University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: **4**

Link:
https://github.students.cs.ubc.ca/CPSC304-2023W-T1/project_c1p1k_l6f8n_s4k7q

Date: **December 1, 2023**

Group Number: **82**

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Stanley Cheung	32009722	s4k7q	yinstanleycheung@gmail.com
Matias Gauvin	27796267	l6f8n	matiasgauvin@gmail.com
Sunny Nie	59484840	c1p1k	sunnymingnie@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.

Milestone 4

Project Description

Our project, Biddr, is a combination between a traditional social-media app and a bidding/fundraiser site. A user can search for communities and join them to view posts from those communities in the homepage. The user can click on a community to bring up posts from that community, and can click on a post to see the comments and transaction history of that post. The homepage shows all posts for all subscribed communities of a user which can be filtered. A user can belong in a chat and send messages to other users in that chat. Additionally, there is a search page for users to look up any combination of attributes for each table. There is also a profile page where users can edit their information.

Difference in Final Schema vs. Milestone 3 Schema

We added a refreshToken field of type VARCHAR(256) to the AppUser table because it's a small technical quirk we needed for our JWT authentication system to save user login sessions. We also added a Joins table to represent a relationship between the User and Community relations so we can fetch posts from communities that the user is part of. Additionally, this is an intuitive functionality as users should be able to join communities. We also changed some attribute data types to PostgreSQL types (ie. mediumblob to bytea and datetime to timestamp) because that is our DBMS of choice for our project.

Final Schema + Screenshots of Data in Tables

Final Schema:

- AppUser(email: varchar[256], username: varchar[32], profilePicture: bytea, fullName: varchar[32], hashedPassword: varchar[256], timeJoined: timestamp, bio: varchar[512], **dateOfBirth: timestamp**, **location: varchar[32],** refreshtoken: varchar[256])

	email [PK] character varying (256)	username character varying (32)	profilepicture bytea	fullname character varying (32)	hashedpassword character varying (256)	timejoined timestamp without time zone	bio character varying (512)	dateofbirth date	location character varying (32)	refreshtoken character varying (256)
1	woodgregory@example.net	trouble	[null]	Benjamin Schmitt	09a3871fa15155c510b8b23733c47be8	1987-05-23 12:03:01	million ground human	2017-06-25	Liechtenstein	[null]
2	cgarza@example.org	how	[null]	Karen Ross	533cb108fc75cac8f5c20d6fd8ccdf2e	2017-02-02 03:31:43	federal partner also	1980-06-16	Eritrea	[null]
3	millerdaniel@example.com	financial	[null]	Charles Fox	c6187c365b23b135ff561c8c278cdcf1	1990-04-29 19:16:19	air his develop	1970-06-28	Ukraine	[null]
4	harrislatasha@example.org	yard	[null]	Lori Butler	bcf6738be52090341bfef777bdeb4ff8	1974-12-19 08:27:43	tend policy management	1991-03-18	Nicaragua	[null]
5	karen15@example.com	network	[null]	Kari Griffin	85d80aa21d6735e7b6e7fe1eb7e6050c	2023-05-10 23:57:07	boy PM relate	1988-02-28	Christmas Island	[null]
6	chris36@example.net	start	[null]	Brandon Gonzalez	0094386106b4a99d899c014bb31884e2	2000-08-15 16:40:11	smile join ago	1996-01-30	Turkmenistan	[null]
7	rwalter@example.org	former	[null]	Jacob Mack	d7415cdcc79ea33a41a4b527f9f262ee	1985-02-13 07:32:48	word same game	1974-05-01	Yemen	[null]
8	ronald84@example.org	activity	[null]	Laura Kennedy	7e22c0dee896798a63e251fc4ed55934	1999-06-06 11:11:51	bring final through	1981-06-14	Northern Mariana Islands	[null]
9	hallmatthew@example.org	foot	[null]	Derrick Bridges	3b3ad97a9aee71b4e37df8fdbe4b9beb	2000-03-25 16:17:07	Mrs school until	2020-12-13	Korea	[null]
10	jeffreyreeves@example.org	wear	[null]	Robert Woods	87db52446d964090b4b7c862fc590b0c	1980-05-09 14:54:17	free anything send	2018-09-15	Romania	[null]
11	inallchats1@gmail.com	inallchats1	[null]	Bruce Wayne	$2b$10$tiEHk3V5QNF2zTo6LAOJReDCPYX1ooQsHDGV6Ua3auPWQuZXyP…	1980-05-09 14:54:17	I am awesome.	2023-10-19	Italy	[null]
12	inallchats2@gmail.com	inallchats2	[null]	Peter Parker	$2b$10$79sOg3IpH2Geda7DF6qRE.t14P64QRilDjMrs7ufs.K.5KujBk9la	1980-05-09 14:54:17	I am very chill.	2023-08-09	Nicaragua	[null]
13	sunnynie@example.com	sunnynie	[null]	Sunny Nie	$2b$10$Q5/mRRTG2rIH8kJPrMYLTeY5s1aueCSPk2XmSbhOcufyPMsGGSr…	2000-05-09 14:54:17	Sunny acc	2023-08-09	Nicaragua	[null]

- LocationAgeOfMajority(location: varchar[32], ageOfMajority: int)

	location [PK] character varying (32)	ageofmajority integer
1	Liechtenstein	22
2	Eritrea	20
3	Ukraine	21
4	Nicaragua	18
5	Christmas Island	17
6	Turkmenistan	18
7	Yemen	21
8	Northern Mariana Islands	18
9	Korea	19
10	Romania	20
11	China	18
12	India	18
13	United States	18
14	Indonesia	18
15	Brazil	18
16	Bangladesh	18
17	Russia	18
18	Japan	20
19	Pakistan	18
20	Mexico	18
21	Philippines	18
22	Vietnam	16
23	Germany	18
24	Iran	18
25	Turkey	18
26	United Kingdom	18
27	Thailand	20
28	France	18
29	Italy	18
30	Egypt	21
31	South Korea	19
32	Kenya	18
33	Spain	18
34	Colombia	18
35	Canada	19

- LocationDateOfBirthLegalAge(**location:** varchar[32], dateOfBirth: timestamp, isLegalAge: boolean)

	location [PK] character varying (32)	dateofbirth [PK] date	islegalage boolean
1	Christmas Island	1988-02-28	true
2	Romania	2018-09-15	false
3	Korea	2020-12-13	false
4	Liechtenstein	2017-06-25	false
5	Eritrea	1980-06-16	true
6	Turkmenistan	1996-01-30	true
7	Northern Mariana Islands	1981-06-14	true
8	Ukraine	1970-06-28	true
9	Yemen	1974-05-01	true
10	Nicaragua	1991-03-18	true
11	Italy	2023-10-19	false
12	Nicaragua	2023-08-09	false

- Post(<u>postId: </u>int, **postedEmail: varchar[256]**, **walletName: varchar[32]**, **walletEmail**: **varchar[256]**, **communityName: varchar[32]**, timePosted: timestamp, expiryTime: timestamp, text: varchar[512], image: bytea, title: varchar[32])

	postid [PK] integer	postedemail character varying (256)	walletname character varying (32)	walletemail character varying (256)	communityname character varying (32)	timeposted timestamp without time zone	expirytime timestamp without time zone	text character varying (512)	image bytea	title character varying (32)
1	1	ronald84@example.org	provide customer	millerdaniel@example.com	beautiful	1992-06-05 23:10:28	1993-07-05 23:10:28	expert group decide	[null]	join should
2	2	cgarza@example.org	send man	cgarza@example.org	more	1987-01-20 23:57:12	1988-02-19 23:57:12	thus should now	[null]	physical case
3	3	woodgregory@example.net	Mrs increase	chris36@example.net	among	1985-03-25 03:32:03	1986-04-24 03:32:03	out education I	[null]	question point
4	4	rwalter@example.org	send man	cgarza@example.org	side	2013-12-04 05:55:04	2015-01-03 05:55:04	edge else recent	[null]	popular perform
5	5	millerdaniel@example.com	Mrs increase	chris36@example.net	among	1979-12-22 04:03:16	1981-01-20 04:03:16	big design low	[null]	with citizen
6	6	harrislatasha@example.org	provide customer	millerdaniel@example.com	side	1998-02-06 02:20:55	1999-03-08 02:20:55	single left determine	[null]	new method
7	7	cgarza@example.org	late themselves	ronald84@example.org	beautiful	2006-03-18 01:21:49	2007-04-17 01:21:49	support rate point	[null]	per figure
8	8	karen15@example.com	late themselves	ronald84@example.org	nature	2019-05-01 16:22:50	2020-05-30 16:22:50	sort hair order	[null]	style responsibility
9	9	hallmatthew@example.org	them computer	cgarza@example.org	star	2004-12-31 02:07:11	2006-01-30 02:07:11	like number word	[null]	water believe
10	10	cgarza@example.org	send man	cgarza@example.org	body	1975-10-09 06:12:48	1976-11-07 06:12:48	notice marriage not	[null]	thus kid
11	11	harrislatasha@example.org	them computer	cgarza@example.org	star	1993-04-27 04:53:54	1994-05-27 04:53:54	democratic amount kid	[null]	room town
12	12	jeffreyreeves@example.org	them computer	cgarza@example.org	body	1976-01-20 14:43:52	1977-02-18 14:43:52	party third teacher	[null]	tonight word
13	13	ronald84@example.org	send man	cgarza@example.org	more	2005-06-10 21:07:31	2006-07-10 21:07:31	early article subject	[null]	shake food
14	14	rwalter@example.org	help involve	chris36@example.net	benefit	2010-06-09 21:03:07	2011-07-09 21:03:07	car chance project	[null]	represent organization
15	15	woodgregory@example.net	late themselves	ronald84@example.org	body	2005-09-11 15:18:11	2006-10-11 15:18:11	movement for defense	[null]	major campaign
16	16	millerdaniel@example.com	send man	cgarza@example.org	more	2020-07-01 05:43:26	2021-07-31 05:43:26	citizen history force	[null]	check avoid
17	17	karen15@example.com	much seven	harrislatasha@example.org	beautiful	1991-07-11 15:01:17	1992-08-09 15:01:17	add success home	[null]	guy leg
18	18	rwalter@example.org	official raise	cgarza@example.org	everybody	2000-07-08 11:27:35	2001-08-07 11:27:35	environment weight think	[null]	material week
19	19	harrislatasha@example.org	provide customer	millerdaniel@example.com	more	2000-01-24 15:23:20	2001-02-22 15:23:20	son himself Republican	[null]	half meeting
20	20	karen15@example.com	help involve	chris36@example.net	among	2023-01-25 02:40:39	2024-02-24 02:40:39	guy indeed and	[null]	nearly argue

- Auction(**<u>postID:</u> int**, minBid: float)

	postid [PK] integer	minbid double precision
1	1	3550.95
2	2	5392.71
3	3	4.46
4	4	[null]
5	5	9.63
6	6	0.01
7	7	42235.16
8	8	55563.29
9	9	0.92
10	10	76496.81

- Fundraiser(**postID: int**, goal: float)

	postid [PK] integer	goal double precision
1	11	6202.05
2	12	5506.01
3	13	73668.05
4	14	[null]
5	15	97992.89
6	16	0.96
7	17	0.57
8	18	4775
9	19	0.73
10	20	85.72

- Chat(chatID: int, chatName: varchar[32])

	chatid [PK] integer	chatname character varying (32)
1	1	per truth article
2	2	day activity process
3	3	tell oil front
4	4	stock democratic federal
5	5	religious summer between
6	6	significant attorney investment
7	7	head site benefit
8	8	improve thing eight
9	9	door expert sea
10	10	white remain quality

- Wallet(<u>walletName:</u> varchar[32], **<u>email</u>: varchar[256]**, balance: float)

	walletname [PK] character varying (32)	**email** [PK] character varying (256)	**balance** double precision
1	help involve	chris36@example.net	0.68
2	them computer	cgarza@example.org	5.4
3	much seven	harrislatasha@example.org	94.05
4	provide customer	millerdaniel@example.com	62.14
5	send man	cgarza@example.org	59336.81
6	late themselves	ronald84@example.org	8.19
7	official raise	cgarza@example.org	5.45
8	knowledge international	ronald84@example.org	3381.56
9	kitchen especially	harrislatasha@example.org	1059.94
10	Mrs increase	chris36@example.net	7309.75

- Bid(<u>bidID</u>: int, **walletName: varchar[32]**, **email: varchar[256]**, **postID: int,** amount: float, timeCreated: timestamp, status: varchar[16])

	bidid [PK] integer	**walletname** character varying (32)	**email** character varying (256)	**postid** integer	**amount** double precision	**timecreated** timestamp without time zone	**status** character varying (16)
1	1	late themselves	ronald84@example.org	5	183.47	2015-11-01 05:58:42	Paid
2	2	send man	cgarza@example.org	6	751.67	1970-09-29 09:22:46	Highest
3	3	much seven	harrislatasha@example.org	1	362.46	2023-09-24 16:43:20	Paid
4	4	official raise	cgarza@example.org	10	788.9	1973-12-14 09:28:36	[null]
5	5	official raise	cgarza@example.org	7	78610.8	2009-07-17 17:25:16	Paid
6	6	kitchen especially	harrislatasha@example.org	3	98.35	2006-12-07 15:47:49	[null]
7	7	official raise	cgarza@example.org	1	63.39	1978-11-19 18:28:32	[null]
8	8	provide customer	millerdaniel@example.com	1	0.08	1972-08-14 07:11:11	Paid
9	9	help involve	chris36@example.net	8	1679.43	1976-06-07 06:08:49	Highest
10	10	official raise	cgarza@example.org	8	7.71	1986-11-06 00:31:55	[null]

