# SocialQ

## Stored Procedure Documentation

This is the document explaining SocialQ's stored procedures. How it works, what's their purpose, what parameters they use, etc.

## Members

Ajero, Judah Jess T.
(Post Module)

Canturias, Christian G.
(Auth Module)

Guido, Darryl Adrian D.
(Feed Module)

## Instructor

Mr. Armando Jr. Saguin

## Course

Information Management (CC 105)
T/TH (10:00–11:00) • Lec
T/TH (13:00–14:30) • Lab

# 1. Auth Module

Assigned to: Canturias, Christian G.

This contains all stored procedures in the Authentication module.

## 1-1. User_Register

This procedure registers a new user if the chosen username is available.

How to call:

```
CALL User_Register(p_username, p_password)
```

Parameters:

| Name | Data Type | I/O | Info |
|---|---|---|---|
| p_username | VARCHAR(50) | IN | The username |
| p_password | VARCHAR(255) | IN | The hashed password |

Definition:

```
BEGIN
  DECLARE user_count INT DEFAULT 0;

  SELECT COUNT(*) INTO user_count FROM users WHERE username = p_username;

  IF user_count > 0 THEN
    SELECT 'Error: Username already exists' AS Record_Status;
  ELSE
    INSERT INTO users (username, password, joined_date)
    VALUES (p_username, p_password, NOW());
    SELECT 'Success' AS Record_Status;
  END IF;
END
```

This procedure first declares a local variable user_count to store the number of existing users with the same username. It uses the COUNT(*) function, then checks every row's username column of the users table, then compares it to p_username, and then stores the result in the local variable user_count. If user_count is more than 0, that means it found a match, and the username is taken. If user_count is 0, it inserts the values in the username and password columns.

## 1-2. User_Login

This procedure fetches user_id and password for login verification.

How to call:

```
CALL User_Login(p_username)
```

Parameters:

| Name | Data Type | I/O | Info |
|---|---|---|---|
| p_username | VARCHAR(50) | IN | The username to login |

Definition:

```
BEGIN
   SELECT id, username, password FROM users WHERE username = p_username LIMIT 1;
END
```

It selects id, username, and password where username is equal to the p_username input.

## 1-3. User_CreateSession

This procedure creates a session for the logged in user.

How to call:

```
CALL User_CreateSession(p_user_id, p_token)
```

Parameters:

| Name | Data Type | I/O | Info |
|---|---|---|---|
| p_user_id | INT | IN | The user’s ID |
| p_token | VARCHAR(16) | IN | The token PHP generated |

Definition:

```
BEGIN
  INSERT INTO sessions (user_id, token, login_date)
  VALUES (p_user_id, p_token, NOW())
  ON DUPLICATE KEY UPDATE
   token = VALUES(token),
   login_date = VALUES(login_date);

  SELECT 'Success' AS Record_Status;
END
```

It inserts the user's ID and token into a new row in the sessions table. It overwrites any duplicates along the way.

## 1-4. User_Logout

This procedure deletes the session of the current user.

How to call:

```
CALL User_Logout(p_token)
```

Parameters:

| Name | Data Type | I/O | Info |
|---|---|---|---|
| p_token | VARCHAR(16) | IN | Session token |

Definition:

```
User_Logout:
BEGIN
  DELETE FROM sessions WHERE token = p_token;
  SELECT 'Success' AS Record_Status;
END
```

It deletes the current user session.

## 2. Post Module

Assigned to: Ajero, Judah Jess T.

This contains all stored procedures in the Post creation module.

### 2-1. Post_Create

This procedure creates a new post.

How to call:

```
CALL Post_Create(p_token, p_post_content)
```

Parameters:

| Name | Data Type | I/O | Info |
|---|---|---|---|
| p_token | VARCHAR(16) | IN | Current session ID of user |
| p_post_content | TEXT | IN | Content of the post |

Definition:

```
BEGIN
    DECLARE v_user_id INT;

    SELECT user_id INTO v_user_id FROM sessions WHERE token = p_token LIMIT 1;

    IF v_user_id IS NULL THEN
        SELECT 'Error: Invalid or expired token' AS Record_Status;
    ELSE
        INSERT INTO posts (user_id, post_content, posted_date)
        VALUES (v_user_id, p_post_content, NOW());

        SELECT 'Success: Post created' AS Record_Status;
    END IF;
END
```

Inserts the new post into the posts table with current timestamp using NOW() function.

## 2-2. Post_Delete

This procedure deletes a post.

How to call:

```
CALL Post_Delete(p_token, p_post_id)
```

Parameters:

| Name | Data Type | I/O | Info |
|---|---|---|---|
| p_token | VARCHAR(16) | IN | Current session ID of user |
| p_post_id | INT | IN | Content of the post |

Definition:

```
BEGIN
   DECLARE v_user_id INT;
   DECLARE v_user_post_id INT;

   SELECT user_id INTO v_user_id FROM sessions WHERE token = p_token LIMIT 1;
   SELECT user_id INTO v_user_post_id FROM posts WHERE id = p_post_id AND user_id =
v_user_id LIMIT 1;

   IF v_user_id IS NULL THEN
      SELECT 'Error: Invalid or expired token' AS Record_Status;
   ELSEIF v_user_post_id IS NULL THEN
      SELECT 'Error: This is not your post' AS Record_Status;
   ELSE
      DELETE FROM posts WHERE id = p_post_id;
      SELECT 'Success: Post deleted' AS Record_Status;
   END IF;
END
```

Delete the post in the posts table if the current user owns it.

## 3. Feed Module

Assigned to: Guido, Darryl Adrian D.

This contains all stored procedures in the Post creation module.

### 3-1. User_Posts

This procedure returns posts from a specific user, from newest to oldest.

How to call:

```
CALL User_Posts(p_username)
```

Parameters:

| Name | Data Type | I/O | Info |
|---|---|---|---|
| p_username | VARCHAR(50) | IN | Username to filter by |

Definition:

```
BEGIN
  SELECT
    p.id AS post_id,
    u.username,
    p.post_content,
    p.posted_date
  FROM posts p
  JOIN users u ON p.user_id = u.id
  WHERE u.username = in_username
  ORDER BY p.posted_date DESC;
END
```

Selects user_id, username, created_at, content, and post_id from a username that equals to p_username. Joins posts and users table by user_id. It orders by created_at, newest to oldest.

## 3-2. All_Posts

This procedure returns all posts from the database, newest to oldest.

How to call:

```
CALL All_Posts()
```

Parameters:

None

Definition:

```
BEGIN
  SELECT
    p.id AS post_id,
    u.username,
    p.post_content,
    p.posted_date
  FROM posts p
  JOIN users u ON p.user_id = u.id
  ORDER BY p.posted_date DESC;
END
```

Selects user_id, username, created_at, content, and post_id. It orders by created_at, newest to oldest.