SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)
Constituent of Symbiosis International (Deemed University), Pune
(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of India)
Re-Accredited by NAAC with 'A++' Grade

A DCDSL PROJECT REPORT

ON

SQL-Based Analysis of Social Network Data

Submitted By

Alabhya Sharma	23070126010
Amannyu Gondkar	23070126011
Anusri Kadam	23070126016

UNDER THE GUIDANCE OF

Prof. Archana Chaudhari

Assistant Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

A.Y. 2024-25

SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)

Constituent of Symbiosis International (Deemed University), Pune

(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of India)

Re-Accredited by NAAC with 'A++' Grade

वसुधैव कुटुम्बकम्

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

.

CERTIFICATE

This is to certify that the DCDSL Project work entitled **SQL-Based Analysis of Social Network Data** is carried out by **Anusri Kadam, Amannyu Gondkar and Alabhya Sharma** in **Artificial Intelligence & Machine Learning**, Symbiosis International (Deemed University), Pune during the academic year 2024-2025.

Dr.Archana Chaudari, Dr.Kalyani Kadam
Name and Signature of the Guide

Dr. Shruti Patil
Head, Department of AI&ML

TABLE OF CONTENTS

Introduction

In today's digital age, social media has revolutionized the way people communicate, share ideas, and build communities worldwide. It has become a dynamic platform where conversations unfold, trends emerge, and global connections are forged, producing a vast amount of data every second. Analyzing this data opens a window into society's behaviors, interests, and sentiments, revealing powerful insights that can guide businesses, influencers, and everyday users alike.

This project aims to design and implement a database system for a social media application, focused on capturing user interactions and content-sharing dynamics in a structured, efficient way. By creating an organized framework for storing information about users, posts, comments, likes, tags, and trends, this database will enable deeper insights into social media activity.

For businesses and organizations, understanding these social interactions is invaluable. It allows them to respond to emerging trends, adapt marketing strategies in real-time, and enhance user engagement through targeted interactions. Furthermore, by examining patterns in user-generated content, companies can stay attuned to customer sentiment and optimize their online presence effectively.

The system developed in this project provides a foundation for analyzing user engagement and content popularity. By capturing data on user relationships, interactions with content, and activity patterns, this database is built to support applications seeking real-time insights into social trends. It represents a critical tool for interpreting the fast-evolving landscape of social media, paving the way for more meaningful connections and informed decision-making in a world driven by online interactions.

Problem Statement

This project aims to create a scalable database system alongside an intuitive UI that efficiently stores, organizes, and analyzes social media user activity, enabling better insights into user engagement, trending topics, and overall platform usage.

System architecture and Modules

System architecture

1. **Presentation Layer**:
    - This layer provides the user interface, built with Python's Tkinter. It enables users to interact with the system, perform CRUD operations on social media data, and view user activity and content.
    - Features include a graphical interface for creating accounts, uploading photos, commenting, liking posts, and managing hashtags.
2. **Business Logic Layer**:
    - This layer handles the core functionality, including data validation, user activity processing, and enforcing rules around user interactions (such as follow relationships and tag associations).
    - This layer processes commands from the Presentation Layer, ensuring that user actions comply with business rules, such as preventing duplicate follows or managing valid content tagging.
3. **Data Access Layer**:
    - This layer acts as an intermediary between the business logic and the database. It performs CRUD operations, sending requests to the database and retrieving data to pass back to the Business Logic Layer.
    - It abstracts direct database interactions, making the system more maintainable and secure.
4. **Database Layer**:
    - This layer consists of the MySQL database where all data is stored, including tables for users, photos, comments, likes, tags, and hashtags.
    - It manages data persistence, ensuring that information on user interactions, content, and tagging is efficiently stored and accessible for analysis and reporting.

Modules

1. **User Management Module**:
    - **Purpose**: Enables user authentication and account creation, allowing users to interact with the application and manage their profile.
    - **Features**:
        - **Account Creation**: Users can register by providing a username and other details.

- **Follow Management**: Users can follow/unfollow each other, stored in the follows table to establish user connections.

2. **Content Management Module**:
   - **Purpose**: Allows users to upload, tag, and organize media content.
   - **Features**:
     - **Photo Upload**: Users can upload photos, which are stored in the photos table with user and timestamp data.
     - **Tagging and Hashtags**: Users can tag photos, managed through the tags and photo_tags tables, as well as follow specific hashtags for curated content feeds.
     - **Commenting and Liking**: Users can comment on and like photos, linking comments and likes to photos and users in the respective tables.
3. **Data Retrieval and Display Module**:
   - **Purpose**: Facilitates the retrieval and viewing of social media data for users to browse and interact with content.
   - **Features**:
     - **Search and Filter**: Users can search for specific users, hashtags, or posts.
     - **Trends Analysis**: Enables viewing of popular tags and posts, using data aggregation on likes and hashtags.
4. **Error Handling and Notifications Module**:
   - **Purpose**: Ensures smooth user experience by providing clear notifications and handling errors during data entry and interactions.
   - **Features**:
     - **Input Validation**: Prevents invalid data entries by checking inputs for fields such as username and image URLs.
     - **Notifications**: Displays success and error messages for actions like creating posts or following a user (e.g., "Photo uploaded successfully" or "Follow request failed").
     - **Real-Time Feedback**: Immediate error notifications guide users to correct any input issues, helping them interact effectively with the application.