- Donation(<u>donationID</u>: int, **walletName: varchar[32]**, **email: varchar[256]**, **postID: int,** amount: float, timeCreated: timestamp)

	donationid [PK] integer	walletname character varying (32)	email character varying (256)	postid integer	amount double precision	timecreated timestamp without time zone
1	1	Mrs increase	chris36@example.net	11	96447.01	1983-07-15 13:57:45
2	2	late themselves	ronald84@example.org	12	37.95	1994-07-22 08:03:24
3	3	much seven	harrislatasha@example.org	17	4.95	1974-05-26 23:36:59
4	4	help involve	chris36@example.net	11	0.87	2016-10-28 02:08:57
5	5	knowledge international	ronald84@example.org	14	726.49	1980-10-30 02:44:52
6	6	help involve	chris36@example.net	15	16.56	2010-10-29 22:56:36
7	7	kitchen especially	harrislatasha@example.org	14	87044.52	1973-03-29 16:55:58
8	8	provide customer	millerdaniel@example.com	11	7.98	1977-11-27 12:13:54
9	9	help involve	chris36@example.net	19	2167.84	1971-05-28 10:16:57

- Community(<u>communityName</u>: varchar[32], **email: varchar[256]**, longName: varchar[64], description: varchar[256])

	communityname [PK] character varying (32)	email character varying (256)	longname character varying (64)	description character varying (256)
1	more	karen15@example.com	north member	may professor clear community in
2	among	ronald84@example.org	green response	audience quickly federal respond hou…
3	body	hallmatthew@example.org	national onto	than former they unit draw
4	nature	jeffreyreeves@example.org	son bit	new organization not husband identify
5	benefit	jeffreyreeves@example.org	all adult	share return decade event professional
6	beautiful	ronald84@example.org	quickly term	person think anything war traditional
7	side	millerdaniel@example.com	time four	bad prepare young anything it
8	everybody	hallmatthew@example.org	hour particularly	wait poor sort economy our
9	star	ronald84@example.org	fly become	series many draw nor strong

- Comment(<u>commentID</u>: int, **email: varchar[256], postID: int,** text: varchar[256], timeSent: timestamp)

	commentid [PK] integer	email character varying (256)	postid integer	text character varying (256)	timesent timestamp without time zone
1	1	woodgregory@example.net	5	indeed war fact	2019-03-31 12:41:15
2	2	rwalter@example.org	9	degree same sort	1979-01-27 17:13:38
3	3	woodgregory@example.net	4	rise system sense	1999-08-13 23:01:18
4	4	harrislatasha@example.org	6	huge clearly car	1976-08-13 19:43:55
5	5	cgarza@example.org	5	former upon sport	1971-02-20 11:26:26
6	6	ronald84@example.org	2	for onto stay	1983-10-21 03:16:56
7	7	cgarza@example.org	4	brother church our	1996-10-15 18:22:07
8	8	ronald84@example.org	10	mission media enough	2014-08-04 06:15:16
9	9	ronald84@example.org	10	by movement quite	1992-01-13 01:35:47

- PrivateMessage(<u>messageID</u>: int, **email: varchar[256], chatID: int,** text: varchar[256], timeSent: timestamp)

	messageid [PK] integer	**email** character varying (256)	**chatid** integer	**text** character varying (256)	**timesent** timestamp without time zone
1	1	hallmatthew@example.org	8	life popular million	1988-03-22 06:53:17
2	2	jeffreyreeves@example.org	9	despite medical safe	2003-08-31 22:31:43
3	3	woodgregory@example.net	4	expect watch right	2002-11-24 10:16:00
4	4	harrislatasha@example.org	10	foot understand little	1972-07-16 19:26:26
5	5	millerdaniel@example.com	7	authority open over	2016-10-10 05:51:15
6	6	jeffreyreeves@example.org	9	pull best chair	2013-02-23 11:30:18
7	7	jeffreyreeves@example.org	10	lose return customer	2015-04-30 20:26:52
8	8	woodgregory@example.net	4	star process election	2004-07-30 07:56:19
9	9	woodgregory@example.net	4	case book spring	1970-04-23 22:22:17

- Likes(**<u>email</u>: varchar[256], <u>postID</u>: int**)

	email [PK] character varying (256)	**postid** [PK] integer
1	karen15@example.com	8
2	millerdaniel@example.com	9
3	chris36@example.net	3
4	ronald84@example.org	4
5	jeffreyreeves@example.org	6
6	cgarza@example.org	5
7	harrislatasha@example.org	7
8	rwalter@example.org	2
9	rwalter@example.org	10

- EngagedIn(**<u>email</u>: varchar[256], <u>chatID</u>: int**)

	email [PK] character varying (256)	chatid [PK] integer
1	harrislatasha@example.org	10
2	millerdaniel@example.com	7
3	chris36@example.net	5
4	jeffreyreeves@example.org	9
5	hallmatthew@example.org	8
6	chris36@example.net	1
7	jeffreyreeves@example.org	10
8	woodgregory@example.net	4
9	millerdaniel@example.com	4
10	inallchats1@gmail.com	1
11	inallchats1@gmail.com	2
12	inallchats1@gmail.com	3
13	inallchats1@gmail.com	4
14	inallchats1@gmail.com	5
15	inallchats1@gmail.com	6
16	inallchats1@gmail.com	7
17	inallchats1@gmail.com	8
18	inallchats1@gmail.com	9
19	inallchats1@gmail.com	10
20	inallchats2@gmail.com	1
21	inallchats2@gmail.com	2
22	inallchats2@gmail.com	3
23	inallchats2@gmail.com	4
24	inallchats2@gmail.com	5
25	inallchats2@gmail.com	6
26	inallchats2@gmail.com	7
27	inallchats2@gmail.com	8
28	inallchats2@gmail.com	9
29	inallchats2@gmail.com	10

- Joins(**<u>communityName</u>: varchar[32], <u>email</u>: varchar[256]**)

	communityname [PK] character varying (32)	email [PK] character varying (256)
1	more	woodgregory@example.net

SQL Queries + Before/During/After Screenshots

INSERT QUERY

Code location:

- **Path:** server/services/AppUserTable.js
- **Line number(s)**
  - 83-101 (CreateAppUser function)
  - 103-116 (CreateLocationDateOfBirthIsLegalAge function)

```
export const CreateAppUser = async (email, data) => {
    try {
        return await query(
            "INSERT INTO AppUser (email, username, profilePicture, fullName, hashedPassword, timeJoined,  bio, dateOfBirth, location) VALUES ($1, $2, $3, $4, $5, CURRENT_TIMESTAMP, $6, $7, $8)",
            [
                email,
                data.username,
                data.profilePicture,
                data.fullName,
                data.password,
                data.bio,
                data.dateOfBirth,
                data.location,
            ]
        );
    } catch (error) {
        throw error;
    }
};

export const CreateLocationDateOfBirthIsLegalAge = async (
    location,
    dateOfBirth,
    isLegalAge
) => {
    try {
        return await query(
            "INSERT INTO LocationDateOfBirthLegalAge (location, dateOfBirth, isLegalAge) VALUES ($1, $2, $3)",
            [location, dateOfBirth, isLegalAge]
        );
    } catch (error) {
        throw error;
    }
};
```

- *Logic for inserting tuples (this is the logic used for case handling if the tuple doesn't exist in the LocationDateOfBirthIsLegalAge table):*
  - ***Path:** server/controllers/AppUserController.js*
  - ***Line number(s):** 24-69 (PostAppUser function)*

```
export const PostAppUser = async (req, res) => {
    const { email, username, fullName, password, location, dateOfBirth } = req.body;
    /*
    These are the attributes labeled NOT NULL in the table creation script.
    We are just checking here that they are not missing from the request.
    */

    if (!email || !username || !fullName || !password || !location || !dateOfBirth) {
        return res.status(400).json({error: "Missing fields"})
    }

    try {
        // Ensure that email does not already exist
        const emailExists = await QueryAppUserByEmail(email)

        if (emailExists && emailExists[0]) {
            return res.status(400).json({error: "Email already exists"});
        }

        // Ensure that username does not already exist
        const usernameExists = await QueryAppUserByUsername(username)
        if (usernameExists[0]) {
            return res.status(400).json({error: "Username already exists"});
        }


        // If location and dateOfBirth are defined, we need to add tuple in LocationDateOfBirthIsLegal age.
        if (location && dateOfBirth) {
            const locationAgeOfMajority = await GetLocationAgeOfMajority(location);
            const ageOfMajority = locationAgeOfMajority[0].ageofmajority;
            const userAge = getAge(dateOfBirth);
            const isLegalAge = userAge >= ageOfMajority;
            const tupleExists = await GetLocationDateOfBirthIsLegalAge(location, dateOfBirth);
            // only create tuple if it doesn't already exist.
            if(!tupleExists[0]) {
                await CreateLocationDateOfBirthIsLegalAge(location, dateOfBirth, isLegalAge);
            }
        }
        // Create hashed password and store in req.body - to unhash, call bcrypt.compareSync
        req.body.password = bcrypt.hashSync(password, 10);
        await CreateAppUser(email, req.body)
        res.status(200).json({message: `Created new AppUser with email: ${email}`})
    } catch(err) {
        res.status(400).json({error: err.toString()})
    }
};
```

BEFORE (Insert Query)

- App User table with 14 users (primary key is email).

	email [PK] character varying (256)	username character varying (32)	profilepicture bytea	fullname character varying (32)
1	woodgregory@example.net	trouble	[null]	Benjamin Schmitt
2	cgarza@example.org	how	[null]	Karen Ross
3	millerdaniel@example.com	financial	[null]	Charles Fox
4	harrislatasha@example.org	yard	[null]	Lori Butler
5	karen15@example.com	network	[null]	Kari Griffin
6	chris36@example.net	start	[null]	Brandon Gonzalez
7	rwalter@example.org	former	[null]	Jacob Mack
8	ronald84@example.org	activity	[null]	Laura Kennedy
9	hallmatthew@example.org	foot	[null]	Derrick Bridges
10	jeffreyreeves@example.org	wear	[null]	Robert Woods
11	inallchats2@gmail.com	inallchats2	[null]	Peter Parker
12	inallchats1@gmail.com	inallchats1	[null]	Bruce Wayne
13	a@gmail.com	aaa	[null]	aaa
14	sunnynie@example.com	sunnynie	[null]	Sunny Nie

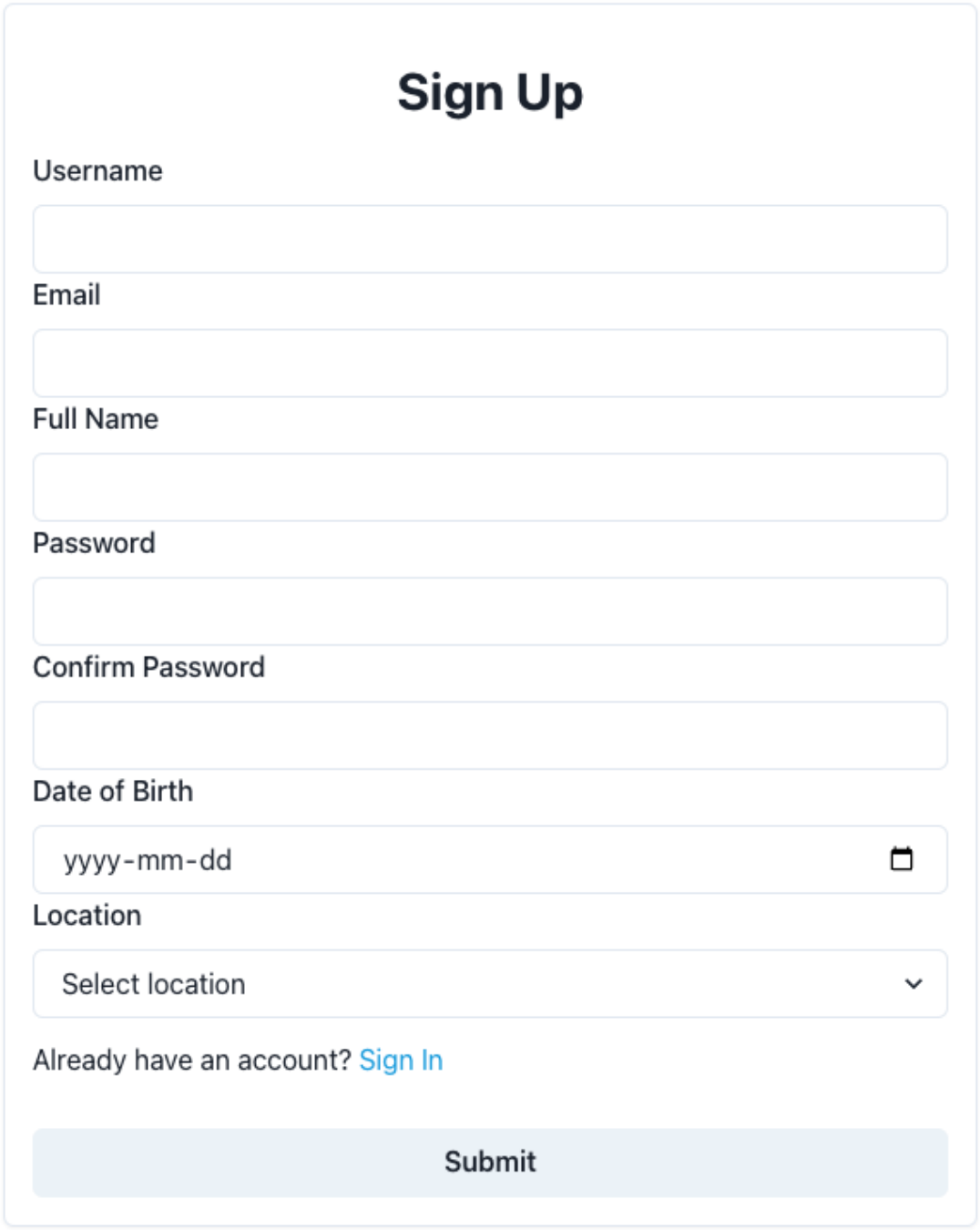
**Some attributes are cut off in the above image for the sake of readability*

- LocationDateOfBirthLegalAge table with 13 tuples (primary key is location/dateofBirth).

