# Final Report

TailorGuide

**Present to**

**Assoc. Prof. Dr. Taratip Suwannasart**

**Dr. Duangdao Wichadakul**

**Assoc. Prof. Dr. Proadpran Punyabukkana**

**Assoc. Prof. Dr. Chotirat Ratanamahatana**

**By**

**Group 11 : New Dragon**

| | |
|---|---|
| Jirapat Phetlertanan | 6130074121 |
| Natchanon Voraruth | 6131013021 |
| Nattapat Boonwirotrit | 6131014721 |
| Nattaphum Jampalee | 6131015321 |
| Natthawut Thongsai | 6131016021 |
| Thira Khunrak | 6131017621 |

**2110322, semester 1/2020**

**Department of Computer Engineering,**

**Faculty of Engineering, Chulalongkorn University**

# Contents

## Background of the project

Many people usually have problems with dressing clothes for their special events such as dating or meeting with some influenced people. To make them be more confident in their looks, we see a chance to help the other to solve these kinds of problems.

Therefore, we, the TailorGuide team, create a “TailorGuide” application to solve this problem. This application makes customers convenient to select stylists who they are interested in consulting about dressing for their special event. Also, this application provides online clothes shops where customers can choose to buy clothes from them. Moreover, TailorGuide also has a blog system, where customers and stylists can make a discussion, recommend clothes, also, stylists and clothes shops can make an advertised blog.

The chatting system provides a chat between customers and stylists to consult or customer and clothes shop to ask for the product’s information, which we record the conversation and keep it security, privacy, and traceback ability.

Online shop systems, each shop has a lot of clothes which we need to store clothes information. Additionally, we need to store the history of the customer's order to trace back.

Furthermore, we need to store user’s profile information, such as SSN, first name, last name, address. Since our system needs to store a lot of information with high security, so we use a database system for our project.

## Objective

1. To create a database system that can handle information of the TailorGuide system effectively.
2. To increase security and privacy of user’s data.
3. To make the system database easy to manage.
4. To make the access to the database more accurate.

## System functionalities

1. Registration system
   1.1. The system shall allow users to create a new account with a unique username on the system, password, user type (customer, stylist or shop).
   1.2. The system shall allow users to verify their identity via phone number and identification card to proceed.

2. Account management system

   2.1. The system shall allow customers to manage their profile with following detail :
   - Public profile
     - Profile picture
     - First Name
     - Last Name
   - Private profile
     - Birth Date
     - Sex
     - Height
     - Weight
     - Bust
     - Hip
     - Waist
     - Closet
     - Phone number

   2.2. The system shall allow stylists to manage their profile.
   2.3. The system shall allow shops to manage their profile.

3. System for users
   3.1. The system shall allow users to login from the system with following methods :
   - validates username and password when users try to login.
   - classify type of user (customer, stylist or shop).

   3.2. The system shall allow users to logout from the system.
   3.3. The system shall allow users to view their profile.

4. Consulting system
    4.1. The system shall allow customers to search stylists and view their profile and rating.
    4.2. The system shall allow customers to request the consultation to the stylist which lasts for 1 hour.
    4.3. The system shall allow stylists to accept or reject customers.
    4.4. The system shall cancel all customer’s requests when one of the stylists accepts the request.
    4.5. The system shall allow customers to consult with the stylist they matched via personal chat.
    4.6. The system shall allow the stylist to view the matched customer's private profile while consulting.
    4.7. The system shall allow the stylist to end the consultation.
    4.8. The system shall end the consultation in 12 hrs.
    4.9. The system shall allow customers to review stylists with their own satisfaction.

5. Shopping system
    5.1. The system shall allow customers to search products or shops and view their item and rating.
    5.2. The system shall allow customers to chat with the shop.
    5.3. The system shall allow customers to order the item from the shop.
    5.4. The system shall allow customers to make payment for the order with the method we provide.
    5.5. The system shall allow customers to review shops with their own satisfaction.

6. Blog system
    6.1. The system shall allow users to post in community blogs.
    6.2. The system shall allow stylists to post recommendation blogs.
    6.3. The system shall allow stylists and shops to post advertisement blogs when they pay.
    6.4. The system shall allow all blogs to show in blog feed.
    6.5. The system shall allow users to comment in blogs.

7. Payment System
    7.1. The system shall allow users to update their debit card with following information:

- Cardholder’s name
- Card type
- 16 digits card number
- Expiration date
- CVV

7.2. The system shall allow customers to pay for their consultation cost or order via debit card to TailorGuide’s bank account.
7.3. The system shall transfer 80% of consultation cost to stylist.
7.4. The system shall allow stylists and shops to pay for advertising expenses via debit card to TailorGuide’s bank account.
7.5. The system shall transfer 80% of order payment to shop after customers confirm the delivery.
7.6. The system shall transfer 80% of order payment to shop if customer didn’t confirm the delivery for 7 days after the delivery status is complete.

