Bulletin Board: Announceit

Yosiris Andújar
Christian Lorenzo
Armando Ortiz
Yosiris.andujar@upr.edu, Christian.lorenzo1@upr.edu, Armando.ortiz1@upr.edu

1. Introduction

College campus have a lot of life going on and many opportunities to take. People and organizations want to announce their events, used books, tutoring, housing and many other things. They mostly use physical bulletin boards to reach the campus community. This require much effort such as manual labor, printing papers, time, limited space and accessibility.

The target clients are students, professors, and faculty members in the campus of UPRM. This application will be both a mobile and web-app because it will reach a broader clientele without difficulty.

The technologies that will be used are HTML5, CSS3, Javascript, AngularJs, Ionic, and PostgreSQL. This will result in a consistent development with both mobile and desktop applications.

2. Client App Description

1. **Home page**: The app will show all announcements regardless of not having a user account.
2. **Searching**: Any user can search any post by keyword.
3. **Sign-up**: Users will register giving their email and personal information.
4. **Creating an announcement**: A registered user will be able to post his announcement. The user must choose a category of the announcement by books, tutoring, events, housing, or others. The description of the announcement will be in a textbox.

Note: Registered users can only three announcements in the home page. Premium users can have unlimited announcements in the home page.

1. **Messaging**: Registered user can privately message with the owner of the announcement.
1. **Payment**: Registered user can pay the owner of the announcement according to their own arrangements.
1. **Settings**: The registered user can edit their attributes.
1. **Profile**: The registered user can see their history of payments, messages, and posts.
2. **Reports**: The registered user can report a post if the content is inadequate.

