Dhirubhai Ambani
Institute of Information and Communication Technology

Subject: DBMS

Subject code: IT214

PHOTO MANAGEMENT SYSTEM

Date of submission: 23/11/2022

Due date of submission: 23/11/2022

Name: **ADITYA KOTHARI (202001115)**

GAURAV SHAH (202001116)

Group - 2

Section - 7

Id - 7.6

CONTENTS

Software Requirement Specifications

SRS

1. INTRODUCTION

1.1 PURPOSE

- The purpose of this document is to describe a Photo Management System in which users can store any number of photos from different authorised devices, how users can have easy access to their photos from anywhere with internet services and features which enable different users to collaborate in a single database.

- Provides users with sorting functionality based on different categories like time, date, location, user, size of photo, etc.

1.2 INTENDED AUDIENCE AND READING SUGGESTIONS

This document is intended for all the developers of the products, professors of the university, and teaching assistants of the corresponding course. The target audience for this product is anyone or group with multiple photos or videos in

need of systematic storage of data. This document contains a description for a prototype of the photograph management system. It can be useful to get familiar with the product.

1.3 PRODUCT SCOPE

The Photo Management System provides the users to store/organise photos and access them from different devices.

- can be used for various functionalities like filtering the data by different parameters like date-time, location, categories, tags, etc.

- provides the user a secure and interactive interface that stores all his/her data at one place.

- Users can store a high number of pictures on a monthly subscription basis.

1.4 DESCRIPTION

Requirements :
The system will keep track of all the photos uploaded or downloaded by every user or collaboratively uploaded photos. It will store the places, date, time, size of the picture and all the basic information regarding a photo. It will handle upload and download of multiple photos by multiple users. The software will ensure the safety of the data.

Possible functions that can be implemented :

- ☐ Upload photo
- ☐ Delete photo
- ☐ Sort by time
- ☐ Sort by place
- ☐ Sort by user
- ☐ Sort by tag
- ☐ Sort by size
- ☐ Sort by device
- ☐ Create Album
- ☐ Share album
- ☐ Photo detail
- ☐ Comment on a photo
- ☐ View photos of specific person
- ☐ Download photo

Possible relations we would be using :

- ☐ User_detail
- ☐ Photo_detail
- ☐ Subscription_plan
- ☐ Comment_detail
- ☐ Friends
- ☐ Login_credentials
- ☐ User_class
- ☐ Album

Real-World Work Flow of Photo Management System

- ☐ An existing user, who has already registered, may log into the system, or create a new user account for the new user.

- ☐ They will have the option to upload or delete photos from their available set of photographs, or so called profile.
- ☐ They will be able to create their own album from the existing photos or by adding new ones.
- ☐ If the user has a lot of photos and wants to select photos by certain criteria, he/she will be able to do that by filtering with respect to time, place, device, user, tags, etc.
- ☐ The user can download their pictures from any device they log into.
- ☐ If the user has an active subscription plan, he/she can share their albums to their close ones.
- ☐ A user with an active subscription plan can also comment on the pictures of their close contacts.
- ☐ The user will be able to see the expiry date of the subscription plan and/or renew their subscription.

2. Fact Finding Phase

2.1 Background Readings

Description of readings

- ☐ Photobucket.com https://photobucket.com/

  This site has a good user interface.It helped us as a guide for our project and it showed a precise way to implement some features into our project.

- ☐ https://qdraw.github.io/starsky/

  Photo Management system by Dion van Velde.

  It is a free photo-management tool, which runs on your server or web-space, which allows you to organise your photos according to your choice. Although it does not allow the user to either share albums or access photo albums of their contacts. From this we learnt what a photo database management system requires and how we can add more features to it.

- ☐ Database System Concepts, by Abraham Silberschatz, Henry F. Korth, S. Sudarshan.

  From this article, we understood the DBMS concepts like E-R Model, Database designing etc.

2.2 Interviews

Photo Dump: Interview Plan-I

System: Photo Dump Database

Interviewee: 1) Somin Gandhi **Designation:** Student

Interviewer: 1) Gaurav Shah **Designation** - System Developer

2) Aditya Kothari **Designation** - System Developer

Date: 04/10/2022 **Time:** 7:00 p.m.

Duration: 30 minutes **Place:** C-118 HOR Men

Purpose of Interview:

To understand the perspective of a normal student regarding Photo Storage System and what are his thoughts about the existing functionality of the system.And to judge the importance of the system in his daily life.

Agenda:

- Problems faced during transferring photos and videos from one device to another.
- Expected features from user point of view.
- Difficulties in sharing the photos to others.
- Security and privacy issues concerning the user.
- Views regarding subscription plan as a student.

Documents to be brought to the interview:

Rough sketch of the tables and functions of the system.

Photo Dump : Interview Summary

System: Photo Dump Database

Interviewee: 1) Somin Gandhi **Designation**: student

Interviewer: 1) Gaurav Shah **Designation** - System Developer

2) Aditya Kothari **Designation -** System Developer

Date: 4/10/2022 **Time:** 7:00 p.m.

Duration: 30 min **Place:** C-118 HOR Men

Purpose of Interview:

To understand the perspective of a normal student regarding Photo Storage System and what are his thoughts about the existing functionality of the system.And to judge the importance of the system in his daily life.

Result of the interview :

- What does a normal student expect from the system.
- Multiple devices are used by a user and how the system can help in it.

- How the system would help in sorting and organising the photos.
- On what basis would the system create different albums like tags and categories.
- How the security of the photos should be approached.
- How the tagging of the photos should be implemented and at what phase of uploading.
- Sharing of their photos to other users, interacting with them and commenting on the shared photos.
- How to contribute with other users to upload a photo or multiple photos.
- Different range of plans suiting the users.

Photo Dump: Interview Plan - II

System: Photo Dump Database

Interviewee: 1) Darva Patel **Designation:** Photographer(role play)

Interviewer: 1) Gaurav Shah **Designation:** System Developer

2) Aditya Kothari **Designation:** System Developer

Date: 4/10/22 **Time:** 9:00 p.m.

Duration: 20 mins **Place:** D-115 HOR Men

Purpose of Interview:

Necessary meeting to state and identify the problems and requirements of the Photo storage system regarding all basic features and features should exist and how it can change the view of people in this particular domain over management of their photos entirely.

Agenda:

- Problems faced in storing abundant photos.
- Setting the range of pricing for the subscription plan
- How to manage the data more proficiently and make it easily accessible.
- Access to the system on different platforms.

Documents to be brought to the interview:

Rough sketch of the tables and functions of the system.

Photo Dump : Interview Summary

System: Photo Dump Database

Interviewee: 1) Dharva Patel **Designation**: Photographer

Contact Details:

Interviewer: 1) Gaurav Shah **Designation**: System Developer

2) Aditya Kothari **Designation**: System Developer

Date: 4/10/22 **Time:** 9:00 p.m.

Duration: 20 minutes **Place:** D- 115 HOR Men

Purpose of Interview:

Necessary meeting to state and identify the problems and requirements of the Photo storage system regarding all basic features and features should exist and how it can change the view of people in this particular domain over management of their photos entirely.

Results of the interview :

- Uploading of the photos in large quantities simultaneously.
- Removing duplicate photos.
- Sorting of photos by time, location and tags.
- Differentiating between different types of photographs.
- Subscription plan for large storage users.

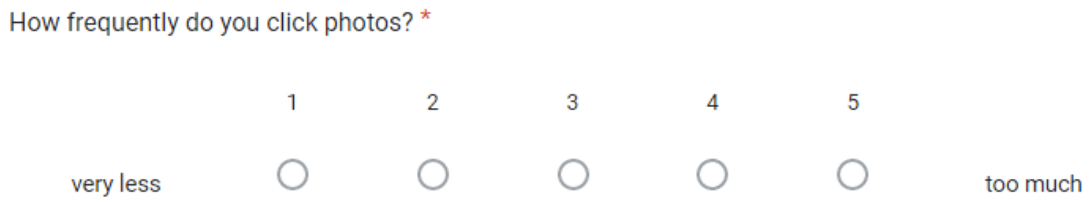
- Subscription plan for storing photos in high quality and its subsequent download.
- Access to photos on different platforms supporting the system.

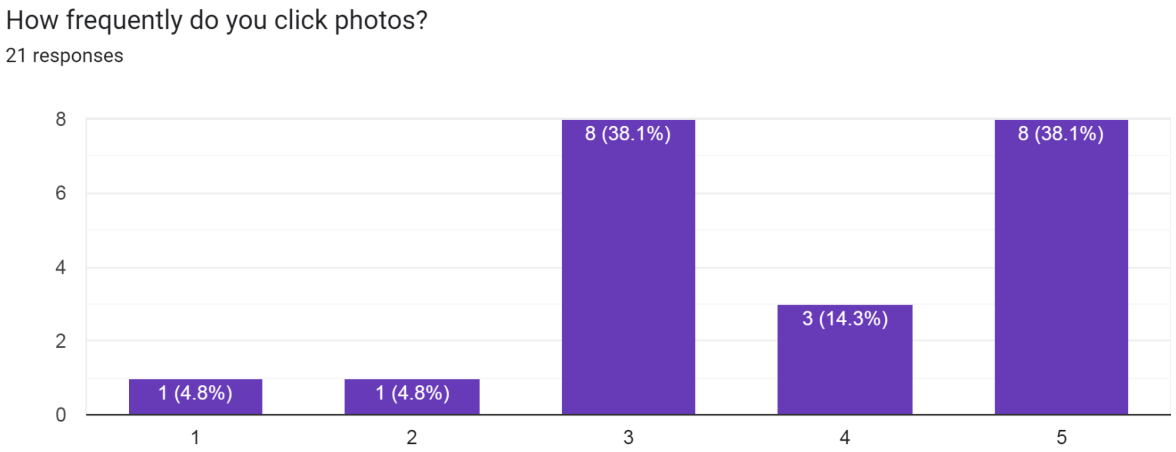
Combined requirements gathered from Interviews

- Fast system to upload multiple photos simultaneously.
- Efficient sorting and organisation of photos according to users needs and removal of duplicates.
- Ways to communicate between users through comments.
- Secure sharing of photos with other users and smooth interaction with each other.
- Subscription plans that should be interesting and allow users to store a large number of high quality photos.

2.3 Questionnaires

1.





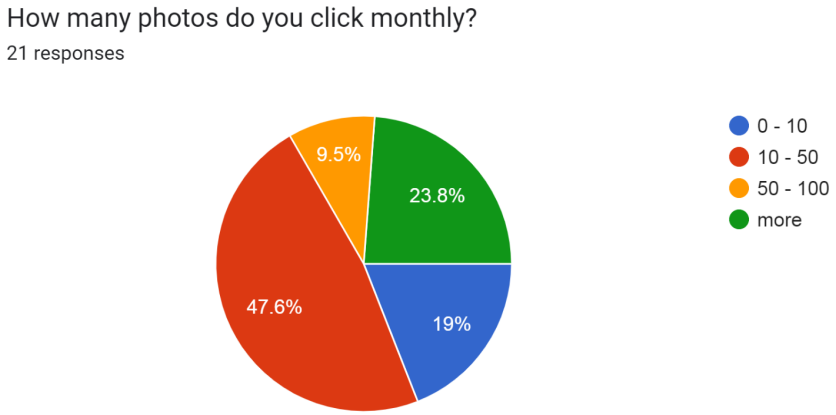
Intent of the question :

to see the frequency of the photos clicked by a person.

Observation from responses :

We can observe that users technically click a high number of photos per day. It shows the potential users for the software.

2.



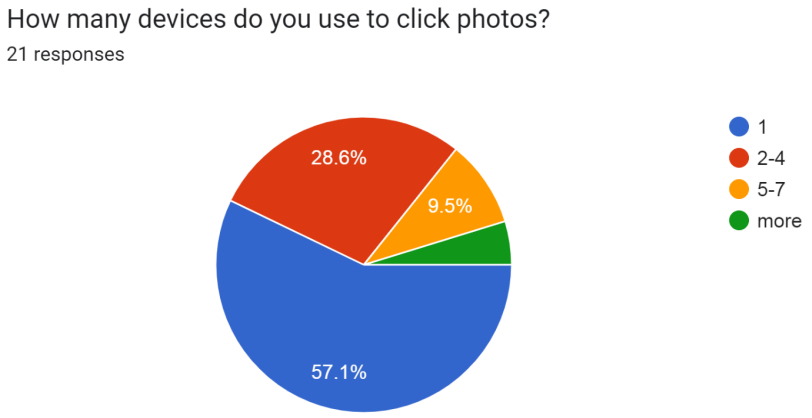
Intent of the questions :

get a range of photographs clicked per month so we can set the limit for free storage.

Observation from responses :

More than 33% of people click more than 50 photos per month.

3.



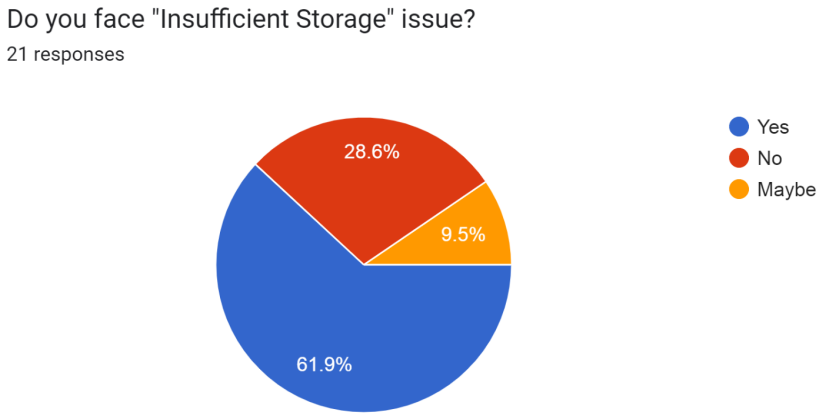
Intent of the question :

To get to know how many devices a person uses to take photos.

Observation :

More than 42% of people use at least 2 or more devices to take photos.

4.



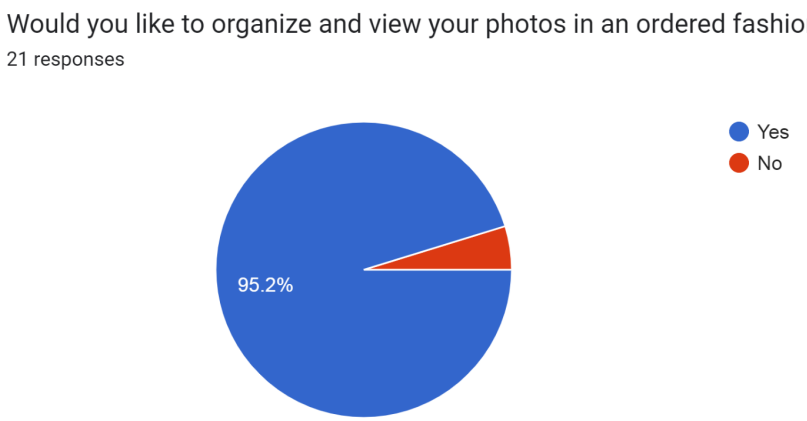
Intent of the question :

To get to know the need for storage of the end users so as to set the storage plans.

Observation :

Majority of the potential users are in need of extra storage.

5.



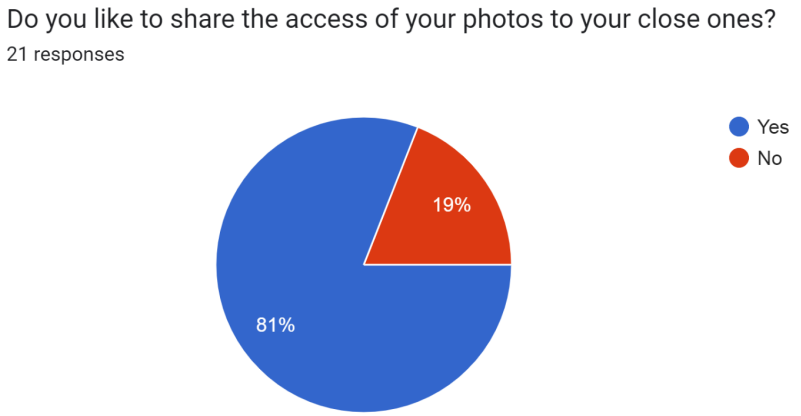
Intent of the question :

Market requirement for the features the software is providing.

Observation :

Almost all the responders want an organised system for their photos.

6.



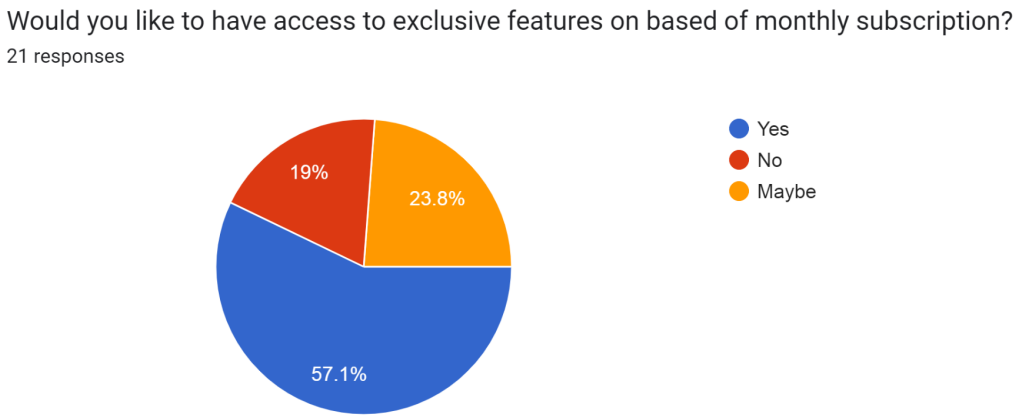
Objective of the question :

to get to know the demand for a feature of the software, which is the ability to share one's photos to another user.

