**Database Systems**

**Lab**

*Section 4A*

**Project Title:** **Social Interaction App**

**Project Deliverable:**

**Stored Procedures and Triggers**

**GROUP MEMBERS:**

● Abdul Ahad Tariq 22L-7866

● Tayyab Kamran Sami 22i-2505

● Saad Tariq 22L-7912

● Imran Ahmed 22L-7983



**Stored Procedures**

1. **Add a New User:**

This procedure inserts a new user into the Users table.

CREATE PROCEDURE AddUser(

@p\_username VARCHAR(50),

@p\_email VARCHAR(255),

@p\_password VARCHAR(255),

@p\_first\_name VARCHAR(50),

@p\_last\_name VARCHAR(50),

@p\_bio TEXT,

@p\_location VARCHAR(100),

@p\_occupation VARCHAR(50),

@p\_interests TEXT,

@p\_birthdate DATE,

@p\_profile\_picture VARCHAR(255)

)

AS

BEGIN

INSERT INTO Users (username, email, password, first\_name, last\_name, bio, location, occupation, interests, birthdate, profile\_picture)

VALUES (p\_username, p\_email, p\_password, p\_first\_name, p\_last\_name, p\_bio, p\_location, p\_occupation, p\_interests, p\_birthdate, p\_profile\_picture);

END

1. **Post a New Post:**

This procedure inserts a new post into the Posts table.

CREATE PROCEDURE AddPost(

@p\_user\_id INT,

@p\_content TEXT,

@p\_post\_type ENUM('text', 'photo', 'video'),

@p\_media\_url VARCHAR(255),

@p\_privacy ENUM('public', 'friends', 'private')

)

AS

BEGIN

INSERT INTO Posts (user\_id, content, post\_type, media\_url, privacy)

VALUES (p\_user\_id, p\_content, p\_post\_type, p\_media\_url, p\_privacy);

END

1. **Add a New Comment:**

This procedure inserts a new comment on a post into the Comments table.

CREATE PROCEDURE AddComment(

@ p\_post\_id INT,

@ p\_user\_id INT,

@ p\_content TEXT

)

AS

BEGIN

INSERT INTO Comments (post\_id, user\_id, content)

VALUES (p\_post\_id, p\_user\_id, p\_content);

END

1. **Like a Post:**

This procedure inserts a new like action on a post into the Likes table.

CREATE PROCEDURE LikePost(

@ p\_post\_id INT,

@ p\_user\_id INT

)

AS

BEGIN

INSERT INTO Likes (post\_id, user\_id)

VALUES (p\_post\_id, p\_user\_id);

END

1. **Update Post Privacy:**

This procedure updates the privacy setting of a post.

CREATE PROCEDURE UpdatePostPrivacy(

@ p\_post\_id INT,

@ p\_privacy ENUM('public', 'friends', 'private')

)

AS

BEGIN

UPDATE Posts SET privacy = p\_privacy WHERE post\_id = p\_post\_id;

END

1. **Add a New Friend:**

This procedure inserts a new friend into the Friends table.

CREATE PROCEDURE AddFriend(

@ p\_user\_id\_1 INT,

@ p\_user\_id\_2 INT

)

AS

BEGIN

DECLARE @user\_count INT;

-- Check if both users exist

SELECT COUNT(\*) INTO user\_count FROM Users WHERE user\_id IN (p\_user\_id\_1, p\_user\_id\_2);

IF (user\_count < 2)

BEGIN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'One or both users do not exist';

END

ELSE

BEGIN

-- Check if users are already friends

IF EXISTS (

SELECT \* FROM Friends

WHERE (friend\_id\_1 = p\_user\_id\_1 AND friend\_id\_2 = p\_user\_id\_2)

OR (friend\_id\_1 = p\_user\_id\_2 AND friend\_id\_2 = p\_user\_id\_1)

)

BEGIN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Users are already friends';

END

ELSE

BEGIN

-- Add friendship

INSERT INTO Friends (friend\_id\_1, friend\_id\_2, created\_at)

VALUES (LEAST(p\_user\_id\_1, p\_user\_id\_2), GREATEST(p\_user\_id\_1, p\_user\_id\_2), NOW());

END

END

END;

**Triggers**

1. **Update User Profile Modification Timestamp:**

This trigger automatically updates the updated\_at timestamp in the Users table whenever a user's profile is modified.

CREATE TRIGGER UpdateUserTimestamp ON Users

AFTER UPDATE

FOR EACH ROW

AS

BEGIN

UPDATE Users SET updated\_at = CURRENT\_TIMESTAMP WHERE user\_id = NEW.user\_id;

END

1. **Delete Comments on Post Deletion:**

This trigger automatically deletes comments associated with a post when the post is deleted.

CREATE TRIGGER DeleteCommentsOnPostDelete ON Posts

AFTER DELETE

FOR EACH ROW

AS

BEGIN

DELETE FROM Comments WHERE post\_id = OLD.post\_id;

END

1. **Update Post Modification Timestamp:**

This trigger automatically updates the timestamp field in the Posts table whenever a post is modified.

CREATE TRIGGER UpdatePostTimestamp ON Posts

AFTER UPDATE

FOR EACH ROW

AS

BEGIN

UPDATE Posts SET timestamp = CURRENT\_TIMESTAMP WHERE post\_id = NEW.post\_id;

END

1. **Delete Likes on Post Deletion:**

This trigger automatically deletes likes associated with a post when the post is deleted.

CREATE TRIGGER DeleteLikesOnPostDelete ON Posts

AFTER DELETE

FOR EACH ROW

AS

BEGIN

DELETE FROM Likes WHERE post\_id = OLD.post\_id;

END

1. **Don’t Make a Profile if Age is Less Than 18:**

This trigger automatically doesn’t Make a Profile if the User's Age is Less Than 18 when the post is created.

CREATE TRIGGER CheckUserAge ON Users

BEFORE INSERT

FOR EACH ROW

AS

BEGIN

DECLARE @user\_age INT;

SET user\_age = TIMESTAMPDIFF(YEAR, NEW.birthdate, CURDATE());

IF (user\_age < 18)

BEGIN

PRINT 'You can’t make a profile because you are under 18';

ROLLBACK TRANSACTION;

END

END;

1. **Check Comments for Harsh or Abusive words:**

This trigger automatically checks Comments for Harsh or Abusive Words:

CREATE TRIGGER CheckCommentContent ON Comments

BEFORE INSERT

FOR EACH ROW

AS

BEGIN

IF (

NEW.content LIKE '%Shit%'

OR NEW.content LIKE '%Bastard%'

OR NEW.content LIKE '%Bitch%'

OR NEW.content LIKE '%Sexy%'

OR NEW.content LIKE '%fat%'

OR NEW.content LIKE '%ugly%'

)

BEGIN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Comment contains inappropriate language';

END

END;

These are basic Stored Procedures and triggers for our App. Further additions or deletions can be made.