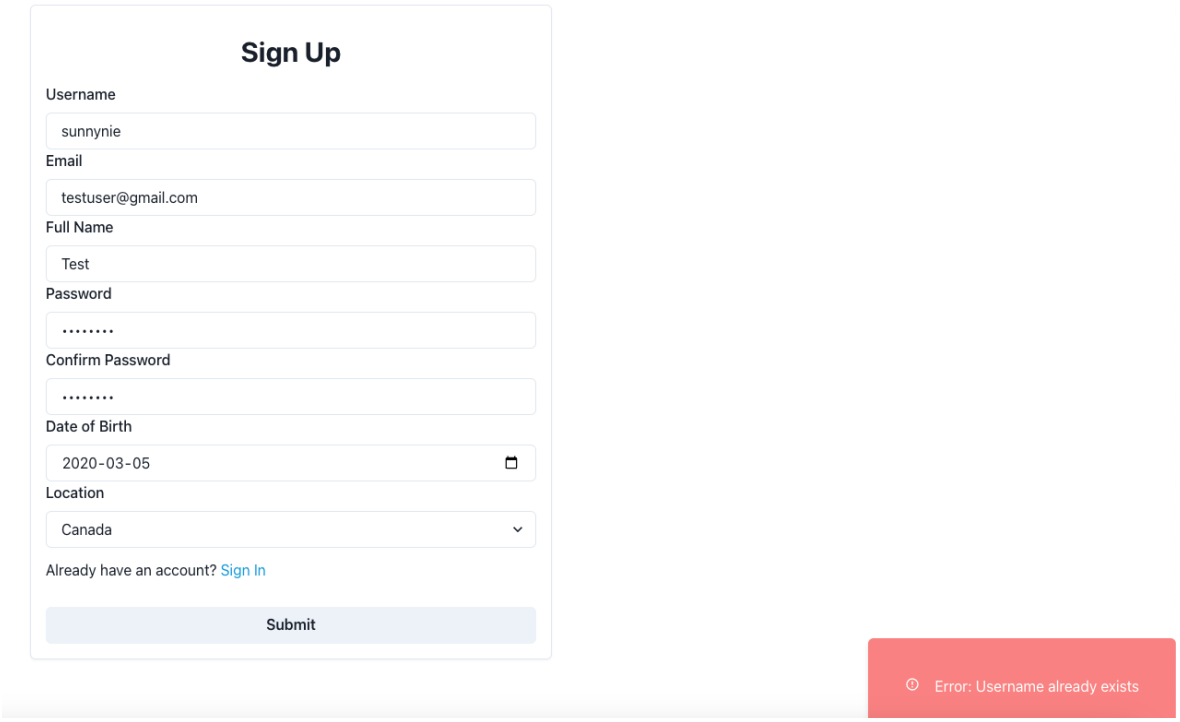
	location [PK] character varying (32)	dateofbirth [PK] date	islegalage boolean
1	Christmas Island	1988-02-28	true
2	Romania	2018-09-15	false
3	Korea	2020-12-13	false
4	Liechtenstein	2017-06-25	false
5	Eritrea	1980-06-16	true
6	Turkmenistan	1996-01-30	true
7	Northern Mariana Islands	1981-06-14	true
8	Ukraine	1970-06-28	true
9	Yemen	1974-05-01	true
10	Nicaragua	1991-03-18	true
11	Italy	2023-10-19	false
12	Nicaragua	2023-08-09	false
13	Bangladesh	2023-12-13	false

DURING (Insert Query)

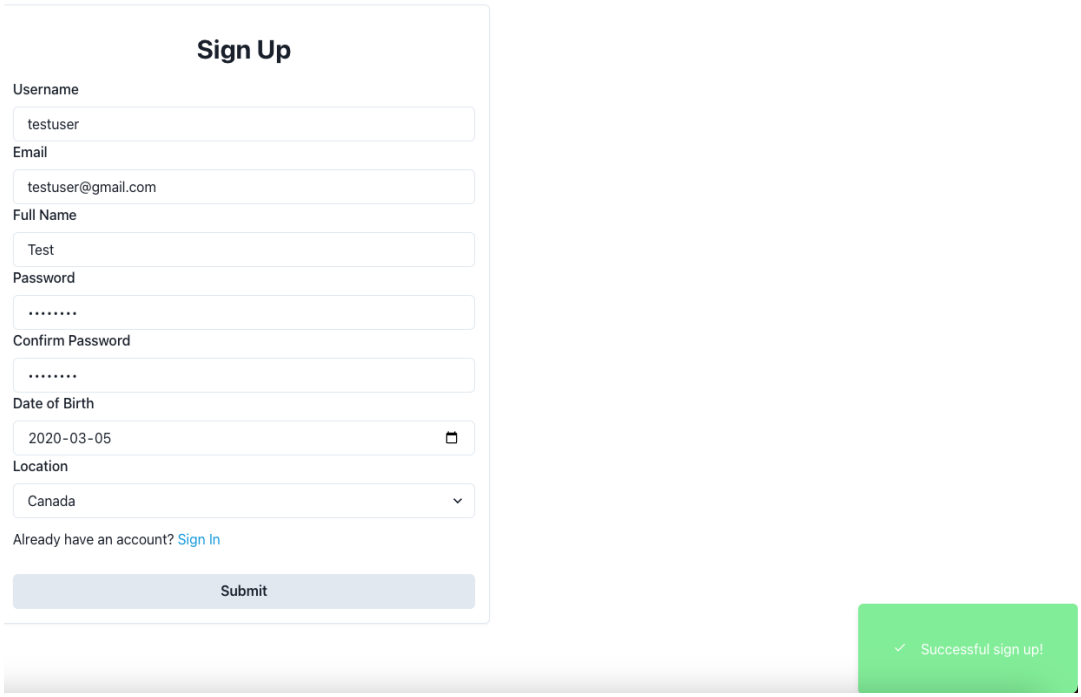
Sign up page that prompts user input for email, username, password, fullname, dateofBirth and location.



After user inputs all fields and presses submit, the backend will check for any errors (ie. missing fields, duplicate primary keys etc.) In the example below, user has entered a duplicate username (which is specified as UNIQUE in the database) and an error notification will popup:



Once a user has submitted valid information, tuple will be created and a success notification popups on user's screen (take note of the location/dateofbirth combo as this will be inserted in the LocationDateOfBirthLegalAge table which is referenced via. foreign key by the AppUser table):



AFTER (Insert Query):

AppUser table with 15 tuples (including the one we just inserted). The foreign key in this tuple connecting to the LocationDateOfBirthLegalAge table is cut off in the below image.

	email [PK] character varying (256)	username character varying (32)	profilepicture bytea	fullname character varying (32)
1	woodgregory@example.net	trouble	[null]	Benjamin Schmitt
2	cgarza@example.org	how	[null]	Karen Ross
3	millerdaniel@example.com	financial	[null]	Charles Fox
4	harrislatasha@example.org	yard	[null]	Lori Butler
5	karen15@example.com	network	[null]	Kari Griffin
6	chris36@example.net	start	[null]	Brandon Gonzalez
7	rwalter@example.org	former	[null]	Jacob Mack
8	ronald84@example.org	activity	[null]	Laura Kennedy
9	hallmatthew@example.org	foot	[null]	Derrick Bridges
10	jeffreyreeves@example.org	wear	[null]	Robert Woods
11	inallchats2@gmail.com	inallchats2	[null]	Peter Parker
12	inallchats1@gmail.com	inallchats1	[null]	Bruce Wayne
13	a@gmail.com	aaa	[null]	aaa
14	sunnynie@example.com	sunnynie	[null]	Sunny Nie
15	testuser@gmail.com	testuser	[null]	Test

LocationDateOfBirthLegalAge table with 14 tuples (including the new one we just inserted).

	location [PK] character varying (32)	dateofbirth [PK] date	islegalage boolean
1	Christmas Island	1988-02-28	true
2	Romania	2018-09-15	false
3	Korea	2020-12-13	false
4	Liechtenstein	2017-06-25	false
5	Eritrea	1980-06-16	true
6	Turkmenistan	1996-01-30	true
7	Northern Mariana Islands	1981-06-14	true
8	Ukraine	1970-06-28	true
9	Yemen	1974-05-01	true
10	Nicaragua	1991-03-18	true
11	Italy	2023-10-19	false
12	Nicaragua	2023-08-09	false
13	Bangladesh	2023-12-13	false
14	Canada	2020-03-05	false

UPDATE QUERY

Code location:

- **Path**: server/services/CommunityTable.js
- **Function name**: UpdateCommunity
- **Line numbers**: 15-29

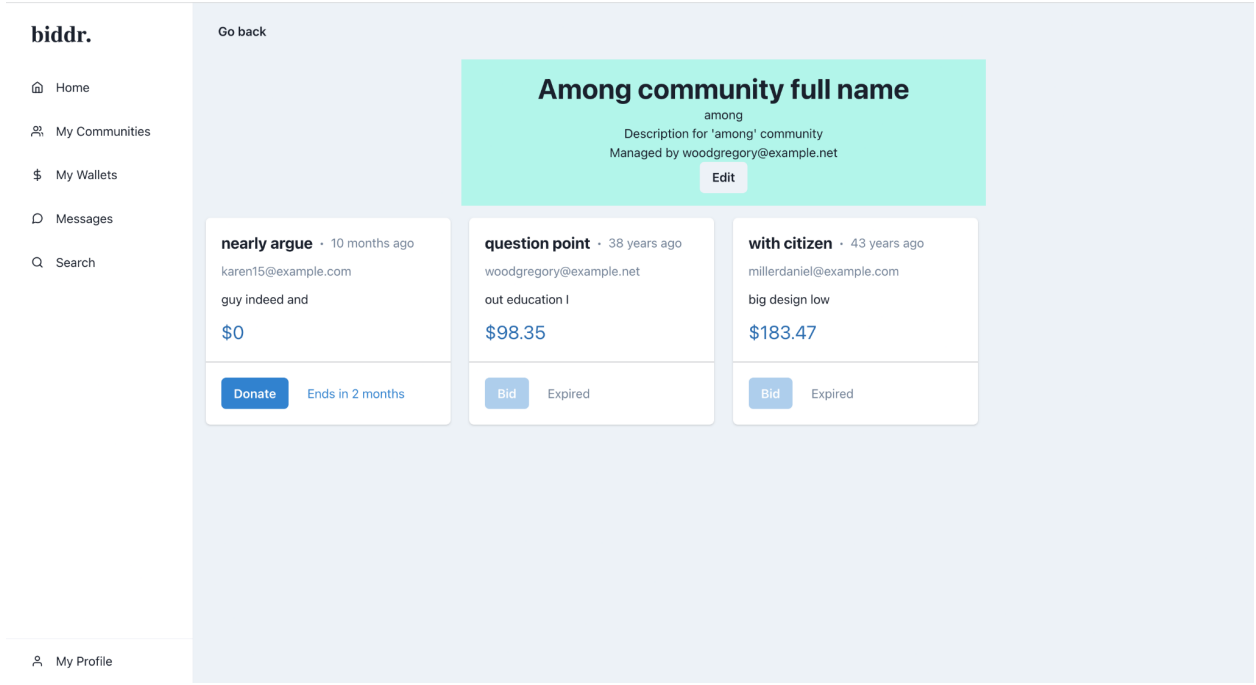
```
15  export const UpdateCommunity = async (
16    communityName,
17    email,
18    longName,
19    description
20  ) => {
21    try {
22      return await query(
23        "UPDATE Community SET email = $1, longName = $2, description = $3 WHERE communityName = $4",
24        [email, longName, description, communityName]
25      );
26    } catch (error) {
27      throw error;
28    }
29  };
```

BEFORE (Update query)

Community table for community with primary key 'among':

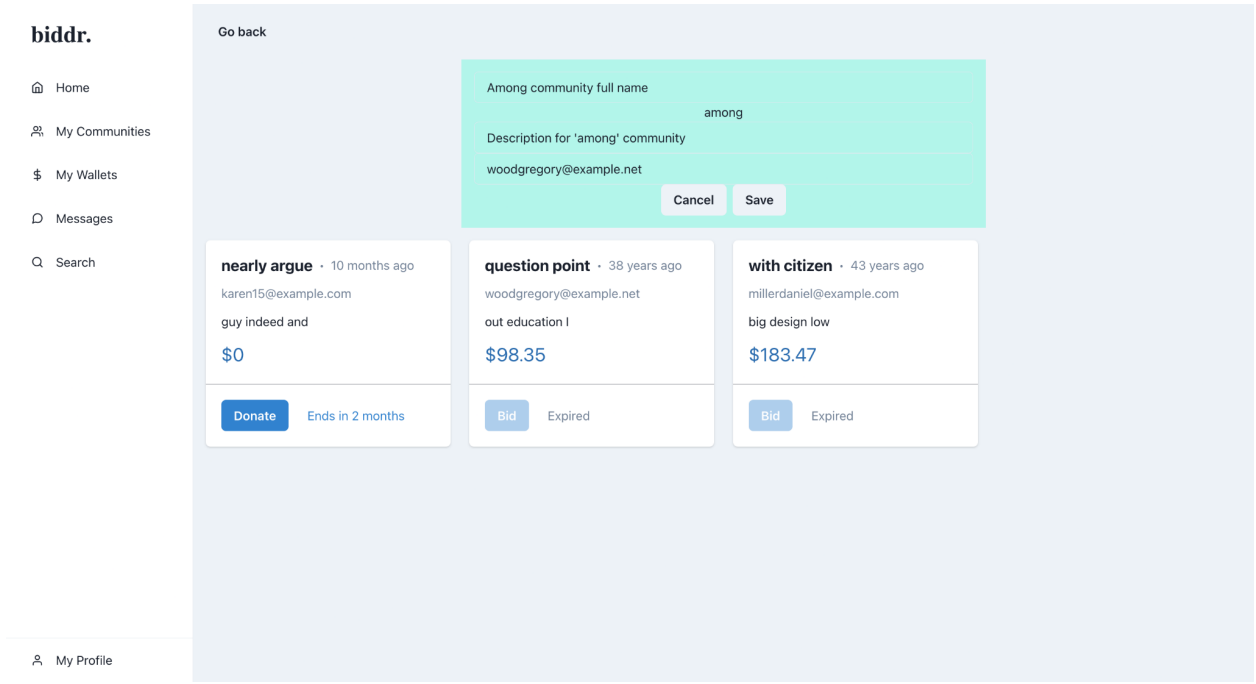
	communityname [PK] character varying (32)	email character varying (256)	longname character varying (64)	description character varying (256)
1	among	woodgregory@example.net	Among community full name	Description for 'among' community