Functional Requirements

1. User Account and Relationship Management

- **Purpose**: Allows users to create accounts, manage their profiles, and establish connections with other users.
- **Features**:
    - **Account Creation and Authentication**:
        - Users can register accounts and log in with unique usernames. User details are stored in the users table.
    - **Follow/Unfollow Functionality**:
        - Users can follow and unfollow each other, establishing relationships stored in the follows table.
    - **Login Tracking**:
        - User login information, including IP addresses and login times, is stored in the login table for tracking access patterns.

2. Content Management

- **Purpose**: Enables users to upload, tag, and interact with content on the platform.
- **Features**:
    - **Photo Upload**:
        - Users can upload photos, which are stored in the photos table, with fields for image URLs, upload timestamps, and associated user IDs.
    - **Commenting on Photos**:
        - Users can add comments to photos, which are recorded in the comments table with links to the photo and user who commented.
    - **Liking Photos**:
        - Users can like photos, which are tracked in the likes table, allowing for a record of each user's interaction with each photo.

3. Tag and Hashtag Management

- **Purpose**: Allows users to categorize and organize content for easier discovery and interaction.
- **Features**:
    - **Tagging Photos**:
        - Users can add tags to photos, which are stored in the tags and photo_tags tables, facilitating better content categorization and searchability.

- **Following Hashtags**:
  - Users can follow specific hashtags of interest, which is managed in the hashtag_follow table, enabling users to view content tagged with their favorite topics.
- **Organizing Hashtags**:
  - Hashtags are maintained in the hashtags table with timestamps for easy management and usage tracking across the platform.

4. Data Retrieval and Feed Display

- **Purpose**: Enables users to access content based on their interests, connections, and trending topics.
- **Features**:
  - **User Feed**:
    - Displays a personalized feed for users, showing content from followed users and tags they are interested in.
  - **Content Discovery**:
    - Users can search for specific tags, users, or posts and filter content based on tags or upload dates.
  - **Trending Topics**:
    - Aggregates popular tags and posts, giving users insights into trending content by analyzing likes and hashtag data.

5. Error Handling and Notifications

- **Purpose**: Ensures user interactions are smooth by providing feedback on actions and guiding users through the application.
- **Features**:
  - **Input Validation**:
    - Ensures only valid data is entered, checking fields such as usernames, URLs, and text entries for comments and tags.
  - **User Notifications**:
    - Provides feedback for each action, with success messages (e.g., "Follow request sent") or error alerts (e.g., "Invalid input").
  - **Real-Time Error Feedback**:
    - Immediate error messages for invalid actions, such as attempting to follow a nonexistent user or uploading unsupported content formats.

Entities, relationships and attributes

1. User

- **Attributes**: user_id (Primary Key), username, created_date
- **Relationships**:
    - Follows: Users follow others via the follows table.
    - Login: User login records in login.
    - Likes: Users like photos, connected through likes.

2. Photo

- **Attributes**: photo_id (Primary Key), image_url, user_id, photo_created_at
- **Relationships**:
    - User: Each photo is uploaded by a user.
    - Comments: Photos can have comments in comments.
    - Likes: Users can like photos, through likes.
    - Tags: Linked with tags through photo_tags.
    -

3. Comment

- **Attributes**: comment_id (Primary Key), comment_text, user_id, photo_id, comment_created_at
- **Relationships**:
    - User: Each comment belongs to a user.
    - Photo: Each comment is attached to a photo.

4. Follow

- **Attributes**: follower_id, followee_id
- **Relationship**: Connects users in a many-to-many relationship.

5. Tag

- **Attributes**: tag_id (Primary Key), tag_name
- **Relationship**: Linked to photos through photo_tags.

6. Photo Tags

- **Attributes**: phtid (Primary Key), photo_id, tag_id
- **Relationships**: Connects tags and photos in a many-to-many format.

7. Hashtag

- **Attributes**: hashtag_id (Primary Key), hashtag_name, created_at

- **Relationship**: Users can follow hashtags via hashtag_follow.

8. Hashtag Follow

- **Attributes**: user_id, hashtag_id, created_at
- **Relationship**: Connects users and hashtags in a many-to-many relationship.

9. Login

- **Attributes**: login_id (Primary Key), user_id, ip, login_time
- **Relationship**: Tracks each user's login activity.