Technologies. HTML5, CSS3, Javascript, AngularJs, and Ionic

3. Server side description

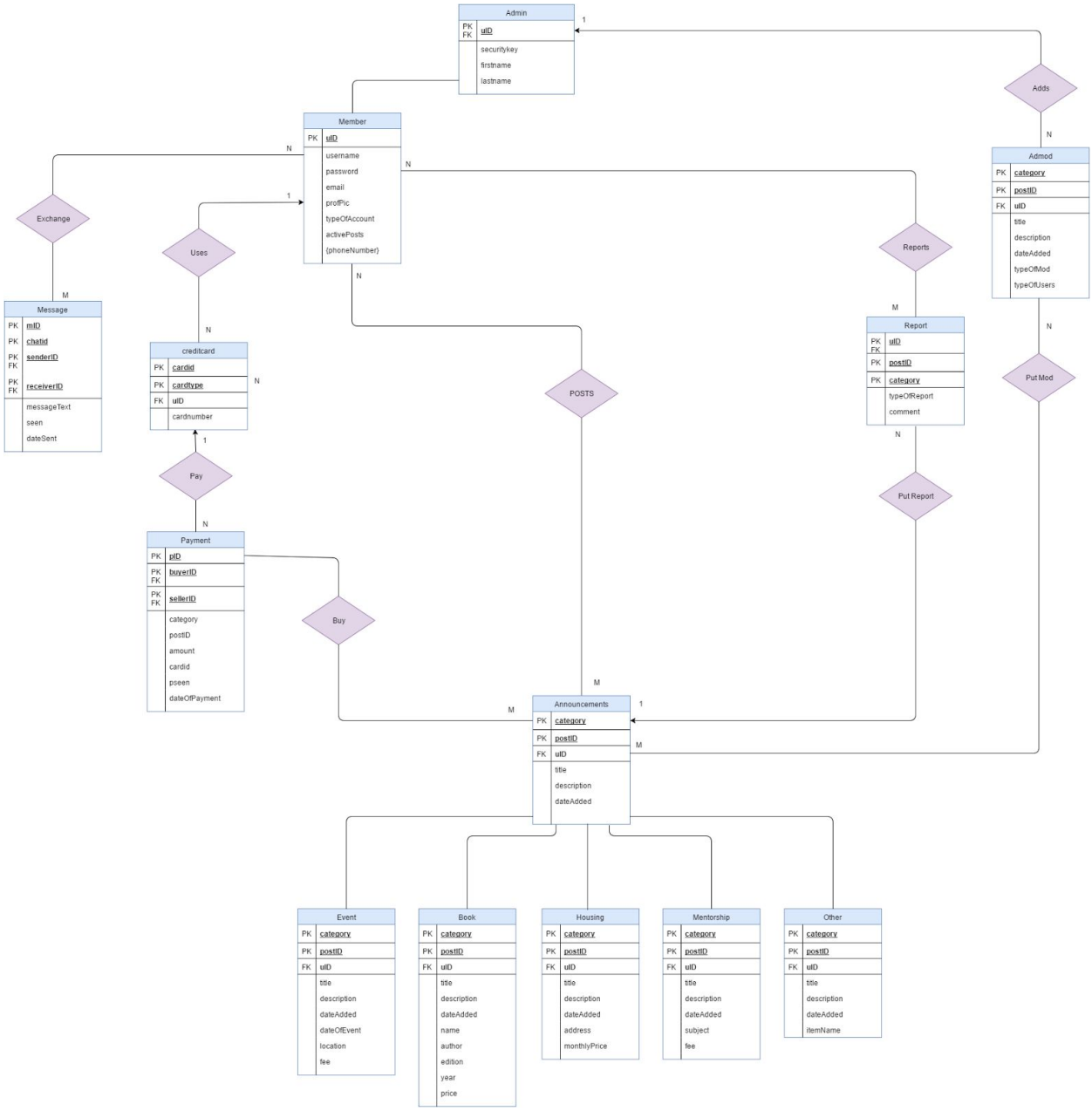
Server Capabilities

- The system shall allow the user to sign in into an existing account or create a new account.
- The signup process will send validation email after registering an account using nodemailer.
- The administrator will be able to delete any reported post.
- User registration is not necessary for browsing the main page.

- A notification should be displayed when announcements have been correctly submitted.
- A notification should be displayed when there is an error while submitting the announcement.

The server side tier is going to be developed using nodemailer and PostgreSQL.

4. E-R Diagram



The E-R Diagram shown above explains the relationships in the system.

The relationship between “Member” and “Message” is N by M. Multiple “Member” can exchange multiple “Messages”.

The relationship between “Payment” and “creditcard” is N by 1. Multiple “Payment” are paid with one “credit card”

The relationship between “Member” and “Announcement” is 1 by N. A “Person” posts multiple “Announcements”.

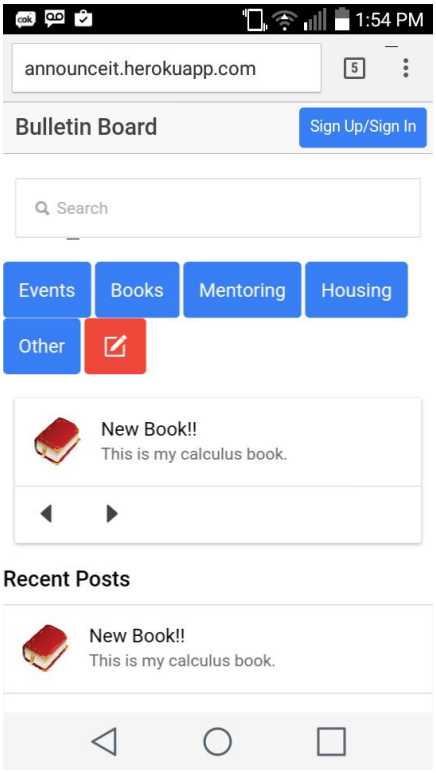
The relationship between “Member” and “Report” is N by M. Multiple “Person” reports an “Announcement”. A “Person” reports multiple “Announcement”.

The relationship between “Admin” and “Admod” is 1 by N. The “Admin” adds multiple “Admod” (Admin Modifications).

