**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 varchar(1000),

@p\_post\_type varchar(10),

@p\_media\_url VARCHAR(255)

)

AS

BEGIN

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

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

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. **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 @user\_count = COUNT(\*) FROM Users WHERE user\_id IN (@p\_user\_id\_1, @p\_user\_id\_2);

IF (@user\_count = 2)

BEGIN

-- Check if users are already friends

IF NOT 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

-- Add friendship

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

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

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

AS

BEGIN

UPDATE Users

SET updated\_at = CURRENT\_TIMESTAMP

FROM inserted

WHERE Users.user\_id = inserted.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

AS

BEGIN

-- Delete comments associated with the deleted post

DELETE FROM Comments

WHERE post\_id IN (SELECT post\_id FROM deleted);

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

AS

BEGIN

-- Update the timestamp for modified posts

UPDATE Posts

SET timestamp = CURRENT\_TIMESTAMP

FROM inserted

WHERE Posts.post\_id = inserted.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

AS

BEGIN

-- Delete likes associated with the deleted post

DELETE FROM Likes

WHERE post\_id IN (SELECT post\_id FROM deleted);

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 CheckAgeBeforeInsert

ON Users

INSTEAD OF INSERT

AS

BEGIN

-- Check the age before insertion

IF EXISTS (

SELECT 1

FROM inserted

WHERE DATEDIFF(year, birthdate, GETDATE()) < 18

)

BEGIN

-- If age is less than 18, don't insert the record

RAISERROR ('User must be at least 18 years old.', 16, 1)

ROLLBACK TRANSACTION;

END

ELSE

BEGIN

-- If age is 18 or older, proceed with insertion

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

SELECT username, email, password, first\_name, last\_name, bio, location, occupation, interests, birthdate, profile\_picture, cover\_picture, created\_at, updated\_at

FROM inserted;

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

INSTEAD OF INSERT

AS

BEGIN

-- Check comment content before insertion

IF EXISTS (

SELECT 1

FROM inserted

WHERE

content LIKE '%Shit%'

OR content LIKE '%Bastard%'

OR content LIKE '%Bitch%'

OR content LIKE '%Sexy%'

OR content LIKE '%fat%'

OR content LIKE '%ugly%'

)

BEGIN

-- If comment contains inappropriate language, raise an error

RAISERROR ('Comment contains inappropriate language.', 16, 1);

ROLLBACK TRANSACTION;

END

ELSE

BEGIN

-- If comment is appropriate, proceed with insertion

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

SELECT user\_id, post\_id, content, timestamp

FROM inserted;

END

END;

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