10. Likes

- **Attributes**: user_id, photo_id
- **Relationships**: Connects users and photos they liked.

11. Bookmark

- **Attributes**: bmid (Primary Key), post_id, user_id, created_at
- **Relationships**: Connects users and bookmarked photos.

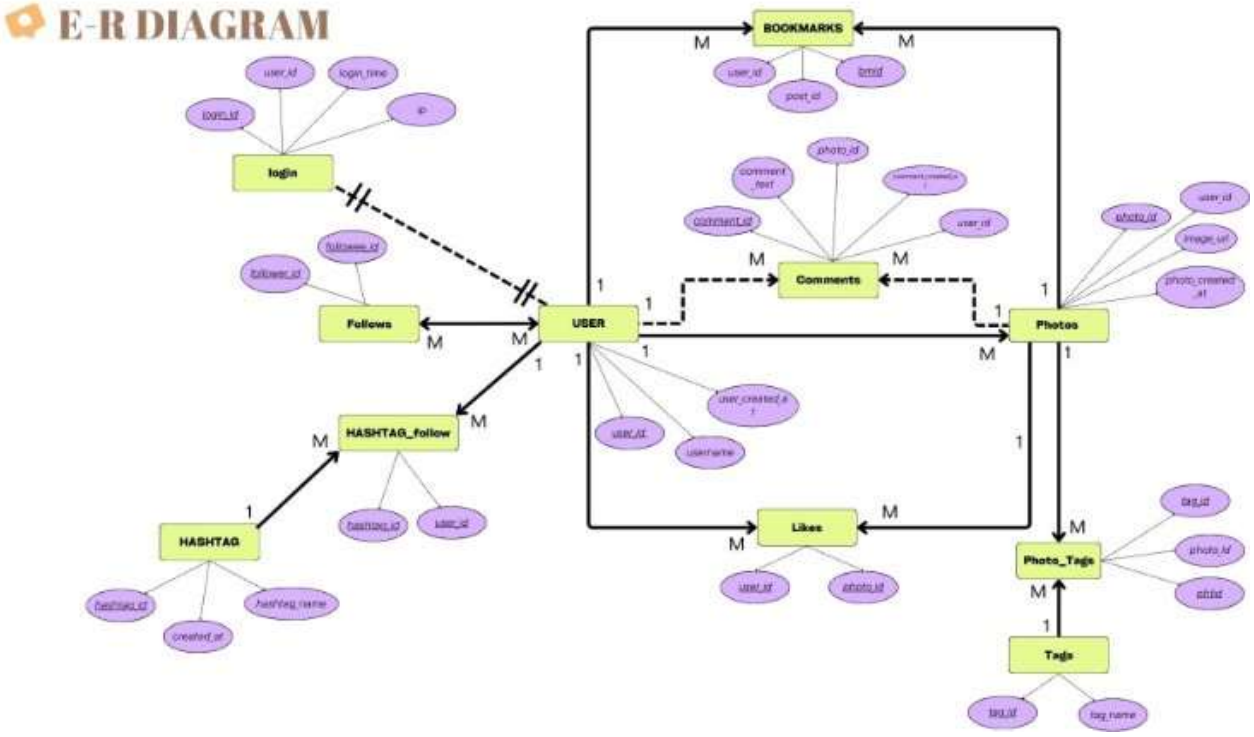


Fig 1 : ER diagram

Relational schema

1. users (user_id , username, user_created_at)
2. photos (photo_id, image_url, **user_id**, photo_created_at)
3. comments(comment_id,comment_text,**user_id**,**photo_id**, comment_created_at)
4. likes (**user_id, photo_id**)
5. follows (**follower_id, followee_id**)
6. tags (tag_id, tag_name)
7. photo_tags (phtid, **photo_id, tag_id**)
8. login (login_id, **user_id**, ip, login_time)
9. hashtags (hashtag_id, hashtag_name, created_at)
10. hashtag_follow (**user_id, hashtag_id)**
11. bookmarks (bmid, **post_id, user_id**)

Implementation

Database Setup:

1. Download MySQL Server [2]
   Go to the MySQL Community Downloads page and download the installer.
2. Run the Installer
   Launch the installer and select MySQL Server and MySQL Workbench. Proceed through the prompts.
3. Configuration Setup
   Choose a setup type (e.g., Developer Default).
   Set a root password when prompted.
   Optionally, create a dedicated user account for added security.
4. Complete Installation
   Finish the installation and open MySQL Workbench or command-line client to confirm it works.
5. Verify Installation
   Open MySQL Workbench or command line and connect to the server using the root password: mysql -u root -p

After completing these steps, MySQL will be installed and ready for database creation and management.

