Najrin Sultana-1605042

Abrar Nadib-1605043

Bikroy.com

Functionality:

• Users Can Sign up.

• Users Can Sign in.

• Users Can Post a product.

• Users will get notifications if he needs to pay.

• Users will get notifications after sending his product for admin verification.

• Admin can see pending products.

• Admin needs to verify products.

• Users will get notifications when admin verifies his product.

• Admin can removes a product.

• Users will get notifications if admin removes his.

• Registered Users can report a product.

• Every admin will be notified if someone reports.

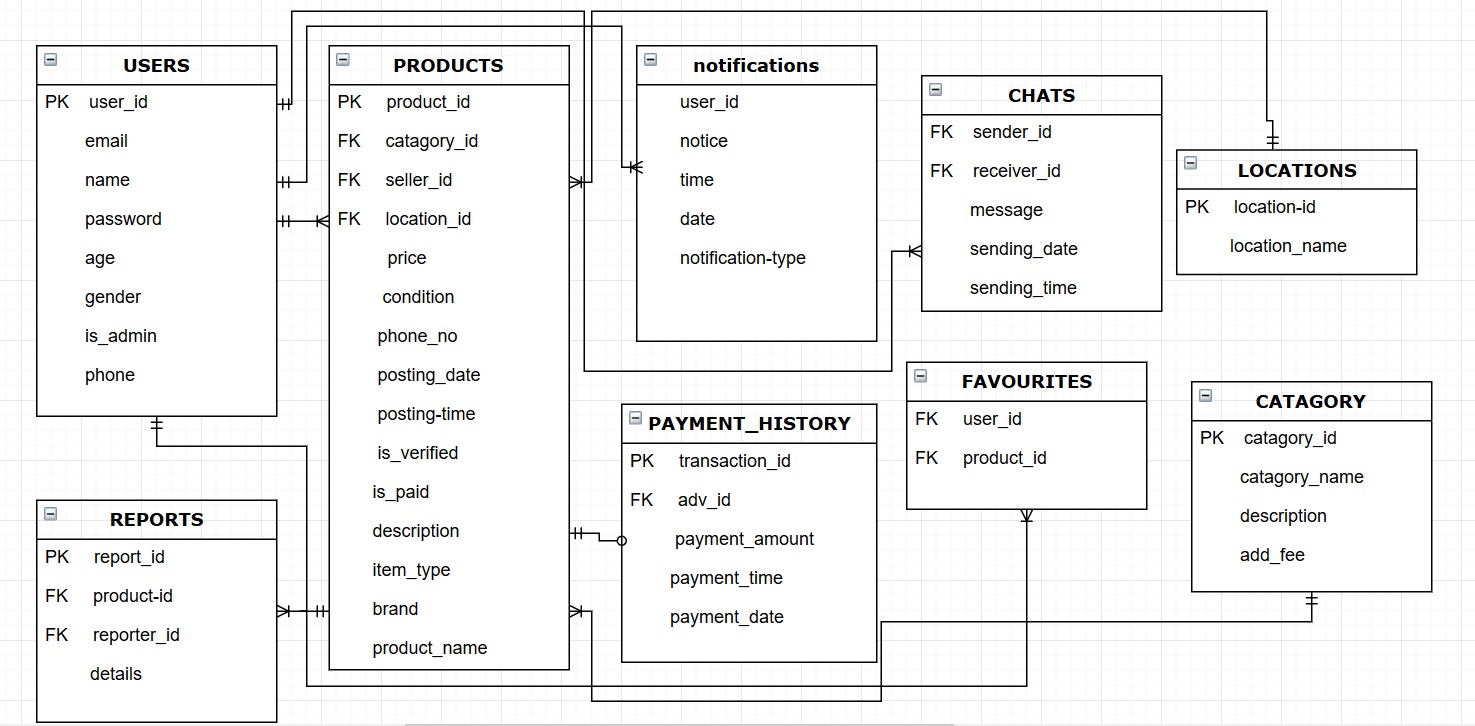
• Registered Users can send message to each other.

