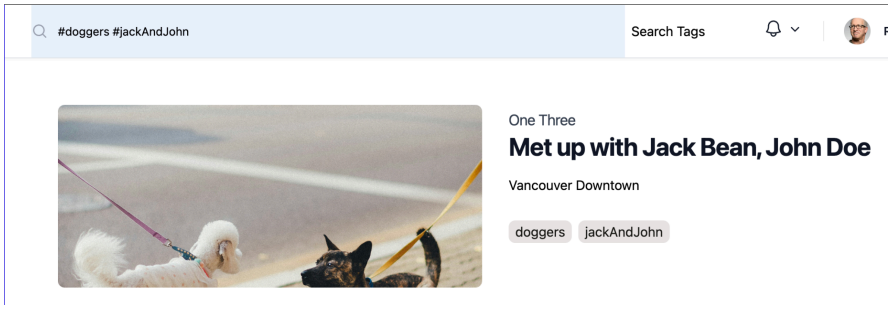
During:

The user will type in the tags they are searching for, we will use #doggers and #jackAndJohn to demonstrate only post 3 will be returned.

First we search #doggers and two posts are returned



Now we add #jackAndJohn and the user see's only post 3



After:

The output is a table containing posts with ALL the matching tags (note in the program there is a temporary table created with the user searched tags, and then subsequently dropped. That table has been added manually to show the db change)

Query | Query History

```
SELECT
        pw.postID,
        w.walkID,
        array_agg(DISTINCT od.name) AS dogs,
        own.ownerID,
        CONCAT(own.firstName, ' ', own.lastName) AS owner_name,
        w.location,
        array_agg(DISTINCT pm.url) AS urls,
        array_agg(DISTINCT CONCAT(own1.firstName, ' ', own1.lastName))
          FILTER (WHERE CONCAT(own1.firstName, ' ', own1.lastName) IS NOT NULL
          AND own1.ownerID <> own.ownerID)
            AS met_up_owners,
        array_agg(DISTINCT pwt.tag) AS tags
        FROM Post_Walk pw
        JOIN Walk w ON pw.walkID = w.walkID
        JOIN WentFor wf ON w.walkID = wf.walkID
        JOIN Owns_Dog od ON wf.dogID = od.dogID
        JOIN Post_Walk_Owner pwo ON pw.postID = pwo.postID
        JOIN Owner_Name own ON own.ownerID = pwo.ownerID

        LEFT JOIN Post_Media pm ON pw.postID = pm.postID
        LEFT JOIN Post_Walk_Tag pwt ON pw.postID = pwt.postID
        LEFT JOIN On_MeetUp omu ON w.walkID = omu.walkID
        LEFT JOIN Walk_Date wd ON w.walkID = wd.walkID
        LEFT JOIN Schedules s ON omu.meetUpID = s.meetUpID
        LEFT JOIN Owner_Name own1 ON s.ownerID = own1.ownerID
        WHERE EXISTS (SELECT DISTINCT p.postid
                      FROM post_walk_tag p
                      WHERE NOT EXISTS (SELECT tag
```

Data Output | Messages | Notifications

	postid integer	walkid integer	dogs character varying[]	ownerid integer	owner_name text	location character varying (255)	urls character varying[]	met_up_owners text[]	tags character v
1	3	3	{Arfy,Blanky,Doggers}	3	One Three	Vancouver Downtown	{"photo (3).jpg","photo (9).jpg","video (2).mp4"}	{"Jack Bean","John Doe"}	{doggers,ja