Table Creation Scripts :

```
CREATE DATABASE project_data;
USE project_data;

CREATE TABLE users(
    id INT AUTO_INCREMENT UNIQUE PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    user_created_at TIMESTAMP DEFAULT NOW()
);
describe users;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
username	varchar(255)	NO		NULL	
user_created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
CREATE TABLE photos(
    id INT AUTO_INCREMENT PRIMARY KEY,
    image_url VARCHAR(355) NOT NULL,
    user_id INT NOT NULL,
    photo_created_at TIMESTAMP,
    FOREIGN KEY(user_id) REFERENCES users(id)
);
describe photos;
CREATE TABLE comments(
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
image_url	varchar(355)	NO		NULL	
user_id	int	NO	MUL	NULL	
photo_created_at	timestamp	YES		NULL	

```
CREATE TABLE comments(
    id INT AUTO_INCREMENT PRIMARY KEY,
    comment_text VARCHAR(255) NOT NULL,
    user_id INT NOT NULL,
    photo_id INT NOT NULL,
    comment_created_at TIMESTAMP,
    FOREIGN KEY(user_id) REFERENCES users(id),
    FOREIGN KEY(photo_id) REFERENCES photos(id)
);
describe comments;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
comment_text	varchar(255)	NO		NULL	
user_id	int	NO	MUL	NULL	
photo_id	int	NO	MUL	NULL	
comment_created_at	timestamp	YES		NULL	

```
CREATE TABLE likes(
    user_id INT NOT NULL,
    photo_id INT NOT NULL,
    FOREIGN KEY(user_id) REFERENCES users(id),
    FOREIGN KEY(photo_id) REFERENCES photos(id)
    PRIMARY KEY(user_id,photo_id)
);
describe likes;
CREATE TABLE follows(
    follower id INT NOT NULL,
```

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
photo_id	int	NO	PRI	NULL	

```
CREATE TABLE follows(
    follower_id INT NOT NULL,
    followee_id INT NOT NULL,
    FOREIGN KEY (follower_id) REFERENCES users(id),
    FOREIGN KEY (followee_id) REFERENCES users(id),
    PRIMARY KEY(follower_id,followee_id)
);
describe follows;
```

Field	Type	Null	Key	Default	Extra
follower_id	int	NO	PRI	NULL	
followee_id	int	NO	PRI	NULL	

```
CREATE TABLE tags(
    id INTEGER AUTO_INCREMENT PRIMARY KEY,
    tag_name VARCHAR(255) UNIQUE NOT NULL
);
describe tags;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
tag_name	varchar(255)	NO	UNI	NULL	

```
CREATE TABLE photo_tags(
    photo_id INT NOT NULL,
    tag_id INT NOT NULL,
    FOREIGN KEY(photo_id) REFERENCES photos(id),
    FOREIGN KEY(tag_id) REFERENCES tags(id),
    PRIMARY KEY(photo_id,tag_id)
);
describe photo_tags;
```

Field	Type	Null	Key	Default	Extra
photo_id	int	NO	PRI	NULL	
tag_id	int	NO	PRI	NULL	
phtid	int	NO	UNI	NULL	auto_increment

```
CREATE TABLE login (
    login_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    ip VARCHAR(50) NOT NULL,
    login_time TIMESTAMP,
    FOREIGN KEY(user_id) REFERENCES users(id)
);
describe login;
```

Field	Type	Null	Key	Default	Extra
login_id	int	NO	PRI	NULL	auto_increment
user_id	int	NO	MUL	NULL	
ip	varchar(50)	NO		NULL	
login_time	timestamp	YES		NULL	

```sql
CREATE TABLE hashtags (
    hashtag_id INT AUTO_INCREMENT PRIMARY KEY,
    hashtag_name VARCHAR(255) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT NOW()
);
describe hashtags;
```

Field	Type	Null	Key	Default	Extra
hashtag_id	int	NO	PRI	NULL	auto_increment
hashtag_name	varchar(255)	NO	UNI	NULL	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```sql
CREATE TABLE hashtag_follow (
    user_id INT NOT NULL,
    hashtag_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT NOW(),
    PRIMARY KEY(user_id, hashtag_id),
    FOREIGN KEY(user_id) REFERENCES users(id),
    FOREIGN KEY(hashtag_id) REFERENCES hashtags(hashtag_id)
);
describe hashtag_follow;
```

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
hashtag_id	int	NO	PRI	NULL	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```sql
CREATE TABLE bookmarks (
    post_id INT NOT NULL,
    user_id INT NOT NULL,
    PRIMARY KEY(user_id, post_id),
    FOREIGN KEY(post_id) REFERENCES photos(id)
    FOREIGN KEY(user_id) REFERENCES users(id)
);
describe bookmarks;
```

Field	Type	Null	Key	Default	Extra
post_id	int	NO	PRI	NULL	
user_id	int	NO	PRI	NULL	
bmid	int	NO	UNI	NULL	auto_increment