## ER diagram

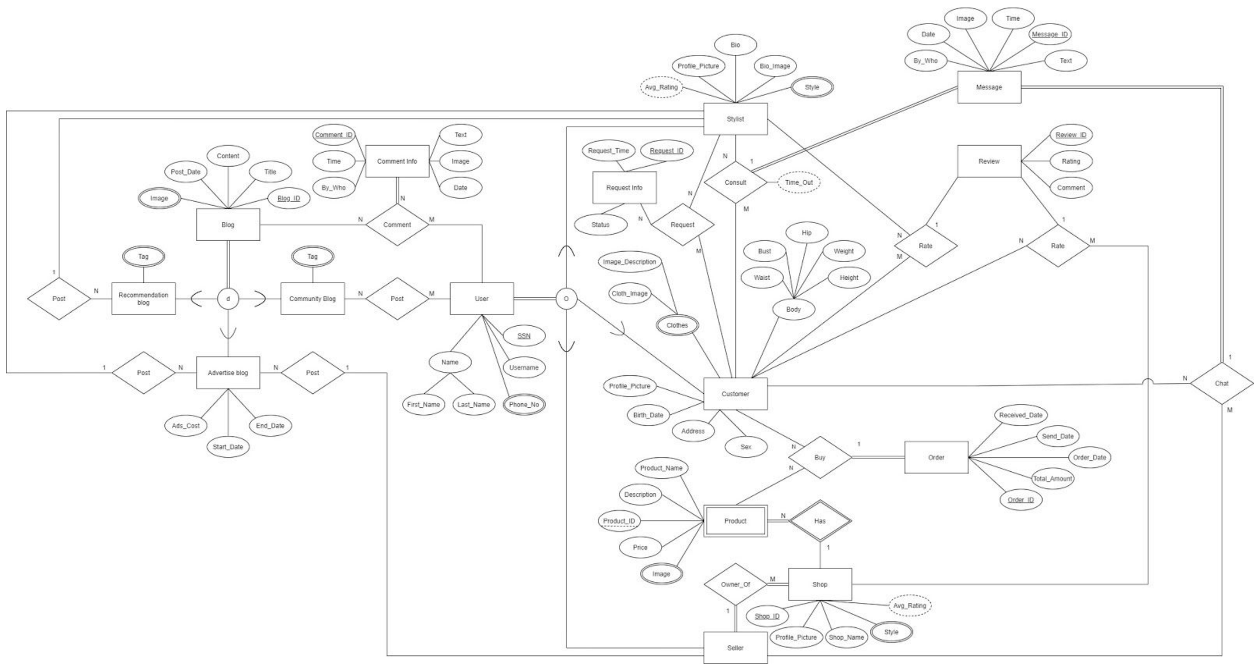


Figure 1 : ER diagram

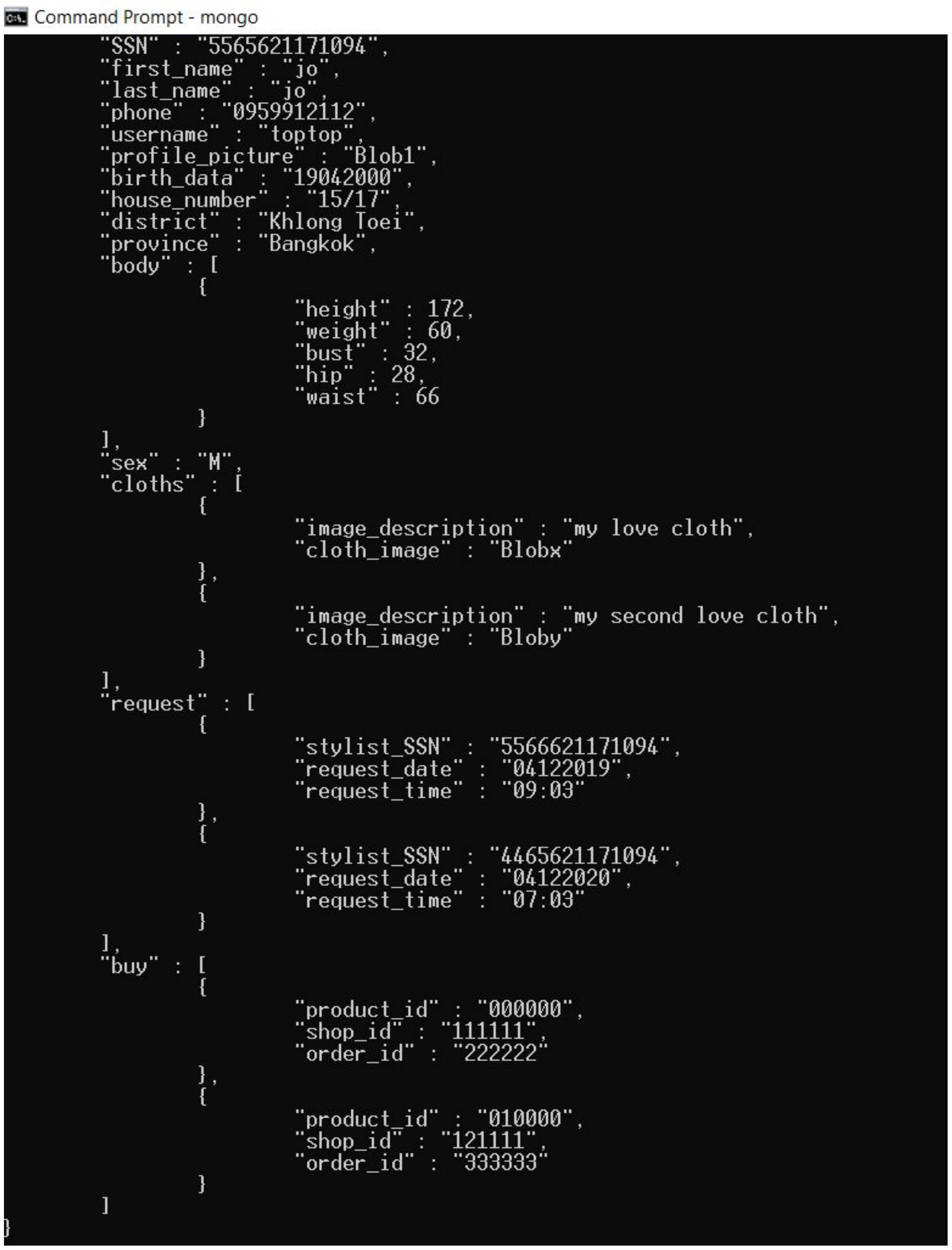
https://drive.google.com/file/d/1FBgBrHSbONjts4n79WO0sN-egxB57B0_/view?usp=sharing

## Document-based design schema

- **Customer description**

```
{
        SSN:SSN_value,
        first_name:FIRST_NAME_value,
        last_name:LAST_NAME_value,
        phone:PHONE_value,
        username:USER_NAME_value,
        profile_picture:PROFILE_PICTURE_value,
        birth_data:BIRTH_DATA_value,
        house_number:HOUSE_NUMBER_value,
        district:DISTRICT_value,
        province:PROVINCE_value,
        sex:SEX_value,
        body:[
                {
                        height:HEIGHT_value,
                        weight:WEIGHT_value,
                        bust:BUST_value,
                        hip:HIP_value,
                        waist:WAIST_value
                }
        ],
        cloths:[
                {
```

```
        image_description:IMAGE_DESCRIPTION_value,
        cloth_image:CLOTH_IMAGE_value
        },
        {
        image_description:IMAGE_DESCRIPTION_value,
        cloth_image:CLOTH_IMAGE_value
        },....
],

request:[
        {
                stylist_SSN:SSN_value,
                request_date:REQUEST_DATE_value,
                request_time:REQUEST_TIME_value
        },
        {
                stylist_SSN:SSN_value,
                request_date:REQUEST_DATE_value,
                request_time:REQUEST_TIME_value
        },....
],

buy:[
        {
                product_id:PRODUCT_ID_value,
                shop_id:'SHOP_ID_value,
                order_id:ORDER_ID_value
            },
            {
                product_id:PRODUCT_ID_value,
                shop_id:'SHOP_ID_value,
                order_id:ORDER_ID_value
            },....
        ]
    }
```



Command Prompt - mongo

```
        "SSN" : "5565621171094",
        "first_name" : "jo",
        "last_name" : "jo",
        "phone" : "0959912112",
        "username" : "toptop",
        "profile_picture" : "Blob1",
        "birth_data" : "19042000",
        "house_number" : "15/17",
        "district" : "Khlong Toei",
        "province" : "Bangkok",
        "body" : [
                {
                        "height" : 172,
                        "weight" : 60,
                        "bust" : 32,
                        "hip" : 28,
                        "waist" : 66
                }
        ],
        "sex" : "M",
        "cloths" : [
                {
                        "image_description" : "my love cloth",
                        "cloth_image" : "Blobx"
                },
                {
                        "image_description" : "my second love cloth",
                        "cloth_image" : "Bloby"
                }
        ],
        "request" : [
                {
                        "stylist_SSN" : "5566621171094",
                        "request_date" : "04122019",
                        "request_time" : "09:03"
                },
                {
                        "stylist_SSN" : "4465621171094",
                        "request_date" : "04122020",
                        "request_time" : "07:03"
                }
        ],
        "buy" : [
                {
                        "product_id" : "000000",
                        "shop_id" : "111111",
                        "order_id" : "222222"
                },
                {
                        "product_id" : "010000",
                        "shop_id" : "121111",
                        "order_id" : "333333"
                }
        ]
}
```

This picture shows a sample of Customer’s data that is normalized.