Observation :

Most of them have a positive response to function.

7.



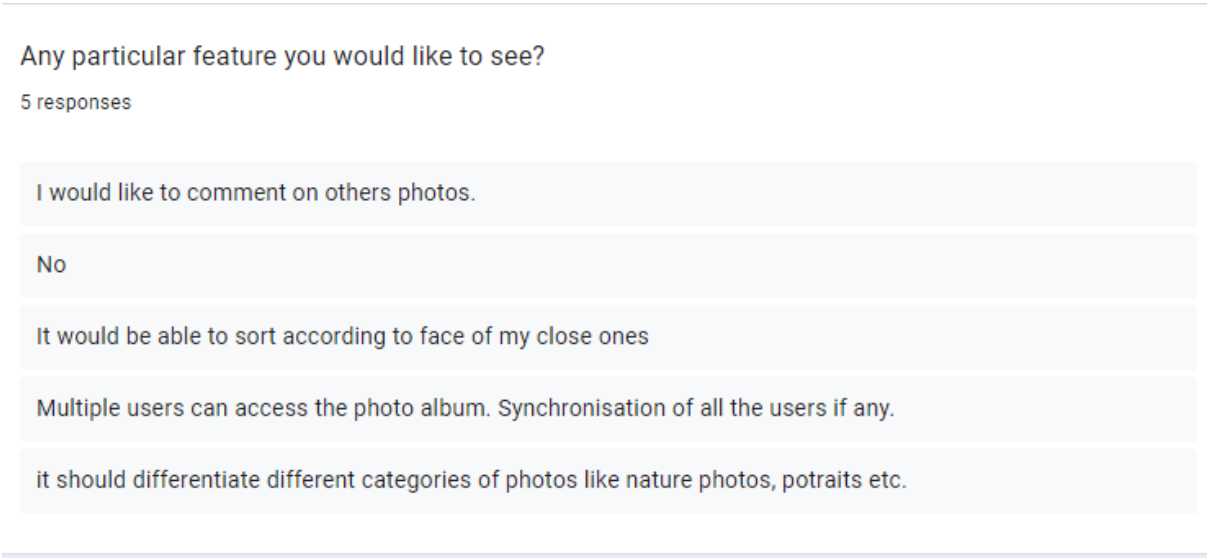
Objective of the question :

Current market perspective to paid subscriptions for photo storage and additional features.

Observation :

>50% of people agree to paid benefits and 23% more of them are potential customers for it if the plan is satisfactory enough.

8.



Combined requirements gathered from Questionnaire :

- Our system should be able to handle big databases as the number of photos clicked and the quality of them can be high.
- The system should be able to function on multiple platforms as more than 40% participants use multiple devices.
- The system should have a great UI and should be easy to use as many participants agree to sharing photos and videos.
- The subscription plan should be interesting and reasonable enough so that all the potential customers can access it.
- The system should be open to updates in the future as participants request additional features.
- The system would require large cloud storage to store all the photos of users.

2.4 Observations

Photo Storage System : Observations

System: Photo Dump Database

Observations by: Gaurav Shah

Date: 4/10/2022 **Time:** 2:30 p.m.

Duration: 45 minutes **Place:** Wedding Hall

Observations:

- Multiple photos were clicked in quick succession.
- Different quality photos were stored, some of high quality and some of low quality.
- Previous photos and videos were referred to constantly.
- Many photos were searched in a short period of time.
- A lot of photos were taken of many different people.

Combined Requirements from Observation :

- Fast and reliable searching and sorting functionality.
- Able to store a large number of photos quickly.
- Store high quality photos.
- Identification of different tags for different persons.

3. Fact Finding Chart

Objective	Technique	Subject	Time commitment
To get the background knowledge of the photo management	Background Readings	Few already existing systems and requirements for the management system	24 hrs
Market analysis for average income of a person	Observation	Customer	3 hrs
To get the basis requirement and problems faced by an aspiring photographer	Interview	photographer	0.5 hrs
To understand the end user's needs	Questionnaire	Google form	24 hrs
To setup reasonable plans for the end users	Observation	Owner of the database	2 hrs
Interaction between the users	Interview	Student	0.5 hrs
To determine what relations would the management system comprise of	Background Readings	Already existing systems of different categories	1 hr

4. List of Requirements

- Efficient sorting and organisation of photos according to users needs and removal of duplicates. [2]
- System structure and functions should be designed in such a way that it ensures the fast performance of the system. [2]
- Secure sharing of photos with other users and smooth interaction with each other. [2]
- The system should have a great UI and should be easy to use as many participants agree to sharing photos and videos. [1]
- Feedback should be received at particular stages of the system. [1]
- The system should be open to updates in the future as participants request additional features. [1]
- The subscription plan should be interesting and reasonable enough so that all the potential customers can access it. [2]
- Devices capable of uploading and downloading data. [1]
- We should also provide backup for our database. [1]

** frequency of requirements are mentioned alongside.*

5. User Categories and Privileges

- **Photo Management Owner -** This will be the admin of the database. He will have all the privileges .
- **Free User (without subscription) -** It will be the end user and they will have the privileges to upload, delete, create albums, filter the pictures with different tags.
- **Paid User (with subscription) -** It will get all the privileges of the free users along with ability to share albums, add comments to shared photos and more storage space.

6. Assumptions

- We assume that all the details regarding the photo are already provided.
- All the users have ID and passwords safe and protected.
- The user has a good internet connection available to him.
- Friends and families of the user also are subscribed to the systems.

7. Business Constraints

- Availability of devices with cameras.
- The Internet is not available at all places.
- Pricing for data storage on the server side.
- Price range for subscription plans is limited.

Noun Analysis and ER Diagram

PROBLEM DESCRIPTION

We have to create a photo database management system. It will keep track of all the photos uploaded or downloaded by every user or collaboratively uploaded photos.

WORKFLOW :

- The user will be allotted a unique login id to login to the system. He will be able to log into the system from multiple devices that are connected to the internet.

- After logging in, he/she will be able to upload the photos from that particular device. He/She can store all the pictures he has uploaded from multiple devices. This way users can manage their photos.

- The user will be able to do multiple functions on their collection of photos. He/She can create their own album from their already existing collection of photos or by adding new ones.

- They can filter the photos with respect to time of photo being uploaded, the location where the photo was clicked, which will be mentioned by the user themself, the device from which the photo

was uploaded, the tags associated with the photos, which will be added by the user, and the type or category of the image.

- The user will be able to download images onto the devices he is logged into. He/She can also delete photos from his/her collection.
- The system will provide the user with an option of subscription plan which will have additional features along with the already existing features for the normal user.

- The subscribed user will be able to share their album to their freind's user id. They can also comment on the photos shared with them by their friends and vica versa.

- A comment from any user can be seen by only the user who shared his/her photo.

- The user will be able to view the expiry date of their subscription.

In the implementation of the database, there will be certain important relationship tables. Some of the possible relations will be user_details that will contain user information like id, password, name, age, etc,

photo_details that will contain information about the photo uploaded and will have attributes like photo_id, user_id, size, category, type, time, location etc, tag_details which will contain a wide range of tags,

subscription_details that will list users who have an active subscription plan along with their plans expiry date, comment_details that will contain the comments posted by the users on the particular picture that was set to them, user_friends table that contains the list of friends of users and login
_credentials which will store the login details of the user. User_class will divide users into different groups of admin, normal user and subscribed user.

The requirements of the database are as follows -

- Efficient sorting and organization of photos according to users needs and removal of duplicates.
- System structure and functions should be designed in such a way that it ensures the fast performance of the system.
- Secure sharing of photos with other users and smooth interaction with each other.
- The system should have a great UI and should be easy to use as many participants agree to sharing photos and videos.
- Feedback should be received at particular stages of the system.
- The system should be open to updates in the future as participants request additional features.
- The subscription plan should be interesting and reasonable enough so that all the potential customers can access it.
- Devices capable of uploading and downloading data.
- We should also provide backup for our database.

The database will have an admin that will have all the privileges of the system. There will be two categories of users : normal user and subscribed user. The former will have a limited number of uploading capacity and will not be able to share albums to his/her friends. The subscribed user will have greater upload capacity than the normal user. He/she can also comment on the images of his/her friends and can also share with them his/her custom albums. The comments will only be visible to the user who has shared the album or picture.

UNIQUE FEATURES :

1) **Commenting on shared images/albums**

   Any subscribed user will be able to share images or their albums to their friends or close ones. He can share them using their id which will be allotted to them by the system. The comments on the shared photos/albums will only be visible to the user who has sent them and the one who has commented on them.

   Implementation

   We will add an option of share, for which the user will have to enter an id of the user whom he wants to share. The id has to be valid and for that it will be checked in the system. A user cannot share images to his own id. The user with whom content is shared will get a notification and he will get an option of commenting on that content. Only the two users will have the view access to the comment and no one else will be able to see that comment. A user

can receive multiple comments on the same image by the same users or different users.

Noun and Verb Analysis Table :

Serial no.	Nouns	Verb
1	photo	create
2	database management system	keep
3	user	upload
4	data	download
5	storage	allot
6	location	login
7	time	access
8	size	view
9	device	share
10	database	select
11	system	add
12	album	filter
13	relation	delete
14	subscription	buy
15	date	provide
16	internet	store
17	friends	upload
18	features	update
19	functions	comment
20	image	share
21	admin	allot
22	customers	receive

23	user	delete
24	collection	upload
25	backup	send
26	comment	get
27	performance	commenting
28	participants	access
29	image	enter
30	user information	sharing
31	category	create
32	tags	search
33	comment	write
34	album	collect
35	user	send
36	friends	share
37	subscription	add
38	image	view
39	system	receive
40	notification	post
41	access	download
42	picture	delete
43	content	create
44	photo	keep
45	option	comment
46	device	provide
47	user	share

48	feature	check
49	categories	manages
50	time	
51	location	
52	date	
53	comment	
54	friends	
55	credentials	
56	database	
57	system	
58	user	
59	interaction	
60	data	
61	capacity	
62	album	
63	content	
64	user	
65	image	
66	option	
67	age	
68	class	

Accepted Noun and Verbs list

Candidate entity set	**Candidate attribute set**	**Candidate relationship set**
photo	photo_id date time location device category size	Create, manages, tag
user	user_id name age contact_number	Subscribes, manages, post, edit, login
subscription_plan	sub_plan_id sub_plan_cost sub_plan_duration	subscribes
	album_id	

album	user_id photo_id number_of_photos	Includes, edit
friends	parent_user_id child_user_id photo_id	access
comment	comment_id content photo_id user_id	post
user_class	class_id class_name	
login_credentials	login_user_id password	login

Rejected Noun and Verbs list

Noun	Reject Reason
Database management system	Irrelevant
data	General
storage	Vague
location	Attribute
date	Attribute
age	Attribute
time	Attribute
size	Attribute
device	Attribute
database	General
system	Vague
relation	General
internet	General
features	General
function	General
image	Duplicate
picture	Duplicate
admin	Vague
customers	General
collection	Vague
backup	Vague
performance	Vague

participants	Vague
User information	Vague
category	Attribute
Notification	Vague
content	Attribute
credentials	Irrelevant
interaction	Vague
capacity	Irrelevant
option	Vague

Verb	**Reject Reason**
create	Irrelevant
keep	General
upload	Irrelevant
download	Irrelevant
allot	Duplicate
login	Irrelevant
view	General
select	Vague
share	Irrelevant
add	Vague
filter	General
delete	Irrelevant
buy	General

provide	General
store	Vague
update	Vague
receive	General
send	Duplicate
get	General
enter	General
search	General
write	General
collect	Vague
check	General

Identifying Entity types

The weak entity set includes the “Album” and “Friends” entities.
Only Simple Association links are present in our ER diagram.

Identifying Relationship types

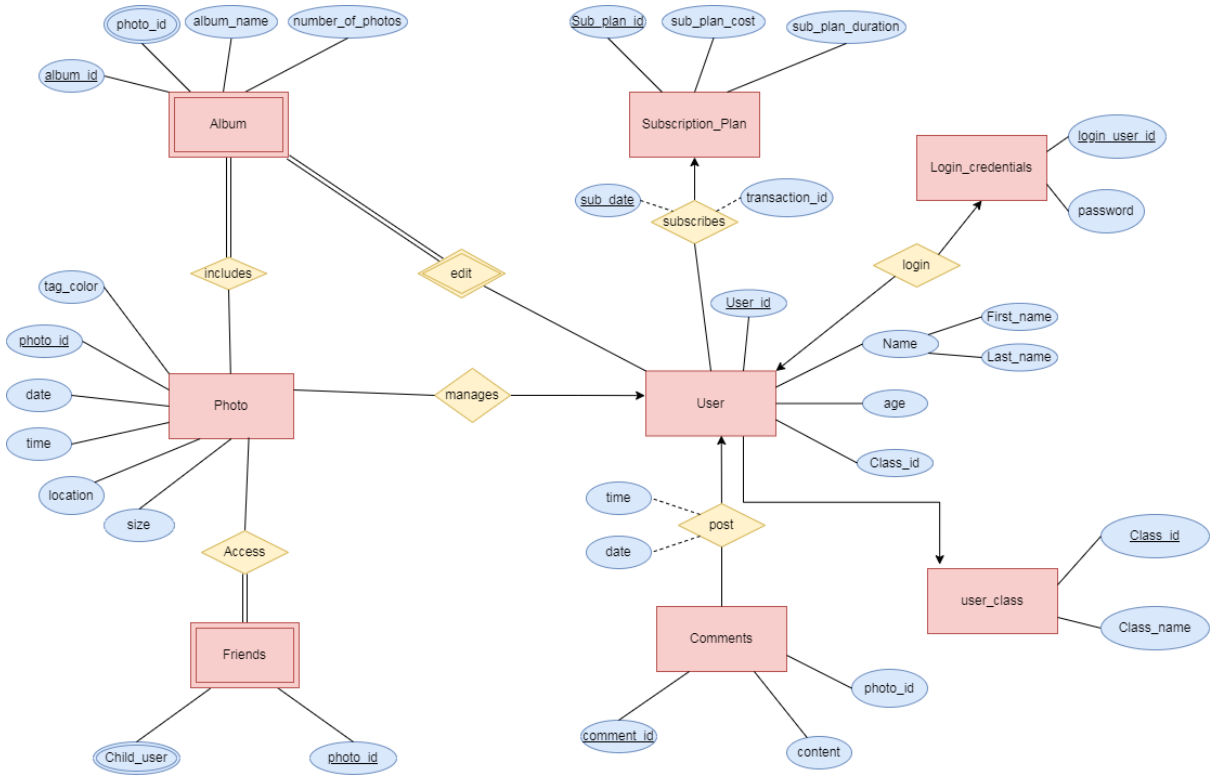
Entities - Attributes

Photo
photo_id, date, time, location, size, tag_color
User
User_id, Name, age, class_id
Subscription_Plan
sub_plan_id, sub_plan_cost, sub_plan_duration
Comments
comment_id, photo_id, user_id, content
Friends
Photo_id, parent_user, child_user
Album
Album_id, Photo_id, number_of_photos
User_class
Class_id, Class_name
Login_credentials
Login_user_id, password

Binary Relationship :

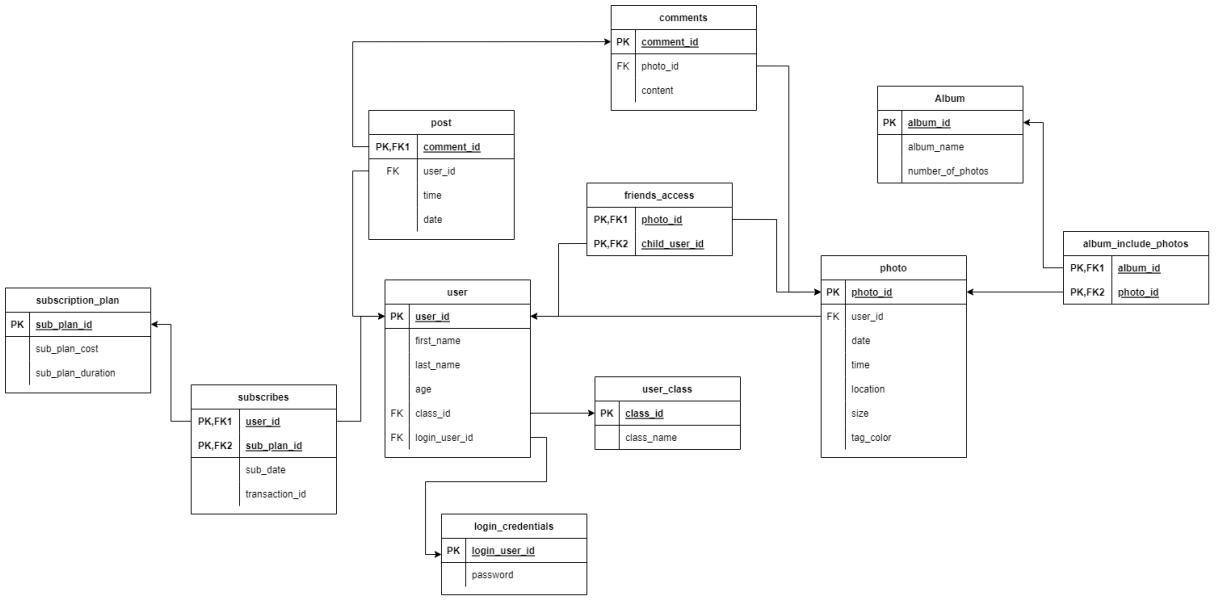
Includes
Edit
Subscribes
Manages
Tag
Access
Post
Login

