MINI PROJECT SRS

Social Media Management System

PEDDINENI BAVITHA – PES1UG21CS410
NEHA MIRIAM THOMAS – PES1UG21CS374

USER REQUIREMENT SPECIFICATION

Table of Contents:

1. Introduction

● Purpose of the project

The purpose of this project is for the operation of social media platforms, they provide the foundation for storing, managing, and analysing the vast amount of data that is generated by social media users. This data is used for a variety of purposes, including user management, content storage and retrieval, insights.

● Scope of the project

1. Scale: System can cater to small businesses, large enterprises, or a global user base.
2. Third-party Integrations: Integration with third-party tools and platforms, such as social media analytics services or CRM systems, to enhance the system's functionality.
3. User Base: Define the target audience and user base for the system. Can be designed for individual users, businesses, marketing agencies, or a combination of these. Understanding your user base is crucial for tailoring features and usability.
4. Platform Integration: The system can support multiple social media platforms. Either a specific platform (e.g., Facebook, Twitter) or provide integration with multiple platforms.

2. Project Description

● Project overview

The Social Media Management System (SMMS) project is dedicated to creating a centralized platform for effective and efficient management of multiple social media accounts. Our SMMS aims to simplify the complexities of social media marketing, providing a user-friendly interface that caters to businesses, individuals, and marketing agencies.

1. Centralized Account Management: Our SMMS will streamline the process of managing various social media accounts, allowing users to access and control multiple platforms from a single dashboard.
2. User Accessibility: The platform will be accessible via both web and mobile devices, providing users with the flexibility to manage their social media presence on the go.
3. Scalability: The system will be designed to scale seamlessly as user bases grow, maintaining performance and responsiveness.

● Major project functionalities

1. User Registration and Authorization:
    - Allow users to register accounts and log in securely.
    - Implement single-factor authorization for added security.
2. Content Management:
    - Create, edit, and delete posts.
    - Categorize and tag content for better organization.
3. User Profile Management:

    - Allow users to customize their profiles.
    - Upload profile pictures and cover photos.

3. System Features and Function Requirements

- System Feature 1: Content Management

Description of Feature: Users can create, edit, and organize their content.

Users should make sure that their photos and videos stay within the maximum size.

Entity – Post

● System Feature 2: Social Media Account Integration

Description of Feature: Users can link and manage their social media accounts within the SMMS.

Functional Requirements:

Support integration with major social media platforms, including Facebook, Twitter, Instagram, and LinkedIn.

Allow users to connect multiple accounts of the same platform.

Entity - Login

- System Feature 3: User Registration and Authentication

Description of Feature: This feature allows users to create accounts and securely log in to the SMMS.