## Schema diagram

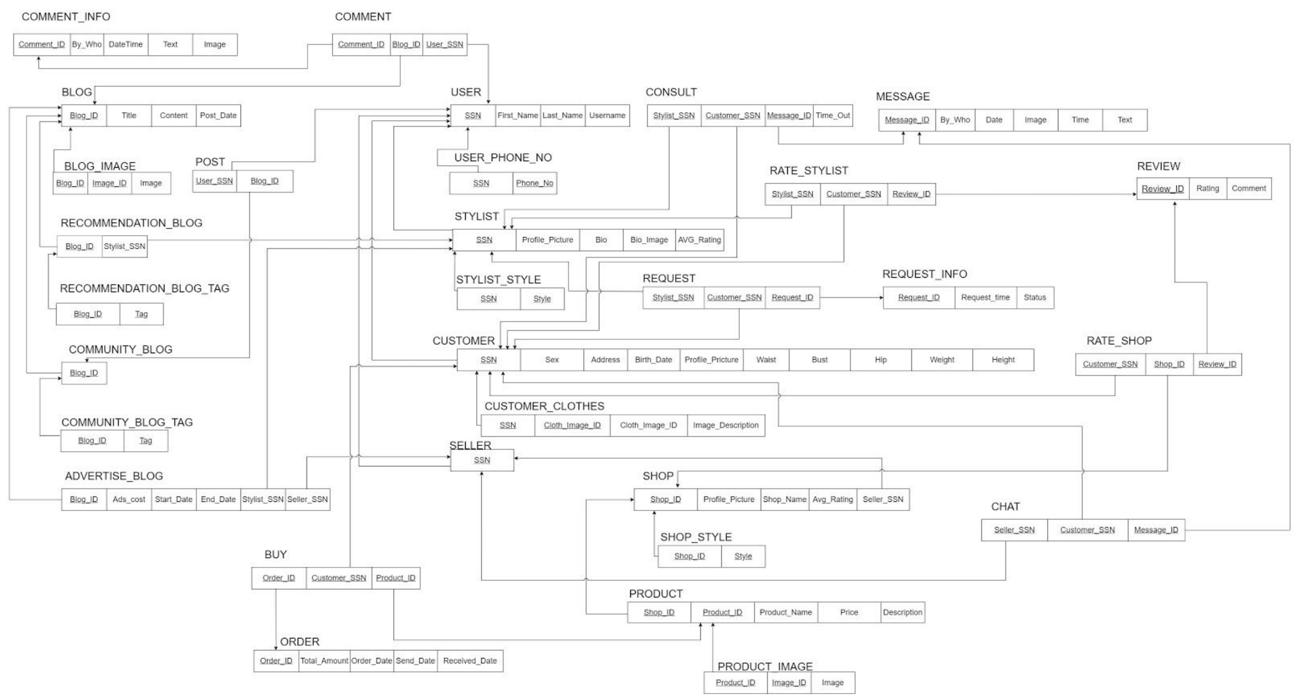

COMMENT_INFO
Comment_ID | By_Who | DateTime | Text | Image
COMMENT
Comment_ID | Blog_ID | User_SSN
BLOG
Blog_ID | Title | Content | Post_Date
USER
SSN | First_Name | Last_Name | Username
CONSULT
Stylist_SSN | Customer_SSN | Message_ID | Time_Out
MESSAGE
Message_ID | By_Who | Date | Image | Time | Text
BLOG_IMAGE
Blog_ID | Image_ID | Image
POST
User_SSN | Blog_ID
USER_PHONE_NO
SSN | Phone_No
RATE_STYLIST
Stylist_SSN | Customer_SSN | Review_ID
REVIEW
Review_ID | Rating | Comment
RECOMMENDATION_BLOG
Blog_ID | Stylist_SSN
STYLIST
SSN | Profile_Picture | Bio | Bio_Image | AVG_Rating
STYLIST_STYLE
SSN | Style
REQUEST
Stylist_SSN | Customer_SSN | Request_ID
REQUEST_INFO
Request_ID | Request_time | Status
RECOMMENDATION_BLOG_TAG
Blog_ID | Tag
COMMUNITY_BLOG
Blog_ID
CUSTOMER
SSN | Sex | Address | Birth_Date | Profile_Pricture | Waist | Bust | Hip | Weight | Height
RATE_SHOP
Customer_SSN | Shop_ID | Review_ID
CUSTOMER_CLOTHES
SSN | Cloth_Image_ID | Cloth_Image_ID | Image_Description
COMMUNITY_BLOG_TAG
Blog_ID | Tag
SELLER
SSN
ADVERTISE_BLOG
Blog_ID | Ads_cost | Start_Date | End_Date | Stylist_SSN | Seller_SSN
SHOP
Shop_ID | Profile_Picture | Shop_Name | Avg_Rating | Seller_SSN
CHAT
Seller_SSN | Customer_SSN | Message_ID
SHOP_STYLE
Shop_ID | Style
BUY
Order_ID | Customer_SSN | Product_ID
PRODUCT
Shop_ID | Product_ID | Product_Name | Price | Description
ORDER
Order_ID | Total_Amount | Order_Date | Send_Date | Received_Date
PRODUCT_IMAGE
Product_ID | Image_ID | Image

Figure 2 : schema diagram
https://drive.google.com/file/d/1uRwxooRmv2wy6JLp3tc0oBfxC42kfW-P/view?usp=sharing

## Normalization

The schema diagram above was made by ER-to-Relational Mapping Algorithm which reduces the redundancy that may occur in the database system, has the decomposition in step 6 : Mapping of Multivalued attributes, and also has an efficient relationship mapping which makes no functional dependencies or no transitive functional dependencies in the database system.

Since the schema diagram has no composition of attributes, no functional dependencies and no transitive functional dependencies, we can conclude that this schema diagram is already in Third Normal Form (3NF).

## Data dictionary of entity and relationship type

### Entity Type

| No. | Entity Name | Description |
|---|---|---|
| **1** | User | User information |
| **2** | Customer | Customer information |
| **3** | Stylist | Stylist information |
| **4** | Shop | Shop information |
| **5** | Order | Order information |
| **6** | Product | Product information |
| **7** | Blog | Post detail |
| **8** | Community Blog | Discussion blog information |
| **9** | Recommendation Blog | Information about recommendation blog by stylists |
| **10** | Advertise Blog | Information about advertise blog by stylists or shop |
| **11** | Review | Review detail |
| **12** | Message | Message content and detail |
| **13** | Comment Info | Comment Information |
| **14** | Request Info | Consult request information |

**Relationship Type**

| No. | Entity Name | Description |
|---|---|---|
| **15** | Consult | Consult information |

**Entity Type Name: User**

| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
|---|---|---|---|---|
| SSN | Char(13) | Social Security number | | No |
| Username | Varchar(20) | Username of user | | No |
| First_Name | Varchar(30) | First name of user | | No |
| Last_Name | Varchar(30) | Last name of user | | No |
| Phone_No | Char(10) | Phone number of the user | | No |

**Entity Type Name: Customer**

| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
|---|---|---|---|---|
| Profile_Picture | BLOB | Profile picture of the customer | | Yes |
| Birth_Date | Date | Birthdate of the customer | | Yes |
| Address | Varchar(100) | Address of the customer | | No |
| Height | Float(5) | Height of customer | xx.xxx | Yes |
| Weight | Float(5) | Weight of customer | xx.xxx | Yes |
| Sex | Varchar(50) | Sex of customer | | Yes |
| Bust | Float(5) | Bust size in centimeters | xx.xxx | Yes |
| Waist | Float(5) | Waist size in centimeters | xx.xxx | Yes |
| Hip | Float(5) | Hip size in centimeters | xx.xxx | Yes |
| Image_Description (complex) (Multivalue) | Varchar(1000) | Description of cloth images | | Yes |
| Cloth_Image (complex) (Multivalue) | LONGBLOB | Image of cloth in closet | | Yes |

**Entity Type Name: Stylist**

| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
|---|---|---|---|---|
| Profile_Picture | BLOB | Profile picture of the stylist | | Yes |
| Style (Multivalue) | Varchar(30) | A specific style of the stylist | | No |
| Bio | Varchar(1000) | A short biography about stylist | | Yes |
| Bio_Image | LONGBLOB | Profile image of stylist | | Yes |
| Avg_Rating (derived) | Float | Average rating score | | No |

**Entity Type Name: Shop**

| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
|---|---|---|---|---|
| <u>Shop_ID</u> | Char(10) | ID of shop | | No |
| Profile_Picture | BLOB | Profile picture of shop | | Yes |
| Shop_Name | Varchar(30) | Name of Shop | | No |
| Shop_Type (Multivalue) | Varchar(30) | Type of cloth that sells in shops. | | No |
| Avg_Rating | Float | Average rating score | | No |

**Entity Type Name: Order**

| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
|---|---|---|---|---|
| <u>Order_ID</u> | Char(14) | Purchase code to identify the order | | No |
| Order_Date | Date | Date that this order had been placed. | | No |
| Send_Date | Date | Date that product had been sent to courier | | No |
| Recived_Date | Date | Date that customer received the product | | Yes |
| Total_Amount | Int | Total amount of product in the order | | No |

**Entity Type Name: Product**

| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
|---|---|---|---|---|
| Product_ID | Char(10) | ID of product | | No |
| Product_Name | Varchar(30) | Name of Product | | No |
| Price | Int | Price of this product | Positive number | No |
| Description | Varchar(100) | Description about product | | No |
| Image (Multivalue) | LONGBLOB | Picture of product | | No |