ER Diagram


photo_id
album_name
number_of_photos
album_id
Album
Sub_plan_id
sub_plan_cost
sub_plan_duration
Subscription_Plan
login_user_id
Login_credentials
password
sub_date
transaction_id
subscribes
includes
edit
login
tag_color
photo_id
date
time
location
size
Photo
manages
User_id
First_name
Name
Last_name
User
age
Class_id
Access
time
post
date
Class_id
user_class
Class_name
Friends
Comments
photo_id
comment_id
content
Child_user
photo_id

ER Diagram to Relational Model

Relational Model



DDL Script

1. album table

```
-- Table: photo_management.album

-- DROP TABLE IF EXISTS photo_management.album;

CREATE TABLE IF NOT EXISTS photo_management.album
(
    album_id bigint NOT NULL,
    album_name character varying COLLATE pg_catalog."default" NOT NULL DEFAULT 'newfolder'::character varying,
    no_of_photos bigint,
    CONSTRAINT album_pkey PRIMARY KEY (album_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management.album
    OWNER to postgres;
```

2. comment table

```
-- Table: photo_management.comments

-- DROP TABLE IF EXISTS photo_management.comments;

CREATE TABLE IF NOT EXISTS photo_management.comments
(
    comment_id bigint NOT NULL,
    content text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT comments_pkey PRIMARY KEY (comment_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management.comments
    OWNER to postgres;
```

3. login_credentials table

```
-- Table: photo_management.login_credentials

-- DROP TABLE IF EXISTS photo_management.login_credentials;

CREATE TABLE IF NOT EXISTS photo_management.login_credentials
(
    login_user_id bigint NOT NULL,
    password character varying COLLATE pg_catalog."default" NOT NULL DEFAULT
o,
    CONSTRAINT login_credentials_pkey PRIMARY KEY (login_user_id)
)
```

```
TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management.login_credentials
    OWNER to postgres;
```

4. subscription_plan table

```
-- Table: photo_management.subscription_plan

-- DROP TABLE IF EXISTS photo_management.subscription_plan;

CREATE TABLE IF NOT EXISTS photo_management.subscription_plan
(
    sub_plan_id bigint NOT NULL,
    sub_plan_cost bigint,
    sub_plan_duration bigint,
    CONSTRAINT subscription_plan_pkey PRIMARY KEY (sub_plan_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management.subscription_plan
    OWNER to postgres;
```

5. user_class table

```
-- Table: photo_management.user_class

-- DROP TABLE IF EXISTS photo_management.user_class;

CREATE TABLE IF NOT EXISTS photo_management.user_class
```

```
(
    class_id bigint NOT NULL,
    class_name character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT user_class_pkey PRIMARY KEY (class_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management.user_class
    OWNER to postgres;
```

6. user table

```
-- Table: photo_management.user

-- DROP TABLE IF EXISTS photo_management."user";

CREATE TABLE IF NOT EXISTS photo_management."user"
(
    user_id bigint NOT NULL,
    first_name character varying COLLATE pg_catalog."default" NOT NULL,
    last_name character varying COLLATE pg_catalog."default",
    age bigint,
    class_id bigint,
    login_user_id bigint,
    CONSTRAINT user_pkey PRIMARY KEY (user_id),
    CONSTRAINT user_fkey1 FOREIGN KEY (class_id)
        REFERENCES photo_management.user_class (class_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
```

```
    NOT VALID,
  CONSTRAINT user_fkey2 FOREIGN KEY (login_user_id)
    REFERENCES photo_management.login_credentials (login_user_id) MATCH
SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management."user"
  OWNER to postgres;
```

7. subscribes table

```
-- Table: photo_management.subscribes

-- DROP TABLE IF EXISTS photo_management.subscribes;

CREATE TABLE IF NOT EXISTS photo_management.subscribes
(
  user_id bigint NOT NULL,
  sub_plan_id bigint NOT NULL,
  sub_date date NOT NULL,
  transaction_id bigint,
  CONSTRAINT subscribe_pkey PRIMARY KEY (user_id, sub_plan_id),
  CONSTRAINT subscribe_fkey1 FOREIGN KEY (user_id)
    REFERENCES photo_management."user" (user_id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE,
```

```
    CONSTRAINT subscribes_fkey2 FOREIGN KEY (sub_plan_id)
        REFERENCES photo_management.subscription_plan (sub_plan_id) MATCH
SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management.subscribes
    OWNER to postgres;
```

8. photo table

```
-- Table: photo_management.photo

-- DROP TABLE IF EXISTS photo_management.photo;

CREATE TABLE IF NOT EXISTS photo_management.photo
(
    photo_id bigint NOT NULL,
    date date NOT NULL,
    "time" time without time zone NOT NULL,
    location character varying COLLATE pg_catalog."default",
    size character varying COLLATE pg_catalog."default" NOT NULL,
    tag_color character varying COLLATE pg_catalog."default",
    user_id bigint NOT NULL,
    CONSTRAINT photo_pkey PRIMARY KEY (photo_id),
    CONSTRAINT photo_fkey FOREIGN KEY (user_id)
        REFERENCES photo_management."user" (user_id) MATCH SIMPLE
        ON UPDATE CASCADE
```

```
    ON DELETE CASCADE
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management.photo
  OWNER to postgres;
```

9. post table

```
    -- Table: photo_management.post

    -- DROP TABLE IF EXISTS photo_management.post;

    CREATE TABLE IF NOT EXISTS photo_management.post
    (
      "time" time without time zone NOT NULL,
      date date NOT NULL,
      comment_id bigint NOT NULL,
      user_id bigint,
      CONSTRAINT post_pkey PRIMARY KEY (comment_id),
      CONSTRAINT post_fkey1 FOREIGN KEY (user_id)
        REFERENCES photo_management."user" (user_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
      CONSTRAINT post_fkey2 FOREIGN KEY (comment_id)
        REFERENCES photo_management.comments (comment_id) MATCH
    SIMPLE
```

```
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
    )

    TABLESPACE pg_default;

    ALTER TABLE IF EXISTS photo_management.post
      OWNER to postgres;
```

10. friend_access

```
    -- Table: photo_management.friend_access

-- DROP TABLE IF EXISTS photo_management.friend_access;

CREATE TABLE IF NOT EXISTS photo_management.friend_access
(
  photo_id bigint NOT NULL,
  child_user_id bigint NOT NULL,
  CONSTRAINT access_pkey PRIMARY KEY (photo_id, child_user_id),
  CONSTRAINT access_fkey1 FOREIGN KEY (photo_id)
    REFERENCES photo_management.photo (photo_id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  CONSTRAINT access_fkey2 FOREIGN KEY (child_user_id)
    REFERENCES photo_management."user" (user_id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
)
```

```
TABLESPACE pg_default;

ALTER TABLE IF EXISTS photo_management.friend_access
    OWNER to postgres;
```

11. album_include_photos

```
-- Table: photo_management.album_include_photos

-- DROP TABLE IF EXISTS photo_management.album_include_photos;

CREATE TABLE IF NOT EXISTS photo_management.album_include_photos
(
    album_id bigint NOT NULL,
    photo_id bigint NOT NULL,
    CONSTRAINT include_pkey PRIMARY KEY (album_id, photo_id),
    CONSTRAINT include_fkey1 FOREIGN KEY (album_id)
        REFERENCES photo_management.album (album_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT include_fkey2 FOREIGN KEY (photo_id)
        REFERENCES photo_management.photo (photo_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

TABLESPACE pg_default;
ALTER TABLE IF EXISTS photo_management.album_include_photos
    OWNER to postgres;
```

Normalization and Schema Refinement

The original Relational Model is as follows -

user(<u>user_id</u>, class_id, login_user_id, first_name, last_name, age)
photo(<u>photo_id</u>, user_id, date, time, location, size, tag_color)
comments(<u>comment_id</u>, photo_id, content)
subscription_plan(<u>sub_plan_id</u>, sub_plan_cost, sub_plan_duration)
album(<u>album_id</u>, album_name, number_of_photos)
post(<u>comment_id</u>,user_id, time, date)
friends_access(<u>photo_id</u>, <u>child_user_id</u>)
user_class(<u>class_id</u>, class_name)
login_credentials(<u>login_user_id</u>, password)
subscribes(<u>user_id</u>, <u>sub_plan_id</u>, sub_date, transaction_id)
album_includes_photos(<u>album_id</u>, <u>photo_id</u>)

Now we list down the functional dependencies in all the tables.

photo(photo_id, user_id, date, time, location, size, tag_color)

PK dependency :
Photo_id -> user_id, date, time, location, size, tag_color

Functional dependencies :

Photo_id -> user_id
Photo_id -> date
Photo_id -> time
Photo_id -> location
Photo_id -> size
Photo_id -> tag_color

Partial key dependencies : NONE

Transitive dependencies : NONE

Redundancies : NONE

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

user(<u>user_id</u>, class_id, login_user_id, first_name, last_name, age)

PK dependency :
User_id -> class_id, login_user_id, first_name, last_name, age

Functional dependencies :

User_id -> class_id
User_id -> login_user_id
User_id -> first_name
User_id -> last_name
User_id -> age

Partial key dependencies : None

Transitive dependencies : None

Redundancies : None

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

comments(<u>comment_id</u>, photo_id, content)

PK dependency :
Comment_id -> photo_id, content

Functional dependencies :

Comment_id -> photo_id
Comment_id -> content

Partial key dependencies : None

Transitive dependencies : None

Redundancies : None

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

subscription_plan(sub_plan_id, sub_plan_cost, sub_plan_duration)

PK dependency :
Sub_plan_id -> sub_plan_cost, sub_plan_duration

Functional dependencies :

Sub_plan_id -> sub_plan_cost
Sub_plan_id -> sub_plan_duration

Partial key dependencies : None

Transitive dependencies : None

Redundancies : None

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

album(<u>album_id</u>, album_name, number_of_photos)

PK dependency :
Album_id -> album_name, number_of_photos

Functional dependencies :

Album_id -> album_name
Album_id -> number_of_photos

Partial key dependencies : None

Transitive dependencies : None

Redundancies : None

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

post(<u>comment_id</u>,user_id, time, date)

PK dependency :
Comment_id -> user_id, date, time

Functional dependencies :

Comment_id -> user_id
Comment_id -> date
Comment_id -> time

Partial key dependencies : None

Transitive dependencies : None

Redundancies : None

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

user_class(class_id, class_name)

PK dependency :
Class_id -> class_name

Functional dependencies :

Class_id -> class_name

Partial key dependencies : None

Transitive dependencies : None

Redundancies : None

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

login_credentials(<u>login_user_id</u>, password)

PK dependency :
Login_user_id -> password

Functional dependencies :

Login_user_id -> password

Partial key dependencies : NONE

Transitive dependencies : NONE

Redundancies : NONE

Anomalies :
Insert - NONE

Update- NONE

Delete- NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

subscribes(user_id, sub_plan_id, sub_date, transaction_id)

PK dependency :
user_id + sub_plan_id -> sub_date, transaction_id

Functional dependencies :
user_id + sub_plan_id -> sub_date
user_id + sub_plan_id -> transaction_id

Partial key dependencies : NONE

Transitive dependencies : NONE

Redundancies : NONE

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

album_includes_photos(album_id, photo_id)

PK dependency :
album_id + photo_id

Functional dependencies :
album_id + photo_id

Partial key dependencies : NONE

Transitive dependencies : NONE

Redundancies : NONE

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

friends_access(photo_id, child_user_id)

PK dependency :
Photo_id + child_user_id

Functional dependencies :
Photo_id + child_user_id

Partial key dependencies : NONE

Transitive dependencies : NONE

Redundancies : NONE

Anomalies :
Insert : NONE

Update : NONE

Delete : NONE

In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.

Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.

Hence, this relation is in **BCNF** as well.

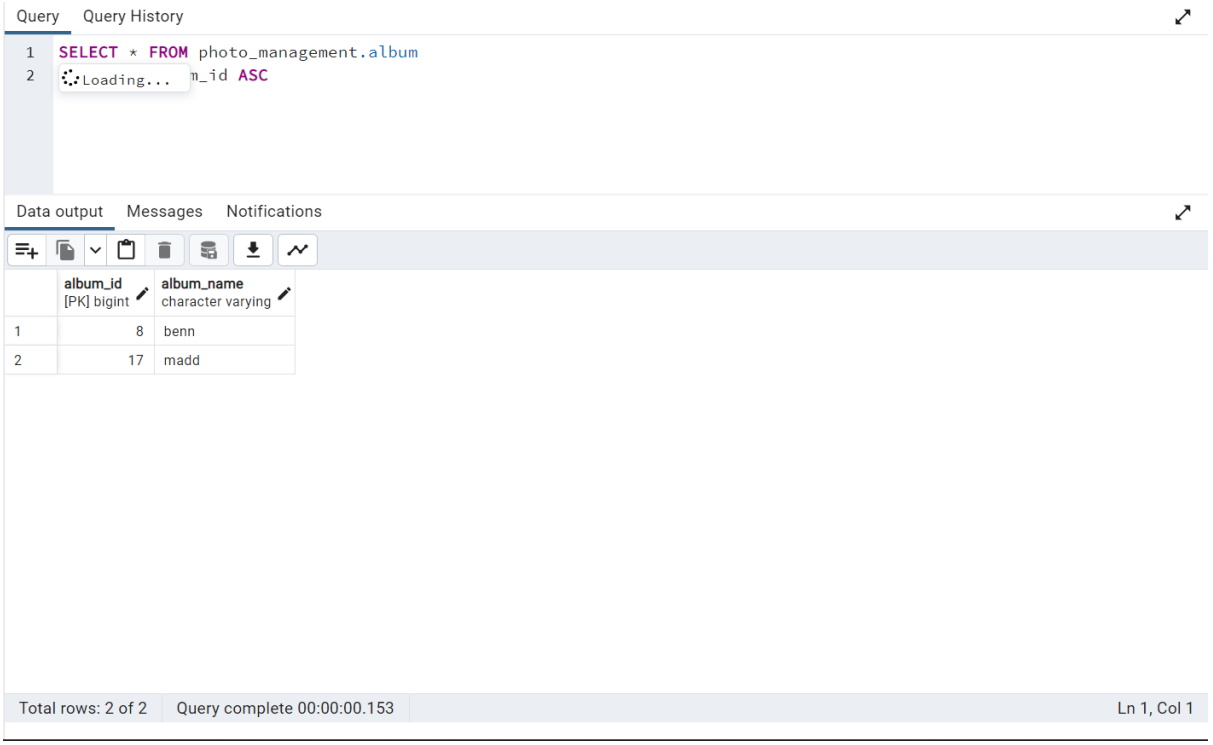
SQL Queries

Inserting entries into the tables

```
SET SEARCH_PATH to photo_management;
COPY "user_class" FROM 'D:\data\user_class.csv' (DELIMITER(','));
COPY "subscription_plan" FROM 'D:\data\sub_plan_data.csv' (DELIMITER(','));
COPY "login_credentials" FROM 'D:\data\login_credentials_data.csv' (DELIMITER(','));
COPY "comments" FROM 'D:\data\comment_data_v2.csv' (DELIMITER(','));
COPY "user" FROM 'D:\data\user_data_v2.csv' (DELIMITER(','));
COPY "post" FROM 'D:\data\post_data_v2.csv' (DELIMITER(','));
COPY "photo" FROM 'D:\data\photo_data_v5.csv' (DELIMITER(','));
COPY "subscribes" FROM 'D:\data\subscribes_data_v2.csv' (DELIMITER(','));
COPY "friend_access" FROM 'D:\data\friend_access_data.csv' (DELIMITER(','));
COPY "album" FROM 'D:\data\album_data.csv' (DELIMITER(','));
COPY "album_include_photos" FROM 'D:\data\album_photo_data.csv' (DELIMITER(','));
select * from subscribes;
```

THE ENTRIES OF THE TABLES ARE AS FOLLOWS -

1 album



2 album_include_photos

Query | Query History

```
1  SELECT * FROM photo_management.album_include_photos
2  ORDER BY album_id ASC, photo_id ASC
```

Loading...

Data output | Messages | Notifications

	album_id [PK] bigint	photo_id [PK] bigint
1	8	286
2	17	581
3	17	712

Total rows: 3 of 3 | Query complete 00:00:00.140 | Ln 2, Col 37

3 comments

Query | Query History

```
1  SELECT * FROM photo_management.comments
2  ORDER BY comment_id ASC
```

Data output | Messages | Notifications

	comment_id [PK] bigint	content text	photo_id bigint
20	552	sagittis.	293
21	563	sit amet	617
22	592	Nunc ma…	677
23	605	Suspendi…	246
24	625	nunc sit a…	394
25	678	aliquet. P…	882
26	681	urna nec l…	361
27	695	magna tel…	293
28	728	et ultrices…	801
29	746	Proin eget	925
30	827	orci adipi…	286
31	830	a feugiat …	581
32	854	nibh. Don…	712
33	893	amet dia…	423
34	922	et netus e…	108
35	923	malesuad…	293

Total rows: 35 of 35 | Query complete 00:00:00.294 | Ln 1, Col 1

4 friend_access

Query | Query History

```
1  SELECT * FROM photo_management.friend_access
2  ORDER BY photo_id ASC, user_id ASC
```

Data output | Messages | Notifications

Loading...

	photo_id [PK] bigint	user_id [PK] bigint
1	108	52440
2	246	77681
3	286	21408
4	293	21408
5	293	57869
6	361	87688
7	394	81195
8	423	40476
9	581	27773
10	617	57890
11	677	69011
12	712	40149
13	801	27773
14	882	84178
15	925	40149

Total rows: 15 of 15 | Query complete 00:00:00.167 | Ln 2, Col 36

5 login_credentials

Query | Query History

```
1  SELECT * FROM photo_management.login_credentials
2  ORDER BY login_user_id ASC
```

Data output | Messages | Notifications

	login_user_id [PK] bigint	password character varying
19	41755	9wvXZRBJ
20	45400	e2yaEeE6UXxn
21	46491	NpulvSnhj5
22	46713	PAjo5P5fXYDX
23	48426	xJpONSLp
24	49211	riWPVf
25	50068	uhpPkR3
26	50189	XJsKsJgLdDn
27	51231	MUHcB1IKTtX2
28	52193	eQwyvlWQTTWM
29	53878	iV4w1ioFqJ3
30	62175	rSi0ILXgov
31	63001	esIR9jDCqU
32	67324	4rE1EOiOgT
33	68951	yjkVelNh
34	71698	z0LMwdV
35	74883	SMJoXCfqgJ

Total rows: 45 of 45 | Query complete 00:00:00.136 | Ln 1, Col 1

6 photo

Query | Query History

```
1  SELECT * FROM photo_management.photo
2  ORDER BY photo_id ASC
```

Data output | Messages | Notifi Loading...

	photo_id [PK] bigint	date date	time time without time zone	location character varying	size character varying	tag_color character varying	user_id bigint
60	771	2022-08-07	02:52:46	Oamaru	10	#ed1a72	25436
61	785	2022-10-09	22:30:01	Sicuani	7	#1f7cba	33746
62	788	2022-08-30	14:39:34	Tyumen	4	#a9ef8b	52923
63	789	2022-09-22	13:43:29	Wellington	4	#6c5cbc	99690
64	791	2022-01-31	02:06:47	San Rafael	7	#f461be	40476
65	801	2022-10-20	08:32:58	Assen	1	#aef495	84178
66	817	2022-07-18	12:35:29	Canoas	3	#653aaf	33574
67	854	2022-03-21	08:40:47	Trondheim	1	#999de8	26545
68	856	2022-04-21	20:19:07	Changi Bay	3	#30a7b2	39693
69	872	2022-05-11	20:08:04	Varash	4	#2e83dd	89126
70	875	2022-03-04	21:38:22	Otukpo	1	#e0bd64	34144
71	879	2021-11-22	00:39:33	Rockingham	1	#6e8ec9	87239
72	882	2021-12-20	13:07:24	Tropea	2	#b5f79b	69011
73	896	2022-04-27	01:14:49	Hohen Neuendorf	5	#447ec4	33574
74	912	2021-11-20	18:20:49	Logroño	2	#068287	52916
75	925	2022-03-22	08:54:00	Saint-Sébastien-s...	4	#0033ff	84178

Total rows: 75 of 75 | Query complete 00:00:00.137 | Ln 2, Col 23

7 post

Query Query History

```
SELECT * FROM photo_management.post
ORDER BY comment_id ASC
```

Data output Messages Notifications

	time time without time zone	date date	comment_id [PK] bigint	user_id bigint
20	05:18:44	2021-12-04	552	69011
21	16:20:48	2022-04-02	563	77681
22	09:29:03	2022-07-08	592	81195
23	03:04:16	2022-06-23	605	84178
24	19:48:28	2022-04-05	625	87688
25	20:52:01	2022-08-29	678	21408
26	20:15:24	2022-10-29	681	27773
27	18:43:33	2021-11-23	695	40149
28	20:54:37	2022-05-25	728	40476
29	11:27:53	2022-01-30	746	52440
30	01:58:38	2022-04-21	827	57869
31	15:17:36	2022-11-08	830	57890
32	22:19:28	2021-12-09	854	69011
33	02:27:32	2022-08-07	893	77681
34	14:11:12	2022-04-16	922	81195
35	02:54:40	2021-11-27	923	84178

Total rows: 35 of 35 Query complete 00:00:00.200 Ln 1, Col 1

8 subscribes

Query Query History

```
SELECT * FROM photo_management.subscribes
ORDER BY user_id ASC, sub_plan_id ASC, sub_date ASC
```

Data output Messages Notifications

	user_id [PK] bigint	sub_plan_id [PK] bigint	sub_date [PK] date	transaction_id bigint
1	21408	3	2021-11-24	158775
2	27773	1	2021-12-09	328793
3	40149	1	2021-12-08	862238
4	40476	3	2022-10-13	869102
5	52440	2	2022-08-27	543844
6	57869	1	2022-05-09	247655
7	57890	1	2022-01-13	204624
8	69011	2	2022-04-01	311441
9	77681	1	2021-12-08	804399
10	81195	2	2022-05-23	441292
11	84178	3	2022-06-16	779386
12	87688	1	2022-02-20	404950

Total rows: 12 of 12 Query complete 00:00:00.277 Ln 1, Col 1

9 subscription_plan

Query | Query History

```
1  SELECT * FROM photo_management.subscription_plan
2  ORDER BY sub_plan_id ASC
```

Data output | Messages | Notifications

	sub_plan_id [PK] bigint	sub_plan_cost bigint	sub_plan_duration bigint
1	1	89	15
2	2	899	200
3	3	4999	1300

Total rows: 3 of 3 | Query complete 00:00:00.203 | Ln 1, Col 1

10 user

Query Query History

```
SELECT * FROM photo_management."user"
ORDER BY user_id ASC
```

Data output Messages Notifications

	user_id [PK] bigint	first_name character varying	last_name character varying	age bigint	class_id bigint	login_user_id bigint
29	61532	Patrizio	Probert	59	1	63001
30	64591	Lezlie	Greatorex	18	4	80129
31	69011	Ros	Ricold	50	2	95437
32	77681	Rozanne	Harryman	30	2	30525
33	78222	Meier	Lighton	52	4	62175
34	79793	Reinaldos	Waryk	34	1	28324
35	81195	Abagael	Klampk	15	2	89850
36	81603	Waldo	Arrighini	15	3	41068
37	84178	Aarika	Blemings	44	2	68951
38	85829	Daveen	Bartomeu	14	4	28588
39	86439	Joceline	Silvester	27	1	21499
40	87239	Rustin	Gutman	47	1	13660
41	87688	Maridel	Matresse	22	2	46713
42	89126	Bren		46	4	50068
43	94019	Oralee	Collyns	32	4	48426
44	97364	Cati	Doddridge	54	1	13322
45	99690	Seth	Banbrigge	53	1	78835

Total rows: 45 of 45 Query complete 00:00:00.146 Ln 1, Col 1

11 user_class

Query Query History

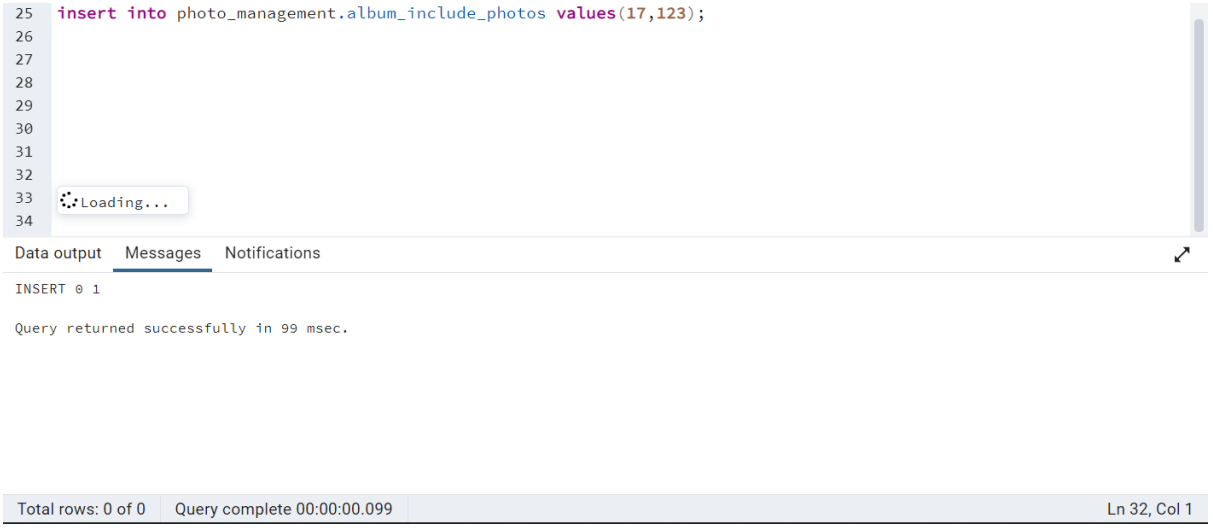
```
SELECT * FROM photo_management.user_class
ORDER BY class_id ASC
```

Data output Messages Notifications

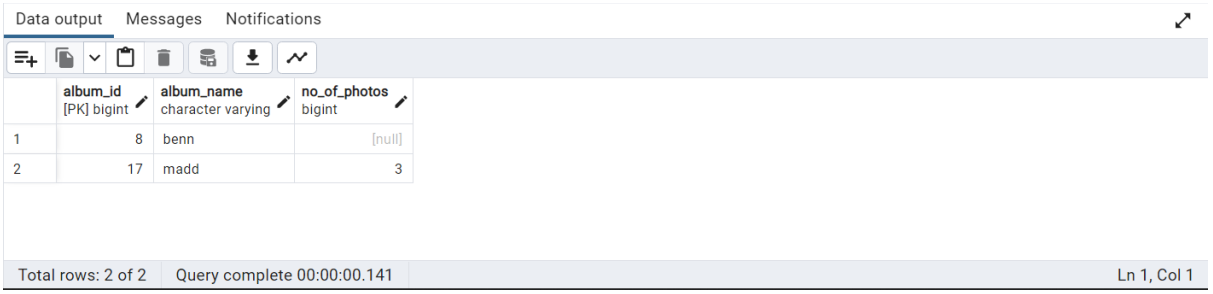
	class_id [PK] bigint	class_name character varying
1	1	free
2	2	subscribed
3	3	admin
4	4	developer

TRIGGER AND FUNCTION

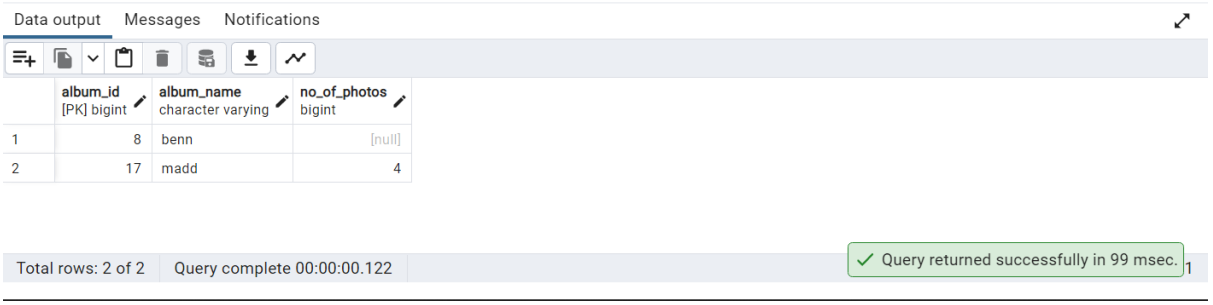
```
SET SEARCH_PATH TO photo_management;
CREATE OR REPLACE FUNCTION photo_sum()
  RETURNS trigger AS
$$
BEGIN
    update photo_management.album
    set no_of_photos = (    SELECT count(photo_id)
        FROM album_include_photos NATURAL JOIN album
        WHERE album_include_photos.album_id = album.album_id
        GROUP BY album.album_id
        HAVING album.album_id = new.album_id
    )
    WHERE album_id = new.album_id;
    RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER photo_sum
  AFTER INSERT
  ON photo_management.album_include_photos
  FOR EACH ROW
  EXECUTE PROCEDURE photo_sum();
```



BEFORE INSERT



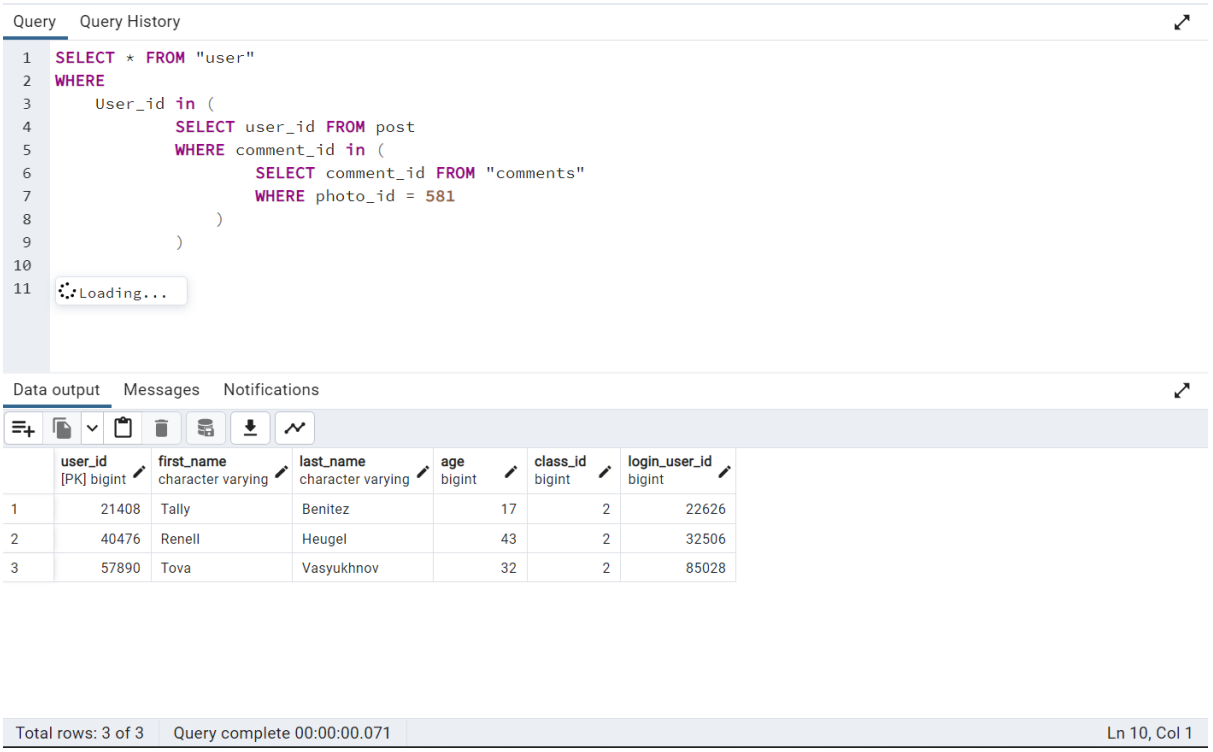
AFTER INSERT



SQL QUERIES

Q1 Display the details of the user who commented on the photo with photo_id = 581.

```
SELECT * FROM "user"
WHERE
	User_id in (
			SELECT user_id FROM post
			WHERE comment_id in (
					SELECT comment_id FROM "comments"
					WHERE photo_id = 581
				)
			)
```



Q2 Display all users who posted photos on 1st January 2022.

```
SELECT * FROM "user"
WHERE user_id in
	(SELECT  user_id FROM "photo"
WHERE photo."date" = '2022-01-01'
)
```

```
SELECT * FROM "user"
WHERE user_id in
    (SELECT  user_id FROM "photo"
WHERE photo."date" = '2022-01-01'
)
```

Data output | Messages | Notifications

	user_id [PK] bigint	first_name character varying	last_name character varying	age bigint	class_id bigint	login_user_id bigint
1	40149	Brita	Geaves	20	2	49211

Total rows: 1 of 1 | Query complete 00:00:01.941 | Ln 11, Col 1

Q3 Display the class details of users who have access to photo with photo_id = 108.

```
SELECT user_id, class_id FROM "user"
WHERE user_id in
        (
            SELECT user_id FROM "friend_access"
            WHERE photo_id = 108
        )
```

Query | Query History

```
SELECT user_id, class_id FROM "user"
WHERE user_id in
        (
            SELECT user_id FROM "friend_access"
            WHERE photo_id = 108
        )
```

Loading...

Data output | Messages | Notifications

	user_id [PK] bigint	class_id bigint
1	52440	2

Total rows: 1 of 1 | Query complete 00:00:22.192 | Ln 27, Col 4

Q4 Display album_id of all albums who have at least 1 photo of size >= 5.

```
SELECT DISTINCT album_id FROM album_include_photos
WHERE
    photo_id in (
            SELECT photo_id FROM photo
            WHERE "size" >= '5'
        )
```

Data output | Messages | Notifications

	album_id bigint
1	8

Total rows: 1 of 1 | Query complete 00:00:05.854 | Ln 17, Col 1

Q5 Display details of all comments given a photo_id = 108.

```
SELECT post.comment_id, user_id, "content", "time", "date" FROM post,"comments"
WHERE
    post.comment_id in (
            SELECT comment_id FROM "comments"
            WHERE photo_id = 108
        )
```

Data output | Messages | Notifications

	comment_id bigint	user_id bigint	content text	time time without time zone	date date
96	922	81195	nibh. Don...	14:11:12	2022-04-16
97	202	40476	amet dia...	05:55:42	2022-03-14
98	543	57890	amet dia...	14:45:49	2021-12-19
99	922	81195	amet dia...	14:11:12	2022-04-16
100	202	40476	et netus e...	05:55:42	2022-03-14
101	543	57890	et netus e...	14:45:49	2021-12-19
102	922	81195	et netus e...	14:11:12	2022-04-16
103	202	40476	malesuad...	05:55:42	2022-03-14
104	543	57890	malesuad...	14:45:49	2021-12-19
105	922	81195	malesuad...	14:11:12	2022-04-16

Total rows: 105 of 105 | Query complete 00:00:00.047 | Ln 24, Col 1

Q6 Display user_id and sub_plan details of user who have access to photo with photo_id = 423.

```
SELECT * FROM subscribes
WHERE
    user_id in (

            SELECT user_id FROM friend_access
            WHERE photo_id = 423
)
```

Data output | Messages | Notifications

	user_id [PK] bigint	sub_plan_id [PK] bigint	sub_date [PK] date	transaction_id bigint
1	40476	3	2022-10-13	869102

Total rows: 1 of 1 | Query complete 00:00:03.793 | Ln 31, Col 1

Q7 Display details of the users who have subscribed to the most expensive plan.

```
SELECT * FROM "user"
WHERE user_id in (
    SELECT user_id Loading... bes
    WHERE sub_plan_id =
(
SELECT sub_plan_id FROM subscription_plan
WHERE sub_plan_cost = (
            SELECT max(sub_plan_cost) FROM subscription_plan
)
)
)
```

Data output | Messages | Notifications

	user_id [PK] bigint	first_name character varying	last_name character varying	age bigint	class_id bigint	login_user_id bigint
1	21408	Tally	Benitez	17	2	22626
2	84178	Aarika	Blemings	44	2	68951
3	40476	Renell	Heugel	43	2	32506

Total rows: 3 of 3 | Query complete 00:00:05.957 | Ln 41, Col 19

Q8 Display details of the users who have not posted any photo.

```
51  SELECT * FROM "user"
52  WHERE user_id not in (
53      SELECT user_id FROM photo
54  )
55
```

Loading...

Data output | Messages | Notifications

user_id [PK] bigint	first_name character varying	last_name character varying	age bigint	class_id bigint	login_user_id bigint

Total rows: 0 of 0 | Query complete 00:00:05.334 | Ln 54, Col 2

Q9 Display details of subscription_plan tha has most users subscribed for.

```
SELECT * FROM subscription_plan
WHERE sub_plan_id =
(
SELECT sub_plan_id
FROM subscribes
GROUP BY sub_plan_id
ORDER BY count(user_id) desc limit 1);
```

Data output | Messages | Notifications

	sub_plan_id [PK] bigint	sub_plan_cost bigint	sub_plan_duration bigint
1	1	89	15

Total rows: 1 of 1 | Query complete 00:00:00.188 | Ln 60, Col 1

Q10 Display the sub_plan_id and sub_date of users who has not posted photos in 2021.

```
SELECT sub_plan_id, sub_date FROM subscribes
WHERE user_id not in (
    SELECT user_id FROM photo
    WHERE date BETWEEN '2021-01-01' and '2021-12-31'
)
```

Loading...

Data output | Messages | Notifications

	sub_plan_id bigint	sub_date date
1	3	2021-11-24
2	3	2022-06-16
3	3	2022-10-13
4	2	2022-08-27
5	1	2021-12-09
6	1	2022-02-20
7	1	2021-12-08
8	2	2022-05-23
9	1	2021-12-08
10	1	2022-05-09

Total rows: 10 of 10 | Query complete 00:00:07.728 | Ln 73, Col 1

Q11 Display the comments on the photo with id = 4 in ascending order of time and date.

```
75  SELECT * FROM post
76  WHERE comment_id in (
77      SELECT comment_id FROM "comments"
78      WHERE photo_id = 423
79  )
80  ORDER BY "date","time"
81
82
```

Data output | Messages | Notifications

	time time without time zone	date date	comment_id [PK] bigint	user_id bigint
1	21:18:01	2022-01-19	178	40149
2	07:10:12	2022-05-30	538	57869
3	02:27:32	2022-08-07	893	77681

Total rows: 3 of 3 | Query complete 00:00:59.760 | Ln 78, Col 22

Q12 Display user_id and class name of users who commented between October 2021 and October 2022.

```
84  SELECT user_id, class_name FROM "user" natural join user_class
85  WHERE user_id in (
86      SELECT user_id FROM post
87      WHERE "date" BETWEEN '2021-10-01' and '2022-10-01'
88  )
89
```

Loading...

Data ... Notifications

	user_id bigint	class_name character varying
7	87688	subscribed
8	77681	subscribed
9	57890	subscribed
10	81195	subscribed
11	40149	subscribed
12	57869	subscribed

Total rows: 12 of 12 | Query complete 00:00:06.698 | Ln 89, Col 1

Q13 Display photo details sorted by the size.

```
98  SELECT * FROM photo
99  ORDER BY "size"
100
101
102
103
104
105
```

Loading...

Data output | Messages | Notifications

	photo_id bigint	date date	time time without time zone	location character varying	size character varying	tag_color character varying	user_id bigint
70	646	2022-07-23	23:36:24	Maiduguri	9	#cfd865	59526
71	688	2021-11-29	13:02:56	Astore	9	#56d3b0	81603
72	413	2021-11-24	13:27:06	Chepén	9	#efd873	50986
73	380	2022-09-18	15:49:32	Swabi	9	#f7278f	97364
74	271	2021-12-23	21:11:34	Pachuca	9	#ff68fc	28053
75	661	2021-12-22	04:48:39	Bremen	9	#41d836	10221

Total rows: 75 of 75 | Query complete 00:00:24.684 | Ln 100, Col 1

Q14 Display users who have access to the photo with id = 688

```
102 SELECT user_id FROM friend_access
103 WHERE photo_id = 688
104
105
```

Data output | Messages | Notifications

	user_id bigint

Total rows: 0 of 0 | Query complete 00:00:03.377 | Ln 102, Col 1

Q15 Display details of the photos where the colour is code '#ff524c'.

```
105  SELECT * FROM photo
106  WHERE tag_color = '#ff524c'
107
108
109
110  Loading...
111
```

Data output | Messages | Notifications

	photo_id [PK] bigint	date date	time time without time zone	location character varying	size character varying	tag_color character varying	user_id bigint
1	108	2022-01-01	08:43:37	Nampa	3	#ff524c	40149

Total rows: 1 of 1 | Query complete 00:00:15.087 | Ln 109, Col 1

Q16 Display user details of subscribed users in ascending order of their name.

```
109  SELECT * FROM "user"
110  WHERE class_id = 2
111  ORDER BY first_name,last_name
112
113  Loading...
114
115
```

Data output | Messages | Notifications

	user_id [PK] bigint	first_name character varying	last_name character varying	age bigint	class_id bigint	login_user_id bigint
6	87688	Maridel	Matresse	22	2	46713
7	40476	Renell	Heugel	43	2	32506
8	69011	Ros	Ricold	50	2	95437
9	77681	Rozanne	Harryman	30	2	30525
10	27773	Sidonnie	Northen	56	2	89602
11	21408	Tally	Benitez	17	2	22626
12	57890	Tova	Vasyukhnov	32	2	85028

Total rows: 12 of 12 | Query complete 00:00:12.485 | Ln 112, Col 1

Q17 Display the details of subscribed users in ascending order of sub_date.

```
114 SELECT * FROM subscribes
115 ORDER BY sub_date
116
117
```

Loading...

Data output | Messages | Notifications

	user_id [PK] bigint	sub_plan_id [PK] bigint	sub_date [PK] date	transaction_id bigint
1	21408	3	2021-11-24	158775
2	40149	1	2021-12-08	862238
3	77681	1	2021-12-08	804399
4	27773	1	2021-12-09	328793
5	57890	1	2022-01-13	204624
6	87688	1	2022-02-20	404950
7	69011	2	2022-04-01	311441
8	57869	1	2022-05-09	247655
9	81195	2	2022-05-23	441292

Total rows: 12 of 12 | Query complete 00:00:08.128 | Ln 116, Col 1

Q18 Display details of users whose age lies between 15 and 25.

```
124 SELECT * FROM "user"
125 WHERE age BETWEEN 15 and 25
126
127
```

Data output | Messages | Notifications

	user_id [PK] bigint	first_name character varying	last_name character varying	age bigint	class_id bigint	login_user_id bigint
1	33746	Rosella	Aleswell	16	3	22160
2	81603	Waldo	Arrighini	15	3	41068
3	21408	Tally	Benitez	17	2	22626
4	87688	Maridel	Matresse	22	2	46713
5	49958	Diannne	Jerzykiewicz	18	4	45400
6	25436	Jobina	O'Luney	15	1	21831
7	64591	Lezlie	Greatorex	18	4	80129
8	81195	Abagael	Klampk	15	2	89850
9	40149	Brita	Geaves	20	2	49211

Total rows: 10 of 10 | Query complete 00:00:02.873 | Ln 119, Col 1

Q19 Display details of users belonging to class developer.

```
119 SELECT * FROM "user"
120 WHERE class_id = 4
121
122
123
```

Data output | Messages | Notifications

	user_id [PK] bigint	first_name character varying	last_name character varying	age bigint	class_id bigint	login_user_id bigint
1	10221	Tilly	Adame	14	4	53878
2	85829	Daveen	Bartomeu	14	4	28588
3	89126	Bren		46	4	50068
4	78222	Meier	Lighton	52	4	62175
5	49958	Diannne	Jerzykiewicz	18	4	45400
6	52916	Vernon		39	4	81693
7	53754	Kevin	Bradlaugh	36	4	74883
8	64591	Lezlie	Greatorex	18	4	80129
9	94019	Oralee	Collyns	32	4	48426

Total rows: 9 of 9 | Query complete 00:00:21.710 | Ln 115, Col 11

Q20 Display comments posted between may 2020 and may 2022.

```
123 SELECT * FROM post
124 WHERE "date" between '2020-05-01' and '2022-05-01'
125
126
```

Data output | Messages | Notifications

	time time without time zone	date date	comment_id [PK] bigint	user_id bigint
9	17:14:50	2022-01-20	325	81195
10	06:39:24	2022-01-08	468	27773
11	17:02:42	2022-03-15	536	52440
12	14:45:49	2021-12-19	543	57890
13	05:18:44	2021-12-04	552	69011
14	16:20:48	2022-04-02	563	77681
15	19:48:28	2022-04-05	625	87688
16	18:43:33	2021-11-23	695	40149
17	11:27:53	2022-01-30	746	52440
18	01:58:38	2022-04-21	827	57869
19	22:19:28	2021-12-09	854	69011
20	14:11:12	2022-04-16	922	81195
21	02:54:40	2021-11-27	923	84178

Total rows: 21 of 21 | Query complete 00:00:00.752 | Ln 120, Col 19

Front End Development

back end code :

model code :

```
class photo(models.Model):

    photo_id = models.IntegerField(primary_key=True)

    date = models.CharField(max_length=11)

    time = models.CharField(max_length = 20)

    location = models.CharField(max_length = 100)

    size = models.IntegerField()

    tag_color = models.CharField(max_length = 10)

    user_id = models.IntegerField()


    class Meta:

        managed = False

        db_table = 'photo'
```

Database code :

```
DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql',

        'NAME': '202001116_db',

        'USER': 'postgres',

        'PASSWORD': 'admin',

        'HOST': 'localhost',

        'PORT':'5432',

    }

}
```

URL code :

```
urlpatterns = [

    path('admin/', admin.site.urls),

    path('',views.HomePage,name="HomePage"),

    path('Insert',views.insertphoto,name="insertphoto"),

    path('showphoto',views.showphoto,name="showphoto"),

    path('Edit/<int:id>',views.editphoto,name="editphoto"),

    path('Update/<int:id>',views.updatephoto,name="updatephoto"),

    path('Delete/<int:id>',views.deletephoto,name="deletephoto"),

    path('sortasc',views.sortasc,name="sortasc"),

    path('sortdesc',views.sortdesc,name="sortdesc"),

    path('runquery',views.runquery,name="runquery"),

]
```

FUNCTIONS :

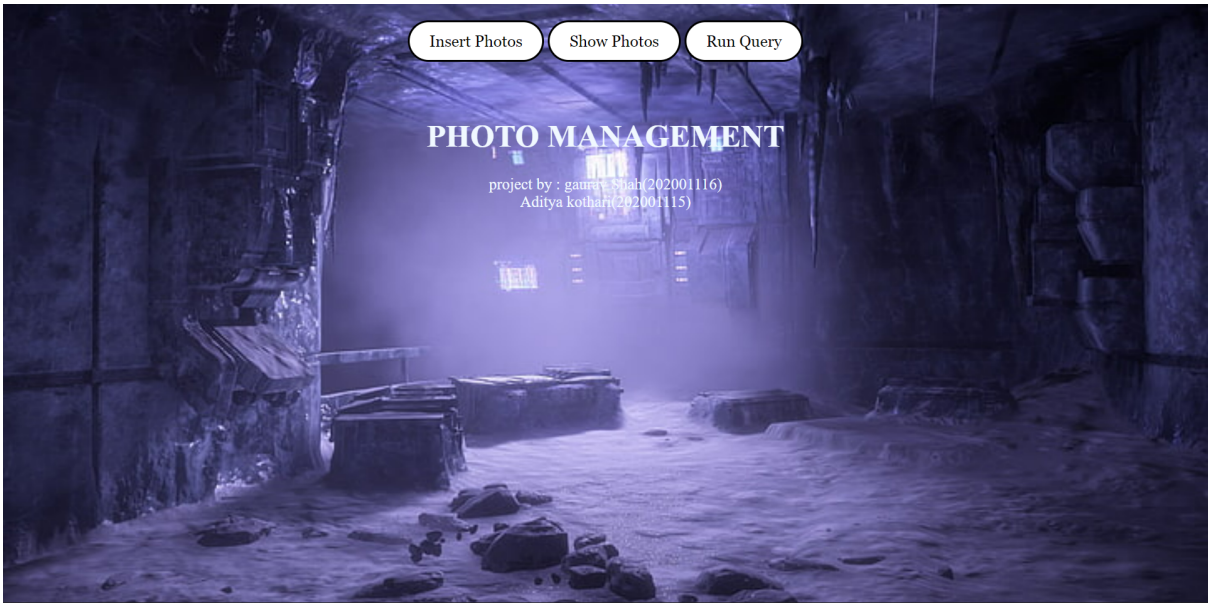
```
def HomePage(request):
    showall=photo.objects.all()
    return render(request,'HomePage.html',{"data":showall})



def insertphoto(request):
    if request.method=="POST":
        if request.POST.get('photo_id') and request.POST.get('date')
and request.POST.get('time') and request.POST.get('location') and
request.POST.get('size') and request.POST.get('tag_color') and
request.POST.get('user_id'):
            saverecord=photo()
            saverecord.photo_id=request.POST.get('photo_id')
            saverecord.date=request.POST.get('date')
            saverecord.time=request.POST.get('time')
            saverecord.location=request.POST.get('location')
            saverecord.size=request.POST.get('size')
            saverecord.tag_color=request.POST.get('tag_color')
            saverecord.user_id=request.POST.get('user_id')

            allval=photo.objects.all()

            for i in allval:
                if int(i.photo_id)==int(request.POST.get('photo_id')):
                    messages.warning(request,'photo already
exists....!');
                    return render(request,'insertphoto.html')
```

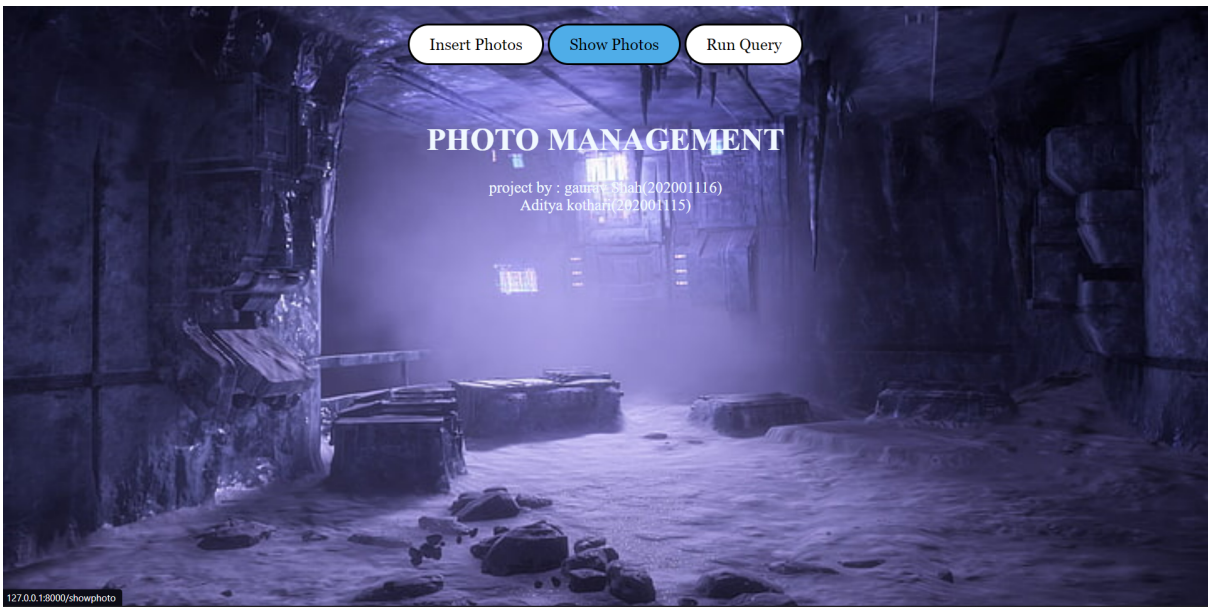
```
            saverecord.save()
            messages.success(request,'photo is saved succesfully!!')
            return render(request,'insertphoto.html')
    else:
            return render(request,'insertphoto.html')



def showphoto(request):
    showall=photo.objects.all()
    return render(request,'showphoto.html',{"data":showall})


def editphoto(request,id):
    editpic=photo.objects.get(photo_id=id)
    return render(request,'editphoto.html',{"photo":editpic})


def updatephoto(request,id):
    Updatephoto=photo.objects.get(photo_id=id)
    form=photoforms(request.POST,instance=Updatephoto)
    if form.is_valid():
        form.save()
        messages.success(request,'Photo updated successfully')
        return render(request,'editphoto.html',{"photo":Updatephoto})


def deletephoto(request,id):
    delphoto=photo.objects.get(photo_id=id)
    delphoto.delete()
    showdata=photo.objects.all()
    return render(request,'showphoto.html',{"data":showdata})
```

```
def runascphoto(request):
    raw_query = "select * from photo_management.photo order by date
,time"
    cursor = connection.cursor()
    cursor.execute(raw_query)
    alldata=cursor.fetchall()
    return render(request,'sortasc.html',{'data':alldata})


def rundescphoto(request):
    raw_query = "select * from photo_management.photo order by date
desc ,time desc "
    cursor = connection.cursor()
    cursor.execute(raw_query)
    alldata=cursor.fetchall()
    return render(request,'sortdesc.html',{'data':alldata})


def sortasc(request):
    raw_query = "select * from photo_management.photo order by date
,time"
    cursor = connection.cursor()
    cursor.execute(raw_query)
    alldata=cursor.fetchall()
    return render(request,'sortasc.html',{'data':alldata})


def sortdesc(request):
    raw_query = "select * from photo_management.photo order by date
desc ,time desc "
    cursor = connection.cursor()
```