•Registered users can mark a product as favorite and view the favorite products later.

• Users can view products by category and locations.

• Users can search for products.

ERD



DDL-Bikroy.com

For Products Table:

CREATE TABLE public.products(

product\_id integer NOT NULL DEFAULT nextval('products\_product\_id\_seq'::regclass),

catagory\_id integer NOT NULL,seller\_id integer NOT NULL,

price integer NOT NULL,

location\_id integer NOT NULL,

condition character varying(255) COLLATE pg\_catalog."default",

phone\_no bigint NOT NULL,

posting\_date date NOT NULL,

posting\_time time(6) without time zone NOT NULL,

is\_varified boolean DEFAULT false,

is\_paid boolean description character varying COLLATE pg\_catalog."default" NOT NULL,

item\_type character varying(255) COLLATE pg\_catalog."default",

brand character varying(255) COLLATE pg\_catalog."default",

product\_name character varying(255) COLLATE pg\_catalog."default",

CONSTRAINT products\_pkey PRIMARY KEY (product\_id),

CONSTRAINT products\_catagory\_id\_fkey FOREIGN KEY (catagory\_id)

REFERENCES public.catagory (catagory\_id) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT products\_location\_id\_fkey FOREIGN KEY (location\_id)

REFERENCES public.location (location\_id) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT products\_seller\_id\_fkey FOREIGN KEY (seller\_id)

REFERENCES public.users (user\_id) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE)

For Users Table:

CREATE TABLE public.users(

email character varying(255) COLLATE pg\_catalog."default" NOT NULL,

name character varying(255) COLLATE pg\_catalog."default" NOT NULL,

password character varying(255) COLLATE pg\_catalog."default" NOT NULL,

age integer NOT NULL,

user\_id integer NOT NULL DEFAULT nextval('users\_user\_id\_seq'::regclass),

gender character varying(255) COLLATE pg\_catalog."default" NOT NULL,

is\_admin boolean DEFAULT false,

phone bigint NOT NULL,

CONSTRAINT users\_pkey PRIMARY KEY (user\_id),

CONSTRAINT users\_email\_key UNIQUE (email))

For Location Table:

CREATE TABLE public.location

(

location\_id integer NOT NULL DEFAULT nextval('location\_location\_id\_seq'::regclass),

location\_name character varying(255) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT location\_pkey PRIMARY KEY (location\_id),

CONSTRAINT location\_location\_name\_key UNIQUE (location\_name)

)

For Catagory Table:

CREATE TABLE public.catagory

(

catagory\_id integer NOT NULL DEFAULT nextval('catagory\_catagory\_id\_seq'::regclass),

catagory\_name character varying(255) COLLATE pg\_catalog."default" NOT NULL,

description character varying(255) COLLATE pg\_catalog."default" NOT NULL,

add\_fee integer DEFAULT 0,

CONSTRAINT catagory\_pkey PRIMARY KEY (catagory\_id),

CONSTRAINT catagory\_catagory\_name\_catagory\_id\_key UNIQUE (catagory\_name)

INCLUDE(catagory\_id)

)

For Chats Table:

CREATE TABLE public.chats

(

sender\_id integer NOT NULL,

reciever\_id integer NOT NULL,

message character varying COLLATE pg\_catalog."default" NOT NULL,

sending\_date date NOT NULL,

sending\_time time without time zone NOT NULL,

CONSTRAINT chats\_reciever\_id\_fkey FOREIGN KEY (reciever\_id)

REFERENCES public.users (user\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT chats\_sender\_id\_fkey FOREIGN KEY (sender\_id)

REFERENCES public.users (user\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

For Favourites Table:

CREATE TABLE public.favourites

(

user\_id integer NOT NULL,

product\_id integer NOT NULL,

CONSTRAINT favourites\_product\_id\_fkey FOREIGN KEY (product\_id)