**Entity Type Name: Blog**

| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
|---|---|---|---|---|
| Blog_ID | Char(10) | ID of blog | | No |
| Title | Varchar(30) | Title of blog | | No |
| Content | Varchar(500) | Content that are in the blog | | No |
| Post_Date | Date | Date that blog had been posted | | No |
| Image (Multivalue) | LONGBLOB | An image that inserts in blog | | Yes |

**Entity Type Name: Community Blog**

| Attribute Name | Type | Descriptive Name | Valid Values | Allow Nulls |
|---|---|---|---|---|
| Tag (multivalue) | Varchar(100 ) | Category of content | | Yes |

**Entity Type Name: Recommendation Blog**

| Attribute Name | Type | Descriptive Name | Valid Values | Allow Nulls |
|---|---|---|---|---|
| Tag (multivalue) | Varchar(100) | Category of content | | **Yes** |

**Entity Type Name: Advertise Blog**

| Attribute Name | Type | Descriptive Name | Valid Values | Allow Nulls |
|---|---|---|---|---|
| Ads_Cost | Int | Advertisement fee | | No |
| Start_Date | Date | The date of advertising that had been post | | No |
| End_Date | Date | Advertise's expired date | | No |

**Entity Type Name: Review**

| Attribute Name | Type | Descriptive Name | Valid Values | Allow Nulls |
|---|---|---|---|---|
| Review_ID | Char(10) | ID of rating | | No |
| Rating | Float | Rating score that customer gives to stylist or shop | 0.0 - 5.0 | No |
| Comment | Varchar(1000) | Rating description | | Yes |

**Entity Type Name: Message**

| Attribute Name | Type | Descriptive Name | Valid Values | Allow Nulls |
|---|---|---|---|---|
| Message_ID | Char(10) | ID of message | | No |
| By_Who | Char(13) | Message had sent by who | | No |
| Text | Varchar(1000) | Text content of message | | Yes |
| Image | BLOB | Image in message | | Yes |
| Date | Date | Date that message had been sent | | No |
| Time | Time | Time that message had been sent | | No |

**Entity Type Name: Comment Info**

| Attribute Name | Type | Descriptive Name | Valid Values | Allow Nulls |
|---|---|---|---|---|
| Comment_ID | Char(10) | ID of comment | | No |
| By_Who | Char(10) | Comment has been post by who | | No |
| Text | Varchar(1000) | Text content of comment | | Yes |
| Image | BLOB | Image in comment | | Yes |
| Date | Date | Date that comment has been post | | No |
| Time | Time | Time that comment has been post | | No |

**Entity Type Name: Request Info**

| Attribute Name | Type | Descriptive Name | Valid Values | Allow Nulls |
|---|---|---|---|---|
| Request_ID | Date | ID of request | | No |
| Request_Time | Time | Time that request has been send | | No |
| Status | Varchar(20) | Status of the request | | Yes |

**Relationship Type Name: Consult**

| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
|---|---|---|---|---|
| Time_Out | Time | Time that consult will end |  | No |

## Indexing

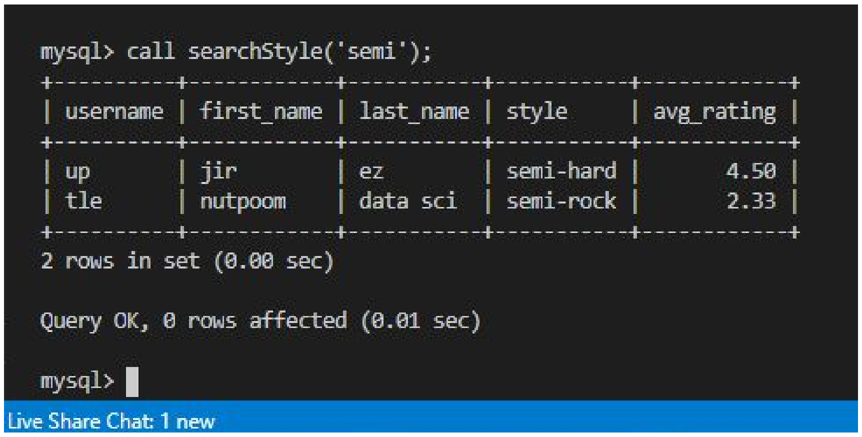
Relation:Stylist
Search key: avg_rating
Index structure: B+ Tree indexes
เพราะว่าเราจะทำการค้นหาเป็นช่วง จึงใช้ B+ Tree ที่เหมาะแก่การทำ Range search เช่น “ WHERE avg_rating > 4 ”

## Stored procedures

### searchStyle procedure

Show up to 3 stylists that their style contains the keyword that we input order by their average rating.

```
DELIMITER //
CREATE PROCEDURE searchStyle (IN name varchar(30))
    BEGIN
        SELECT DISTINCT username, first_name, last_name, style, avg_rating
        from Usert natural join Stylist natural join Stylist_style
        where style LIKE CONCAT('%',name,'%')
        ORDER BY avg_rating DESC
        LIMIT 3;
    END //
DELIMITER ;
```



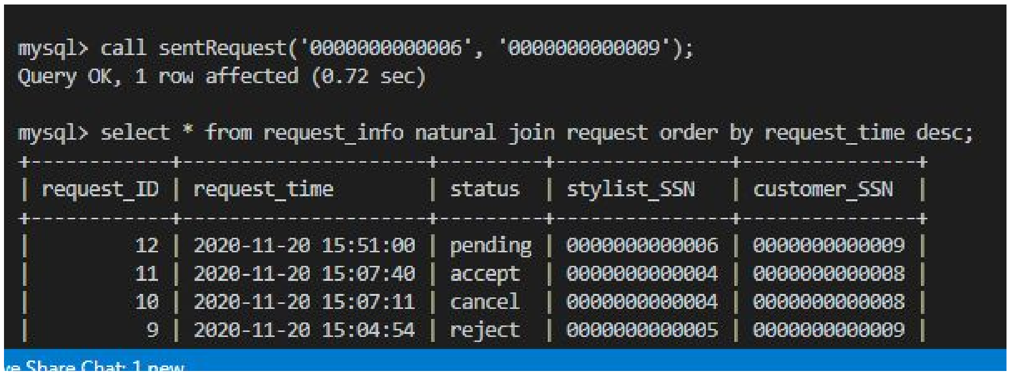
```
mysql> call searchStyle('semi');
+----------+------------+-----------+-----------+------------+
| username | first_name | last_name | style     | avg_rating |
+----------+------------+-----------+-----------+------------+
| up       | jir        | ez        | semi-hard |       4.50 |
| tle      | nutpoom    | data sci  | semi-rock |       2.33 |
+----------+------------+-----------+-----------+------------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql>
```

Figure 3 : call searchStyle procedure and example of result

**sentRequest procedure**

Insert consultation request information from customer (cus) to stylist (sty) that we input to sentRequest() into Request_info and Request table.

```
DELIMITER //
CREATE PROCEDURE sentRequest(IN sty CHAR(13), IN cus CHAR(13))
        BEGIN
                INSERT INTO Request_info(request_ID, request_time, status) VALUES
                        (null, now(), DEFAULT);

                INSERT INTO Request(stylist_SSN, customer_SSN, request_ID) VALUES
                        (sty, cus, lastid());
        END //
DELIMITER ;
```



```
mysql> call sentRequest('0000000000006', '0000000000009');
Query OK, 1 row affected (0.72 sec)

mysql> select * from request_info natural join request order by request_time desc;
+------------+---------------------+---------+---------------+---------------+
| request_ID | request_time        | status  | stylist_SSN   | customer_SSN  |
+------------+---------------------+---------+---------------+---------------+
|         12 | 2020-11-20 15:51:00 | pending | 0000000000006 | 0000000000009 |
|         11 | 2020-11-20 15:07:40 | accept  | 0000000000004 | 0000000000008 |
|         10 | 2020-11-20 15:07:11 | cancel  | 0000000000004 | 0000000000008 |
|          9 | 2020-11-20 15:04:54 | reject  | 0000000000005 | 0000000000009 |
```