```sql
SET SQL_SAFE_UPDATES = 0;
ALTER TABLE photo_tags
MODIFY COLUMN phtid INT;
SET @counter = 1;
UPDATE photo_tags
SET phtid = @counter := @counter + 1;
SET SQL_SAFE_UPDATES = 1;
ALTER TABLE photo_tags
MODIFY COLUMN phtid INT AUTO_INCREMENT UNIQUE;
ALTER TABLE bookmarks
ADD COLUMN bmid INT;
SET @counter = 1;
UPDATE bookmarks
SET bmid = @counter := @counter + 1;
```

```sql
INSERT INTO users (username, user_created_at) VALUES ('Kenton_Kirlin', '2017-02-16 18:22:10.846'), ('Andre_Purdy85', '2017-04-02 17:11:21.417'), ('Harley_Lind18', '2017-02-21 11:12:32.574'),
('Arely_Bogan63', '2016-08-13 01:28:43.085'), ('Aniya_Hackett', '2016-12-07 01:04:39.298'), ('Travon.Waters', '2017-04-30 13:26:14.496'),
('Kasandra_Homenick', '2016-12-12 06:50:07.996'), ('Tabitha_Schamberger11', '2016-08-20 02:19:45.512'), ('Gus93', '2016-06-24 19:36:30.978'), ('Presley_McClure', '2016-08-07 16:25:48.561'),
('Justina.Gaylord27', '2017-05-04 16:32:15.577'), ('Dereck65', '2017-01-19 01:34:14.296'), ('Alexandro35', '2017-03-29 17:09:02.344'), ('Jaclyn81', '2017-02-06 23:29:16.394'),
('Billy52', '2016-10-05 14:10:20.453'), ('Annalise.McKenzie16', '2016-08-02 21:32:45.646'), ('Norbert_Carroll35', '2017-02-06 22:05:43.425'), ('Odessa2', '2016-10-21 18:16:56.390'),
('Hailee26', '2017-04-29 18:53:39.650'), ('Delpha.Kihn', '2016-08-31 02:42:30.288'), ('Rocio33', '2017-01-23 11:51:15.467'), ('Kenneth64', '2016-12-27 09:48:17.380'),
('Eveline95', '2017-01-23 23:14:18.569'), ('Maxwell.Halvorson', '2017-04-18 02:32:43.597'), ('Tierra.Trantow', '2016-10-03 12:49:20.774'), ('Josianne.Friesen', '2016-06-07 12:47:00.703'), ('Darwin29', '20
```

```sql
INSERT INTO photos(image_url, user_id,photo_created_at) VALUES
('https://picsum.photos/100', 1, '2017-07-10 07:45:00'),('https://picsum.photos/101', 1, '2018-01-15 10:20:00'),('https://picsum.photos/102', 1, '2018-06-22 12:35:00'),
('https://picsum.photos/103', 1, '2019-03-05 14:50:00'),('https://picsum.photos/104', 1, '2019-11-30 16:05:00'),('https://picsum.photos/105', 2, '2020-02-25 09:15:00'),
('https://picsum.photos/106', 2, '2020-07-10 18:45:00'),('https://picsum.photos/107', 2, '2020-09-14 11:30:00'),('https://picsum.photos/108', 2, '2021-04-08 08:25:00'),('https://picsum.ph

INSERT INTO follows(follower_id, followee_id) VALUES (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (2, 10), (2, 11), (2, 12), (2, 13), (2, 14), (2, 15), (2, 16), (2, 17)
INSERT INTO comments(comment_text, user_id, photo_id) VALUES ('unde at dolorem', 2, 1), ('quae ea ducimus', 3, 1), ('alias a voluptatum', 5, 1), ('facere suscipit sunt', 14, 1), ('totam e
INSERT INTO likes(user_id,photo_id) VALUES (2, 1), (5, 1), (9, 1), (10, 1), (11, 1), (14, 1), (19, 1), (21, 1), (24, 1), (35, 1), (36, 1), (41, 1), (46, 1), (47, 1), (54, 1), (55, 1), (57
INSERT INTO tags(tag_name) VALUES ('sunset'), ('photography'), ('sunrise'), ('landscape'), ('food'), ('foodie'), ('delicious'), ('beauty'), ('stunning'), ('dreamy'), ('lol'), ('happy'), (
INSERT INTO photo_tags(photo_id, tag_id) VALUES (1, 18), (1, 17), (1, 21), (1, 13), (1, 19), (2, 4), (2, 3), (2, 20), (2, 2), (3, 8), (4, 12), (4, 11), (4, 21), (4, 13), (5, 15), (5, 14),
INSERT INTO hashtags (hashtag_name) VALUES
('travel'), ('food'), ('nature'), ('photography'), ('music'),
('art'), ('fitness'), ('lifestyle'), ('fashion'), ('technology'),
('love'), ('fun'), ('adventure'), ('inspiration'), ('health'),
('family'), ('friends'), ('animals'), ('books'), ('movies'),
('sports'), ('quotes'), ('education'), ('science'), ('gaming'),
('cooking'), ('wellness'), ('skincare'), ('makeup'), ('design'),
('crafts'), ('diy'), ('hiking'), ('running'), ('cycling'),
('beach'), ('mountains'), ('citylife'), ('selfcare'), ('mindfulness'),
('positivity'), ('mentalhealth'), ('growthmindset'), ('success'), ('entrepreneurship'),
```