REFERENCES public.products (product\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT favourites\_user\_id\_fkey FOREIGN KEY (user\_id)

REFERENCES public.users (user\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

For Notifications Table:

CREATE TABLE public.notifications

(

user\_id integer NOT NULL,

notice character varying COLLATE pg\_catalog."default" NOT NULL,

"time" time without time zone NOT NULL,

date date NOT NULL,

notification\_type character varying COLLATE pg\_catalog."default",

CONSTRAINT notifications\_user\_id\_fkey FOREIGN KEY (user\_id)

REFERENCES public.users (user\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

For Reports Table:

CREATE TABLE public.reports

(

report\_id integer NOT NULL DEFAULT nextval('reports\_report\_id\_seq'::regclass),

product\_id integer NOT NULL,

details character varying COLLATE pg\_catalog."default" NOT NULL,

reporter\_id integer NOT NULL,

CONSTRAINT reports\_pkey PRIMARY KEY (report\_id),

CONSTRAINT reports\_product\_id\_fkey FOREIGN KEY (product\_id)

REFERENCES public.products (product\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT reports\_reporter\_id\_fkey FOREIGN KEY (reporter\_id)

REFERENCES public.users (user\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

For paymnt\_history Table:

CREATE TABLE public.paymnt\_history

(

adv\_id integer NOT NULL,

transaction\_id integer NOT NULL DEFAULT nextval('paymnt\_history\_transaction\_id\_seq'::regclass),

payment\_amount integer NOT NULL,

payment\_date date NOT NULL,

payment\_time time without time zone NOT NULL,

CONSTRAINT paymnt\_history\_pkey PRIMARY KEY (transaction\_id),

CONSTRAINT paymnt\_history\_adv\_id\_fkey FOREIGN KEY (adv\_id)

REFERENCES public.products (product\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

FUNCTIONS

**post\_product()**

**This function will post a product.**

**The function will return false if the user has to pay for the product.  
A notification will be sent the user to pay for the product through trigger**

**CREATE OR REPLACE FUNCTION public.post\_product(pcatagory\_id integer, pseller\_id integer, pprice integer, plocation\_id   
integer,pcondition character varying,pphone\_no bigint,pdescription character varying, pitem\_type character varying,   
pbrand character varying, pproduct\_name character varying )  
RETURNS bool   
LANGUAGE 'plpgsql'  
declare  
pis\_paid bool;  
amount integer;  
begin  
select catagory.add\_fee into amount from catagory where catagory.catagory\_id=pcatagory\_id;  
if amount=0 then   
pis\_paid:=null;  
else  
pis\_paid:=false;  
end if;  
INSERT INTO public.products(  
catagory\_id, seller\_id, price, location\_id, condition, phone\_no, posting\_date, posting\_time, is\_varified, is\_paid, description, item\_type, brand, product\_name)  
VALUES (pcatagory\_id, pseller\_id, pprice, plocation\_id, pcondition, pphone\_no, current\_date, localtime, false,pis\_paid,pdescription, pitem\_type, pbrand, pproduct\_name);  
return pis\_paid;  
end;**

#### register\_user()

##### This function will register the user if no user with the same email id exists.

###### CREATE OR REPLACE FUNCTION public.register\_user( page integer, pemail character varying, pgender character varying, pname character varying, ppassword character varying, pphone bigint) RETURNS boolean AS $BODY$ declare coun integer; BODY boolean; begin select count(\*) into coun from users where pemail=users.email; if(coun=0) then BODY:=true; INSERT INTO public.users( email, name, password, age, gender, phone) VALUES (pemail, pname, ppassword, page, pgender, pphone); else BODY:=false; end if; return BODY; end;

#### add\_favourite()

##### This function will add a product as favourite

###### CREATE OR REPLACE FUNCTION public.add\_favourite(u\_id in integer,p\_id in integer) RETURNS void begin if p\_id not in (select product\_id from favourites where favourites.user\_id=u\_id) then INSERT INTO public.favourites(user\_id, product\_id) VALUES (u\_id,p\_id); end if; end;