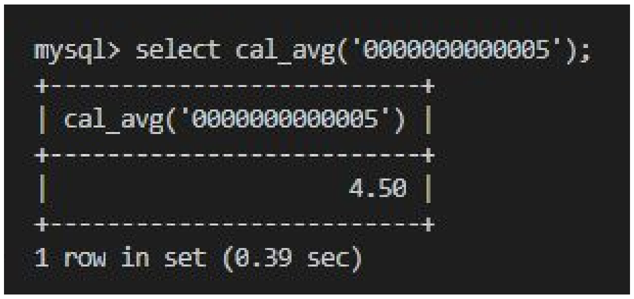
Figure 4 : call sentRequest procedure and example of result

## Stored functions

### cal_avg function

Calculate the average rating of the stylist (s) that we input to cal_avg() that we will use to update the avg_rating in the stylist table.

```
DELIMITER //
CREATE FUNCTION cal_avg(s CHAR(13)) RETURNS float(3,2)
   DETERMINISTIC READS SQL DATA
   BEGIN
      DECLARE result float(3,2);
      SET result = (
         SELECT avg(rating)
         FROM  rate_stylist natural join review
         WHERE stylist_SSN = s
         GROUP by stylist_SSN
      );
      RETURN result;
   END //
DELIMITER ;
```



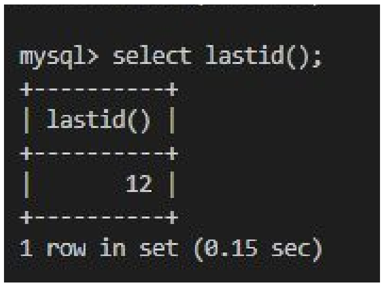
```
mysql> select cal_avg('0000000000005');
+--------------------------+
| cal_avg('0000000000005') |
+--------------------------+
|                     4.50 |
+--------------------------+
1 row in set (0.39 sec)
```

Figure 5 : example of cal_avg function use and result

**lastid function**

return latest request ID from request_info table to use with request table.

```
DELIMITER //
CREATE FUNCTION lastid() RETURNS MEDIUMINT
   DETERMINISTIC READS SQL DATA
   BEGIN
      DECLARE result MEDIUMINT;
      SET result = (
         SELECT max(request_ID)
         FROM request_info
      );
      RETURN result;
   END //
DELIMITER ;
```



```
mysql> select lastid();
+----------+
| lastid() |
+----------+
|       12 |
+----------+
1 row in set (0.15 sec)
```

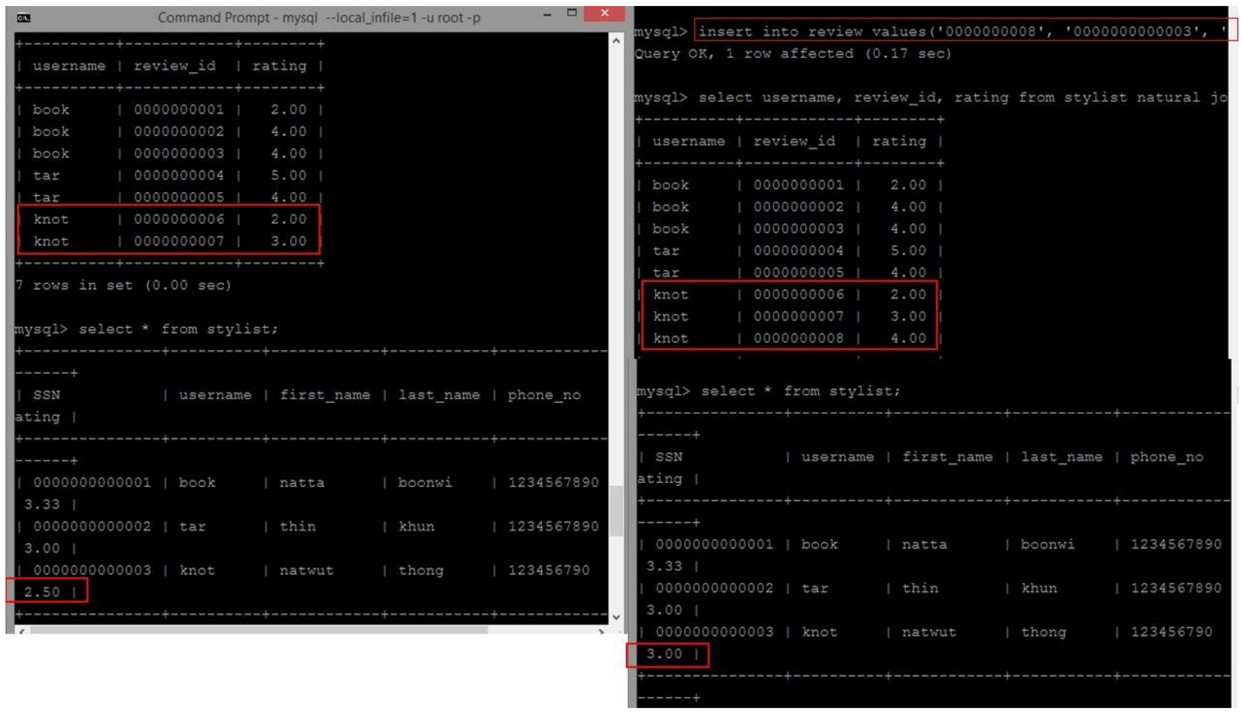
Figure 6 : example of lastid function use and result

## Triggers

### update_rating trigger

When a customer adds a new review for a stylist. The trigger will make an update to the avg_rating attribute in the stylist table, because we need to update the avg_rating every time that customer makes a review.

```
DELIMITER $$
CREATE TRIGGER update_rating
AFTER INSERT
ON rate_stylist FOR EACH ROW
BEGIN
        CALL updateRating(NEW.stylist_SSN);
END$$
DELIMITER ;
```



```
Command Prompt - mysql  --local_infile=1 -u root -p
+----------+------------+--------+
| username | review_id  | rating |
+----------+------------+--------+
| book     | 0000000001 |   2.00 |
| book     | 0000000002 |   4.00 |
| book     | 0000000003 |   4.00 |
| tar      | 0000000004 |   5.00 |
| tar      | 0000000005 |   4.00 |
| knot     | 0000000006 |   2.00 |
| knot     | 0000000007 |   3.00 |
+----------+------------+--------+
7 rows in set (0.00 sec)

mysql> select * from stylist;
+---------------+----------+------------+-----------+-----------
------+
| SSN           | username | first_name | last_name | phone_no
ating |
+---------------+----------+------------+-----------+-----------
------+
| 0000000000001 | book     | natta      | boonwi    | 1234567890
 3.33 |
| 0000000000002 | tar      | thin       | khun      | 1234567890
 3.00 |
| 0000000000003 | knot     | natwut     | thong     | 123456790
 2.50 |
+---------------+----------+------------+-----------+-----------
```