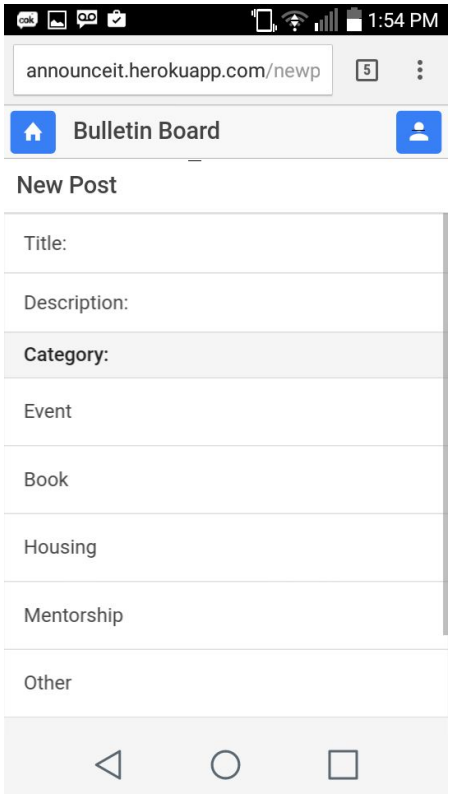
The relationship between “Member” and “creditcard” is 1 by N. The “Member” uses multiple “credit cards”.

The entities “Event”, “Book”, “Housing”, “Mentorship”, and “Other” are sub-entities of the “Announcement” entity.

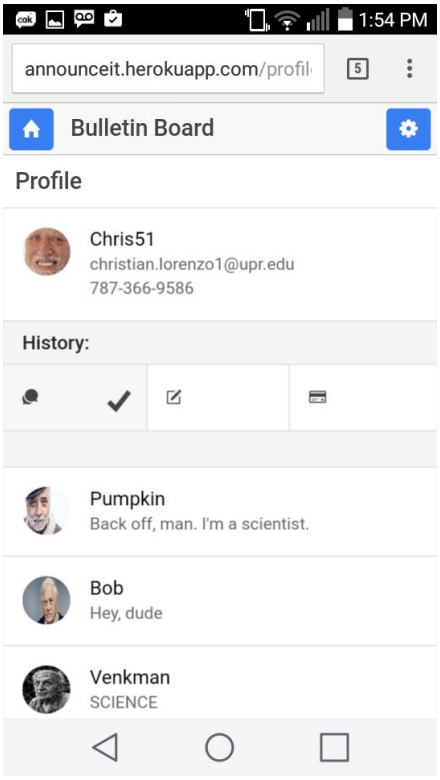
5. Client – Side Screen Shots



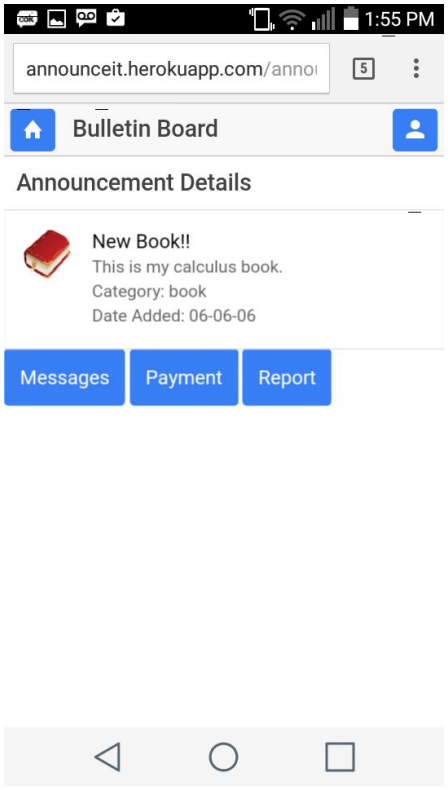
- This is the homepage for Bulletin Board.
- The "Sign Up/Sign In" button takes the user to a form where the user has to input all the necessary information to create an account or to log in into the system.
- The search will filter the results by their title or description, but is not enabled yet.
- The "Events", "Book", "Mentoring", "Housing", "Other" buttons, will filter the results by category. For example, clicking on the "Books" button will result in the view of posts that fall under the books category only.
- The button with the "write symbol" will take the user to the New Post page.