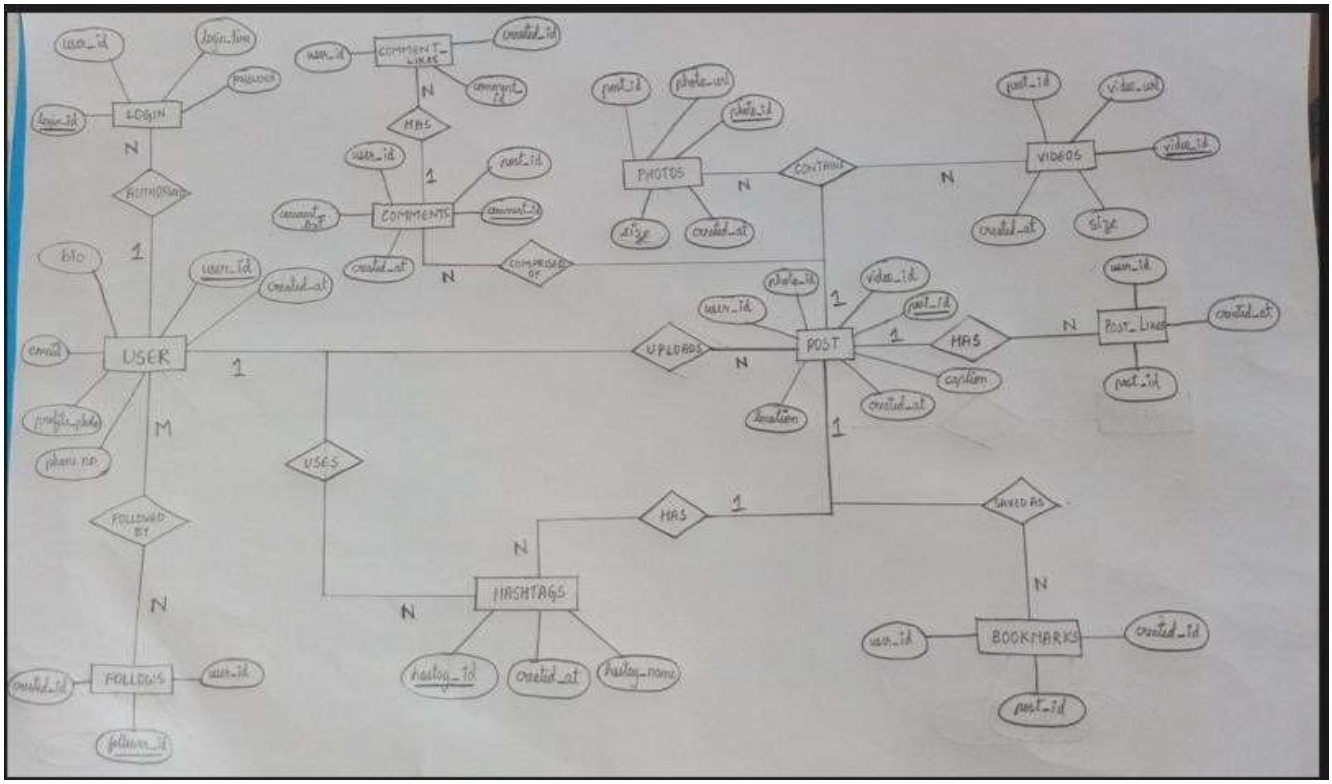
Functional Requirements:

Users must provide a valid email address and choose a unique username during registration.

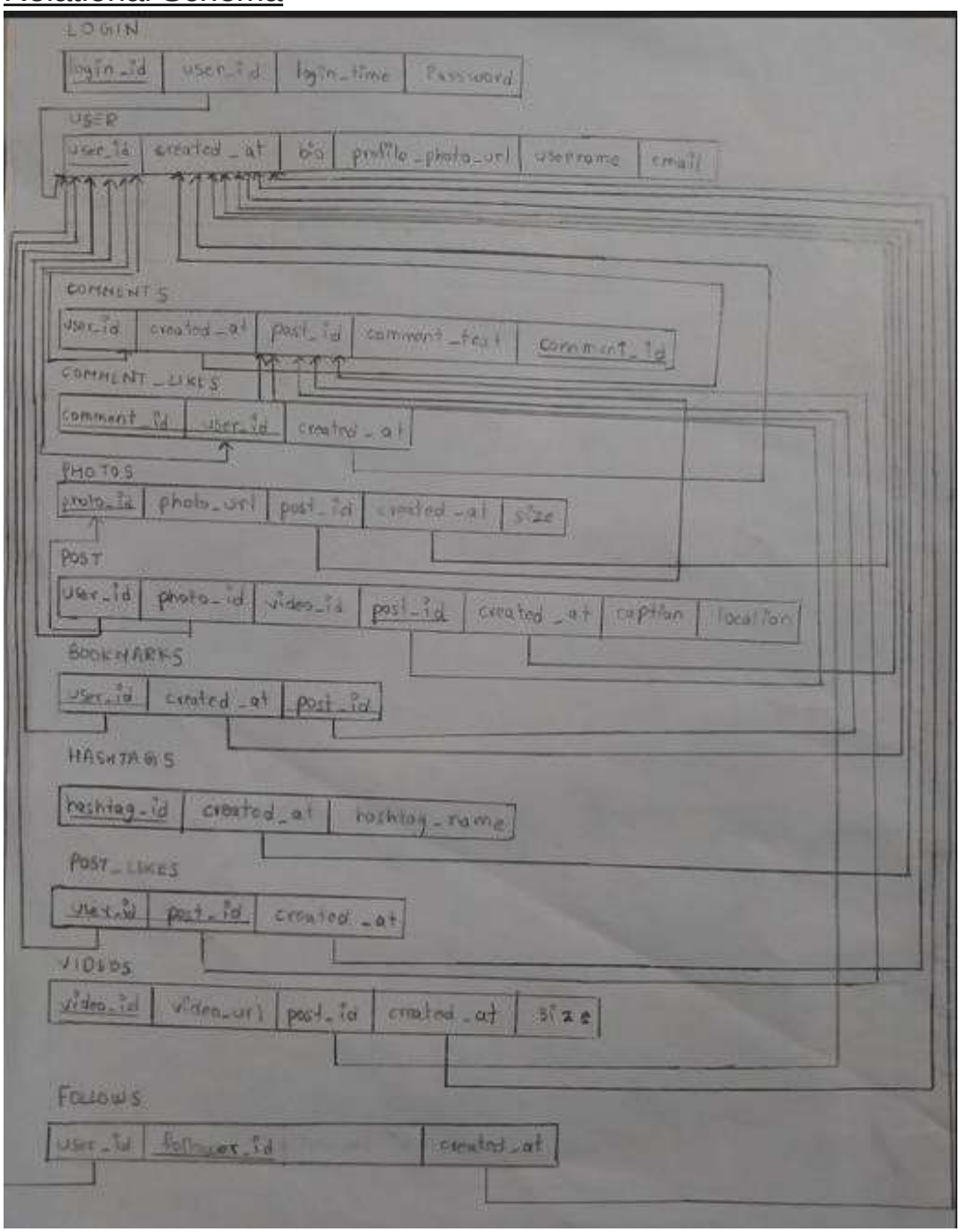
Passwords must meet security standards, including minimum length and complexity.