Same community on the front-end (info all in cyan box)

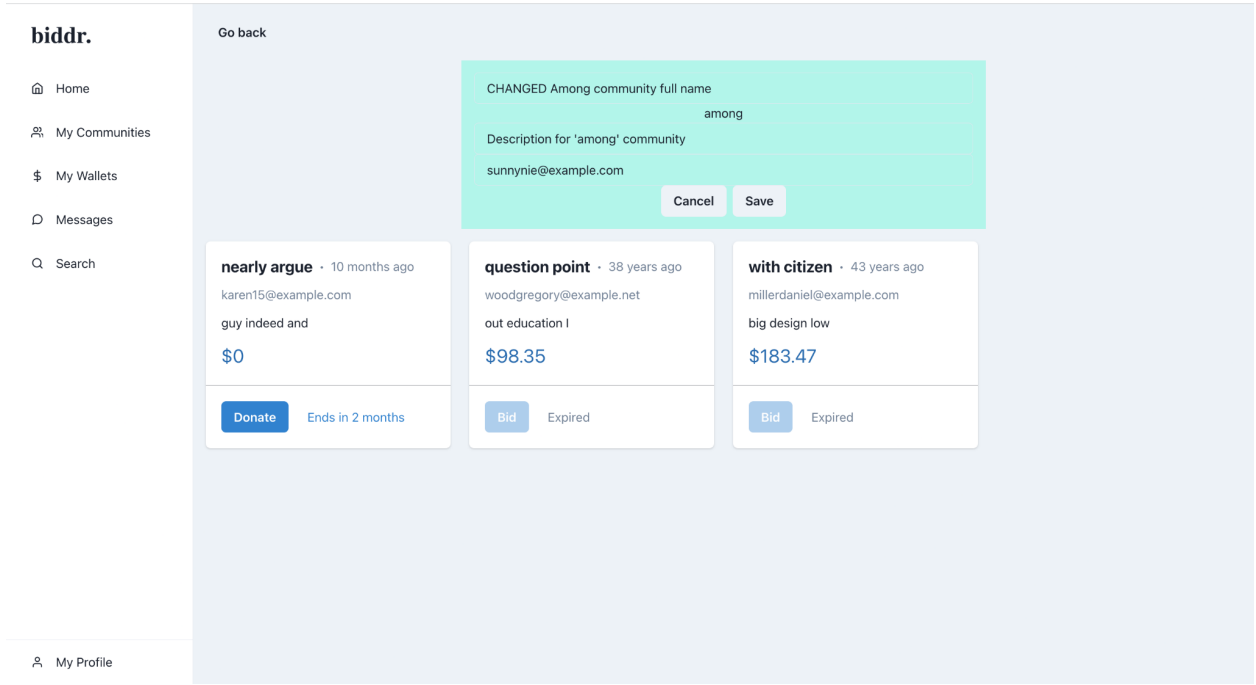


DURING (Update query)

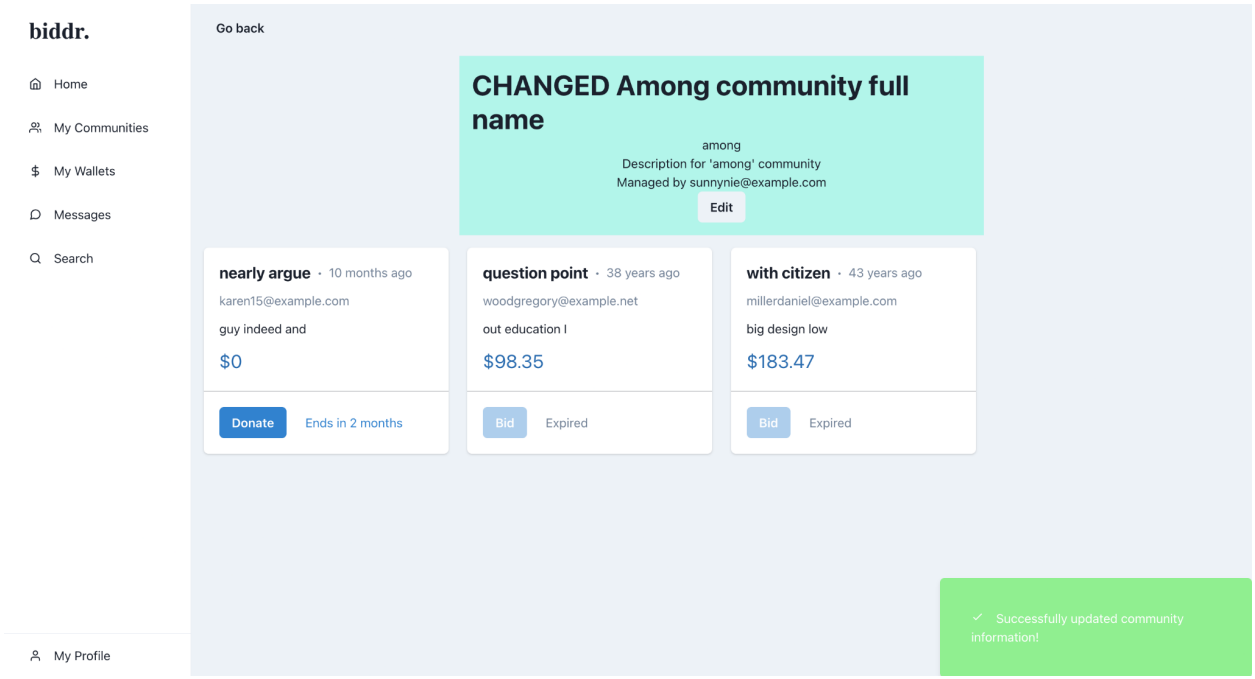
User clicks Edit button, all editable text become inputs that the user can edit



Assume user changes community full name and foreign-key email to sunnynie@example.com



User clicks save, gets success notification



AFTER (Update query)

Changed among table

	communityname [PK] character varying (32)	email character varying (256)	longname character varying (64)	description character varying (256)
1	among	sunnynie@example.com	CHANGED Among community full name	Description for 'among' community

DELETE QUERY

Code Location: server/services/ChatTable.js | Function: DeleteChatByID() | Line: 21.

I will delete the Chat with ID = 9. It will lead to a cascade of deletions in the PrivateMessage and EngagedIn tables as well.

Before:

Table - Chat:

	chatid [PK] integer	chatname character varying (32)
1	1	per truth article
2	2	day activity process
3	3	tell oil front
4	4	stock democratic federal
5	5	religious summer between
6	6	significant attorney investment
7	7	head site benefit
8	8	improve thing eight
9	9	door expert sea
10	10	white remain quality

Table - PrivateMessage:

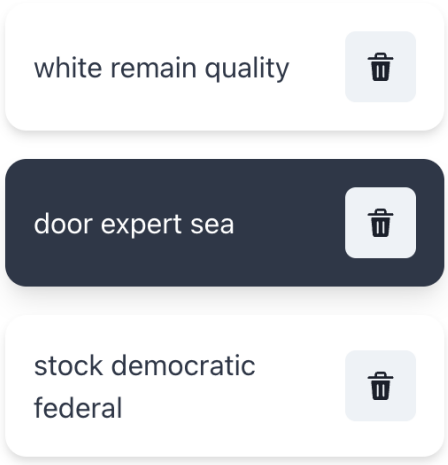
	messageid [PK] integer	email character varying (256)	chatid integer	text character varying (256)	timesent timestamp without time zone
1	1	hallmatthew@example.org	8	life popular million	1988-03-22 06:53:17
2	2	jeffreyreeves@example.org	9	despite medical safe	2003-08-31 22:31:43
3	3	woodgregory@example.net	4	expect watch right	2002-11-24 10:16:00
4	4	harrislatasha@example.org	10	foot understand little	1972-07-16 19:26:26
5	5	millerdaniel@example.com	7	authority open over	2016-10-10 05:51:15
6	6	jeffreyreeves@example.org	9	pull best chair	2013-02-23 11:30:18
7	7	jeffreyreeves@example.org	10	lose return customer	2015-04-30 20:26:52
8	8	woodgregory@example.net	4	star process election	2004-07-30 07:56:19
9	9	woodgregory@example.net	4	case book spring	1970-04-23 22:22:17
10	11	inallchats1@gmail.com	3	hello	2023-12-01 10:00:19.172164
11	12	inallchats1@gmail.com	3	hello hello hello	2023-12-01 10:00:34.717351
12	13	inallchats2@gmail.com	9	Hello	2023-12-01 17:09:52.390666
13	14	inallchats2@gmail.com	9	Deleting this leads to a cascade.	2023-12-01 17:10:00.71612

Table - EngagedIn:

	email [PK] character varying (256)	chatid [PK] integer
1	chris36@example.net	1
2	chris36@example.net	5
3	hallmatthew@example.org	8
4	harrislatasha@example.org	10
5	inallchats1@gmail.com	1
6	inallchats1@gmail.com	2
7	inallchats1@gmail.com	3
8	inallchats1@gmail.com	4
9	inallchats1@gmail.com	5
10	inallchats1@gmail.com	6
11	inallchats1@gmail.com	7
12	inallchats1@gmail.com	8
13	inallchats1@gmail.com	9
14	inallchats1@gmail.com	10
15	inallchats2@gmail.com	1
16	inallchats2@gmail.com	2
17	inallchats2@gmail.com	3
18	inallchats2@gmail.com	4
19	inallchats2@gmail.com	5
20	inallchats2@gmail.com	6
21	inallchats2@gmail.com	7
22	inallchats2@gmail.com	8
23	inallchats2@gmail.com	9
24	inallchats2@gmail.com	10
25	jeffreyreeves@example.org	9
26	jeffreyreeves@example.org	10

During:

In the Message page of the app, click on the Trash Can button next to the chat you want to delete.



After:

We see that any tuple referencing chatid = 9 in Chat, PrivateMessage, and EngagedIn has been deleted.

Table - Chat:

	chatid [PK] integer	chatname character varying (32)
1	1	per truth article
2	2	day activity process
3	3	tell oil front
4	4	stock democratic federal
5	5	religious summer between
6	6	significant attorney investment
7	7	head site benefit
8	8	improve thing eight
9	10	white remain quality

Table - PrivateMessage:

	messageid [PK] integer	email character varying (256)	chatid integer	text character varying (256)	timesent timestamp without time zone
1	1	hallmatthew@example.org	8	life popular million	1988-03-22 06:53:17
2	3	woodgregory@example.net	4	expect watch right	2002-11-24 10:16:00
3	4	harrislatasha@example.org	10	foot understand little	1972-07-16 19:26:26
4	5	millerdaniel@example.com	7	authority open over	2016-10-10 05:51:15
5	7	jeffreyreeves@example.org	10	lose return customer	2015-04-30 20:26:52
6	8	woodgregory@example.net	4	star process election	2004-07-30 07:56:19
7	9	woodgregory@example.net	4	case book spring	1970-04-23 22:22:17
8	11	inallchats1@gmail.com	3	hello	2023-12-01 10:00:19.172164
9	12	inallchats1@gmail.com	3	hello hello hello	2023-12-01 10:00:34.717351

Table - EngagedIn:

	email [PK] character varying (256)	chatid [PK] integer
1	chris36@example.net	1
2	chris36@example.net	5
3	hallmatthew@example.org	8
4	harrislatasha@example.org	10
5	inallchats1@gmail.com	1
6	inallchats1@gmail.com	2
7	inallchats1@gmail.com	3
8	inallchats1@gmail.com	4
9	inallchats1@gmail.com	5
10	inallchats1@gmail.com	6
11	inallchats1@gmail.com	7
12	inallchats1@gmail.com	8
13	inallchats1@gmail.com	10
14	inallchats2@gmail.com	1
15	inallchats2@gmail.com	2
16	inallchats2@gmail.com	3
17	inallchats2@gmail.com	4
18	inallchats2@gmail.com	5
19	inallchats2@gmail.com	6
20	inallchats2@gmail.com	7
21	inallchats2@gmail.com	8
22	inallchats2@gmail.com	10
23	jeffreyreeves@example.org	10
24	millerdaniel@example.com	4
25	millerdaniel@example.com	7
26	woodgregory@example.net	4

SELECTION QUERY

Code location:

- **Path**: server/services/CommunityTable.js
- **Function name**: FilterCommunities
- **Line numbers**: 74-95