TRIGGER

FUNCTIONS

#### please\_pay()

##### This trigger function will send notification to user to pay.

###### CREATE FUNCTION public.please\_pay() RETURNS trigger begin if new.is\_paid=false then INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES (new.seller\_id, 'Please pay for your product '||new.product\_name, localtime, current\_date, 'payment'); end if; return null; end;

#### product\_verified()

##### This trigger function will send notification to seller after admin verification.

###### CREATE FUNCTION public.product\_verified() RETURNS trigger begin if old.is\_varified=false and new.is\_varified=true then INSERT INTO public.notifications( user\_id, notice, "time", date, notification\_type) VALUES (new.seller\_id, ' your product '||new.product\_name||' is verified by admin', localtime, current\_date, 'verification'); end if; return null; end;

#### sent\_for\_verification\_paid()

##### This trigger function will send notification to seller after his payment.

###### CREATE FUNCTION public.sent\_for\_verification\_paid() RETURNS trigger declare amount integer; begin select add\_fee into amount from catagory where catagory.catagory\_id=( select products.catagory\_id from products where products.product\_id=new.adv\_id); if new.payment\_amount>=amount then UPDATE public.products SET is\_paid=true WHERE products.product\_id=new.adv\_id; INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES ((select products.seller\_id from products where products.product\_id=new.adv\_id), 'Your product '||((select products.product\_name from products where products.product\_id=new.adv\_id))||'is sent for admin verification', localtime, current\_date, 'verification'); end if; return null; end;

#### sent\_for\_verification\_unpaid()

##### This trigger function will send notification to seller after posting add if he doesn't need to pay.

###### CREATE FUNCTION public.sent\_for\_verification\_unpaid() RETURNS trigger begin if new.is\_paid is null then INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES (new.seller\_id, ' your product '||new.product\_name||' is sent for admin verification ', localtime, current\_date, 'verification'); end if; return null; end;

#### public.someone\_reported()

##### This trigger function will send notification to every admin if someone reports.

###### CREATE FUNCTION public.someone\_reported() RETURNS trigger declare r record; begin for r in(select user\_id from users where users.is\_admin=true) loop INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES (r.user\_id, 'the product with id '||new.product\_id||' is reported', localtime, current\_date, 'report'); end loop; return null; end;

#### public.product\_deleted()

##### This trigger function will send notification to seller if his product is removed.

###### CREATE FUNCTION public.product\_deleted() RETURNS trigger begin INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES (old.seller\_id, ' your product '||old.product\_name||' is removed by admin', localtime, current\_date, 'removal'); return null; end;

TRIGGERS

#### public.log\_reported

##### This trigger will fire someone\_reported() function.

###### CREATE TRIGGER log\_reported AFTER INSERT ON public.reports FOR EACH ROW EXECUTE PROCEDURE public.someone\_reported();

#### public.log\_sent\_for\_verification\_unpaid

##### This trigger will fire sent\_for\_verification\_unpaid() function

###### CREATE TRIGGER log\_sent\_for\_verification\_unpaid AFTER INSERT ON public.products FOR EACH ROW EXECUTE PROCEDURE public.sent\_for\_verification\_unpaid();

#### public.log\_product\_verified

##### This trigger will fire log\_product\_verified() function..

###### CREATE TRIGGER log\_product\_verified AFTER UPDATE OF is\_varified ON public.products FOR EACH ROW EXECUTE PROCEDURE public.product\_verified();

#### public.log\_sent\_for\_verification\_paid

##### This trigger will fire sent\_for\_verification\_paid() function .

###### CREATE TRIGGER log\_sent\_for\_verification\_paid AFTER INSERT ON public.paymnt\_history FOR EACH ROW EXECUTE PROCEDURE public.sent\_for\_verification\_paid();