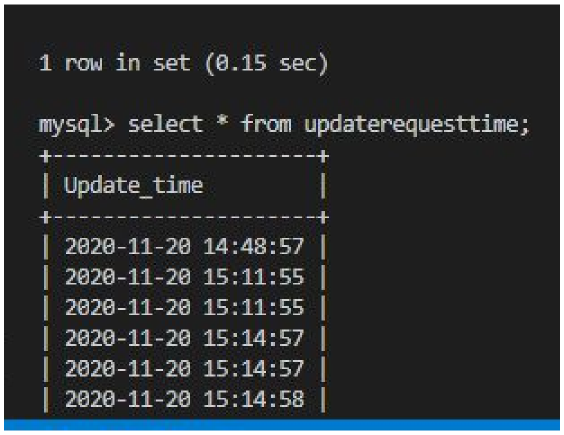
```
mysql> insert into review values('0000000008', '0000000000003', '
Query OK, 1 row affected (0.17 sec)

mysql> select username, review_id, rating from stylist natural jo
+----------+------------+--------+
| username | review_id  | rating |
+----------+------------+--------+
| book     | 0000000001 |   2.00 |
| book     | 0000000002 |   4.00 |
| book     | 0000000003 |   4.00 |
| tar      | 0000000004 |   5.00 |
| tar      | 0000000005 |   4.00 |
| knot     | 0000000006 |   2.00 |
| knot     | 0000000007 |   3.00 |
| knot     | 0000000008 |   4.00 |

mysql> select * from stylist;
+---------------+----------+------------+-----------+-----------
------+
| SSN           | username | first_name | last_name | phone_no
ating |
+---------------+----------+------------+-----------+-----------
------+
| 0000000000001 | book     | natta      | boonwi    | 1234567890
 3.33 |
| 0000000000002 | tar      | thin       | khun      | 1234567890
 3.00 |
| 0000000000003 | knot     | natwut     | thong     | 123456790
 3.00 |
+---------------+----------+------------+-----------+-----------
------+
```

Figure 7 : example of update_rating trigger being used

**requestLOG trigger**

We have to save the time stamp every time stylists accept or reject requests for data to work with the consult table.

```
DELIMITER $$
CREATE TRIGGER requestLOG
AFTER UPDATE
ON request_info FOR EACH ROW
BEGIN
        INSERT INTO UpdateRequestTime values(now());
END$$
DELIMITER ;
```



```
1 row in set (0.15 sec)

mysql> select * from updaterequesttime;
+---------------------+
| Update_time         |
+---------------------+
| 2020-11-20 14:48:57 |
| 2020-11-20 15:11:55 |
| 2020-11-20 15:11:55 |
| 2020-11-20 15:14:57 |
| 2020-11-20 15:14:57 |
| 2020-11-20 15:14:58 |
```

Figure 8 : example of requestLOG trigger being used

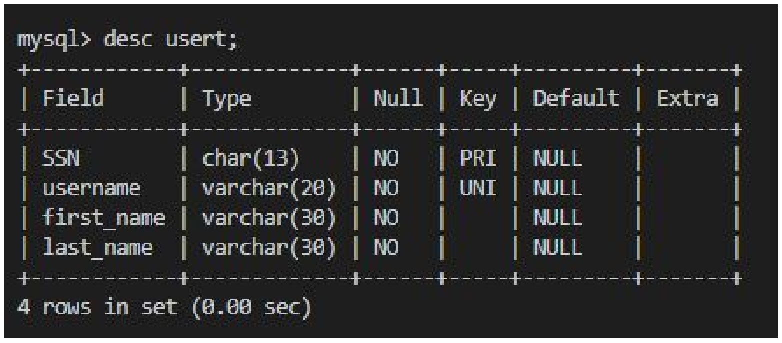
**addStylist transaction**

When stylist fill SSN with a wrong format, the database will be rollback but if stylist fill SSN with the right one, the database will commit the data.

```
        connection.autocommit = False
        objdata = (wdata[0],wdata[5])
        objdata1 = (wdata[0],wdata[6])
        objdata2 = (wdata[0],wdata[4])
        objdata3 = (wdata[0],wdata[1],wdata[2],wdata[3])
        sqlQuery3 = "insert into usert (SSN,username,first_name,last_name) values
(%s,%s,%s,%s)"
        cursor.execute(sqlQuery3, objdata3)
        sqlQuery2 = "insert into user_phone_no (SSN,phone_no) values (%s,%s)"
        cursor.execute(sqlQuery2, objdata2)
        sqlQuery = "insert into stylist(SSN,profile_pic,bio,bio_image,avg_rating) values
(%s,null,%s,null,3.00)"
        cursor.execute(sqlQuery, objdata)
        sqlQuery1 = "insert into stylist_style (SSN,style) values (%s,%s)"
        cursor.execute(sqlQuery1, objdata1)

        if(len(wdata[0])==13):
           connection.commit()
           state = 0
        connection.rollback()
```

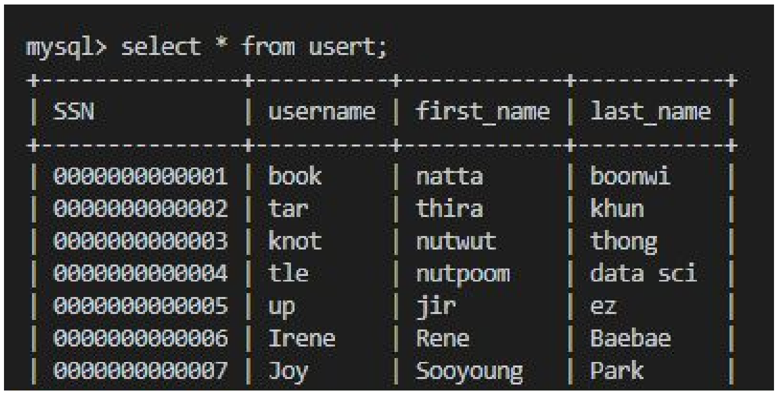
## Integrity

In the Usert table we use SSN as a primary key and we set the username to be unique. As shown in a figure below.



```
mysql> desc usert;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| SSN        | char(13)    | NO   | PRI | NULL    |       |
| username   | varchar(20) | NO   | UNI | NULL    |       |
| first_name | varchar(30) | NO   |     | NULL    |       |
| last_name  | varchar(30) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

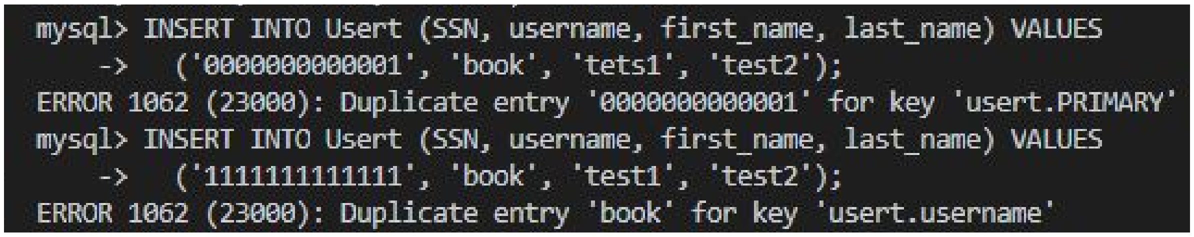
Figure 9 : describe of stylist table



```
mysql> select * from usert;
+---------------+----------+------------+-----------+
| SSN           | username | first_name | last_name |
+---------------+----------+------------+-----------+
| 0000000000001 | book     | natta      | boonwi    |
| 0000000000002 | tar      | thira      | khun      |
| 0000000000003 | knot     | nutwut     | thong     |
| 0000000000004 | tle      | nutpoom    | data sci  |
| 0000000000005 | up       | jir        | ez        |
| 0000000000006 | Irene    | Rene       | Baebae    |
| 0000000000007 | Joy      | Sooyoung   | Park      |
```

Figure 10 : the current data in stylist table

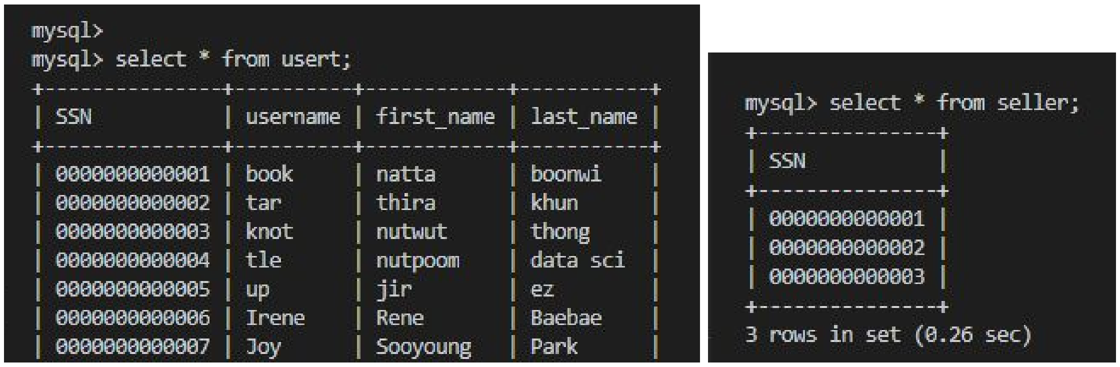
So, it will have an error when we try to insert the Usert that SSN had already existed or insert stylist that has the username that already exists like in the figure below.