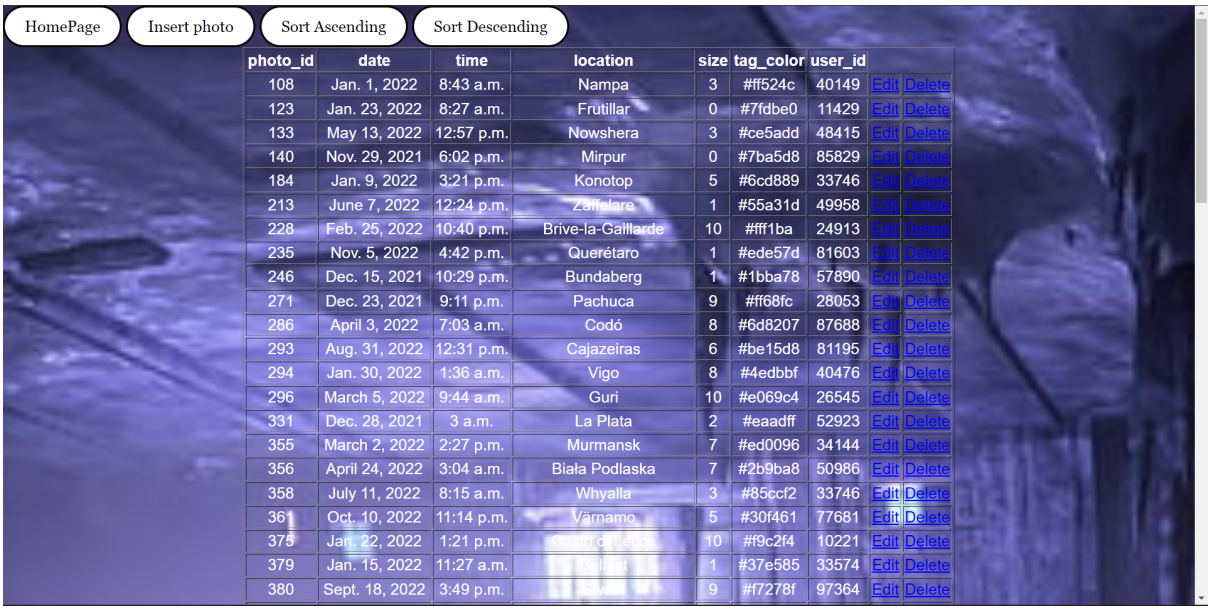
```
    cursor.execute(raw_query)

    alldata=cursor.fetchall()

    return render(request,'sortdesc.html',{'data':alldata})


def runquery(request):

    raw_query = "select * from photo_management.photo where
photo.user_id=33574"

    cursor = connection.cursor()

    cursor.execute(raw_query)

    alldata=cursor.fetchall()

    return render(request,'runquery.html',{'data':alldata})
```

Website Screenshots :



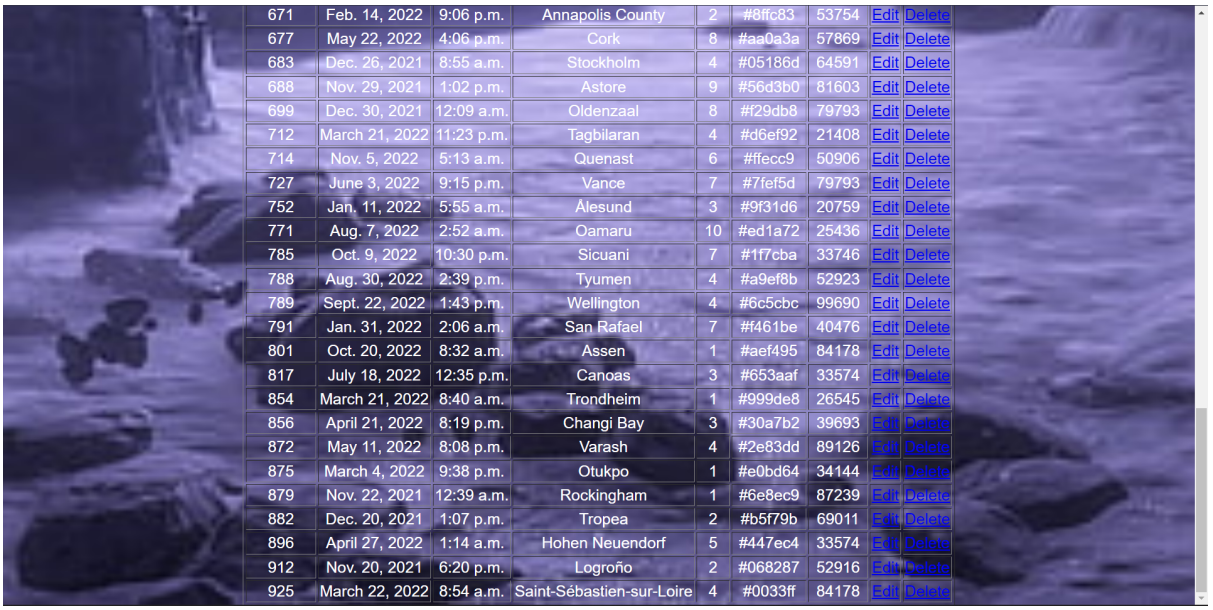


selecting “show photos” :



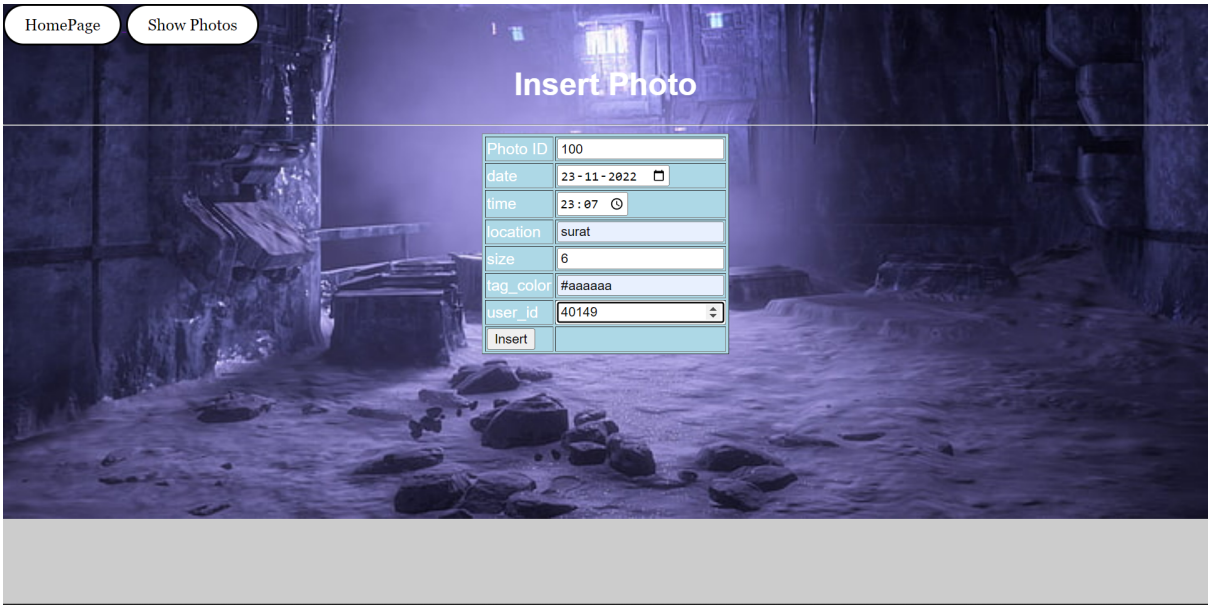
photo_id	date	time	location	size	tag_color	user_id		
108	Jan. 1, 2022	8:43 a.m.	Nampa	3	#ff524c	40149	Edit	Delete
123	Jan. 23, 2022	8:27 a.m.	Frutillar	0	#7fdbe0	11429	Edit	Delete
133	May 13, 2022	12:57 p.m.	Nowshera	3	#ce5add	48415	Edit	Delete
140	Nov. 29, 2021	6:02 p.m.	Mirpur	0	#7ba5d8	85829	Edit	Delete
184	Jan. 9, 2022	3:21 p.m.	Konotop	5	#6cd889	33746	Edit	Delete
213	June 7, 2022	12:24 p.m.	Zaffelare	1	#55a31d	49958	Edit	Delete
228	Feb. 25, 2022	10:40 p.m.	Brive-la-Gaillarde	10	#fff1ba	24913	Edit	Delete
235	Nov. 5, 2022	4:42 p.m.	Querétaro	1	#ede57d	81603	Edit	Delete
246	Dec. 15, 2021	10:29 p.m.	Bundaberg	1	#1bba78	57890	Edit	Delete
271	Dec. 23, 2021	9:11 p.m.	Pachuca	9	#ff68fc	28053	Edit	Delete
286	April 3, 2022	7:03 a.m.	Codó	8	#6d8207	87688	Edit	Delete
293	Aug. 31, 2022	12:31 p.m.	Cajazeiras	6	#be15d8	81195	Edit	Delete
294	Jan. 30, 2022	1:36 a.m.	Vigo	8	#4edbbf	40476	Edit	Delete
296	March 5, 2022	9:44 a.m.	Guri	10	#e069c4	26545	Edit	Delete
331	Dec. 28, 2021	3 a.m.	La Plata	2	#eaadff	52923	Edit	Delete
355	March 2, 2022	2:27 p.m.	Murmansk	7	#ed0096	34144	Edit	Delete
356	April 24, 2022	3:04 a.m.	Biała Podlaska	7	#2b9ba8	50986	Edit	Delete
358	July 11, 2022	8:15 a.m.	Whyalla	3	#85ccf2	33746	Edit	Delete
361	Oct. 10, 2022	11:14 p.m.	Värnamo	5	#30f461	77681	Edit	Delete
375	Jan. 22, 2022	1:21 p.m.	[illegible]	10	#f9c2f4	10221	Edit	Delete
379	Jan. 15, 2022	11:27 a.m.	[illegible]	1	#37e585	33574	Edit	Delete
380	Sept. 18, 2022	3:49 p.m.	[illegible]	9	#f7278f	97364	Edit	Delete

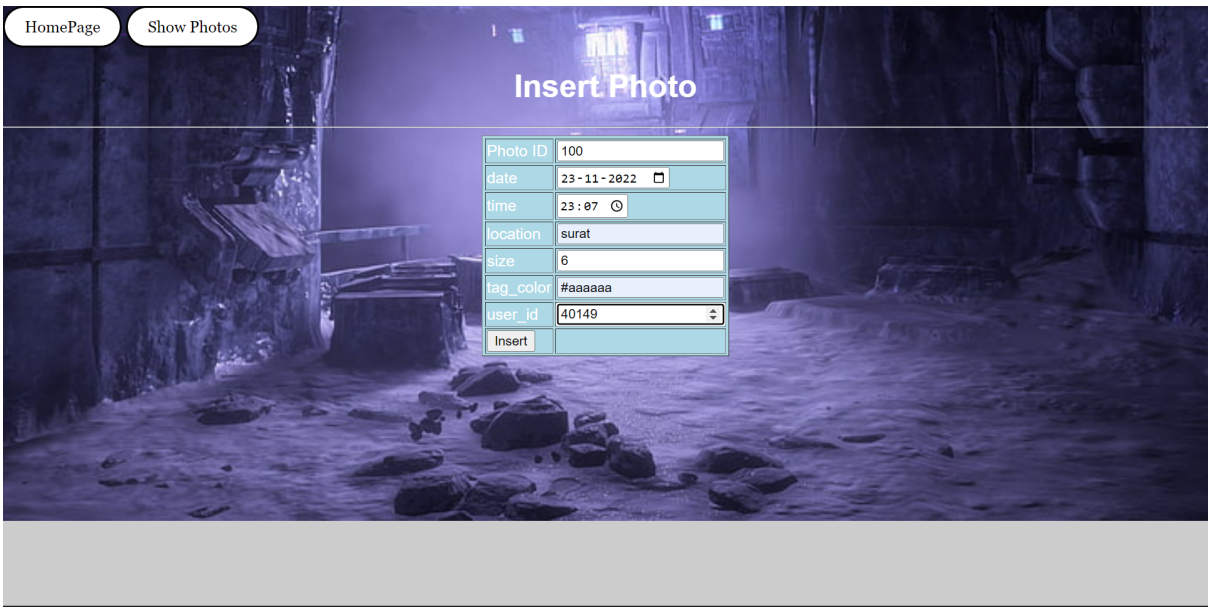
before insert :



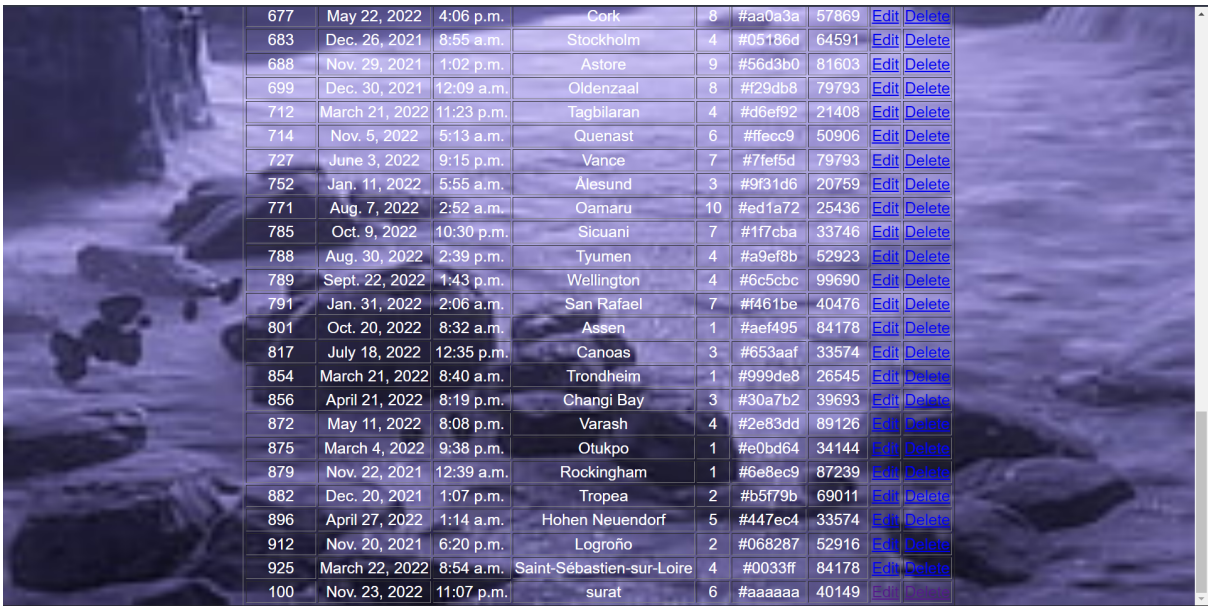
671	Feb. 14, 2022	9:06 p.m.	Annapolis County	2	#8ffc83	53754	Edit Delete
677	May 22, 2022	4:06 p.m.	Cork	8	#aa0a3a	57869	Edit Delete
683	Dec. 26, 2021	8:55 a.m.	Stockholm	4	#05186d	64591	Edit Delete
688	Nov. 29, 2021	1:02 p.m.	Astore	9	#56d3b0	81603	Edit Delete
699	Dec. 30, 2021	12:09 a.m.	Oldenzaal	8	#f29db8	79793	Edit Delete
712	March 21, 2022	11:23 p.m.	Tagbilaran	4	#d6ef92	21408	Edit Delete
714	Nov. 5, 2022	5:13 a.m.	Quenast	6	#ffecc9	50906	Edit Delete
727	June 3, 2022	9:15 p.m.	Vance	7	#7fef5d	79793	Edit Delete
752	Jan. 11, 2022	5:55 a.m.	Ålesund	3	#9f31d6	20759	Edit Delete
771	Aug. 7, 2022	2:52 a.m.	Oamaru	10	#ed1a72	25436	Edit Delete
785	Oct. 9, 2022	10:30 p.m.	Sicuani	7	#1f7cba	33746	Edit Delete
788	Aug. 30, 2022	2:39 p.m.	Tyumen	4	#a9ef8b	52923	Edit Delete
789	Sept. 22, 2022	1:43 p.m.	Wellington	4	#6c5cbc	99690	Edit Delete
791	Jan. 31, 2022	2:06 a.m.	San Rafael	7	#f461be	40476	Edit Delete
801	Oct. 20, 2022	8:32 a.m.	Assen	1	#aef495	84178	Edit Delete
817	July 18, 2022	12:35 p.m.	Canoas	3	#653aaf	33574	Edit Delete
854	March 21, 2022	8:40 a.m.	Trondheim	1	#999de8	26545	Edit Delete
856	April 21, 2022	8:19 p.m.	Changi Bay	3	#30a7b2	39693	Edit Delete
872	May 11, 2022	8:08 p.m.	Varash	4	#2e83dd	89126	Edit Delete
875	March 4, 2022	9:38 p.m.	Otukpo	1	#e0bd64	34144	Edit Delete
879	Nov. 22, 2021	12:39 a.m.	Rockingham	1	#6e8ec9	87239	Edit Delete
882	Dec. 20, 2021	1:07 p.m.	Tropea	2	#b5f79b	69011	Edit Delete
896	April 27, 2022	1:14 a.m.	Hohen Neuendorf	5	#447ec4	33574	Edit Delete
912	Nov. 20, 2021	6:20 p.m.	Logroño	2	#068287	52916	Edit Delete
925	March 22, 2022	8:54 a.m.	Saint-Sébastien-sur-Loire	4	#0033ff	84178	Edit Delete

insert :