```sql
INSERT INTO login (user_id, ip, login_time) VALUES (1, '192.168.1.101', '2020-05-23 09:43:39'), (2, '192.168.1.102', '2018-03-16 18:59:41'), (3, '192.168.1.103', '2021-06-13 21:50:12'),
 (4, '192.168.1.104', '2020-01-19 04:21:41'), (5, '192.168.1.105', '2022-09-26 00:14:49'), (6, '192.168.1.106', '2022-10-14 08:05:14'), (7, '192.168.1.107', '2019-04-06 12:27:31'), (8, '

INSERT INTO hashtag_follow (user_id, hashtag_id) VALUES
(1, 1), (1, 2), (1, 3), (2, 4), (2, 5),
(3, 6), (3, 7), (4, 8), (5, 9), (5, 10),
(6, 11), (6, 12), (7, 13), (7, 14), (8, 15),
(8, 16), (9, 17), (9, 18), (10, 19), (10, 20),
(11, 21), (12, 22), (13, 23), (13, 24), (14, 25),

INSERT INTO bookmarks (post_id, user_id) VALUES
(1, 1), (1, 2), (2, 1), (2, 3), (3, 4),
(3, 5), (4, 1), (4, 2), (5, 6), (5, 7),
(6, 8), (6, 9), (7, 10), (7, 11), (8, 12),
(9, 13), (9, 14), (10, 15), (10, 16), (11, 17),
(12, 18), (12, 19), (13, 20), (13, 21), (14, 22),
(15, 23), (15, 24), (16, 25), (16, 26), (17, 27),
(17, 28), (18, 29), (19, 30), (20, 31), (21, 32),
(22, 33), (23, 34), (24, 35), (25, 36), (26, 37),
(27, 38), (28, 39), (29, 40), (30, 41), (31, 42),
(32, 43), (33, 44), (34, 45), (35, 46), (36, 47),
(37, 48), (38, 49), (39, 50);
```

Stored Procedures, Views, and Indexes:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetPhotosByUsername`
(IN p_username VARCHAR(255))
BEGIN
    SELECT p.image_url
    FROM photos p
    JOIN users u ON p.user_id = u.id
    WHERE u.username = p_username;
END
```

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `project_data`.`photo_likes` AS
    SELECT
        `p`.`id` AS `photo_id`,
        `u`.`username` AS `username`,
        COUNT(`l`.`user_id`) AS `like_count`
    FROM
        ((`project_data`.`photos` `p`
        LEFT JOIN `project_data`.`likes` `l` ON ((`p`.`id` = `l`.`photo_id`)))
        JOIN `project_data`.`users` `u` ON ((`p`.`user_id` = `u`.`id`)))
    GROUP BY `p`.`id` , `u`.`username`
```

```
CREATE DEFINER=`root`@`localhost` FUNCTION `GetTotalLikes`(photoId INT)
 RETURNS int DETERMINISTIC
BEGIN
    DECLARE totalLikes INT;
    SELECT COUNT(*) INTO totalLikes FROM likes WHERE photo_id = photoId;
    RETURN totalLikes;
END
```

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `project_data`.`photo_likes` AS
    SELECT
        `p`.`id` AS `photo_id`,
        `u`.`username` AS `username`,
        COUNT(`l`.`user_id`) AS `like_count`
    FROM
        ((`project_data`.`photos` `p`
        LEFT JOIN `project_data`.`likes` `l` ON ((`p`.`id` = `l`.`photo_id`)))
        JOIN `project_data`.`users` `u` ON ((`p`.`user_id` = `u`.`id`)))
    GROUP BY `p`.`id` , `u`.`username`
```

Sample Data

Users table :

id	username	user_created_at
1	Kenton_Kirlin	2017-02-16 18:22:11
2	Andre_Purdy85	2017-04-02 17:11:21
3	Harley_Lind18	2017-02-21 11:12:33
4	Arely_Bogan63	2016-08-13 01:28:43
5	Aniya_Hackett	2016-12-07 01:04:39
6	Travon.Waters	2017-04-30 13:26:14
7	Kasandra_Homenick	2016-12-12 06:50:08
8	Tabitha_Schamberger11	2016-08-20 02:19:46