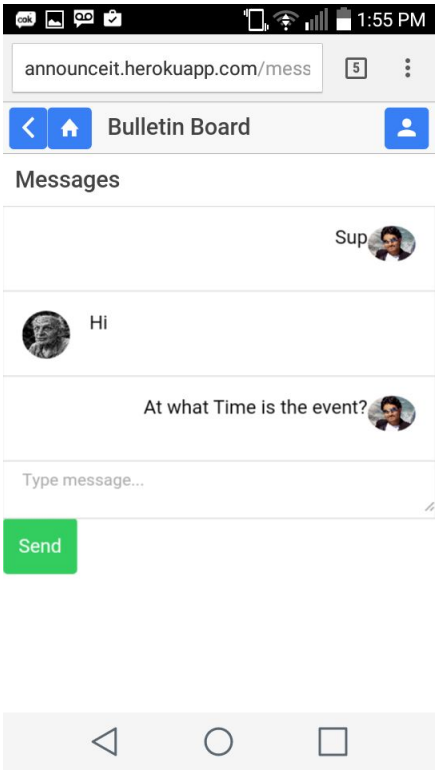
- This is the new post page.
- The user will input all the necessary information to make a new post.
- The user will select the category of the post.
- The button with the “avatar symbol” will take the user to their profile.



- This is the profile page.
- The button with the “home symbol” will take the user to the homepage.
- The button with the “gear symbol” will take the user to the settings page, where the user can edit their information.
- The profile details shown in this page will be the profile picture, username, email, and phone number. The user can select the “message symbol” to view their messages; Or the user can select the “write symbol” to view their previous posts; Or the user can select the “card symbol” to view their payment history.



- This is the announcement details page.
- The user will see the details of their posts.
- The “Messages” button will take the user to the chat page of the original poster and the user.
- The “Payment” button will take the user to a page in which they can make a payment to the original poster.
- The “Report” button will take the user to a page where they can report the post.



- This is the messages page.
- The user can send and receive messages in this page.