```
mysql> INSERT INTO Usert (SSN, username, first_name, last_name) VALUES
    ->   ('0000000000001', 'book', 'tets1', 'test2');
ERROR 1062 (23000): Duplicate entry '0000000000001' for key 'usert.PRIMARY'
mysql> INSERT INTO Usert (SSN, username, first_name, last_name) VALUES
    ->   ('1111111111111', 'book', 'test1', 'test2');
ERROR 1062 (23000): Duplicate entry 'book' for key 'usert.username'
```

Figure 11 : result when trying to insert the duplicate SSN or username

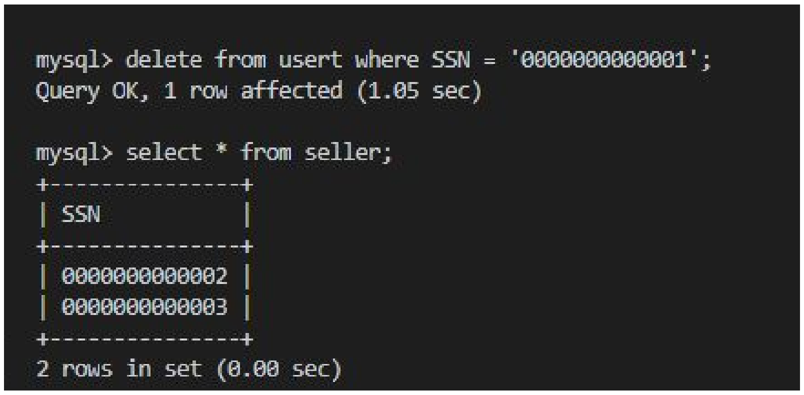
Also, we have foreign key between SSN of Usert and seller that set on delete cascade. So, if we delete the user in the Usert table, the seller with the same SSN will also be deleted from the seller table.



```
mysql>
mysql> select * from usert;
+---------------+----------+------------+-----------+
| SSN           | username | first_name | last_name |
+---------------+----------+------------+-----------+
| 0000000000001 | book     | natta      | boonwi    |
| 0000000000002 | tar      | thira      | khun      |
| 0000000000003 | knot     | nutwut     | thong     |
| 0000000000004 | tle      | nutpoom    | data sci  |
| 0000000000005 | up       | jir        | ez        |
| 0000000000006 | Irene    | Rene       | Baebae    |
| 0000000000007 | Joy      | Sooyoung   | Park      |
```

```
mysql> select * from seller;
+---------------+
| SSN           |
+---------------+
| 0000000000001 |
| 0000000000002 |
| 0000000000003 |
+---------------+
3 rows in set (0.26 sec)
```

Figure 12 : current data in Usert and seller table



```
mysql> delete from usert where SSN = '0000000000001';
Query OK, 1 row affected (1.05 sec)

mysql> select * from seller;
+---------------+
| SSN           |
+---------------+
| 0000000000002 |
| 0000000000003 |
+---------------+
2 rows in set (0.00 sec)
```

Figure 13 : data after we delete Usert SSN '0000000000001'

## Execution path

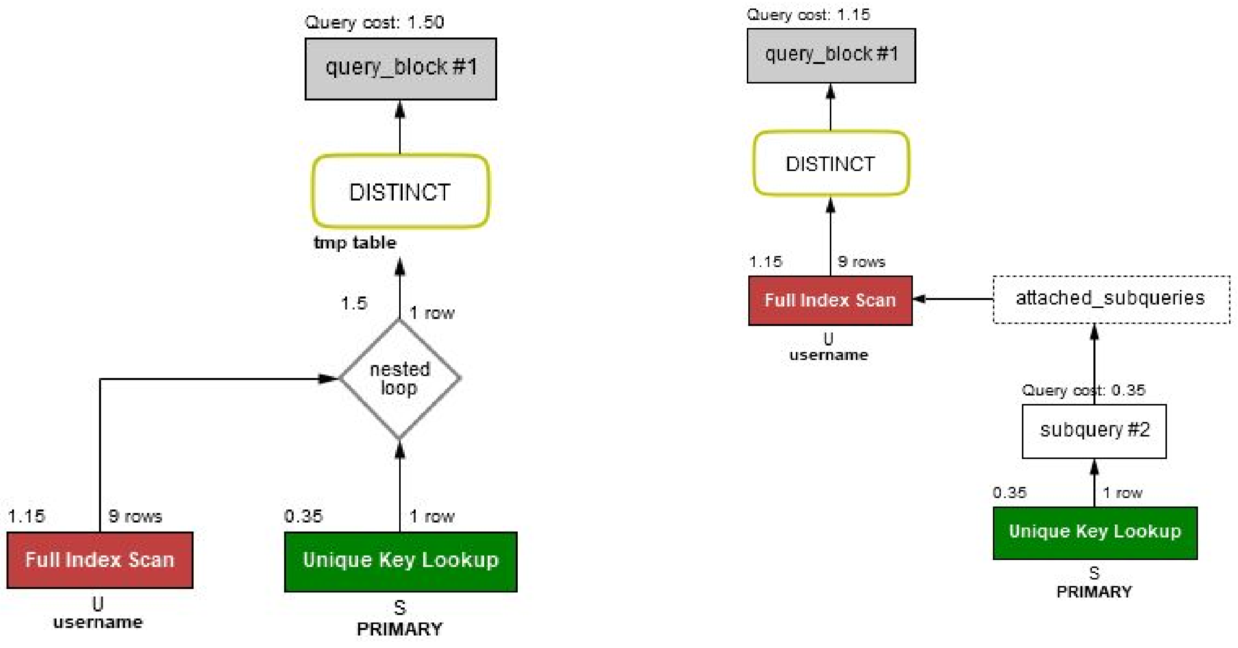
Both procedures show the stylist that their username contains the keyword.

(a.)

```
SELECT DISTINCT U.username
   from Usert U natural join Stylist S
   where U.username LIKE CONCAT('%',"Irene",'%');
```

(b.)

```
SELECT DISTINCT U.username
FROM Usert U
WHERE U.SSN = (SELECT S.SSN
   FROM Stylist S
   WHERE U.SSN = S.SSN
   )
   AND
   U.username LIKE CONCAT('%',"Irene",'%');
```



(a) (b)

Figure 14 : result of both procedures

## Complex query

### searchRe

Show all records of the stylist’s consultation requests.

```
DELIMITER //
CREATE PROCEDURE searchRe (IN name varchar(30))
        BEGIN
        SELECT Re.request_id,Ui.username,Re.status
        from Usert Ui, (SELECT *
                from Usert U
                inner join request R
                on U.SSN = R.stylist_SSN
                natural join request_info RI
                where R.request_id = RI.request_id) as Re
        where Re.username = name and Re.customer_SSN = Ui.SSN;
        END //
DELIMITER ;
```

### Find_Top_Ten

Show the top ten customers that spent the most money by buying products in our system.

```
DELIMITER //
CREATE PROCEDURE Find_Top_Ten()
        BEGIN
        SELECT U.SSN,username,first_name,last_name,total from usert U
        RIGHT JOIN (SELECT *
         FROM (
                SELECT SSN , SUM(total_amount) AS total
                        FROM ((customer c
                INNER JOIN buy b
                ON c.SSN = b.customer_SSN)
                INNER JOIN Ordert o
                ON b.order_id = o.order_id)
                GROUP BY c.SSN
         ) AS T
         ORDER BY T.total DESC
```

```
     LIMIT 10) AS Mx ON U.SSN = Mx.SSN;


    END //
DELIMITER ;
```

### Find_top_3_customer

Find top 3 royalty customers that have the most number of requests.