Photos table:

id	image_url	user_id	photo_created_at
1	https://picsum.photos/100	1	2017-07-10 07:45:00
2	https://picsum.photos/101	1	2018-01-15 10:20:00
3	https://picsum.photos/102	1	2018-06-22 12:35:00
4	https://picsum.photos/103	1	2019-03-05 14:50:00
5	https://picsum.photos/104	1	2019-11-30 16:05:00
6	https://picsum.photos/105	2	2020-02-25 09:15:00
7	https://picsum.photos/106	2	2020-07-10 18:45:00
8	https://picsum.photos/107	2	2020-09-14 11:30:00

Comments table:

id	comment_text	user_id	photo_id	comment_created_at
1	Where at the pain	2	1	NULL
2	Which we lead	3	1	NULL
3	Another from the pleasures	5	1	NULL
4	To do is to be	14	1	NULL
5	Totally choosing is to seek	17	1	NULL
6	Life because somewhat	21	1	NULL
7	Exercise is to be unperturbed	24	1	NULL
8	They are to the complaint	31	1	NULL

Likes table:

user_id	photo_id
2	1
5	1
9	1
10	1
11	1
14	1
19	1
21	1

tags table:

id	tag_name
20	beach
8	beauty
18	concert
7	delicious
10	dreamy
19	drunk
16	fashion
5	food

Photo_tags table:

photo_id	tag_id	phtid
1	18	1
1	17	2
1	21	3
1	13	4
1	19	5
2	4	6
2	20	8
2	2	9

Login table:

login_id	user_id	ip	login_time
1	1	192.168.1.101	2020-05-23 09:43:39
2	2	192.168.1.102	2018-03-16 18:59:41
3	3	192.168.1.103	2021-06-13 21:50:12
4	4	192.168.1.104	2020-01-19 04:21:41
5	5	192.168.1.105	2022-09-26 00:14:49
6	6	192.168.1.106	2022-10-14 08:05:14
7	7	192.168.1.107	2019-04-06 12:27:31
8	8	192.168.1.108	2023-09-07 23:13:23

Hashtag table:

hashtag_id	hashtag_name	created_at
1	travel	2024-11-08 09:45:39
2	food	2024-11-08 09:45:39
3	nature	2024-11-08 09:45:39
4	photography	2024-11-08 09:45:39
5	music	2024-11-08 09:45:39
6	art	2024-11-08 09:45:39
7	fitness	2024-11-08 09:45:39
8	lifestyle	2024-11-08 09:45:39

bookmarks table:

post_id	user_id	bmid
1	1	1
1	2	2
2	1	3
2	3	4
3	4	5
3	5	6
4	1	7
4	2	8

Hashtag_follow table:

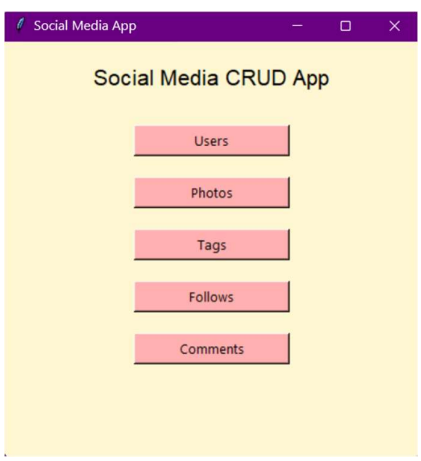
user_id	hashtag_id	created_at
1	1	2024-11-08 09:45:51
1	2	2024-11-08 09:45:51
1	3	2024-11-08 09:45:51
2	4	2024-11-08 09:45:51
2	5	2024-11-08 09:45:51
3	6	2024-11-08 09:45:51
3	7	2024-11-08 09:45:51
4	8	2024-11-08 09:45:51

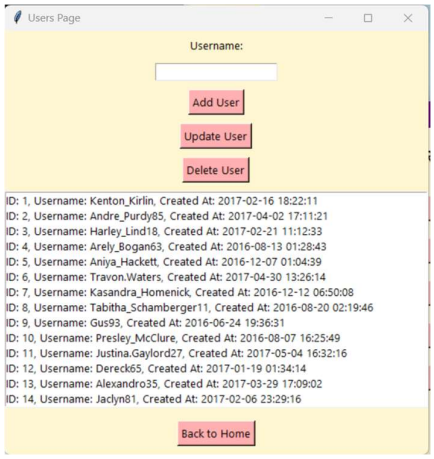
comments table:

