1- Photobook

A web based photo sharing application designed and implemented using PostgreSQL engine and Python Flask.

Data stored in the system includes : Users, Albums, Friends, Photos, Tags, Comments and Likes.
The system support the following use cases:

1. **User management:** becoming a registered user, adding and listing friends, tracking top 10 users.
2. **Album and Photo management:** browse photos as a registered user or visitor, registered users can upload photos and create albums, and delete albums.
3. **Tag Management:** Viewing your/all Photos by Tag Name, Viewing the Most Popular Tags, conjunctive tag queries photo search
4. **Comments:** post a comment, like a photo, search for users who made a comment
5. **Recommendations:** recommend friends of friends, suggest photos users may also like based on common tags.

2- Schema Changes Since April 17th

1. Users
    a. Password length changed to 72 since hashed value is longer | password VARCHAR(72) NULL

2. Albums
    a. Datetime is set to the default timestamp. Users are no longer required to insert time. datetime TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP

3. Photos
    a. Caption is increased from 100 characters to 250 caption VARCHAR(250) NULL
    b. Datetime is set to the default timestamp. Users are no longer required to insert time. datetime TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
    c. Using path URL instead of BYTEA for storing photos. path TEXT NOT NULL,

4. Friends Table - Check legit friendship trigger
    a. A user cannot add himself as a friend (previously a user could - we fixed that)

```
CREATE OR REPLACE FUNCTION check_legit_friend_relationship()
RETURNS TRIGGER AS $$
BEGIN
  IF NOT EXISTS (SELECT 1 FROM users WHERE user_id = NEW.user_id AND
is_visitor = FALSE) OR
     NOT EXISTS (SELECT 1 FROM users WHERE user_id = NEW.friend_id AND
is_visitor = FALSE) OR
     NOT EXISTS (SELECT 1 WHERE NEW.user_id <> NEW.friend_id) THEN
      RAISE EXCEPTION 'Both users must be non-visitors and not the same user.';
  END IF;
  RETURN NEW;
END;
```

5. Comments Table
    a. Datetime is set to the default timestamp. Users are no longer required to insert time. datetime TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
6. Likes Table
    a. Datetime is set to the default timestamp. Users are no longer required to insert time. datetime TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP

3 - Development Process

1. Designing Database Schema
2. Implementation of Database schema in Flask
3. Building APIs
4. Building Frontend pages
5. Connecting Frontend to Backend APIs

❖ Design

The details in designing the database including the ER diagram, Schema, SQLs and report 1 can be found here:
https://github.com/ayushpandeynp/photobook/tree/master/database

❖ Code Structure

Server

- Backend - api calls to DB
- Frontend - routing for pages
- Static - javascript files for interactivity and css files for styling
- Templates - all html that displays webpage

❖ Libraries Used

a. **psycopg2** as PostgreSQL database adapter for the Python
b. **Bcrypt** for modern password hashing
c. **Axios** for asynchronous calls. Axios is a promise based HTTP client.
d. **fontawesome** for styling icons.
e. **JWT** for authentication.
f. **UUID** for unique identifiers generation.
g. **render_template** for displaying the html pages.

❖ Building APIs

Public – doesn't require token authentication
User scope – requires token authentication

The SQL queries were wrapped under api call
Users

1. Login (/login)
2. Register (/signup , /visitor_signup)
   visitor signup allows most of the data fields to be null. This is needed when a visitor wants to add a comment or like.

Friend

1. Add a friend -/add-friend
2. List all friends of a user - /list-friends'
3. Search for a friend by name - /search-users
4. Friend Recommendation - '/friend-recommendation'

Albums & Photos & Tags

1. Create an album (user scope) - '/create-album'
2. List all albums (user scope)) - /list-albums'
3. List all albums (public scope) - /list-albums-public'
4. List all photos, by album (public) - '/list-photos-by-album'
5. Delete album (user scope) - '/delete-album',
6. Delete photo from an album (user scope) -
7. Photo search by tags (public) - '/photos-with-tags', /photos-with-tags-user
8. You may also like - /you-may-also-like'
9. Top 10 users who make the largest contribution (comments + photos count) - '/top-contributors',
10. Most popular tags (public) '/popular-tags'

Comments & Likes

1. Add comment to a photo (user or visitor – should exist on the users table) '/add-comment',
2. Like a photo (user or visitor) '/like-photo',
3. Total likes of a photo, along with their associated users who liked (public) /photo-likes'
4. All comments of a photo, along with their associated users who commented (public) '/photo-comments'
5. Comment search, returns names of users, and photos with matching comments (public) '/comment-search'

❖ Constraints Implemented

1. In signing up , DOB information can be optional.
2. If the user already exists in the database with the same email address an error message is produced.
3. All photos and albums are made public and registered and non-registered users can browse it.
4. Only registered users can upload photos
5. Users can search between their photos or all photos by a toggle.
6. Top contributors are just registered users - we filter out the visitors.
7. Deleting photos and albums. Deleting the album should delete all photos in it.
8. Users can only modify or delete their photos
9. Users search through the photos by specifying conjunctive tag queries
10. Both registered and unregistered users can leave comments
11. Users cannot leave comments on their own photos
12. Search for users whose comments match exactly the entered comment. If multiple users, return in order of the number of comments matched.
13. There is a limit of 10 on popular tags. Clicking on a tag opens the photos associated with it.
14. Any user should be able to see how many likes a photo has and the names of the users who liked this photo.
15. Extra - Visitors or registered users cannot like a photo more than once

❖ Assumptions

1. Friend relationship is undirected which means that if User 1 add User 2 as a friend, then User 2's friend is User 1 and vise versa
2. Deleting an album will delete all data associated with it, photos and even comments.
3. In the You May Also Like functionality:
    a. We order based on conciseness = (the total matched of top popular tags/ number of tags belonging to the photo), the higher the value the higher it is ranked.