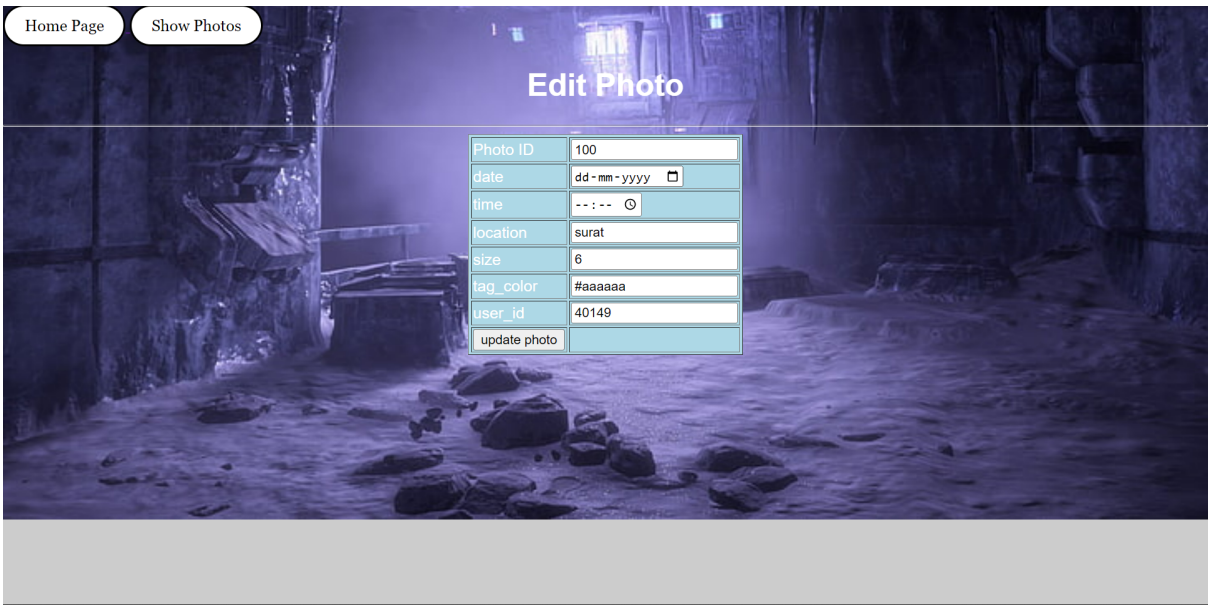


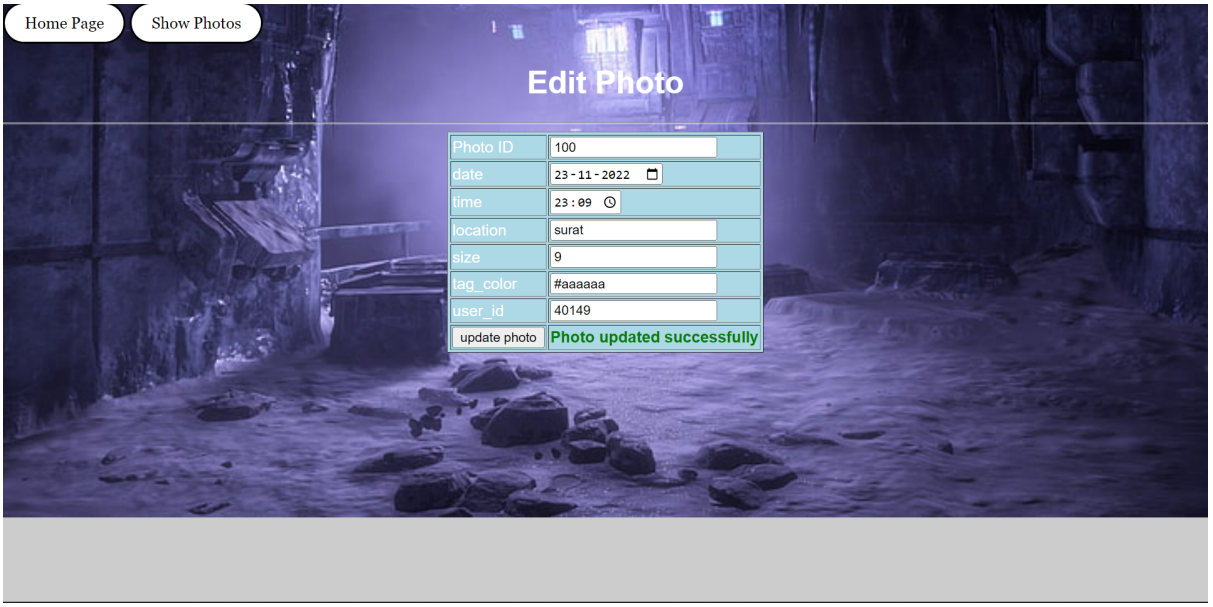
after insert :

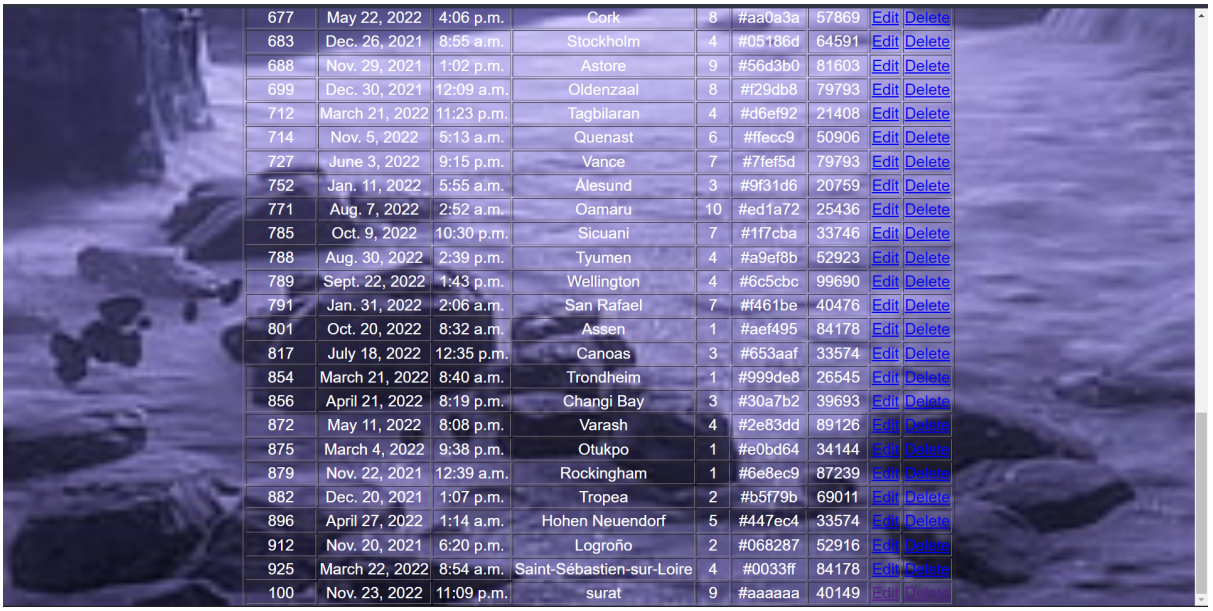


677	May 22, 2022	4:06 p.m.	Cork	8	#aa0a3a	57869	Edit Delete
683	Dec. 26, 2021	8:55 a.m.	Stockholm	4	#05186d	64591	Edit Delete
688	Nov. 29, 2021	1:02 p.m.	Astore	9	#56d3b0	81603	Edit Delete
699	Dec. 30, 2021	12:09 a.m.	Oldenzaal	8	#f29db8	79793	Edit Delete
712	March 21, 2022	11:23 p.m.	Tagbilaran	4	#d6ef92	21408	Edit Delete
714	Nov. 5, 2022	5:13 a.m.	Quenast	6	#ffecc9	50906	Edit Delete
727	June 3, 2022	9:15 p.m.	Vance	7	#7fef5d	79793	Edit Delete
752	Jan. 11, 2022	5:55 a.m.	Ålesund	3	#9f31d6	20759	Edit Delete
771	Aug. 7, 2022	2:52 a.m.	Oamaru	10	#ed1a72	25436	Edit Delete
785	Oct. 9, 2022	10:30 p.m.	Sicuani	7	#1f7cba	33746	Edit Delete
788	Aug. 30, 2022	2:39 p.m.	Tyumen	4	#a9ef8b	52923	Edit Delete
789	Sept. 22, 2022	1:43 p.m.	Wellington	4	#6c5cbc	99690	Edit Delete
791	Jan. 31, 2022	2:06 a.m.	San Rafael	7	#f461be	40476	Edit Delete
801	Oct. 20, 2022	8:32 a.m.	Assen	1	#aef495	84178	Edit Delete
817	July 18, 2022	12:35 p.m.	Canoas	3	#653aaf	33574	Edit Delete
854	March 21, 2022	8:40 a.m.	Trondheim	1	#999de8	26545	Edit Delete
856	April 21, 2022	8:19 p.m.	Changi Bay	3	#30a7b2	39693	Edit Delete
872	May 11, 2022	8:08 p.m.	Varash	4	#2e83dd	89126	Edit Delete
875	March 4, 2022	9:38 p.m.	Otukpo	1	#e0bd64	34144	Edit Delete
879	Nov. 22, 2021	12:39 a.m.	Rockingham	1	#6e8ec9	87239	Edit Delete
882	Dec. 20, 2021	1:07 p.m.	Tropea	2	#b5f79b	69011	Edit Delete
896	April 27, 2022	1:14 a.m.	Hohen Neuendorf	5	#447ec4	33574	Edit Delete
912	Nov. 20, 2021	6:20 p.m.	Logroño	2	#068287	52916	Edit Delete
925	March 22, 2022	8:54 a.m.	Saint-Sébastien-sur-Loire	4	#0033ff	84178	Edit Delete
100	Nov. 23, 2022	11:07 p.m.	surat	6	#aaaaaa	40149	Edit Delete

Editing photos :


Home Page
Show Photos
Edit Photo
Photo ID
100
date
dd-mm-yyyy
time
--:--
location
surat
size
6
tag_color
#aaaaaa
user_id
40149
update photo


Home Page
Show Photos
Edit Photo
Photo ID
100
date
23-11-2022
time
23:09
location
surat
size
9
tag_color
#aaaaaa
user_id
40149
update photo
Photo updated successfully

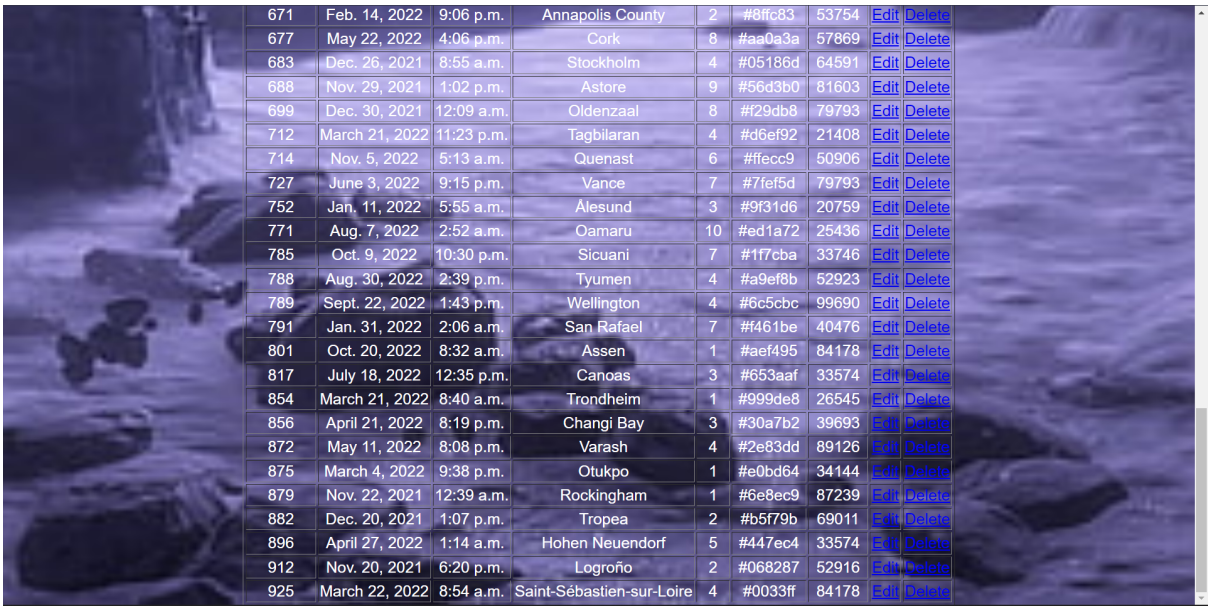


677	May 22, 2022	4:06 p.m.	Cork	8	#aa0a3a	57869	Edit Delete
683	Dec. 26, 2021	8:55 a.m.	Stockholm	4	#05186d	64591	Edit Delete
688	Nov. 29, 2021	1:02 p.m.	Astore	9	#56d3b0	81603	Edit Delete
699	Dec. 30, 2021	12:09 a.m.	Oldenzaal	8	#f29db8	79793	Edit Delete
712	March 21, 2022	11:23 p.m.	Tagbilaran	4	#d6ef92	21408	Edit Delete
714	Nov. 5, 2022	5:13 a.m.	Quenast	6	#ffecc9	50906	Edit Delete
727	June 3, 2022	9:15 p.m.	Vance	7	#7fef5d	79793	Edit Delete
752	Jan. 11, 2022	5:55 a.m.	Ålesund	3	#9f31d6	20759	Edit Delete
771	Aug. 7, 2022	2:52 a.m.	Oamaru	10	#ed1a72	25436	Edit Delete
785	Oct. 9, 2022	10:30 p.m.	Sicuani	7	#1f7cba	33746	Edit Delete
788	Aug. 30, 2022	2:39 p.m.	Tyumen	4	#a9ef8b	52923	Edit Delete
789	Sept. 22, 2022	1:43 p.m.	Wellington	4	#6c5cbc	99690	Edit Delete
791	Jan. 31, 2022	2:06 a.m.	San Rafael	7	#f461be	40476	Edit Delete
801	Oct. 20, 2022	8:32 a.m.	Assen	1	#aef495	84178	Edit Delete
817	July 18, 2022	12:35 p.m.	Canoas	3	#653aaf	33574	Edit Delete
854	March 21, 2022	8:40 a.m.	Trondheim	1	#999de8	26545	Edit Delete
856	April 21, 2022	8:19 p.m.	Changi Bay	3	#30a7b2	39693	Edit Delete
872	May 11, 2022	8:08 p.m.	Varash	4	#2e83dd	89126	Edit Delete
875	March 4, 2022	9:38 p.m.	Otukpo	1	#e0bd64	34144	Edit Delete
879	Nov. 22, 2021	12:39 a.m.	Rockingham	1	#6e8ec9	87239	Edit Delete
882	Dec. 20, 2021	1:07 p.m.	Tropea	2	#b5f79b	69011	Edit Delete
896	April 27, 2022	1:14 a.m.	Hohen Neuendorf	5	#447ec4	33574	Edit Delete
912	Nov. 20, 2021	6:20 p.m.	Logroño	2	#068287	52916	Edit Delete
925	March 22, 2022	8:54 a.m.	Saint-Sébastien-sur-Loire	4	#0033ff	84178	Edit Delete
100	Nov. 23, 2022	11:09 p.m.	surat	9	#aaaaaa	40149	Edit Delete

Deleting photo :