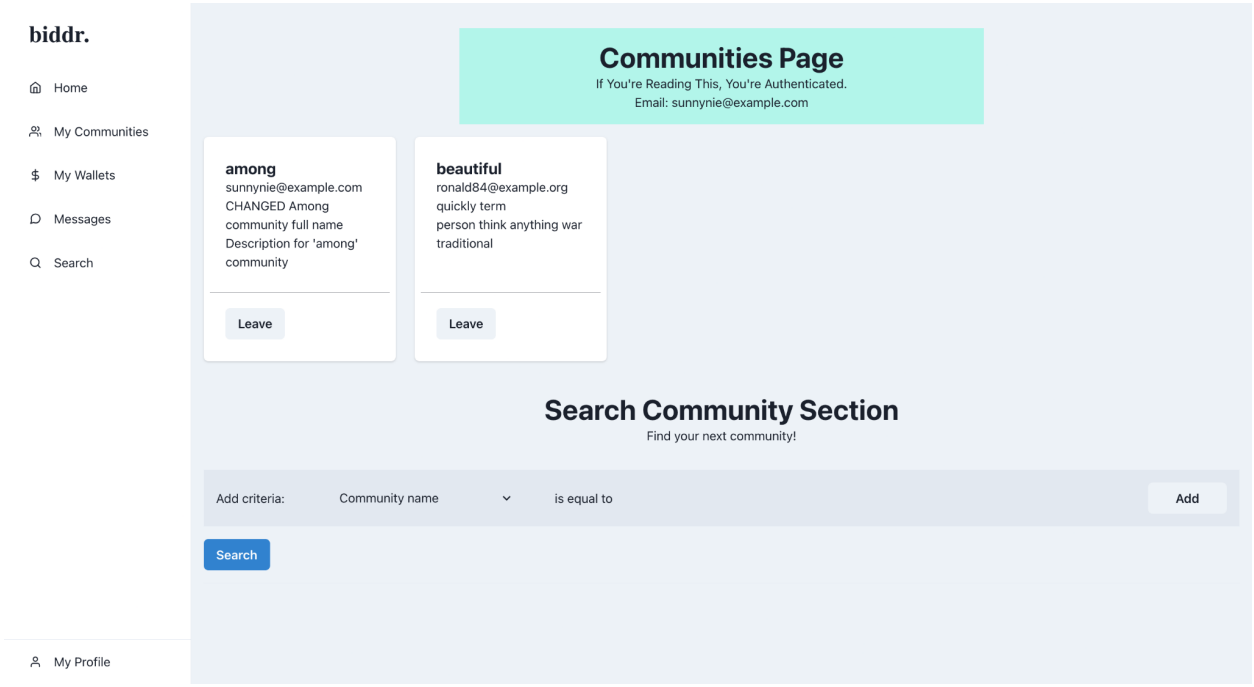
```
74  export const FilterCommunities = async (queries) => {
75    let template = `SELECT * FROM community`;
76    if (queries.length > 0) {
77      template = template.concat(" ", "WHERE");
78    }
79    queries.forEach((item, index) => {
80      if (index !== 0) {
81        template = template.concat(" ", item.type);
82      }
83      template = template.concat(" ", `${item.attribute} = $${index + 1}`);
84    });
85    try {
86      return await query(
87        template,
88        queries.map((item) => {
89          return item.value;
90        })
91      );
92    } catch (error) {
93      return error;
94    }
95  };
96
```

BEFORE (selection query)

All community tables before query (NOTE: select query does not change the tables, this is identical to the after result)

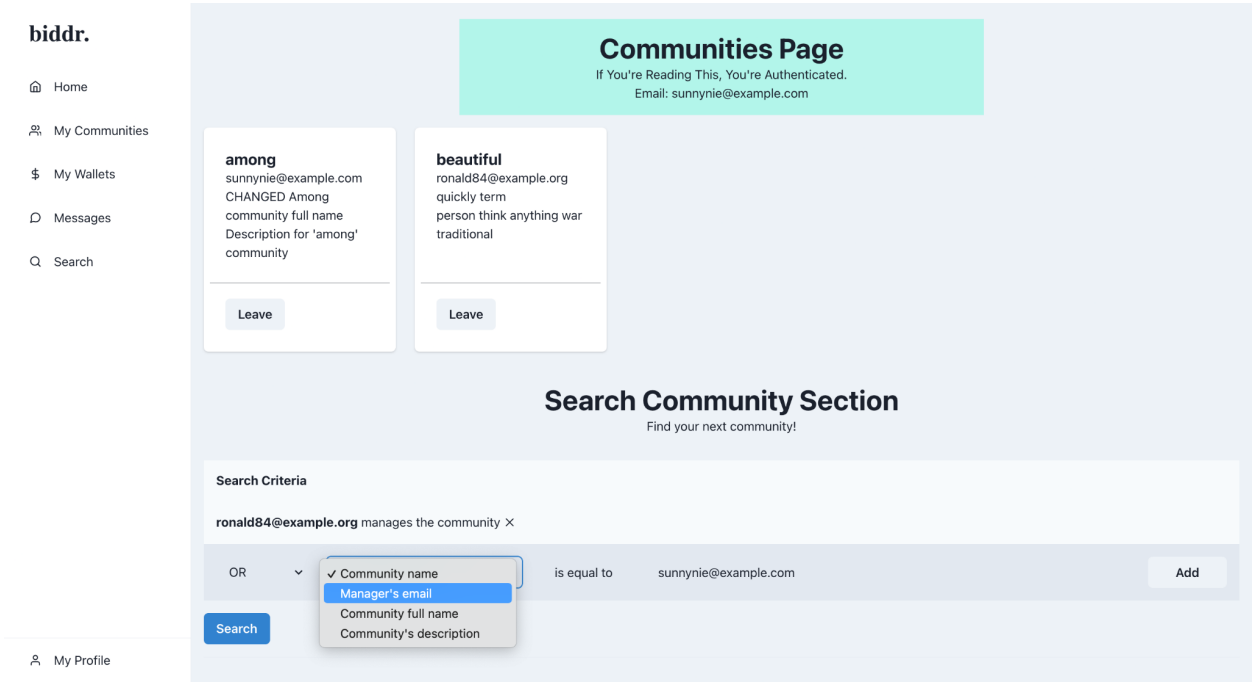
	communityname [PK] character varying (32)	email character varying (256)	longname character varying (64)	description character varying (256)
1	more	karen15@example.com	north member	may professor clear community in
2	body	hallmatthew@example.org	national onto	than former they unit draw
3	nature	jeffreyreeves@example.org	son bit	new organization not husband identify
4	benefit	jeffreyreeves@example.org	all adult	share return decade event professional
5	beautiful	ronald84@example.org	quickly term	person think anything war traditional
6	side	millerdaniel@example.com	time four	bad prepare young anything it
7	everybody	hallmatthew@example.org	hour particularly	wait poor sort economy our
8	star	ronald84@example.org	fly become	series many draw nor strong
9	among	sunnynie@example.com	CHANGED Among community full name	Description for 'among' community

The user can find the Search Community Section at the bottom of the page

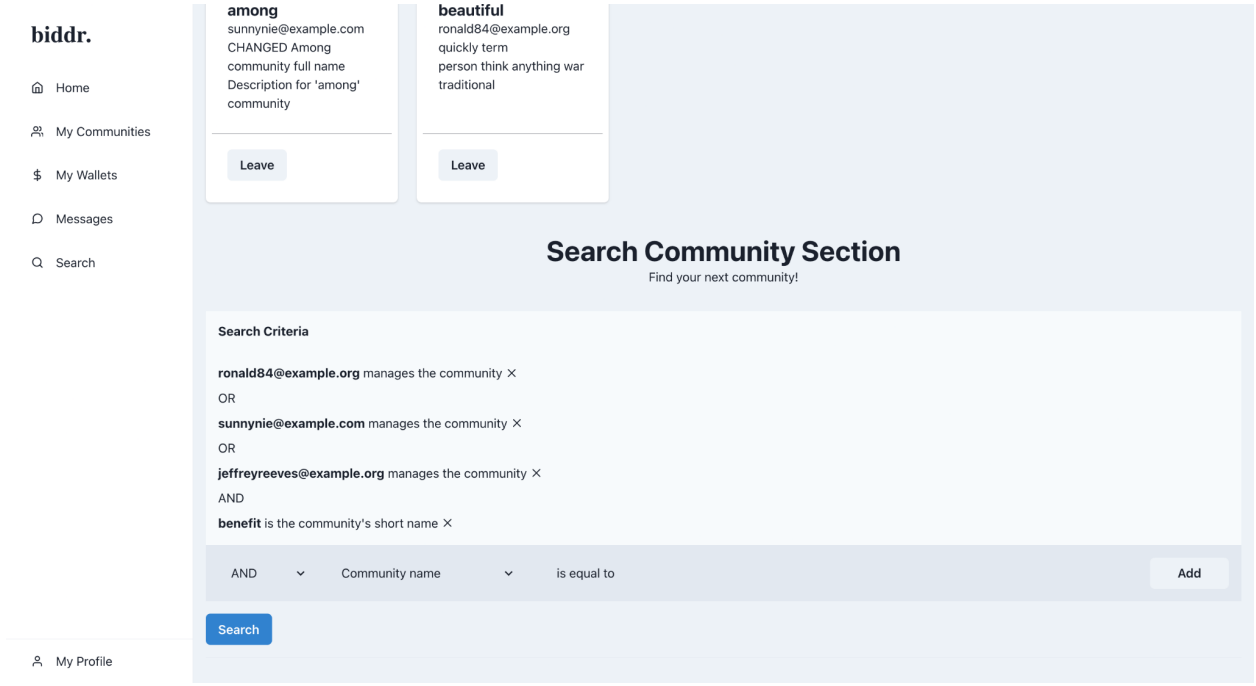


DURING (selection query)

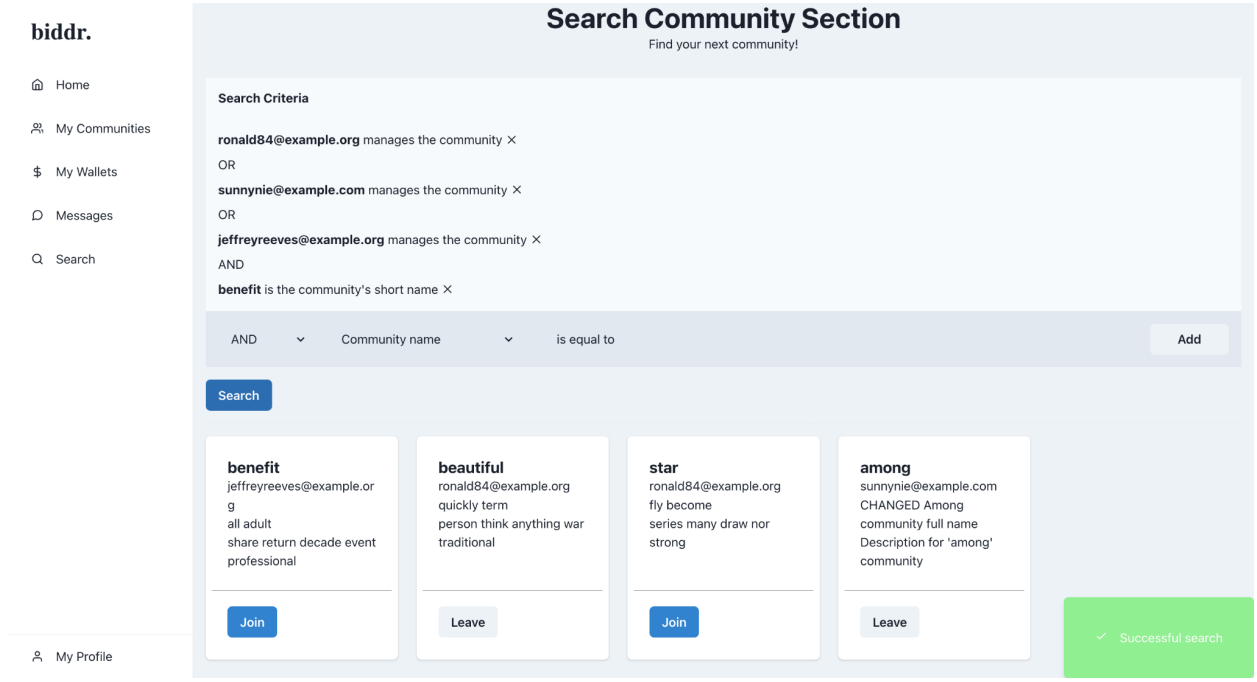
The user can access dropdowns for the type of filter (AND or OR) and attributes for Community they want to filter. The user adds the selection by pressing the Add button.



Assume this is the full filter the user builds



After the user presses search, the app returns the filtered communities with notification.



AFTER (selection query)

The community tables returned by the query

	communityname [PK] character varying (32)	email character varying (256)	longname character varying (64)	description character varying (256)
1	benefit	jeffreyreeves@example.org	all adult	share return decade event professional
2	beautiful	ronald84@example.org	quickly term	person think anything war traditional
3	star	ronald84@example.org	fly become	series many draw nor strong
4	among	sunnynie@example.com	CHANGED Among community full name	Description for 'among' community

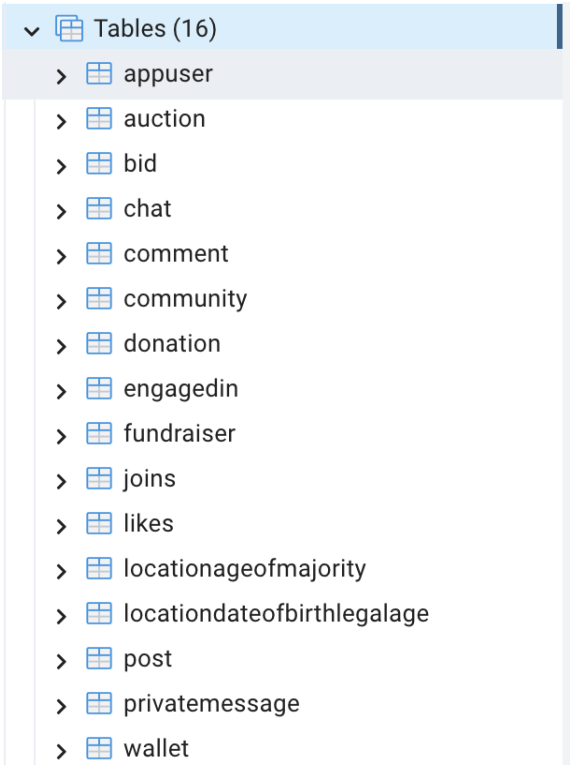
PROJECTION QUERY

Code Location: server/controllers/SearchController.js | Functions: GetAllTables(), GetAllAttributesForTables(), GetSelectAttributesForTables() | Lines: 1–44.

This is a GET query, so the data in our tables doesn't change.

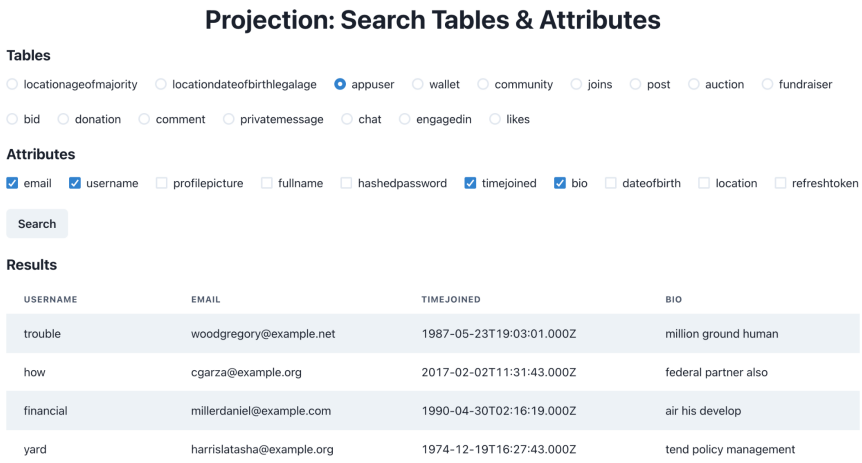
Before:

Our tables:



During:

As you can see, we can choose to view any combination of attributes from any table. The frontend pulls data dynamically, so if we add a new table, it will show up here.

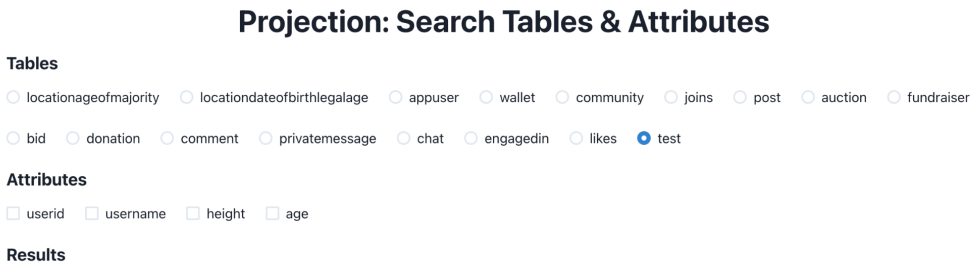


I'll create a Test table:

```sql
CREATE TABLE Test (
    userid INT PRIMARY KEY,
    username VARCHAR(100),
    height INT,
    age INT
);
```

After:

I refresh the page, and now the Test table and its attributes show up.



JOIN QUERY

This query joins the Post and Bid/Donation Tables to display posts based on their types, and further joins the Post and Joins tables to only select posts from communities that the user has joined and that the user has selected from the filter page based on community name.

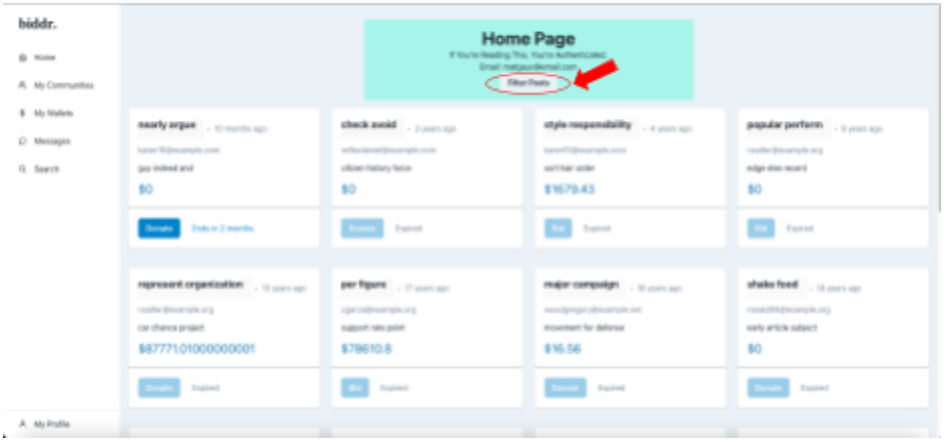
Code Location:

- **Path:** server/services/PostTable.js
- **Line Number(s):** 58-87 (QueryFilteredPostsForEmail function)

```
export const QueryFilteredPostsForEmail = async (email, communities) => {
  try {
    const communitiesString =
      "'" + communities.join("' OR communityName = '") + "'";
    const result = await query(
      `SELECT p.*,
        CASE
          WHEN EXISTS (SELECT 1 FROM auction a WHERE a.postId = p.postId) THEN 'auction'
          WHEN EXISTS (SELECT 1 FROM fundraiser f WHERE f.postId = p.postId) THEN 'fundraiser'
          ELSE 'unknown'
        END AS type,
        COALESCE(MAX(b.amount), 0) AS maxBid,
        COALESCE(SUM(d.amount), 0) AS sumDonations,
        COALESCE(COUNT(b) + COUNT(d), 0) AS totalTransactions
      FROM post p
      LEFT OUTER JOIN bid b ON p.postId = b.postId
      LEFT OUTER JOIN donation d ON p.postId = d.postId
      WHERE p.communityName IN
      (SELECT communityName
      FROM joins
      WHERE email = '${email}' AND communityName = ${communitiesString})
      GROUP BY p.postId
      ORDER BY timePosted DESC
      `
    );
    return result;
  } catch (error) {
    throw error;
  }
};
```

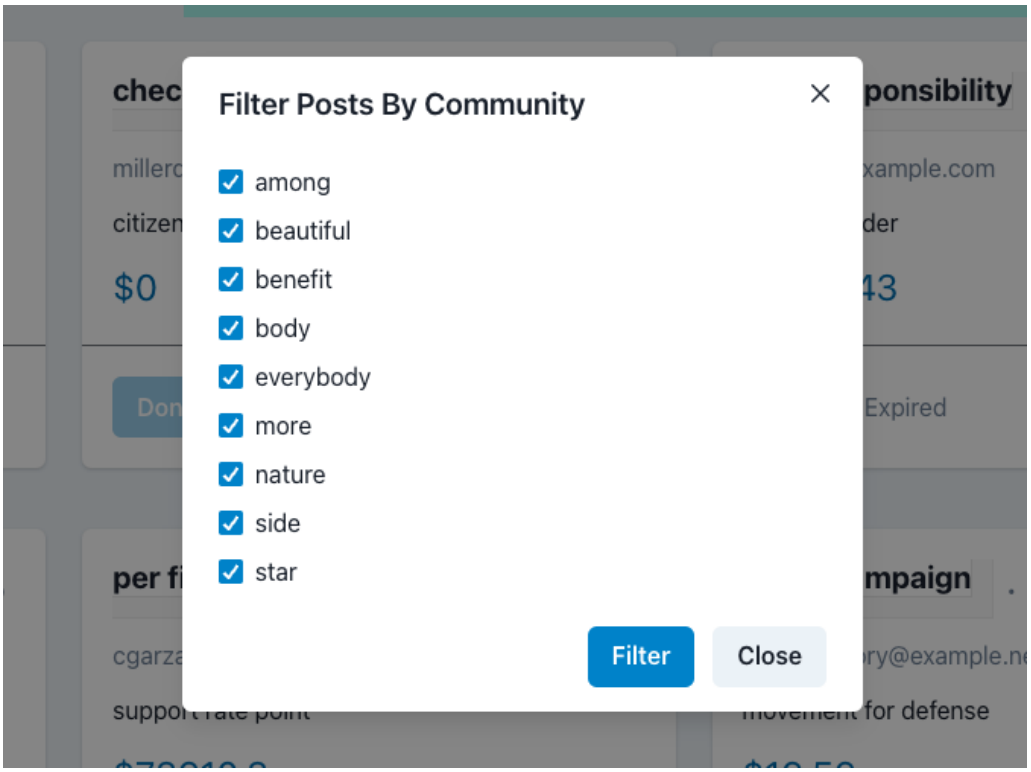
BEFORE (Join Query):

User's homepage displaying all posts in all communities that they are subscribed to. Notice the filter posts button.

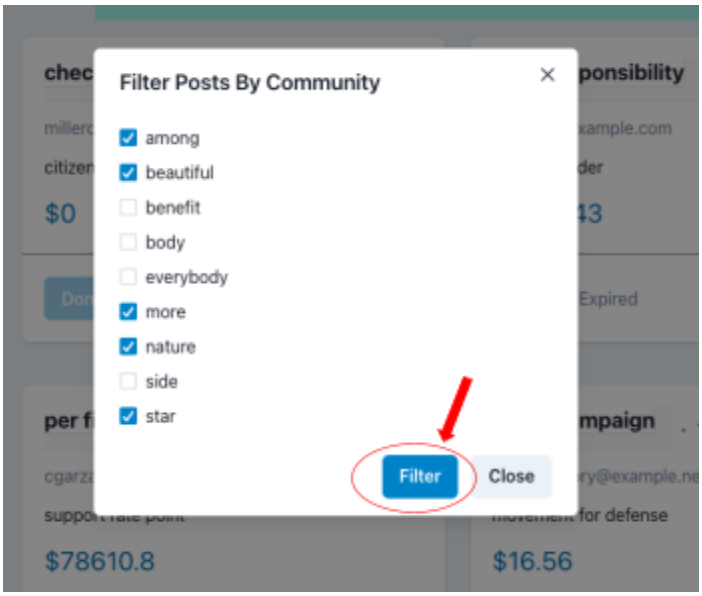


DURING (Join Query)

Upon clicking the filter posts button, users will be presented with a list of communities that they can filter posts through. All are checked by default:

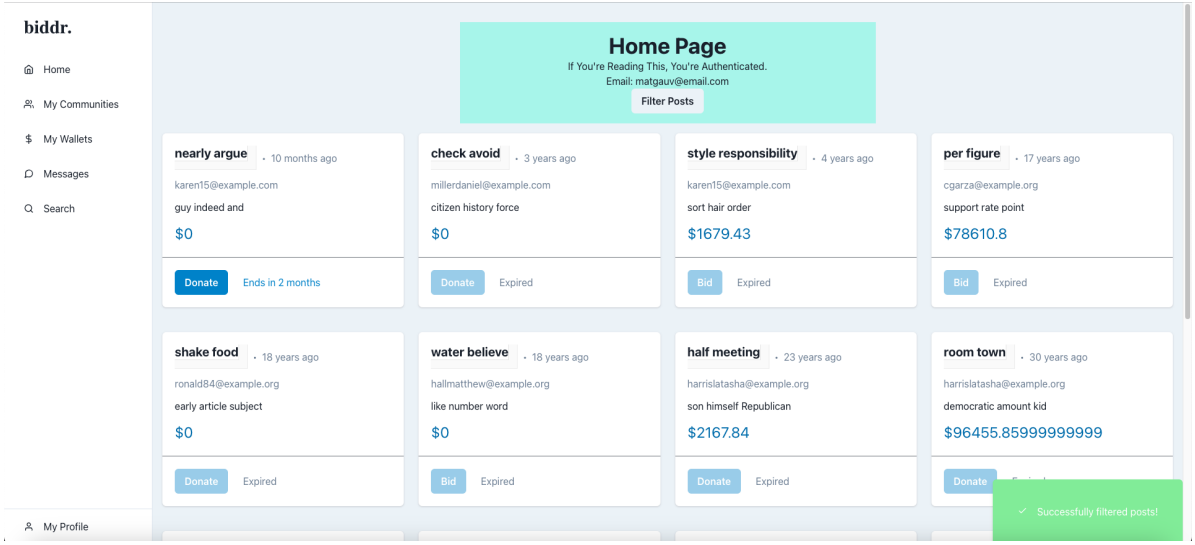


Assume that the user unchecks some random communities and presses the filter button.



AFTER (Join Query):

After filtering, only the posts from the selected communities are displayed and a success notification pops up on the users screen:



The tables returned from this query (split into two for readability):

	postid [PK] integer	postedemail character varying (256)	walletname character varying (32)	walletemail character varying (256)	communityname character varying (32)	timeposted timestamp without time zone
1	20	karen15@example.com	help involve	chris36@example.net	among	2023-01-25 02:40:39
2	16	millerdaniel@example.com	send man	cgarza@example.org	more	2020-07-01 05:43:26
3	8	karen15@example.com	late themselves	ronald84@example.org	nature	2019-05-01 16:22:50
4	7	cgarza@example.org	late themselves	ronald84@example.org	beautiful	2006-03-18 01:21:49
5	13	ronald84@example.org	send man	cgarza@example.org	more	2005-06-10 21:07:31
6	9	hallmatthew@example.org	them computer	cgarza@example.org	star	2004-12-31 02:07:11
7	19	harrislatasha@example.org	provide customer	millerdaniel@example.com	more	2000-01-24 15:23:20
8	11	harrislatasha@example.org	them computer	cgarza@example.org	star	1993-04-27 04:53:54
9	1	ronald84@example.org	provide customer	millerdaniel@example.com	beautiful	1992-06-05 23:10:28
10	17	karen15@example.com	much seven	harrislatasha@example.org	beautiful	1991-07-11 15:01:17
11	2	cgarza@example.org	send man	cgarza@example.org	more	1987-01-20 23:57:12
12	3	woodgregory@example.net	Mrs increase	chris36@example.net	among	1985-03-25 03:32:03
13	5	millerdaniel@example.com	Mrs increase	chris36@example.net	among	1979-12-22 04:03:16

expirytime timestamp without time zone	text character varying (512)	image bytea	title character varying (32)	type text	maxbid double precision	sumdonations double precision	totaltransactions bigint
2024-02-24 02:40:39	guy indeed and	[null]	nearly argue	fundraiser	0	0	0
2021-07-31 05:43:26	citizen history force	[null]	check avoid	fundraiser	0	0	0
2020-05-30 16:22:50	sort hair order	[null]	style responsibility	auction	1679.43	0	2
2007-04-17 01:21:49	support rate point	[null]	per figure	auction	78610.8	0	1
2006-07-10 21:07:31	early article subject	[null]	shake food	fundraiser	0	0	0
2006-01-30 02:07:11	like number word	[null]	water believe	auction	0	0	0
2001-02-22 15:23:20	son himself Republican	[null]	half meeting	fundraiser	0	2167.84	1
1994-05-27 04:53:54	democratic amount kid	[null]	room town	fundraiser	0	96455.85999999999	3
1993-07-05 23:10:28	expert group decide	[null]	join should	auction	362.46	0	3
1992-08-09 15:01:17	add success home	[null]	guy leg	fundraiser	0	4.95	1
1988-02-19 23:57:12	thus should now	[null]	physical case	auction	0	0	0
1986-04-24 03:32:03	out education I	[null]	question point	auction	98.35	0	1
1981-01-20 04:03:16	big design low	[null]	with citizen	auction	183.47	0	1

AGGREGATION WITH GROUP BY

This query selects all posts belonging to a given community. To get the post type (auction or fundraiser), max bid amount, sum of donations, and total transactions for each post, it LEFT OUTER JOINs on bid and donation, then it runs the nested aggregation queries, and finally it GROUP BY the postId. Lastly, it orders the newest posts first.

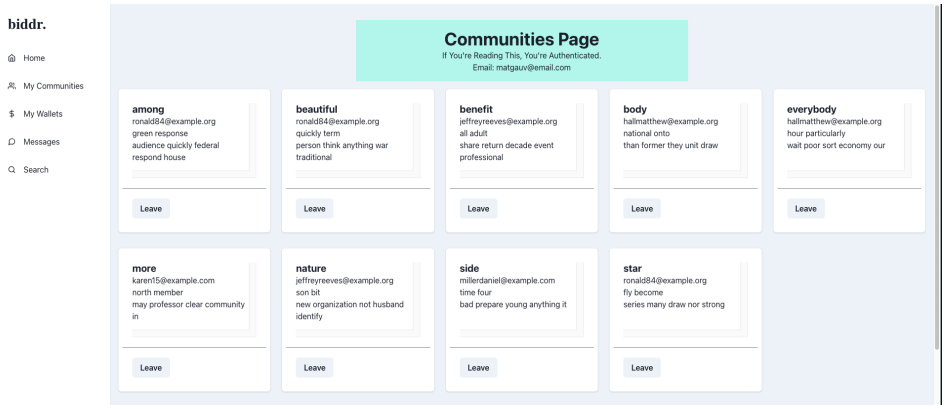
Code Location:

- **Path:** server/services/PostTable.js
- **Line Number(s):** 89-114 (QueryPostsInCommunity function)

```
export const QueryPostsInCommunity = async (name) => {
  try {
    const result = await query(
      `SELECT p.*,
        CASE
          WHEN EXISTS (SELECT 1 FROM auction a WHERE a.postId = p.postId) THEN 'auction'
          WHEN EXISTS (SELECT 1 FROM fundraiser f WHERE f.postId = p.postId) THEN 'fundraiser'
          ELSE 'unknown'
        END AS type,
        COALESCE(MAX(b.amount), 0) AS maxBid,
        COALESCE(SUM(d.amount), 0) AS sumDonations,
        COALESCE(COUNT(b) + COUNT(d), 0) AS totalTransactions
      FROM post p
      LEFT OUTER JOIN bid b ON p.postId = b.postId
      LEFT OUTER JOIN donation d ON p.postId = d.postId
      WHERE communityName = $1
      GROUP BY p.postId
      ORDER BY timePosted DESC
      `,
      [name]
    );
    return result;
  } catch (error) {
    throw error;
  }
};
```

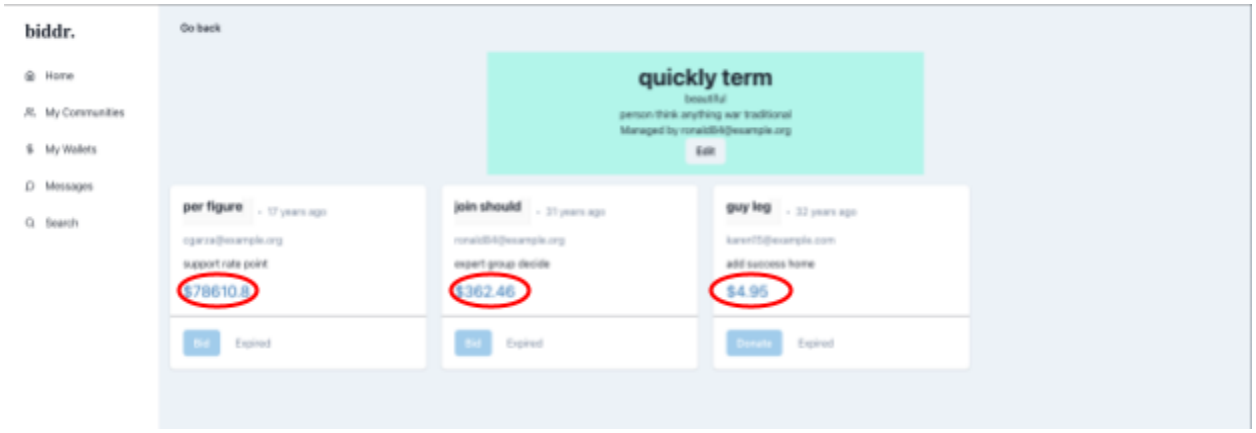
BEFORE (Aggregation w/ GROUP BY Query):

The communities page lists all of the user's joined communities (in this case, the user has joined several communities).



DURING (Aggregation w/ GROUP BY Query)

Upon clicking one of the communities (in this case the community “beautiful”), the community page renders all of the posts within that community and displays their information, including the post-specific aggregated values (ie. the max bid or total donations - circled in red).



AFTER (Aggregation w/ GROUP BY Query)

The table returned from the query when clicking on the “beautiful” community (split into two for readability):

	postid [PK] integer	postedemail character varying (256)	walletname character varying (32)	walletemail character varying (256)	communityname character varying (32)	timeposted timestamp without time zone
1	7	cgarza@example.org	late themselves	ronald84@example.org	beautiful	2006-03-18 01:21:49
2	1	ronald84@example.org	provide customer	millerdaniel@example.com	beautiful	1992-06-05 23:10:28
3	17	karen15@example.com	much seven	harrislatasha@example.org	beautiful	1991-07-11 15:01:17

expirytime timestamp without time zone	text character varying (512)	image bytea	title character varying (32)	type text	maxbid double precision	sumdonations double precision	totaltransactions bigint
2007-04-17 01:21:49	support rate point	[null]	per figure	auction	78610.8	0	1
1993-07-05 23:10:28	expert group decide	[null]	join should	auction	362.46	0	3
1992-08-09 15:01:17	add success home	[null]	guy leg	fundraiser	0	4.95	1

AGGREGATION WITH HAVING

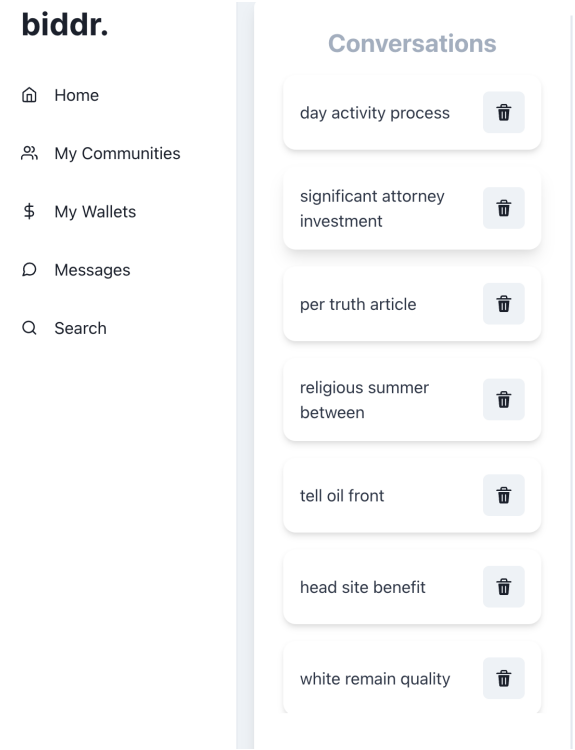
Code Location: server/services/ChatTable.js | Functions: GetAllChatsForUserSorted() | Lines: 68–74.

This query uses HAVING to get all the Chats that a User belongs in, sorted by the most recently sent message in each chat.

```
const response = await query(
    `SELECT chatid, chatname
     FROM chat NATURAL LEFT OUTER JOIN privatemessage
     GROUP BY chatid
     HAVING chatid IN (SELECT chatid
                       FROM chat NATURAL JOIN engagedin
                       WHERE email = $1)
     ORDER BY COALESCE(MAX(timesent), '1900-01-01') DESC;`,
    [email]
);
```

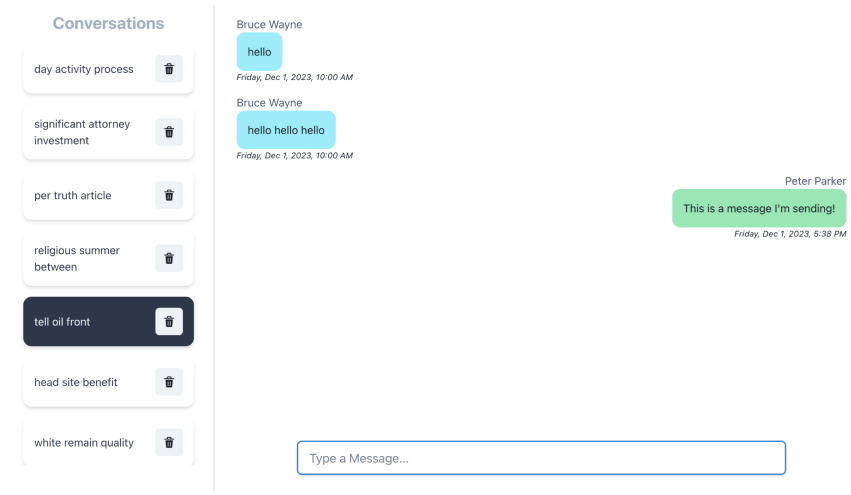
Before:

The User executes this query by clicking on the “Messages” tab, to view all the Chats that they belong in.



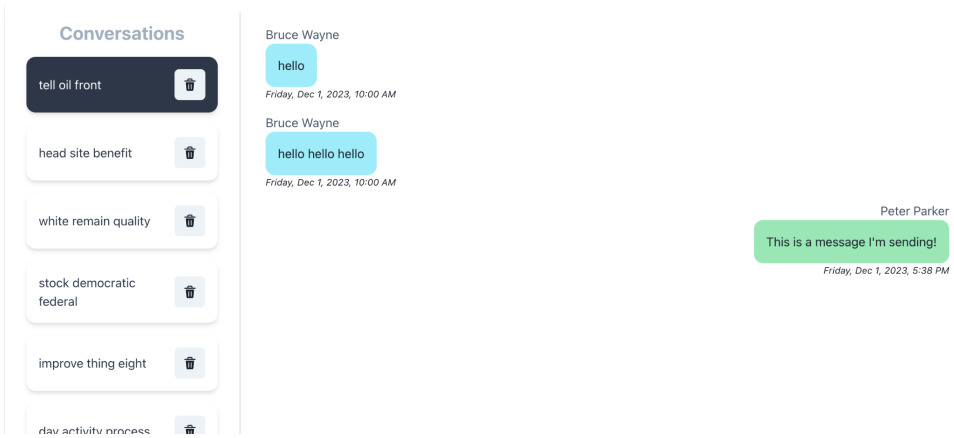
During:

Now, suppose that I send a message in the Chat named “tell oil front”.



After:

I refresh the page, and now “tell oil front” is at the top of the Chat list.



NESTED AGGREGATION WITH GROUP BY QUERY

This query selects all posts belonging to communities the user has joined. It gets the post type (auction or fundraiser), max bid amount, sum of donations, and total transactions for each post by doing LEFT OUTER JOINs on bid and donation, and then running the nested aggregation queries, and grouping by postId. In the WHERE clause, the query filters posts that belong in communities the user has joined. At the end, it orders the newest posts first.

Code location:

- **Path**: server/services/PostTable.js
- **Function name**: QueryHomepagePostsForEmail
- **Line numbers**: 28-56

```
28  export const QueryHomepagePostsForEmail = async (email) => {
29    try {
30      const result = await query(
31        `SELECT p.*,
32          CASE
33            WHEN EXISTS (SELECT 1 FROM auction a WHERE a.postId = p.postId) THEN 'auction'
34            WHEN EXISTS (SELECT 1 FROM fundraiser f WHERE f.postId = p.postId) THEN 'fundraiser'
35            ELSE 'unknown'
36          END AS type,
37          COALESCE(MAX(b.amount), 0) AS maxBid,
38          COALESCE(SUM(d.amount), 0) AS sumDonations,
39          COALESCE(COUNT(b) + COUNT(d), 0) AS totalTransactions
40        FROM post p
41        LEFT OUTER JOIN bid b ON p.postId = b.postId
42        LEFT OUTER JOIN donation d ON p.postId = d.postId
43        WHERE p.communityName IN
44        (SELECT DISTINCT c.communityName
45        FROM joins c
46        WHERE c.email = $1)
47        GROUP BY p.postId
48        ORDER BY timePosted DESC
49        `,
50        [email]
51      );
52      return result;
53    } catch (error) {
54      throw error;
55    }
56  };
```

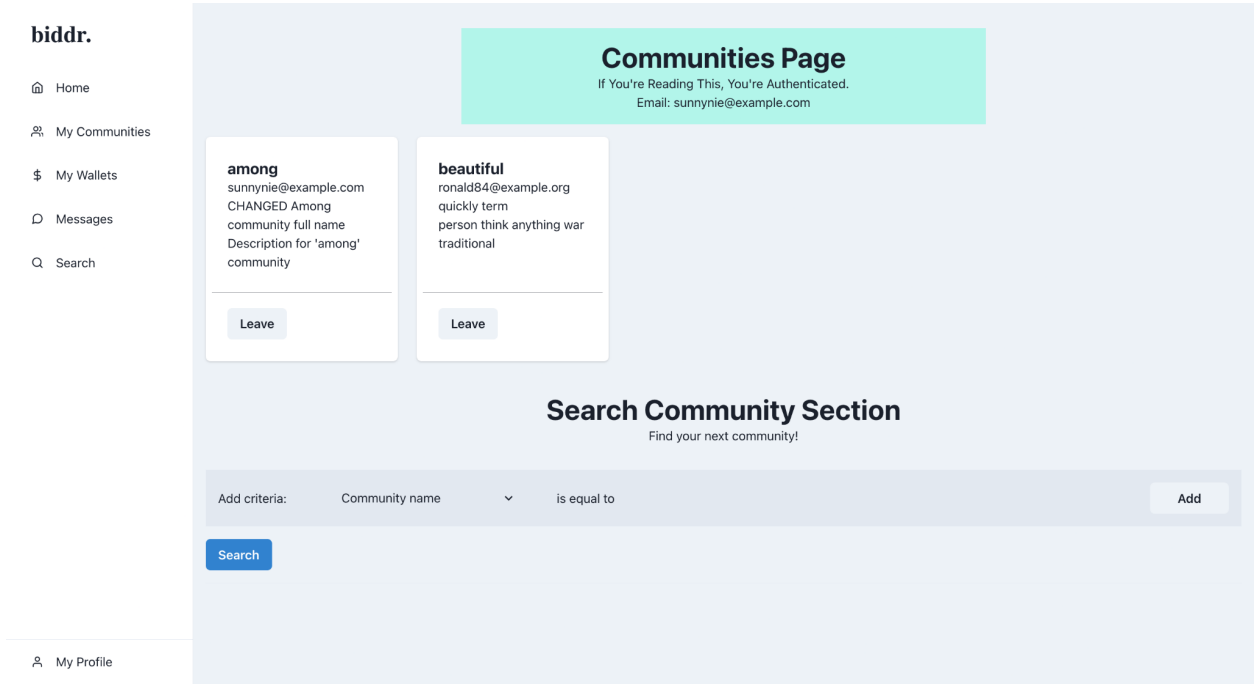
BEFORE (nested aggregation with group by query)

Note: This query does not change any tables. I will attach the 'returned' tables at the end in the 'After' section. Here is the posts tables.

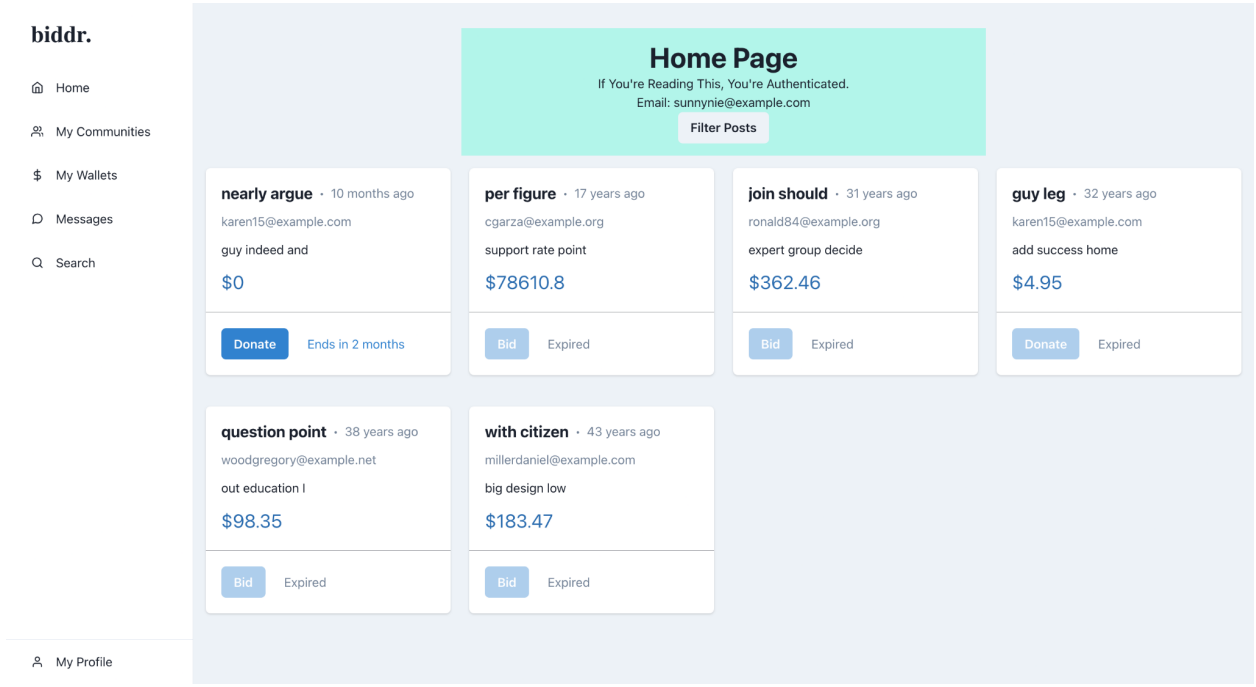
	postid [PK] integer	postedemail character varying (256)	walletname character varying (32)	walletemail character varying (256)	communityname character varying (32)	timeposted timestamp without time zone	expirytime timestamp without time zone	text character varying (512)	image bytea	title character varying (32)
1	1	ronald84@example.org	provide customer	millerdaniel@example.com	beautiful	1992-06-05 23:10:28	1993-07-05 23:10:28	expert group decide	[null]	join should
2	2	cgarza@example.org	send man	cgarza@example.org	more	1987-01-20 23:57:12	1988-02-19 23:57:12	thus should now	[null]	physical case
3	3	woodgregory@example.net	Mrs increase	chris36@example.net	among	1985-03-25 03:32:03	1986-04-24 03:32:03	out education I	[null]	question point
4	4	rwalter@example.org	send man	cgarza@example.org	side	2013-12-04 05:55:04	2015-01-03 05:55:04	edge else recent	[null]	popular perform
5	5	millerdaniel@example.com	Mrs increase	chris36@example.net	among	1979-12-22 04:03:16	1981-01-20 04:03:16	big design low	[null]	with citizen
6	6	harrislatasha@example.org	provide customer	millerdaniel@example.com	side	1998-02-06 02:20:55	1999-03-08 02:20:55	single left determine	[null]	new method
7	7	cgarza@example.org	late themselves	ronald84@example.org	beautiful	2006-03-18 01:21:49	2007-04-17 01:21:49	support rate point	[null]	per figure
8	8	karen15@example.com	late themselves	ronald84@example.org	nature	2019-05-01 16:22:50	2020-05-30 16:22:50	sort hair order	[null]	style responsibility
9	9	hallmatthew@example.org	them computer	cgarza@example.org	star	2004-12-31 02:07:11	2006-01-30 02:07:11	like number word	[null]	water believe
10	10	cgarza@example.org	send man	cgarza@example.org	body	1975-10-09 06:12:48	1976-11-07 06:12:48	notice marriage not	[null]	thus kid
11	11	harrislatasha@example.org	them computer	cgarza@example.org	star	1993-04-27 04:53:54	1994-05-27 04:53:54	democratic amount kid	[null]	room town
12	12	jeffreyreeves@example.org	them computer	cgarza@example.org	body	1976-01-20 14:43:52	1977-02-18 14:43:52	party third teacher	[null]	tonight word
13	13	ronald84@example.org	send man	cgarza@example.org	more	2005-06-10 21:07:31	2006-07-10 21:07:31	early article subject	[null]	shake food
14	14	rwalter@example.org	help involve	chris36@example.net	benefit	2010-06-09 21:03:07	2011-07-09 21:03:07	car chance project	[null]	represent organization
15	15	woodgregory@example.net	late themselves	ronald84@example.org	body	2005-09-11 15:18:11	2006-10-11 15:18:11	movement for defense	[null]	major campaign
16	16	millerdaniel@example.com	send man	cgarza@example.org	more	2020-07-01 05:43:26	2021-07-31 05:43:26	citizen history force	[null]	check avoid
17	17	karen15@example.com	much seven	harrislatasha@example.org	beautiful	1991-07-11 15:01:17	1992-08-09 15:01:17	add success home	[null]	guy leg
18	18	rwalter@example.org	official raise	cgarza@example.org	everybody	2000-07-08 11:27:35	2001-08-07 11:27:35	environment weight think	[null]	material week

DURING (nested aggregation with group by query)

First, the user selects communities whose posts they want to see by joining communities. In this example, the user joined 2 communities: among, and beautiful.



Next, the user clicks the Home button on the left side-bar to execute the query to populate the homepage with all posts from communities they've joined, with nested subqueries to calculate the total amount donated/max bid each post has received so far.



AFTER (nested aggregation with group by query)

Here is the returned query (cutoff so there's two tables)

	postid [PK] integer	postedemail character varying (256)	walletname character varying (32)	walletemail character varying (256)	communityname character varying (32)	timeposted timestamp without time zone	expirytime timestamp without time zone	text character varying (512)	image bytea	title character varying (32)	type text
1	20	karen15@example.com	help involve	chris36@example.net	among	2023-01-25 02:40:39	2024-02-24 02:40:39	guy indeed and	[null]	nearly argue	fundraiser
2	7	cgarza@example.org	late themselves	ronald84@example.org	beautiful	2006-03-18 01:21:49	2007-04-17 01:21:49	support rate point	[null]	per figure	auction
3	1	ronald84@example.org	provide customer	millerdaniel@example.com	beautiful	1992-06-05 23:10:28	1993-07-05 23:10:28	expert group decide	[null]	join should	auction
4	17	karen15@example.com	much seven	harrislatasha@example.org	beautiful	1991-07-11 15:01:17	1992-08-09 15:01:17	add success home	[null]	guy leg	fundraiser
5	3	woodgregory@example.net	Mrs increase	chris36@example.net	among	1985-03-25 03:32:03	1986-04-24 03:32:03	out education l	[null]	question point	auction
6	5	millerdaniel@example.com	Mrs increase	chris36@example.net	among	1979-12-22 04:03:16	1981-01-20 04:03:16	big design low	[null]	with citizen	auction

	)	walletemail character varying (256)	communityname character varying (32)	timeposted timestamp without time zone	expirytime timestamp without time zone	text character varying (512)	image bytea	title character varying (32)	type text	maxbid double precision	sumdonations double precision	totaltransactions bigint
1		chris36@example.net	among	2023-01-25 02:40:39	2024-02-24 02:40:39	guy indeed and	[null]	nearly argue	fundraiser	0	0	0
2		ronald84@example.org	beautiful	2006-03-18 01:21:49	2007-04-17 01:21:49	support rate point	[null]	per figure	auction	78610.8	0	1
3		millerdaniel@example.com	beautiful	1992-06-05 23:10:28	1993-07-05 23:10:28	expert group decide	[null]	join should	auction	362.46	0	3
4		harrislatasha@example.org	beautiful	1991-07-11 15:01:17	1992-08-09 15:01:17	add success home	[null]	guy leg	fundraiser	0	4.95	1
5		chris36@example.net	among	1985-03-25 03:32:03	1986-04-24 03:32:03	out education l	[null]	question point	auction	98.35	0	1
6		chris36@example.net	among	1979-12-22 04:03:16	1981-01-20 04:03:16	big design low	[null]	with citizen	auction	183.47	0	1

DIVISION QUERY

Code Location: server/controllers/SearchController.js | Functions: GetUsersInAllChats() | Lines: 49-55.

For this query, I find a list of users who are in every Chat.

```
const response = await query(`SELECT email, fullname
                              FROM appuser
                              WHERE NOT EXISTS (SELECT chatid
                                                FROM chat
                                                EXCEPT (SELECT chatid
                                                        FROM engagedin
                                                        WHERE engagedin.email = appuser.email))`);
```

Before:

To begin, there are 10 chats in total and users inallchats1@gmail.com and inallchats2@gmail.com are both in every chat.

5	inallchats1@gmail.com	1
6	inallchats1@gmail.com	2
7	inallchats1@gmail.com	3
8	inallchats1@gmail.com	4
9	inallchats1@gmail.com	5
10	inallchats1@gmail.com	6
11	inallchats1@gmail.com	7
12	inallchats1@gmail.com	8
13	inallchats1@gmail.com	9
14	inallchats1@gmail.com	10
15	inallchats2@gmail.com	1
16	inallchats2@gmail.com	2
17	inallchats2@gmail.com	3
18	inallchats2@gmail.com	4
19	inallchats2@gmail.com	5
20	inallchats2@gmail.com	6
21	inallchats2@gmail.com	7
22	inallchats2@gmail.com	8
23	inallchats2@gmail.com	9
24	inallchats2@gmail.com	10

During:

This, the division query returns both their names in the result.

Division: Find Users in All Group Chats

Search

Full Name	Email
Peter Parker	inallchats2@gmail.com
Bruce Wayne	inallchats1@gmail.com

Now, suppose I remove inallchats2@gmail.com from chatID = 9:

21	inallchats2@gmail.com	7
22	inallchats2@gmail.com	8
~~23~~	~~inallchats2@gmail.com~~	~~9~~
24	inallchats2@gmail.com	10

After:

I call the Division query again, and as you would expect, inallchats2@gmail.com is no longer one of the users that are in every chat:

Division: Find Users in All Group Chats

Search

Full Name	Email
Bruce Wayne	inallchats1@gmail.com