#### public.log\_product\_deleted

##### This trigger will fire product\_deleted() function .

###### CREATE TRIGGER log\_product\_deleted AFTER DELETE ON public.products FOR EACH ROW EXECUTE PROCEDURE public.product\_deleted();

#### public.log\_please\_pay

##### This trigger will fire please\_pay() function.

###### CREATE TRIGGER log\_please\_pay AFTER INSERT ON public.products FOR EACH ROW EXECUTE PROCEDURE public.please\_pay();

SIMPLE QUERY

1. This Query Will Show the Catagory:  
  
select \* from catagory;

2. This Query Will Show the Locations:  
  
select \* from location;

3. This Query Will Show the Producs of Mymensingh  
  
select \* from products   
where products.location\_id=1 and products.is\_varified=true   
order by products.is\_paid,products.posting\_date,products.posting\_time;

4. This Query Will Show the Mobile Products  
  
select \* from products  
where products.catagory\_id=6 and products.is\_varified=true  
order by products.is\_paid,products.posting\_date,products.posting\_time;

5. This Query Will Verify a Product  
  
UPDATE public.products  
SET is\_varified=true  
WHERE products.product\_id=22;

6. This Query will show A particular product details  
  
select \* from products  
where products.product\_id=1;

7. This Query will show pending products to admin  
  
select \* from products  
where (products.is\_paid is null or products.is\_paid=true)and products.is\_varified=false  
order by products.posting\_date,products.posting\_time desc ;

8. This Query will show favourites of a user  
  
select \* from favourites  
where favourites.user\_id=2;

9. This Query will send a message  
  
INSERT INTO public.chats(sender\_id, reciever\_id, message, sending\_date, sending\_time)  
VALUES (1, 2,'what time is it', current\_date,localtime );

10. This Query will show the inbox of a user  
  
select \* from chats   
where chats.reciever\_id=2 order by sender\_id, sending\_date,sending\_time;

11. This Query will show notifications of user 2  
  
select \* from notifications   
where notifications.user\_id=2 order by sending\_date,sending\_time;

12. This Query will report a product  
  
INSERT INTO public.reports(product\_id, details, reporter\_id)  
VALUES (16, 'Invalid description',4);

13. This Query will show reports of a product  
  
select \* from reports  
where reports.product\_id=16;

14. This Query will remove a product  
  
DELETE FROM public.products  
WHERE products.product\_id=22;

15. This Query will remove a user  
  
DELETE FROM public.users  
WHERE users.user\_id=2;

16. This Query will pay for a product  
  
INSERT INTO public.paymnt\_history(adv\_id, payment\_amount, payment\_date, payment\_time)  
VALUES (22, 100, current\_date, localtime);

17. This Query will return a user(if exists) with this email and password exists  
  
select users.email,users.name,users.password   
from users whereusers.email='nana@gmail.com' and users.password='nananana';

COMPLEX QUERIES

|  |
| --- |
| 1. This Query Will Show products of 'Mymensingh' if someone searches by this keyword  select \* from products where is\_varified=true and location\_id in (select location\_id from location where position(lower('Mymensingh')in lower(location\_name))>0);  2. This Query Will Show 'Mobile' products if someone searches by this keyword  select \* from products where is\_varified=true and catagory\_id in (select catagory\_id from catagory where position(lower('Mobile')in lower(catagory\_name))>0);  3. This Complex Query was used in sent\_for\_verification\_paid()  Trigger function to insert in the notifications table.  NSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type)  VALUES ((select products.seller\_id from products where products.product\_id=new.adv\_id), 'Your product '||((select  products.product\_name from products where products.product\_id=new.adv\_id))||'is sent for admin verification',  localtime, current\_date, 'verification'); |

4. This Query Will Show the seller details of a product  
  
select \*  
from users  
where users.user\_id =  
(select seller\_id from products where products.product\_id=1);

5. This Query Will Show the produc details of a particular report  
  
select \*  
from products  
where products.product\_id=  
(select reports.product\_id from reports where reports.report\_id=3);

THE END