SQL Tables:

```
-- Table: public.admin

-- DROP TABLE public.admin;

CREATE TABLE public.admin
(
	uid bigint NOT NULL DEFAULT nextval('admin_uid_seq'::regclass),
	securitykey character varying(16) COLLATE pg_catalog."default" NOT NULL,
	firstname text COLLATE pg_catalog."default" NOT NULL,
	lastname text COLLATE pg_catalog."default" NOT NULL,
	CONSTRAINT admin_pkey PRIMARY KEY (uid),
	CONSTRAINT admin_uid_fkey FOREIGN KEY (uid)
	REFERENCES public.member (uid) MATCH SIMPLE
	ON UPDATE NO ACTION
	ON DELETE NO ACTION
)
WITH (
	OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.admin
	OWNER to cggtxtfflkmdrx;




-- Table: public.admod

-- DROP TABLE public.admod;

CREATE TABLE public.admod
(
	category character(1) COLLATE pg_catalog."default" NOT NULL DEFAULT
'a'::bpchar,
	postid bigint NOT NULL DEFAULT nextval('admod_postid_seq'::regclass),
	uid bigint NOT NULL DEFAULT nextval('admod_uid_seq'::regclass),
	title character varying(20) COLLATE pg_catalog."default" NOT NULL,
	description text COLLATE pg_catalog."default",
	dateadded timestamp without time zone NOT NULL DEFAULT now(),
	typeofmod text COLLATE pg_catalog."default",
	typeofusers text COLLATE pg_catalog."default",
```

```
        CONSTRAINT admod_pkey PRIMARY KEY (category, postid),
        CONSTRAINT admod_uid_fkey FOREIGN KEY (uid)
        REFERENCES public.member (uid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        CONSTRAINT admod_typeofusers_check CHECK (typeofusers = ANY (ARRAY['Regular'::text, 'Premium'::text, 'All'::text])),
        CONSTRAINT admod_typeofmod_check CHECK (typeofmod = ANY (ARRAY['Error'::text, 'Warning'::text, 'Attention'::text, 'Announcement'::text, 'Modification'::text, 'Update'::text]))
)
WITH (
        OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.admod
        OWNER to cggtxtfflkmdrx;



-- Table: public.book

-- DROP TABLE public.book;

CREATE TABLE public.book
(
        category character(1) COLLATE pg_catalog."default" NOT NULL DEFAULT 'b'::bpchar,
        postid bigint NOT NULL DEFAULT nextval('book_postid_seq'::regclass),
        uid bigint NOT NULL DEFAULT nextval('book_uid_seq'::regclass),
        title character varying(20) COLLATE pg_catalog."default" NOT NULL,
        description text COLLATE pg_catalog."default",
        dateadded timestamp without time zone NOT NULL DEFAULT now(),
        name text COLLATE pg_catalog."default" NOT NULL,
        author text COLLATE pg_catalog."default" NOT NULL,
        edition character varying(10) COLLATE pg_catalog."default",
        year integer,
        price numeric(9, 2) DEFAULT 0.00,
        CONSTRAINT book_pkey PRIMARY KEY (category, postid),
        CONSTRAINT book_uid_fkey FOREIGN KEY (uid)
        REFERENCES public.member (uid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
WITH (
        OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.book
        OWNER to cggtxtfflkmdrx;



-- Table: public.creditcard

-- DROP TABLE public.creditcard;

CREATE TABLE public.creditcard
(
        cardid bigint NOT NULL DEFAULT nextval('creditcard_cardid_seq'::regclass),
        uid bigint NOT NULL DEFAULT nextval('creditcard_uid_seq'::regclass),
        cardtype text COLLATE pg_catalog."default" NOT NULL,
        cardnumber text COLLATE pg_catalog."default",
        CONSTRAINT creditcard_pkey PRIMARY KEY (cardid, uid),
        CONSTRAINT creditcard_cardid_key UNIQUE (cardid),
        CONSTRAINT creditcard_uid_fkey FOREIGN KEY (uid)
        REFERENCES public.member (uid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        CONSTRAINT creditcard_cardtype_check CHECK (cardtype = ANY (ARRAY['Paypal'::text, 'MasterCard'::text, 'Visa'::text, 'Discover'::text, 'American Express'::text]))
)
WITH (
        OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.creditcard
        OWNER to cggtxtfflkmdrx;



-- Table: public.event

-- DROP TABLE public.event;

CREATE TABLE public.event
```

```
(
	category character(1) COLLATE pg_catalog."default" NOT NULL DEFAULT