Entity – Login

ER Diagram


LOGIN
AUTHORISED
USER
FOLLOWED BY
FOLLOWS
USES
COMMENT_LIKES
HAS
COMMENTS
COMPRISED OF
PHOTOS
CONTAINS
VIDEOS
UPLOADS
POST
HAS
POST_LIKES
HAS
HASHTAGS
SAVED AS
BOOKMARKS

Relational Schema



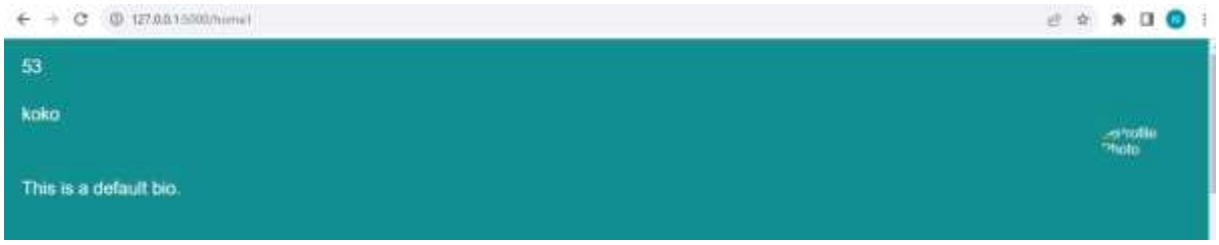
TRIGGER :

1. **CODE FOR TRIGGER:**

```
trig = '''
    CREATE TRIGGER before_user_insert
    BEFORE INSERT ON users
    FOR EACH ROW
    BEGIN
        IF NEW.bio IS NULL OR NEW.bio = '' THEN
            SET NEW.bio = 'This is a default bio.';
        END IF;
    END;

    //
    '''
```

2. **OUTPUT:**



CHANGE IMPLEMENTED IN DATABASE:

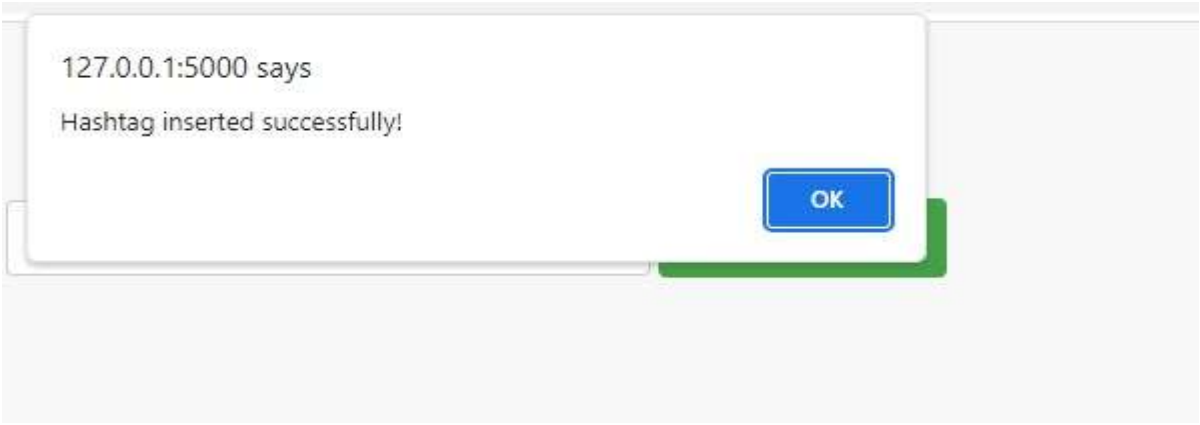
PROCEDURE:

```
proc='''
  CREATE PROCEDURE InsertNewHashtag(
    | IN p_hashtag_name VARCHAR(255)
  )
  BEGIN
    | DECLARE v_hashtag_id INT;
    | -- Check if the hashtag already exists
    | SELECT hashtag_id INTO v_hashtag_id FROM hashtags WHERE hashtag_name =
    | p_hashtag_name;
    |
    | -- If the hashtag doesn't exist, insert it
    | IF v_hashtag_id IS NULL THEN
    |   INSERT INTO hashtags (hashtag_name) VALUES (p_hashtag_name);
    |   SET v_hashtag_id = LAST_INSERT_ID();
    | END IF;
    |
    | -- You can return the ID of the inserted or existing hashtag if needed
    | SELECT v_hashtag_id AS new_hashtag_id;
  END;
  //
  '''
```

SQL STATEMENT WHICH FIRES THIS PROCEDURE:

```
cursor.callproc("InsertNewHashtag", (hashtag_name,))
```

OUTPUT:



CHANGE IMPLEMENTED IN THE DATABASE:

83	#pookie	2023-12-01 12:41:33

QUERIES:

1.

```
query6 = "SELECT u1.username AS follower, u2.username AS followee FROM follows
AS f INNER JOIN users AS u1 ON f.follower_id = u1.user_id INNER JOIN users AS u2
ON f.followee_id = u2.user_id WHERE u2.username = %s"
cursor.execute(query6,(username,))
user_follow=cursor.fetchall()
```

OUTPUT:

2.

```
cursor.execute("INSERT INTO users(username, profile_photo_url, bio,email) VALUES
(%s,%s,%s,%s)",[username, profile_photo_url, bio,email])
cursor.fetchall()
conn.commit()
conn.close()
```

OUTPUT:

3.

```
#Most Likes Posts
mlp_query="SELECT post_likes.user_id, post_likes.post_id, COUNT(post_likes.
post_id) FROM post_likes, post WHERE post.post_id = post_likes.post_id GROUP
BY post_likes.user_id, post_likes.post_id ORDER BY COUNT(post_likes.post_id)
DESC LIMIT 10"
cursor.execute(mlp_query)
mlp=cursor.fetchall()
```

OUTPUT:

User_id, Post_id and Number of likes

- (4, 33, 1)
- (7, 33, 1)
- (8, 33, 1)
- (11, 33, 1)
- (13, 33, 1)
- (17, 33, 1)
- (20, 33, 1)
- (22, 33, 1)
- (24, 33, 1)
- (27, 33, 1)

4.

```
#Most Followed Hashtag
hashtag_q1= "SELECT hashtag_name AS 'Hashtags', COUNT(hashtag_follow.hashtag_id) AS
'Total Follows' FROM hashtag_follow, hashtags  WHERE hashtags.hashtag_id =
hashtag_follow.hashtag_id GROUP BY hashtag_follow.hashtag_id ORDER BY COUNT
(hashtag_follow.hashtag_id) DESC LIMIT 5"
cursor.execute(hashtag_q1)
hashtag_most_followed = cursor.fetchall()
```

OUTPUT: