" به نام خدا "

توضیحات مربوط به پایگاه داده (پروژه قناری)

مریم گلی – شماره دانشجویی : ۹۸۳۱۰۵۴

- ایجاد جدول ها (CREATE TABLE)

شماره	توضیحات	کد ایجاد جدول
۱	user_account	

	← موجودیت : حساب کاربری
	← صفات :
	- نام
	- نام خانوادگی
	- نام کاربری
	- گذرواژه
	- تاریخ تولد
	- زمان عضویت
	- بیوگرافی یا خودنوشته
		`-- user account`
		`CREATE TABLE user_account (`
		`  first_name VARCHAR(20)  NOT NULL,`
		`  last_name  VARCHAR(20)  NOT NULL,`
		`  username  VARCHAR(20)  NOT NULL PRIMARY KEY,`
		`  pass  VARCHAR(128) NOT NULL,`
		`  birth_date DATE NOT NULL,`
		`  reg_date  TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,`
		`  biography  VARCHAR(64) NOT NULL`
		`);`

۲	post	

	← موجودیت : آوا
	← صفات :
	- شماره آوا
	- محتوا
	- کاربر ایجاد کننده ی آوا
	- زمان ایجاد آوا
		`-- post`
		`CREATE TABLE post (`
		`  post_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,`
		`  post_content VARCHAR(256)  NOT NULL,`
		`  post_username  VARCHAR(20)  NOT NULL,`
		`  post_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP`
		`);`

۳	message	

	← موجودیت : پیام
	← صفات :
	- شماره پیام
	- کاربر ارسال کننده
	- کاربر دریافت کننده
	- زمان ارسال
	- نوع محتوا (متن یا آوا)
	- محتوای متنی (شامل متن)
	- محتوای آوایی
	(شامل شناسه آوا)
		`-- message`
		`CREATE TABLE message (`
		`  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,`
		`  sender_username VARCHAR(20)  NOT  NULL,`
		`  receiver_username VARCHAR(20)  NOT NULL,`
		`  send_time  TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,`
		`  content_type CHAR(4) NOT NULL,`
		`  content_text VARCHAR(256),`
		`  content_post_id INT,`
		
		`  FOREIGN KEY (sender_username) REFERENCES user_account(username),`
		`  FOREIGN KEY (receiver_username) REFERENCES user_account(username)`
		`);`

۴ login_users

⟵ موجودیت :
کاربرهای وارد شده به سامانه
⟵ صفات :
- شماره کاربر وارد شده
- نام کاربری
- زمان وارد شدن

```sql
-- login_users
CREATE TABLE login_users (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    username  VARCHAR(20)  NOT NULL,
    login_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    FOREIGN KEY (username) REFERENCES user_account(username)
);
```

۵ hashtag

⟵ موجودیت : علامت ویژه
⟵ صفات :
- محتوا

```sql
-- hashtag
CREATE TABLE hashtag (
    content CHAR(6) NOT NULL PRIMARY KEY CHECK (content LIKE '#_____')
);
```

۶ post_hashtag

⟵ موجودیت :
آوا و هشتگ های موجود در آن
⟵ صفات :
- شماره
- محتوای هشتگ
- شناسه ی آوا (شماره آوا)

```sql
-- post_hashtag

CREATE TABLE post_hashtag (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    hashtag_content CHAR(6) NOT NULL CHECK (hashtag_content LIKE '#_____'),
    post_id INT UNSIGNED NOT NULL,

    FOREIGN KEY (post_id) REFERENCES post(post_id)
);
```

۷ follow

⟵ موجودیت : دنبال کردن
(دنبال کننده و دنبال شونده)
⟵ صفات :
- شماره
- کاربر دنبال کننده
- کاربر دنبال شونده

```sql
-- follow
CREATE TABLE follow (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    follower_username VARCHAR(20)  NOT NULL ,
    following_username VARCHAR(20)  NOT NULL ,

    UNIQUE (follower_username, following_username),
    FOREIGN KEY (follower_username) REFERENCES user_account(username),
    FOREIGN KEY (following_username) REFERENCES user_account(username)
);
```

۸ block

⟵ موجودیت : مسدود کردن
(بلاک کننده و بلاک شده)
⟵ صفات :
- شماره
- کاربر بلاک کننده
- کاربر بلاک شده

```sql
-- block
CREATE TABLE block (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    blocker_username VARCHAR(20)  NOT NULL ,
    blocking_username VARCHAR(20)  NOT NULL ,

    UNIQUE (blocker_username, blocking_username),
    FOREIGN KEY (blocker_username) REFERENCES user_account(username),
    FOREIGN KEY (blocking_username) REFERENCES user_account(username)
);
```

۹	like_post	

	← موجودیت : پسندیدن آوا (لایک کننده و آوای لایک شده)
	← صفات :
	- شماره
	- کاربر لایک کننده
	- شناسه آوا (شماره آوا)
		`-- like_post`
		`CREATE TABLE like_post (`
		`id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,`
		`like_username VARCHAR(20) NOT NULL ,`
		`post_id INT UNSIGNED NOT NULL ,`
		`UNIQUE (like_username, post_id),`
		`FOREIGN KEY (like_username) REFERENCES user_account(username),`
		`FOREIGN KEY (post_id) REFERENCES post(post_id)`
		`);`

۱۰	post_comment	

	← موجودیت : نظر دادن (آوا و نظرات مربوط به آن)
	← صفات :
	- شماره
	- شماره ی آوای اصلی
	- شماره ی آوای مربوط به نظر لایه اول
	- شماره ی آوای مربوط به نظر لایه دوم
		`-- post_comment`
		`CREATE TABLE post_comment (`
		`id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,`
		`post_id INT UNSIGNED NOT NULL ,`
		`comment1_post_id INT UNSIGNED NOT NULL,`
		`comment2_post_id INT UNSIGNED,`
		`FOREIGN KEY (post_id) REFERENCES post(post_id),`
		`FOREIGN KEY (comment1_post_id) REFERENCES post(post_id),`
		`FOREIGN KEY (comment2_post_id) REFERENCES post(post_id)`
		`);`

جداول مربوط به لاگ در trigger ها

۱۱	table_create_an_account	

	← موجودیت : جدول مربوط به لاگ ایجاد حساب کاربری
	← صفات :
	- کاربر ایجاد کننده ی حساب کاربری
	- زمان عضویت
		`-- table -> create_an_account`
		`CREATE TABLE table_create_an_account (`
		`username VARCHAR(20) NOT NULL PRIMARY KEY,`
		`reg_date TIMESTAMP NOT NULL`
		`);`

۱۲	table_new_post	

	← موجودیت : جدول مربوط به لاگ ایجاد آوای جدید
	← صفات :
	- شناسه آوا (شماره آوا)
	- کاربر ایجاد کننده ی آوا
	- زمان ارسال آوا
		`-- table -> new_post`
		`CREATE TABLE table_new_post (`
		`post_id INT UNSIGNED PRIMARY KEY,`
		`post_username VARCHAR(20) NOT NULL,`
		`post_time TIMESTAMP NOT NULL`
		`);`

- **ایجاد فرایند ها (CREATE PROCEDURE)**

فرایند احراز هویت	
توضیحات	**کد ایجاد procedure**

← نام : get_current_user
این فرایند با استفاده از اطلاعات جدول login_ users ، آخرین کاربر وارد شده را به عنوان فاعل، بر می گرداند.
← ورودی: –
← خروجی: نام کاربری فاعل
← جدول نتایج: –
	`-- get current user`
	
	`DELIMITER //`
	`CREATE PROCEDURE get_current_user (OUT out_username VARCHAR(20))`
	`BEGIN`
	`SELECT username INTO out_username`
	`FROM login_users`
	`ORDER BY login_time DESC`
	`LIMIT 1;`
	`END`
	`//`

فرایند مربوط به قابلیت های سامانه	
شماره	**کد ایجاد procedure و توضیحات**

۱	**create_an_account**
	`-- create an account`
	`DELIMITER //`
	`CREATE PROCEDURE create_an_account (IN firstname VARCHAR(20),`
	`IN lastname VARCHAR(20),`
	`IN username VARCHAR(20),`
	`IN pass VARCHAR(128),`
	`IN birth_date DATE,`
	`IN biography VARCHAR(64)`
	`)`
	`BEGIN`
	`INSERT INTO user_account (first_name, last_name, username, pass, birth_date, biography)`
	`VALUES`
	`(firstname, lastname, username, SHA1(pass), birth_date, biography);`
	`END`
	`//`
	
	← قابلیت : ایجاد حساب کاربری
	← ورودی: نام – نام خانوادگی – نام کاربری – گذرواژه – تاریخ تولد – بیوگرافی
	← خروجی: –
	← جدول نتایج: –

۲	← قابلیت: ورود به حساب کاربری
	← ورودی: نام کاربری – گذرواژه
	← خروجی: –
	← جدول نتایج: –
		login
		``
		-- login
		DELIMITER //
		
		CREATE PROCEDURE login (IN in_username VARCHAR(20),
		IN in_pass VARCHAR(128)
		)
		BEGIN
		INSERT INTO `login_users`(`username`)
		SELECT username
		FROM user_account
		WHERE username = in_username AND pass = SHA1(in_pass);
		END
		//
		``

۳	← قابلیت: بررسی ورود های کاربر
	← ورودی: –
	← خروجی: –
	← جدول نتایج:
	- login_time
		get_user_logins
		``
		-- get user logins
		CREATE PROCEDURE get_user_logins ()
		BEGIN
		CALL get_current_user (@out_username);
		
		SELECT login_time
		FROM login_users
		WHERE username = @out_username
		ORDER BY login_time DESC;
		END
		//
		``

۴	← قابلیت: ارسال آوا
	← ورودی: محتوای آوا
	← خروجی: –
	← جدول نتایج: –
		new_post
		``
		-- new post
		DELIMITER //
		CREATE PROCEDURE new_post (IN in_content VARCHAR(256))
		BEGIN
		CALL get_current_user (@current_username);
		
		INSERT INTO `post`(`post_content`, `post_username`)
		VALUES
		(in_content, @current_username);
		END
		//
		``

۵	← قابلیت: دریافت آواهای شخصی
	← ورودی: –
	← خروجی: –
	← جدول نتایج:
	- post_id
	- post_content
		get_user_posts
		``
		-- get user posts
		DELIMITER //
		CREATE PROCEDURE get_user_posts ()
		BEGIN
		CALL get_current_user (@current_username);
		
		SELECT post_id, post_content
		FROM post
		WHERE post_username = @current_username;
		END
		//
		``

۶	← قابلیت: دنبال کردن
	← ورودی: کاربر دنبال شونده
	← خروجی: –
	← جدول نتایج: –
		follow_user
		`-- follow_user`
		`DELIMITER //`
		`CREATE PROCEDURE follow_user (IN in_following_username VARCHAR(20))`
		`BEGIN`
		`CALL get_current_user (@current_username);`
		`INSERT INTO `follow`(`follower_username`, `following_username`)`
		`VALUES`
		`-- follower_username`
		`((SELECT username`
		`FROM user_account`
		`WHERE username = @current_username),`
		`-- following_username`
		`(SELECT username`
		`FROM user_account`
		`WHERE username = in_following_username));`
		`END`
		`//`

۷	← قابلیت: توقف دنبال کردن
	← ورودی: کاربر دنبال شونده
	← خروجی: –
	← جدول نتایج: –
		unfollow_user
		`-- unfollow user`
		`DELIMITER //`
		`CREATE PROCEDURE unfollow_user (IN in_following_username VARCHAR(20))`
		`BEGIN`
		`CALL get_current_user (@current_username);`
		`DELETE FROM `follow``
		`WHERE follower_username = @current_username`
		`AND following_username = in_following_username;`
		`END`
		`//`

۸	← قابلیت: مسدود کردن
	← ورودی: کاربری که قرار است بلاک شود (کاربر بلاک شونده)
	← خروجی: –
	← جدول نتایج: –
		block_user
		`-- block user`
		`DELIMITER //`
		`CREATE PROCEDURE block_user (IN in_blocking_username VARCHAR(20))`
		`BEGIN`
		`CALL get_current_user (@current_username);`
		`INSERT INTO `block`(`blocker_username`, `blocking_username`)`
		`VALUES`
		`-- blocker_username`
		`((SELECT username`
		`FROM user_account`
		`WHERE username = @current_username),`
		`-- blocking username`
		`(SELECT username`
		`FROM user_account`
		`WHERE username = in_blocking_username));`
		`END`
		`//`

۹	unblock_user	← قابلیت: آزاد کردن ← ورودی: کاربر بلاک شده ← خروجی: - ← جدول نتایج: -

```
-- unblock_user
DELIMITER //
CREATE PROCEDURE unblock_user (IN in_blocking_username VARCHAR(20))
    BEGIN
        CALL get_current_user (@current_username);

        DELETE FROM `block`
        WHERE blocker_username = @current_username
              AND blocking_username = in_blocking_username;
    END
//
```

۱۰	get_following_activities	← قابلیت: دریافت فعالیت دنبال شوندگان ← ورودی: - ← خروجی: - ← جدول نتایج: - post_time - post_content

```
-- get following activities
DELIMITER //
CREATE PROCEDURE get_following_activities ()
    BEGIN
        CALL get_current_user (@current_username);

        SELECT post_content, post_time
        FROM post
        WHERE
            post_username IN (SELECT following_username
                              FROM follow
                              WHERE follower_username = @current_username)
            AND
            post_username NOT IN (SELECT blocker_username
                                  FROM block
                                  WHERE blocking_username = @current_username)

        ORDER BY post_time DESC;

    END
//
```

۱۱	get_user_activities	← قابلیت: دریافت فعالیت کاربران ← ورودی: نام کاربری کاربر انتخاب شده ← خروجی: - ← جدول نتایج: - post_content

```
-- get user activities
DELIMITER //
CREATE PROCEDURE get_user_activities (IN in_username VARCHAR(20))
    BEGIN
        CALL get_current_user (@current_username);

        SELECT post_content
        FROM post
        WHERE post_username = in_username
              AND post_username
                  NOT IN (SELECT blocker_username
                          FROM block
                          WHERE blocking_username = @current_username);

    END
//
```

commenting

```
-- commenting
DELIMITER //
CREATE PROCEDURE commenting (IN in_post_id INT,
                             IN comment_type INT,
                             -- comment_type = 1
                             IN comment1_content VARCHAR(256),
                             -- comment_type = 2
                             IN in_comment1_id INT,
                             IN comment2_content VARCHAR(256)
                            )
    BEGIN
        CALL get_current_user (@current_username);

        IF comment_type = 1 THEN
            -- comment 1
            INSERT INTO `post`(`post_content`, `post_username`)
            VALUES (comment1_content, @current_username);

            INSERT INTO `post_comment`(`post_id`, `comment1_post_id`)
            SELECT main_post.post_id, comment1_post.post_id
            FROM post AS main_post, post AS comment1_post
            WHERE main_post.post_id = in_post_id
                  AND comment1_post.post_id = (SELECT MAX(post_id) FROM post)
                  AND (main_post.post_username, comment1_post.post_username)
                      NOT IN (SELECT blocker_username , blocking_username FROM block);

            DELETE FROM `post`
            WHERE post_id = (SELECT MAX(post_id) FROM post)
                  AND NOT EXISTS
                        (
                        SELECT main_post.post_id, comment1_post.post_id
                        FROM post AS main_post, post AS comment1_post
                        WHERE main_post.post_id = in_post_id
                              AND comment1_post.post_id = (SELECT MAX(post_id) FROM post)
                              AND (main_post.post_username, comment1_post.post_username)
                                   NOT IN (SELECT blocker_username , blocking_username FROM block)
                        );

         ELSEIF comment_type = 2 THEN
            -- comment 2
            INSERT INTO `post`(`post_content`, `post_username`)
            VALUES (comment2_content, @current_username);

            INSERT INTO `post_comment`(`post_id`, `comment1_post_id`, `comment2_post_id`)
            SELECT main_post.post_id, comment1_post.post_id, comment2_post.post_id
            FROM post AS main_post, post AS comment1_post, post AS comment2_post
            WHERE main_post.post_id = in_post_id AND comment1_post.post_id = in_comment1_id
                  AND comment2_post.post_id = (SELECT MAX(post_id) FROM post)
                  AND (comment1_post.post_username, comment2_post.post_username)
                       NOT IN (SELECT blocker_username , blocking_username FROM block);

            DELETE FROM `post`
            WHERE post_id = (SELECT MAX(post_id) FROM post)
                  AND NOT EXISTS
                        (
                        SELECT main_post.post_id, comment1_post.post_id, comment2_post.post_id
                        FROM post AS main_post, post AS comment1_post, post AS comment2_post
                        WHERE main_post.post_id = in_post_id AND comment1_post.post_id = in_comment1_id
                              AND comment2_post.post_id = (SELECT MAX(post_id) FROM post)
                              AND (comment1_post.post_username, comment2_post.post_username)
                                   NOT IN (SELECT blocker_username , blocking_username FROM block)
                        );

        END IF;

    END
//
```

۱۲

	← قابلیت: نظر دادن ← ورودی: شماره ی آوای اصلی – نوع نظر (لایه ۱ یا لایه ۲) – محتوای کامنت لایه ۱ – شماره ی آوای کامنت لایه ۱ – محتوای کامنت لایه ۲ ← خروجی: – ← جدول نتایج: –

	get_comments	
۱۳	← قابلیت: دریافت نظرات آوا ← ورودی: شماره ی آوا ← خروجی: – ← جدول نتایج: - c1_post_id - c1_post_content	

```
-- get_comments
DELIMITER //
CREATE PROCEDURE get_comments (IN in_post_id INT)
    BEGIN
        CALL get_current_user (@current_username);

        SELECT post_comment.comment1_post_id, comment1_post.post_content AS comment_1_post_content
        FROM post AS main_post, post AS comment1_post, post_comment
        WHERE main_post.post_id = in_post_id AND main_post.post_id = post_comment.post_id
              AND comment1_post.post_id = post_comment.comment1_post_id
              AND post_comment.comment2_post_id IS NULL
              AND (main_post.post_username, @current_username)
                   NOT IN (SELECT blocker_username , blocking_username FROM block)
              AND (comment1_post.post_username, @current_username)
                  NOT IN (SELECT blocker_username , blocking_username FROM block);
    END
//
```

	add_hashtag	
۱۴	← قابلیت: اضافه کردن علامت ویژه به آوا ← ورودی: شماره آوا – محتوای هشتگ ← خروجی: – ← جدول نتایج: –	

```
-- add hashtag
DELIMITER //
CREATE PROCEDURE add_hashtag (IN in_post_id INT,
                              IN in_hashtag_content CHAR(6)
                             )
    BEGIN
        CALL get_current_user (@current_username);

        INSERT INTO `post_hashtag`(`hashtag_content`, `post_id`)
        VALUES
        (in_hashtag_content, (SELECT post_id
                              FROM post
                              WHERE post_id = in_post_id
                                    AND post_username = @current_username));

        INSERT INTO `hashtag`(`content`)
        SELECT hashtag_content
        FROM post_hashtag
        WHERE hashtag_content = in_hashtag_content
              AND hashtag_content NOT IN (SELECT content FROM hashtag)
              AND EXISTS (SELECT post_id
                          FROM post
                          WHERE post_id = in_post_id
                                AND post_username = @current_username);
    END
//
```

۱۵	← قابلیت: دریافت آواهای یک علامت ویژه ← ورودی: محتوای هشتگ ← خروجی: – ← جدول نتایج: - podt_id - post_time - podt_username - post_content	get_posts_with_specific_hashtag -- get posts with specific hashtag DELIMITER // CREATE PROCEDURE get_posts_with_specific_hashtag (IN in_hashtag_content CHAR(6)) BEGIN CALL get_current_user (@current_username); SELECT post.post_id, post.post_content, post.post_username, post.post_time FROM post_hashtag, post WHERE post_hashtag.post_id = post.post_id AND hashtag_content = in_hashtag_content AND (post_username, @current_username) NOT IN (SELECT blocker_username , blocking_username FROM block) ORDER BY post_time DESC; END //
۱۶	← قابلیت: پسندیدن آوا ← ورودی: شماره آوا ← خروجی: – ← جدول نتایج: –	like_posts -- like posts DELIMITER // CREATE PROCEDURE like_posts (IN in_post_id INT) BEGIN CALL get_current_user (@current_username); INSERT INTO `like_post`(`like_username`, `post_id`) SELECT username, post_id FROM user_account, post WHERE username = @current_username AND post_id = in_post_id AND (post_username, username) NOT IN (SELECT blocker_username , blocking_username FROM block); END //
۱۷	← قابلیت: دریافت تعداد پسند ها ← ورودی: شماره آوا ← خروجی: – ← جدول نتایج: - number_of_likes	get_number_of_likes -- get number of likes DELIMITER // CREATE PROCEDURE get_number_of_likes (IN in_post_id INT) BEGIN CALL get_current_user (@current_username); SELECT COUNT(like_username) AS number_of_likes FROM like_post , post WHERE like_post.post_id = post.post_id AND like_post.post_id = in_post_id AND (post_username, @current_username) NOT IN (SELECT blocker_username , blocking_username FROM block); END //

get_list_of_like_username

۱۸

```
-- get list of like_username
DELIMITER //
CREATE PROCEDURE get_list_of_like_username (IN in_post_id INT)
    BEGIN
        CALL get_current_user (@current_username);

        SELECT like_username
        FROM  like_post , post
        WHERE like_post.post_id = post.post_id
              AND like_post.post_id = in_post_id
              AND (post_username, @current_username)
                   NOT IN  (SELECT blocker_username , blocking_username FROM block)
              AND (like_username, @current_username)
                   NOT IN  (SELECT blocker_username , blocking_username FROM block);
    END
//
```

⟵ قابلیت:

دریافت لیست پسندکنندگان

⟵ ورودی: شماره آوا

⟵ خروجی: –

⟵ جدول نتایج:

- like_username

get_list_of_popular_posts

۱۹

```
-- get list of popular posts
DELIMITER //
CREATE PROCEDURE get_list_of_popular_posts ()
    BEGIN
        CALL get_current_user (@current_username);

        SELECT like_post.post_id, COUNT(like_username) AS number_of_likes
        FROM like_post, post
        WHERE like_post.post_id = post.post_id
              AND (post_username, @current_username)
                   NOT IN  (SELECT blocker_username , blocking_username FROM block)
        GROUP BY like_post.post_id
        ORDER BY number_of_likes DESC;
    END
//
```

⟵ قابلیت:

دریافت آواهای پرطرفدار

⟵ ورودی: –

⟵ خروجی: –

⟵ جدول نتایج:

- podt_id
- number_of_likes

send_message

۲۰

```
-- send message
DELIMITER //
CREATE PROCEDURE send_message (IN in_receiver_username VARCHAR(20),
                               IN in_content_type INT,
                               -- in_content_type = 0 -> text
                               IN in_text VARCHAR(256),
                               -- in_content_type = 1 -> post
                               IN in_post_id INT
                              )
    BEGIN
        CALL get_current_user (@current_username);

        IF in_content_type = 0 THEN
            -- message -> text
            INSERT INTO `message`(`sender_username`, `receiver_username`, `content_type`, `content_text`)
            SELECT sender.username, receiver.username, 'text', in_text
            FROM user_account AS sender, user_account AS receiver
            WHERE sender.username = @current_username AND receiver.username = in_receiver_username
                  AND (receiver.username, sender.username)
                       NOT IN (SELECT blocker_username , blocking_username FROM block);

        ELSEIF in_content_type = 1 THEN
            -- message -> post
            INSERT INTO `message`(`sender_username`, `receiver_username`, `content_type`, `content_post_id`
            SELECT sender.username, receiver.username, 'post', post_id
            FROM user_account AS sender, user_account AS receiver, post
            WHERE sender.username = @current_username AND receiver.username = in_receiver_username
                  AND post_id = in_post_id
                  AND (receiver.username, sender.username)
                       NOT IN (SELECT blocker_username , blocking_username FROM block)
                  AND (post_username, sender.username)
                       NOT IN (SELECT blocker_username , blocking_username FROM block);

        END IF;

    END
//
```

⟵ قابلیت: ارسال پیام

⟵ ورودی:

کاربر دریافت کننده

نوع پیام

محتوای متنی (شامل متن)

محتوای آوایی (شامل شماره آوا)

⟵ خروجی: –

⟵ جدول نتایج: –

۲۱	← قابلیت: دریافت لیست پیام های دریافتی از کاربر ← ورودی: کاربر انتخاب شده (فرستنده پیام) ← خروجی: - ← جدول نتایج: - send_time - content_type - content_text - content_post_id	**get_list_of_received_message_from_user** -- get_list_of_received_message_from_user DELIMITER // CREATE PROCEDURE get_list_of_received_message_from_user (IN in_username VARCHAR(20)) BEGIN CALL get_current_user (@current_username); -- content -> text SELECT send_time, content_type, content_text, content_post_id FROM message WHERE sender_username = in_username AND receiver_username = @current_username AND content_type = 'text' UNION -- content -> post SELECT send_time, content_type, content_text, content_post_id FROM message, post WHERE sender_username = in_username AND receiver_username = @current_username AND content_type = 'post' AND content_post_id = post_id AND (post_username, receiver_username) NOT IN (SELECT blocker_username , blocking_username FROM block) ORDER BY send_time DESC; END //
۲۲	← قابلیت: دریافت لیست ارسال کنندگان پیام ← ورودی: - ← خروجی: - ← جدول نتایج: - sender_username	**get_list_of_sender_username** -- get list of sender_username DELIMITER // CREATE PROCEDURE get_list_of_sender_username () BEGIN CALL get_current_user (@current_username); SELECT DISTINCT senders.sender_username FROM (SELECT sender_username, send_time FROM message WHERE receiver_username = @current_username AND content_type = 'text' UNION SELECT sender_username, send_time FROM message, post WHERE receiver_username = @current_username AND content_type = 'post' AND message.content_post_id = post.post_id AND (post_username, receiver_username) NOT IN (SELECT blocker_username , blocking_username FROM block) ORDER BY send_time DESC) AS senders; END //

- **ایجاد ماشه ها (CREATE Trigger)**

لاگ ایجاد حساب کاربری – log_create_an_account

```
-- log_create_an_account
DELIMITER //
CREATE TRIGGER log_create_an_account
    AFTER INSERT
    ON user_account FOR EACH ROW
        BEGIN
            INSERT INTO `table_create_an_account`(`username`, `reg_date`)
            VALUES (NEW.username, NEW.reg_date);
        END
//
```

← توضیحات:

بعد از ایجاد هر حساب کاربری، نام کاربری و زمان عضویت را در جدول *table_create_an_account* ذخیره می کند.

ایجاد علامت ویژه – hashtag_trigger

```
DELIMITER //
CREATE TRIGGER hashtag_trigger
    AFTER INSERT
    ON post FOR EACH ROW
        BEGIN
            DECLARE p_content VARCHAR(256);
            DECLARE tmp VARCHAR(256);
            DECLARE p_content_length INT;
            DECLARE h_index INT;
            DECLARE h_index_curr INT;
            DECLARE h_index_next INT;
            DECLARE h_content CHAR(6);

            SET p_content = NEW.post_content;
            SELECT LENGTH (p_content) INTO p_content_length;

            SELECT INSTR(p_content, '#') INTO h_index;
            IF h_index > 0 THEN
                WHILE (p_content_length > 0 AND p_content_length IS NOT NULL) DO
                    SELECT INSTR(p_content, '#') INTO h_index_curr;
                    SELECT SUBSTR(p_content, h_index_curr+1) INTO tmp;
                    SELECT INSTR(tmp, '#') INTO h_index_next;
                    IF h_index_next = 0 THEN
                        SET h_index_next = p_content_length;
                    END IF;

                    IF (h_index_next = 6) THEN
                        -- valid hashtag
                        SELECT SUBSTR(p_content, h_index_curr, 6) INTO h_content;
                        CALL add_hashtag (NEW.post_id, h_content);
                    END IF;

                    SELECT SUBSTR(p_content, h_index_curr+h_index_next) INTO p_content;
                    SELECT LENGTH (p_content) INTO p_content_length;

                END WHILE;
            END IF;

        END
//
```

← توضیحات :

بعد از ایجاد هر پست، بررسی می کند که آیا در آن هشتگ معتبری وجود دارد یا خیر و در صورت وجود، عملیات اضافه کردن هشتگ (add_hashtag) را انجام می دهد.

بررسی معتبر بودن هشتگ ها:

ابتدا اندیس اولین علامت # در رشته ی p_content را پیدا می کنیم و در h_index_curr قرار می دهیم.

سپس زیررشته ای از p_content که از اندیس h_index_curr+1 شروع می شود و تا انتهای آن ادامه پیدا میکند را در tmp قرار می دهیم.

حالا اندیس علامت # بعدی در رشته ی p_content ، یا به عبارتی اندیس اولین علامت # را در زیررشته ی tmp پیدا می کنیم و در h_index_next قرار می دهیم.

اگر h_index_next = 6 باشد به این معنی است که دقیقا ۵ کاراکتر بین دو علامت # وجود دارد که نشان دهنده ی معتبر بودن هشتگ است.

لاگ ایجاد آوای جدید – log_new_post

```
-- log_new_post
DELIMITER //
CREATE TRIGGER log_new_post
    AFTER INSERT
    ON post FOR EACH ROW
        BEGIN
            INSERT INTO `table_new_post`(`post_id`, `post_username`, `post_time`)
            VALUES (NEW.post_id, NEW.post_username, NEW.post_time);
        END
//
```

← توضیحات:

بعد از ایجاد هر آوای جدید، شماره آوا، نام کاربری کاربر ایجاد کننده و زمان ایجاد آوا را در جدول *table_new_post* ذخیره می کند.

- **رابط کاربری تحت کنسول**

کد این بخش با استفاده از زبان Java نوشته شده است و تمامی اطلاعات درباره ی چگونگی ورودی دادن در کنسول، از طریق وارد کردن دستور "help:" قابل مشاهده است.

```
help:
▷ List of statements :
   ▸ help:  => show list of statements and arguments

   ▸ create_an_account: <first_name> <last_name> <username> <pass> <birth_date> <biography>
   ▸ login: <username> <pass>
   ▸ get_user_logins:
   ▸ new_post: <post_content>
   ▸ get_user_posts:
   ▸ follow_user: <following_username>
   ▸ unfollow_user: <following_username>
   ▸ block_user: <blocking_username>
   ▸ unblock_user: <blocking_username>
   ▸ get_following_activities:
   ▸ get_user_activities: <username>
   ▸ commenting: <main_post_id> <type> <comment1_content> <comment1_id> <comment2_content>
   ▸ get_comments: <post_id>
   ▸ add_hashtag: <post_id> <hashtag_content>
   ▸ get_posts_with_specific_hashtag: <hashtag_content>
   ▸ like_posts: <post_id>
   ▸ get_number_of_likes: <post_id>
   ▸ get_list_of_like_username: <post_id>
   ▸ get_list_of_popular_posts:
   ▸ send_message: <receiver_username> <content_type> <text> <post_id>
   ▸ get_list_of_received_message_from_user: <username>
   ▸ get_list_of_sender_username:
```

هر دستور متناسب با قابلیت های سامانه نامگذاری شده است.

create_an_account:	**ایجاد حساب کاربری**
login:	**ورود به حساب کاربری**
get_user_logins:	**بررسی ورودهای کاربر**
new_post:	**ارسال آوا**
get_user_posts:	**دریافت آواهای شخصی**
follow_user:	**دنبال کردن**
unfollow_user:	**توقف دنبال کردن**
block_user:	**مسدود کردن**
unblock_user:	**آزاد کردن**
get_following_activities:	**دریافت فعالیت دنبال شوندگان**
get_user_activities:	**دریاقت فعالیت کاربران**
commenting:	**نظر دادن**
get_comments:	**دریافت نظرات آوا**
add_hashtag:	**اضافه کردن علامت ویژه به آوا**
get_posts_with_specific_hashtag:	**دریافت آواهای یک علامت ویژه**
like_posts:	**پسندیدن آوا**
get_number_of_likes:	**دریافت تعداد پسند ها**
get_list_of_like_username:	**دریافت لیست پسندکنندگان**
get_list_of_popular_posts:	**دریافت آواهای پرطرفدار**
send_message:	**ارسال پیام**
get_list_of_received_message_from_user:	**دریافت لیست پیام های دریافتی از کاربر**
get_list_of_sender_username:	**دریافت لیست ارسال کنندگان پیام**