'e'::bpchar,
	postid bigint NOT NULL DEFAULT nextval('event_postid_seq'::regclass),
	uid bigint NOT NULL DEFAULT nextval('event_uid_seq'::regclass),
	title character varying(20) COLLATE pg_catalog."default" NOT NULL,
	description text COLLATE pg_catalog."default",
	dateadded timestamp without time zone NOT NULL DEFAULT now(),
	dateofevent timestamp without time zone,
	location text COLLATE pg_catalog."default",
	fee numeric(9, 2) DEFAULT 0.00,
	CONSTRAINT event_pkey PRIMARY KEY (category, postid),
	CONSTRAINT event_uid_fkey FOREIGN KEY (uid)
	REFERENCES public.member (uid) MATCH SIMPLE
	ON UPDATE NO ACTION
	ON DELETE NO ACTION
)
WITH (
	OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.event
	OWNER to cggtxtfflkmdrx;



-- Table: public.housing

-- DROP TABLE public.housing;

CREATE TABLE public.housing
(
	category character(1) COLLATE pg_catalog."default" NOT NULL DEFAULT
'h'::bpchar,
	postid bigint NOT NULL DEFAULT nextval('housing_postid_seq'::regclass),
	uid bigint NOT NULL DEFAULT nextval('housing_uid_seq'::regclass),
	title character varying(20) COLLATE pg_catalog."default" NOT NULL,
	description text COLLATE pg_catalog."default",
	dateadded timestamp without time zone NOT NULL DEFAULT now(),
	address text COLLATE pg_catalog."default",
	monthlyprice numeric(9, 2) DEFAULT 0.00,
	CONSTRAINT housing_pkey PRIMARY KEY (category, postid),
	CONSTRAINT housing_uid_fkey FOREIGN KEY (uid)
	REFERENCES public.member (uid) MATCH SIMPLE
```

```
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
        OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.housing
        OWNER to cggtxtfflkmdrx;



-- Table: public.member

-- DROP TABLE public.member;

CREATE TABLE public.member
(
        uid bigint NOT NULL DEFAULT nextval('member_uid_seq'::regclass),
        username character varying(16) COLLATE pg_catalog."default" NOT NULL,
        password character varying(20) COLLATE pg_catalog."default" NOT NULL,
        email character varying(50) COLLATE pg_catalog."default" NOT NULL,
        profpic bytea,
        typeofaccount character varying(16) COLLATE pg_catalog."default" DEFAULT
'Regular'::character varying,
        activeposts integer DEFAULT 0,
        phonenumber text COLLATE pg_catalog."default",
        CONSTRAINT member_pkey PRIMARY KEY (uid),
        CONSTRAINT member_email_key UNIQUE (email),
        CONSTRAINT member_username_key UNIQUE (username),
        CONSTRAINT member_check CHECK (activeposts <= 3 AND
typeofaccount::text = 'Regular'::text OR typeofaccount::text = 'Premium'::text)
)
WITH (
        OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.member
        OWNER to cggtxtfflkmdrx;



-- Table: public.mentorship
```

```
-- DROP TABLE public.mentorship;

CREATE TABLE public.mentorship
(
	category character(1) COLLATE pg_catalog."default" NOT NULL DEFAULT 'm'::bpchar,
	postid bigint NOT NULL DEFAULT nextval('mentorship_postid_seq'::regclass),
	uid bigint NOT NULL DEFAULT nextval('mentorship_uid_seq'::regclass),
	title character varying(20) COLLATE pg_catalog."default" NOT NULL,
	description text COLLATE pg_catalog."default",
	dateadded timestamp without time zone NOT NULL DEFAULT now(),
	subject character varying(20) COLLATE pg_catalog."default" NOT NULL,
	fee numeric(9, 2) DEFAULT 0.00,
	CONSTRAINT mentorship_pkey PRIMARY KEY (category, postid),
	CONSTRAINT mentorship_uid_fkey FOREIGN KEY (uid)
	REFERENCES public.member (uid) MATCH SIMPLE
	ON UPDATE NO ACTION
	ON DELETE NO ACTION
)
WITH (
	OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.mentorship
	OWNER to cggtxtfflkmdrx;



-- Table: public.message

-- DROP TABLE public.message;

CREATE TABLE public.message
(
	mid bigint NOT NULL DEFAULT nextval('message_mid_seq'::regclass),
	chatid bigint NOT NULL DEFAULT 1,
	senderid bigint NOT NULL DEFAULT nextval('message_senderid_seq'::regclass),
	receiverid bigint NOT NULL DEFAULT nextval('message_receiverid_seq'::regclass),
	messagetext text COLLATE pg_catalog."default" NOT NULL,
	seen text COLLATE pg_catalog."default" NOT NULL DEFAULT 'Not Seen'::text,
```

```
	datesent timestamp without time zone NOT NULL DEFAULT now(),
	CONSTRAINT message_pkey PRIMARY KEY (chatid, mid, receiverid, senderid),
	CONSTRAINT message_receiverid_fkey FOREIGN KEY (receiverid)
	REFERENCES public.member (uid) MATCH SIMPLE
	ON UPDATE NO ACTION
	ON DELETE NO ACTION,
	CONSTRAINT message_senderid_fkey FOREIGN KEY (senderid)
	REFERENCES public.member (uid) MATCH SIMPLE
	ON UPDATE NO ACTION
	ON DELETE NO ACTION,
	CONSTRAINT message_seen_check CHECK (seen = ANY (ARRAY['Not Seen'::text, 'Seen'::text])),
	CONSTRAINT message_mid_check CHECK (mid > 0)
)
WITH (
	OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.message
	OWNER to cggtxtfflkmdrx;



-- Table: public.other

-- DROP TABLE public.other;

CREATE TABLE public.other
(
	category character(1) COLLATE pg_catalog."default" NOT NULL DEFAULT 'o'::bpchar,
	postid bigint NOT NULL DEFAULT nextval('other_postid_seq'::regclass),
	uid bigint NOT NULL DEFAULT nextval('other_uid_seq'::regclass),
	title character varying(20) COLLATE pg_catalog."default" NOT NULL,
	description text COLLATE pg_catalog."default",
	dateadded timestamp without time zone NOT NULL DEFAULT now(),
	itemname text COLLATE pg_catalog."default" NOT NULL,
	CONSTRAINT other_pkey PRIMARY KEY (category, postid),
	CONSTRAINT other_uid_fkey FOREIGN KEY (uid)
	REFERENCES public.member (uid) MATCH SIMPLE
	ON UPDATE NO ACTION
	ON DELETE NO ACTION
)
```

```
WITH (
        OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.other
        OWNER to cggtxtfflkmdrx;



-- Table: public.payment

-- DROP TABLE public.payment;

CREATE TABLE public.payment
(
        pid bigint NOT NULL DEFAULT nextval('payment_pid_seq'::regclass),
        buyerid bigint NOT NULL DEFAULT nextval('payment_buyerid_seq'::regclass),
        sellerid bigint NOT NULL DEFAULT nextval('payment_sellerid_seq'::regclass),
        cardid bigint NOT NULL DEFAULT nextval('payment_cardid_seq'::regclass),
        category character(1) COLLATE pg_catalog."default" NOT NULL,
        postid bigint NOT NULL DEFAULT nextval('payment_postid_seq'::regclass),
        amount numeric(9, 2) NOT NULL,
        dateofpayment timestamp without time zone NOT NULL DEFAULT now(),
        pseen text COLLATE pg_catalog."default" DEFAULT 'Not Seen'::text,
        CONSTRAINT payment_pkey PRIMARY KEY (buyerid, pid, sellerid),
        CONSTRAINT payment_buyerid_fkey FOREIGN KEY (buyerid)
        REFERENCES public.member (uid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        CONSTRAINT payment_cardid_fkey FOREIGN KEY (cardid)
        REFERENCES public.creditcard (cardid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        CONSTRAINT payment_sellerid_fkey FOREIGN KEY (sellerid)
        REFERENCES public.member (uid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        CONSTRAINT payment_pseen_check CHECK (pseen = ANY (ARRAY['Not
Seen'::text, 'Seen'::text]))
)
WITH (
        OIDS = FALSE
)
TABLESPACE pg_default;
```

```
ALTER TABLE public.payment
	OWNER to cggtxtfflkmdrx;



-- Table: public.report

-- DROP TABLE public.report;

CREATE TABLE public.report
(
	category character(1) COLLATE pg_catalog."default" NOT NULL DEFAULT 'o'::bpchar,
	postid bigint NOT NULL,
	uid bigint NOT NULL DEFAULT nextval('report_uid_seq'::regclass),
	typeofreport text COLLATE pg_catalog."default",
	comment text COLLATE pg_catalog."default" NOT NULL,
	CONSTRAINT report_pkey PRIMARY KEY (category, postid, uid),
	CONSTRAINT report_uid_fkey FOREIGN KEY (uid)
	REFERENCES public.member (uid) MATCH SIMPLE
	ON UPDATE NO ACTION
	ON DELETE NO ACTION,
	CONSTRAINT report_typeofreport_check CHECK (typeofreport = ANY (ARRAY['Inappropriate'::text, 'Spam'::text, 'Offensive'::text, 'Scam'::text, 'Other'::text]))
)
WITH (
	OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.report
	OWNER to cggtxtfflkmdrx;
```