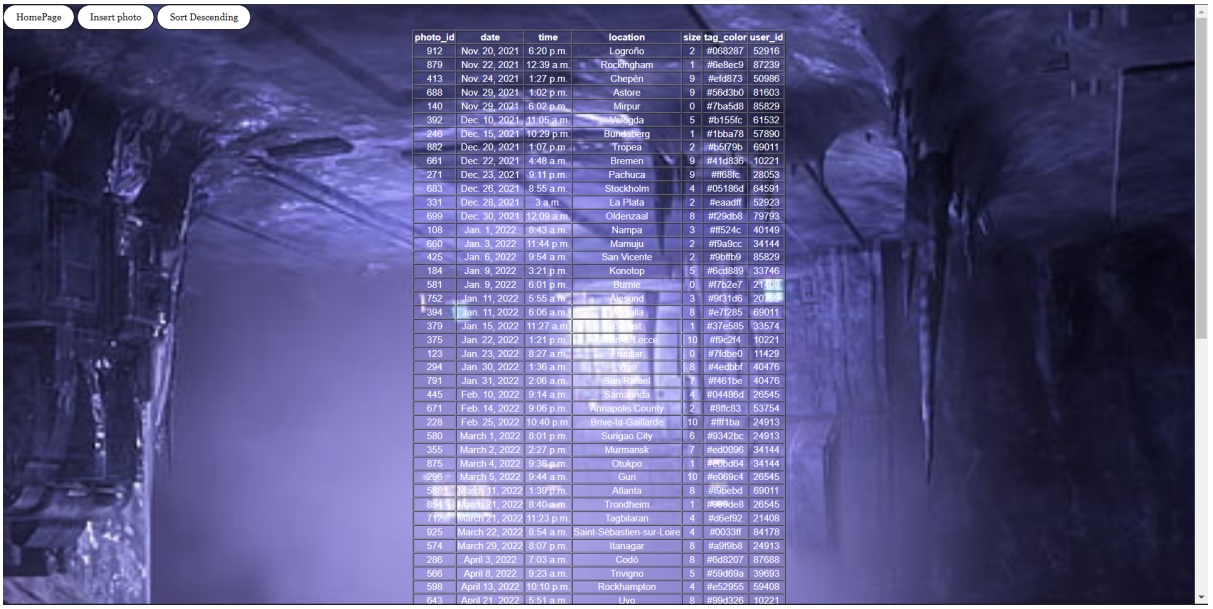
677	May 22, 2022				3a	57869	Edit Delete
683	Dec. 26, 2021				6d	64591	Edit Delete
688	Nov. 29, 2021				b0	81603	Edit Delete
699	Dec. 30, 2021	12:09 a.m.	Oldenzaal	8	#f29db8	79793	Edit Delete
712	March 21, 2022	11:23 p.m.	Tagbilaran	4	#d6ef92	21408	Edit Delete
714	Nov. 5, 2022	5:13 a.m.	Quenast	6	#ffecc9	50906	Edit Delete
727	June 3, 2022	9:15 p.m.	Vance	7	#7fef5d	79793	Edit Delete
752	Jan. 11, 2022	5:55 a.m.	Ålesund	3	#9f31d6	20759	Edit Delete
771	Aug. 7, 2022	2:52 a.m.	Oamaru	10	#ed1a72	25436	Edit Delete
785	Oct. 9, 2022	10:30 p.m.	Sicuani	7	#1f7cba	33746	Edit Delete
788	Aug. 30, 2022	2:39 p.m.	Tyumen	4	#a9ef8b	52923	Edit Delete
789	Sept. 22, 2022	1:43 p.m.	Wellington	4	#6c5cbc	99690	Edit Delete
791	Jan. 31, 2022	2:06 a.m.	San Rafael	7	#f461be	40476	Edit Delete
801	Oct. 20, 2022	8:32 a.m.	Assen	1	#aef495	84178	Edit Delete
817	July 18, 2022	12:35 p.m.	Canoas	3	#653aaf	33574	Edit Delete
854	March 21, 2022	8:40 a.m.	Trondheim	1	#999de8	26545	Edit Delete
856	April 21, 2022	8:19 p.m.	Changi Bay	3	#30a7b2	39693	Edit Delete
872	May 11, 2022	8:08 p.m.	Varash	4	#2e83dd	89126	Edit Delete
875	March 4, 2022	9:38 p.m.	Otukpo	1	#e0bd64	34144	Edit Delete
879	Nov. 22, 2021	12:39 a.m.	Rockingham	1	#6e8ec9	87239	Edit Delete
882	Dec. 20, 2021	1:07 p.m.	Tropea	2	#b5f79b	69011	Edit Delete
896	April 27, 2022	1:14 a.m.	Hohen Neuendorf	5	#447ec4	33574	Edit Delete
912	Nov. 20, 2021	6:20 p.m.	Logroño	2	#068287	52916	Edit Delete
925	March 22, 2022	8:54 a.m.	Saint-Sébastien-sur-Loire	4	#0033ff	84178	Edit Delete
100	Nov. 23, 2022	11:09 p.m.	surat	9	#aaaaaa	40149	Edit Delete

After deletion :



671	Feb. 14, 2022	9:06 p.m.	Annapolis County	2	#8ffc83	53754	Edit	Delete
677	May 22, 2022	4:06 p.m.	Cork	8	#aa0a3a	57869	Edit	Delete
683	Dec. 26, 2021	8:55 a.m.	Stockholm	4	#05186d	64591	Edit	Delete
688	Nov. 29, 2021	1:02 p.m.	Astore	9	#56d3b0	81603	Edit	Delete
699	Dec. 30, 2021	12:09 a.m.	Oldenzaal	8	#f29db8	79793	Edit	Delete
712	March 21, 2022	11:23 p.m.	Tagbilaran	4	#d6ef92	21408	Edit	Delete
714	Nov. 5, 2022	5:13 a.m.	Quenast	6	#ffecc9	50906	Edit	Delete
727	June 3, 2022	9:15 p.m.	Vance	7	#7fef5d	79793	Edit	Delete
752	Jan. 11, 2022	5:55 a.m.	Ålesund	3	#9f31d6	20759	Edit	Delete
771	Aug. 7, 2022	2:52 a.m.	Oamaru	10	#ed1a72	25436	Edit	Delete
785	Oct. 9, 2022	10:30 p.m.	Sicuani	7	#1f7cba	33746	Edit	Delete
788	Aug. 30, 2022	2:39 p.m.	Tyumen	4	#a9ef8b	52923	Edit	Delete
789	Sept. 22, 2022	1:43 p.m.	Wellington	4	#6c5cbc	99690	Edit	Delete
791	Jan. 31, 2022	2:06 a.m.	San Rafael	7	#f461be	40476	Edit	Delete
801	Oct. 20, 2022	8:32 a.m.	Assen	1	#aef495	84178	Edit	Delete
817	July 18, 2022	12:35 p.m.	Canoas	3	#653aaf	33574	Edit	Delete
854	March 21, 2022	8:40 a.m.	Trondheim	1	#999de8	26545	Edit	Delete
856	April 21, 2022	8:19 p.m.	Changi Bay	3	#30a7b2	39693	Edit	Delete
872	May 11, 2022	8:08 p.m.	Varash	4	#2e83dd	89126	Edit	Delete
875	March 4, 2022	9:38 p.m.	Otukpo	1	#e0bd64	34144	Edit	Delete
879	Nov. 22, 2021	12:39 a.m.	Rockingham	1	#6e8ec9	87239	Edit	Delete
882	Dec. 20, 2021	1:07 p.m.	Tropea	2	#b5f79b	69011	Edit	Delete
896	April 27, 2022	1:14 a.m.	Hohen Neuendorf	5	#447ec4	33574	Edit	Delete
912	Nov. 20, 2021	6:20 p.m.	Logroño	2	#068287	52916	Edit	Delete
925	March 22, 2022	8:54 a.m.	Saint-Sébastien-sur-Loire	4	#0033ff	84178	Edit	Delete

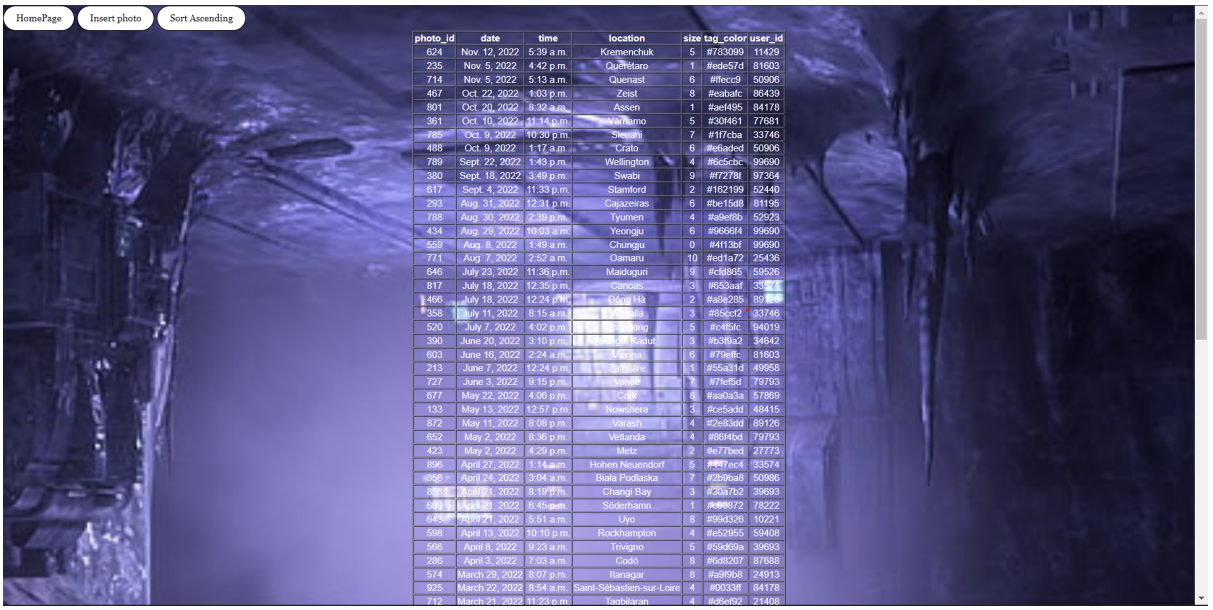
After Applying the query for order by ascending by date and time :



HomePage | Insert photo | Sort Descending

photo_id	date	time	location	size	tag_color	user_id
912	Nov. 20, 2021	6:20 p.m.	Logroño	2	#068287	52916
879	Nov. 22, 2021	12:39 a.m.	Rockingham	1	#6e8ec9	87239
413	Nov. 24, 2021	1:27 p.m.	Chepén	9	#efd873	50986
688	Nov. 29, 2021	1:02 p.m.	Astore	9	#56d3b0	81603
140	Nov. 29, 2021	6:02 p.m.	Mirpur	0	#7ba5d8	85829
392	Dec. 10, 2021	11:05 a.m.	Vologda	5	#b155fc	61532
246	Dec. 15, 2021	10:29 p.m.	Bundaberg	1	#1bba78	57890
882	Dec. 20, 2021	1:07 p.m.	Tropea	2	#b5f79b	69011
661	Dec. 22, 2021	4:48 a.m.	Bremen	9	#41d838	10221
271	Dec. 23, 2021	9:11 p.m.	Pachuca	9	#ff68fc	28053
683	Dec. 26, 2021	8:55 a.m.	Stockholm	4	#05186d	64591
331	Dec. 28, 2021	3 a.m.	La Plata	2	#eaadff	52923
699	Dec. 30, 2021	12:09 a.m.	Oldenzaal	8	#f29db8	79793
108	Jan. 1, 2022	8:43 a.m.	Nampa	3	#ff524c	40149
660	Jan. 3, 2022	11:44 p.m.	Mamuju	2	#f9a9cc	34144
425	Jan. 6, 2022	9:54 a.m.	San Vicente	2	#9bffb9	85829
184	Jan. 9, 2022	3:21 p.m.	Konotop	5	#6cd889	33746
581	Jan. 9, 2022	6:01 p.m.	Burnie	0	#f7b2e7	21408
752	Jan. 11, 2022	5:55 a.m.	Ålesund	3	#9f31d6	20759
394	Jan. 11, 2022	6:06 a.m.	[illegible]	8	#e7f285	69011
379	Jan. 15, 2022	11:27 a.m.	[illegible]	1	#37e585	33574
375	Jan. 22, 2022	1:21 p.m.	[illegible]	10	#f9c2f4	10221
123	Jan. 23, 2022	8:27 a.m.	[illegible]	0	#7fd8e0	11429
284	Jan. 30, 2022	1:36 a.m.	[illegible]	8	#4edbbf	40476
791	Jan. 31, 2022	2:06 a.m.	San Rafael	7	#f461be	40476
445	Feb. 10, 2022	9:14 a.m.	Samandağ	4	#0448d	26545
671	Feb. 14, 2022	9:06 p.m.	Annapolis County	2	#8ffc83	53754
228	Feb. 25, 2022	10:40 p.m.	Brive-la-Gaillarde	10	#ff1fba	24913
580	March 1, 2022	8:01 p.m.	Surigao City	6	#9342bc	24913
355	March 2, 2022	2:27 p.m.	Murmansk	7	#ed0096	34144
875	March 4, 2022	9:38 p.m.	Otukpo	1	#e0bd64	34144
296	March 5, 2022	9:44 a.m.	Guri	10	#c069c4	26545
583	March 11, 2022	1:39 p.m.	Atlanta	8	#f6cebd	69011
854	March 21, 2022	8:40 a.m.	Trondheim	1	#999de8	26545
712	March 21, 2022	11:23 p.m.	Tagbilaran	4	#d6ef92	21408
925	March 22, 2022	8:54 a.m.	Saint-Sébastien-sur-Loire	4	#0033ff	84178
574	March 29, 2022	8:07 p.m.	Itanagar	8	#a9f9b8	24913
286	April 3, 2022	7:03 a.m.	Codó	8	#6d8207	87688
566	April 8, 2022	9:23 a.m.	Trivigno	5	#59d69a	39693
598	April 13, 2022	10:10 p.m.	Rockhampton	4	#e52955	59408
643	April 21, 2022	5:51 a.m.	Uvo	8	#99d326	10221

After applying the query for order by descending by date and time :



photo_id	date	time	location	size	tag_color	user_id
624	Nov. 12, 2022	5:39 a.m.	Kremenchuk	5	#783099	11429
235	Nov. 5, 2022	4:42 p.m.	Querétaro	1	#ede57d	81603
714	Nov. 5, 2022	5:13 a.m.	Quenast	6	#ffecc9	50906
467	Oct. 22, 2022	1:03 p.m.	Zeist	8	#eabafc	86439
801	Oct. 20, 2022	8:32 a.m.	Assen	1	#aef495	84178
361	Oct. 10, 2022	11:14 p.m.	Vanlamo	5	#30f461	77681
785	Oct. 9, 2022	10:30 p.m.	Sleuani	7	#1f7cba	33746
488	Oct. 9, 2022	1:17 a.m.	Crato	6	#e8aded	50906
789	Sept. 22, 2022	1:43 p.m.	Wellington	4	#6c5cbc	99690
380	Sept. 18, 2022	3:49 p.m.	Swabi	9	#f72f8f	97364
617	Sept. 4, 2022	11:33 p.m.	Stamford	2	#162199	52440
293	Aug. 31, 2022	12:31 p.m.	Cajazeiras	6	#be15d8	81195
788	Aug. 30, 2022	2:39 p.m.	Tyumen	4	#a9ef8b	52923
434	Aug. 29, 2022	10:03 a.m.	Yeongju	6	#9666f4	99690
559	Aug. 8, 2022	1:49 a.m.	Chungju	0	#4f13bf	99690
771	Aug. 7, 2022	2:52 a.m.	Oamaru	10	#ed1a72	25436
646	July 23, 2022	11:36 p.m.	Maiduguri	9	#cfd8d5	59526
817	July 18, 2022	12:35 p.m.	Canoas	3	#653aaf	33574
466	July 18, 2022	12:24 p.m.	Đông Hà	2	#a8e285	89126
358	July 11, 2022	8:15 a.m.	Tulla	3	#95ecf2	33746
520	July 7, 2022	4:02 p.m.	Lilong	5	#c4f5fc	94019
390	June 20, 2022	3:10 p.m.	Kadut	3	#b3f9a2	34642
603	June 16, 2022	2:24 a.m.	Verona	6	#79effc	81603
213	June 7, 2022	12:24 p.m.	Zamfara	1	#55a31d	49958
727	June 3, 2022	9:15 p.m.	Vanlo	7	#7fef5d	79793
677	May 22, 2022	4:06 p.m.	Cox	8	#aa0a3a	57869
133	May 13, 2022	12:57 p.m.	Nowshera	3	#ce5add	48415
872	May 11, 2022	8:08 p.m.	Varash	4	#2e83dd	89126
652	May 2, 2022	8:36 p.m.	Vellanda	4	#86f4bd	79793
423	May 2, 2022	4:29 p.m.	Metz	2	#e77bed	27773
896	April 27, 2022	1:14 a.m.	Hohen Neuendorf	5	#4f7ec4	33574
856	April 24, 2022	3:04 a.m.	Biała Podlaska	7	#2b9ba8	50986
852	April 21, 2022	8:19 p.m.	Changi Bay	3	#30a7b2	39693
649	April 21, 2022	6:45 pm	Söderhamn	1	#e00872	78222
643	April 21, 2022	5:51 a.m.	Uyo	8	#99d326	10221
598	April 13, 2022	10:10 p.m.	Rockhampton	4	#e52955	59408
566	April 8, 2022	9:23 a.m.	Trivigno	5	#59d69a	39693
286	April 3, 2022	7:03 a.m.	Codó	8	#6d8207	87688
574	March 29, 2022	8:07 p.m.	Itanagar	8	#a9f9b8	24913
925	March 22, 2022	8:54 a.m.	Saint-Sébastien-sur-Loire	4	#0033ff	84178
712	March 21, 2022	11:23 p.m.	Tagbilaran	4	#d6ef82	21408