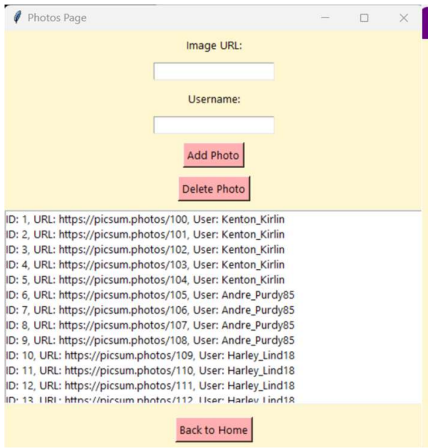
id	comment_text	user_id	photo_id	comment_created_at
1	Where at the pain	2	1	NULL
2	Which we lead	3	1	NULL
3	Another from the pleasures	5	1	NULL
4	To do is to be	14	1	NULL
5	Totally choosing is to seek	17	1	NULL
6	Life because somewhat	21	1	NULL
7	Exercise is to be unperturbed	24	1	NULL
8	They are to the complaint	31	1	NULL

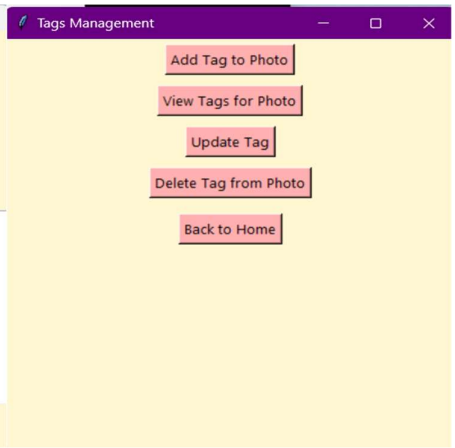
User Interface

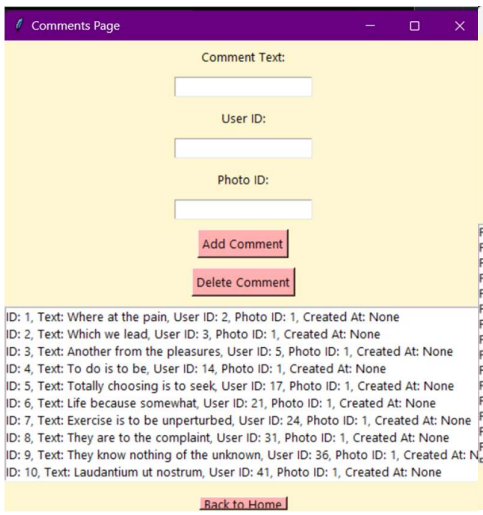
Screenshots of the UI

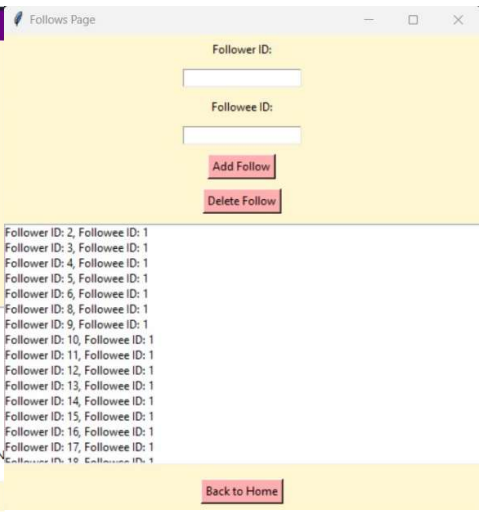












Usage Instructions:

- Home Screen:
  - The main screen offers three options: Users, Photos, and Tags.
  - Click on any button to manage that part of the app.
- Users:
  - Here, you can add, view, update, or delete user accounts.
- Photos:
  - This section allows you to upload, view, or edit photo details.
- Tags:
  - In the Tags section, you can:
  - Add a Tag to a photo.
  - View Tags for a specific photo, which opens a list of tags.
  - Update a Tag or Delete a Tag from a photo.
- Back to Home to return to the main menu.
- Confirmation:
  - After each action, you'll get a confirmation message to confirm it was successful.

Conclusion

The development of this social network application demonstrated the integration of SQL databases with Python's Tkinter for managing a variety of social media functionalities, including user interactions like likes, comments, and follows. Through efficient database design, we established a scalable relational schema that supports complex relationships among users, posts, tags, and other entities, enabling smooth CRUD operations and ensuring data integrity. This project highlights the importance of well-structured databases in handling large volumes of user data while maintaining performance and reliability.

In addition to successfully implementing core features, this project faced challenges related to maintaining data consistency and optimizing query performance. Addressing these issues strengthened our understanding of database indexing, normalization, and foreign key constraints, all critical elements in creating robust social applications. This project not only underscores the technical skills required for database management but also reinforces the significance of social media platforms as powerful tools for digital interaction and engagement.

References

[1] Dataset Link :https://www.kaggle.com/datasets/sanjanamurthy392/instagram-user-analysis?resource=download

[2] MySQL Documentation. (2023). *MySQL 8.0 Reference Manual*. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/