```
DELIMITER //
CREATE PROCEDURE Find_top_3_customer()
    BEGIN
SELECT *
     FROM (
          SELECT SSN , COUNT(*) AS total_count
               FROM (customer c
          INNER JOIN request r
          ON c.SSN = r.customer_SSN)
          GROUP BY c.SSN
     ) AS T
     ORDER BY T.total_count DESC
     LIMIT 3;
END //
DELIMITER ;
```

## Implement GUI

The main menu of the program. The main menu has 4 functions that consist of Request Stylist, Add Stylist, Request Management and Delete Stylist.

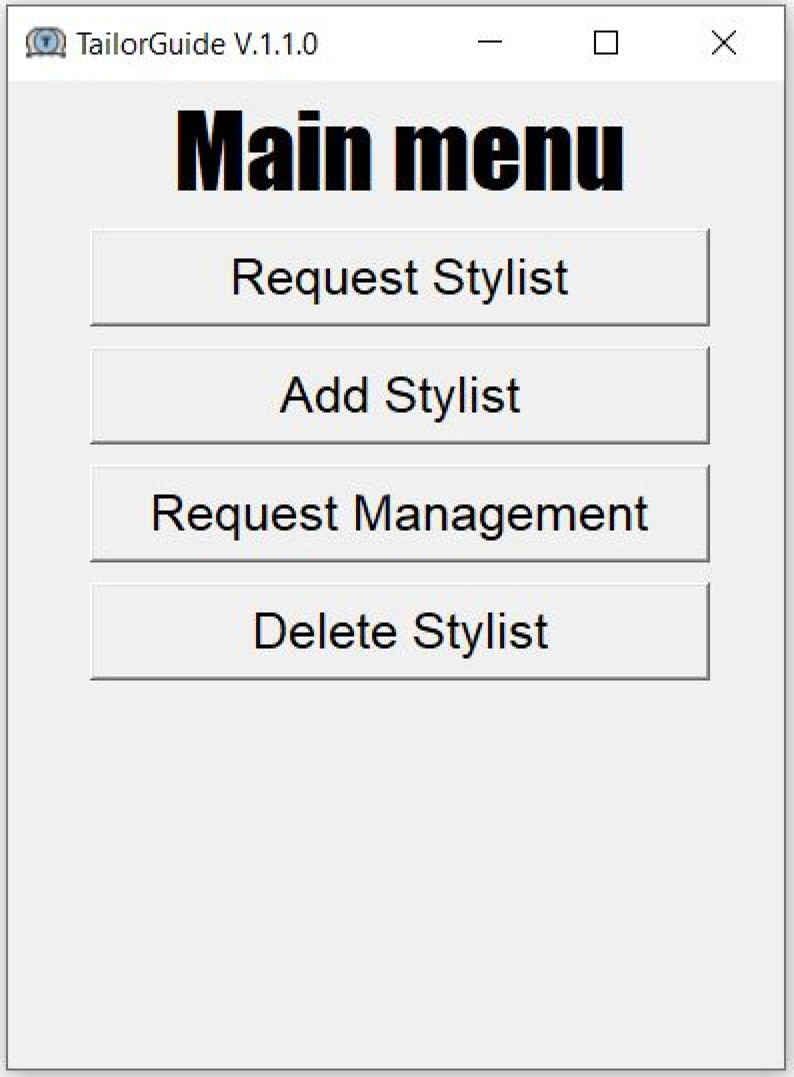
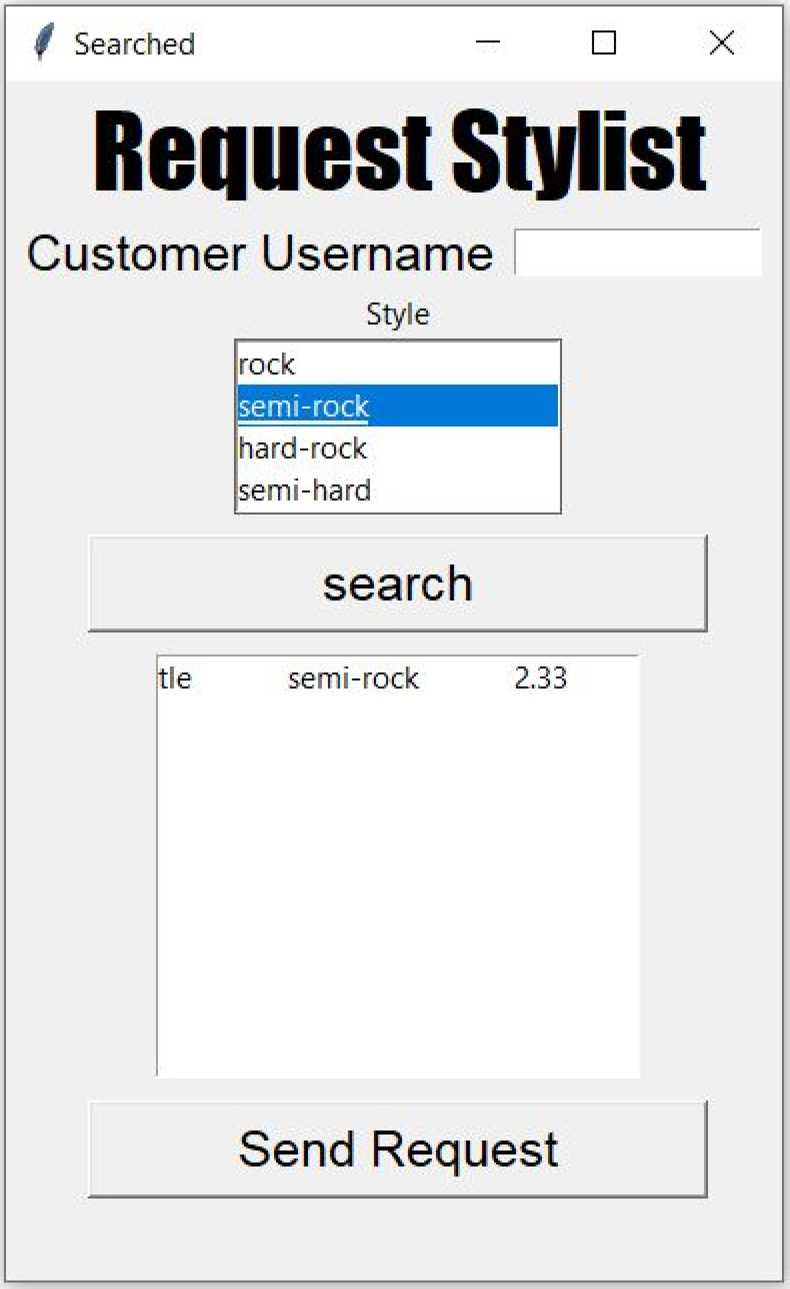


Figure 15 : main menu of GUI

## Request Stylist : QUERY

```
def searchStyleDB(self, databasename, table) :
    try:
       connection = mysql.connector.connect(host='localhost',
                                            database=databasename,
                                            user='root',
                                            password=pw)
       sqlQuery = "select distinct style from stylist_style"
       cursor = connection.cursor()
       cursor.execute(sqlQuery)
       records = cursor.fetchall()
       self.data = records
    except:
       retmsg = ["1", "Error"]
    else :
       retmsg = ["1", "Not Found"]
       if records[1] != "" :
          retmsg = ["0", "Found"]
    finally:
       if (connection.is_connected()):
          connection.close()
          cursor.close()
       return retmsg
```

```
def searchNameDB(self, databasename, table) :
    wkey = str(self.data[0]) #correct here

    try:
      connection = mysql.connector.connect(host='localhost',
                                           database=databasename,
                                           user='root',
                                           password=pw)

      cursor = connection.cursor()
      cursor.callproc('searchStyle', (self.data[0],))

      for result in cursor.stored_results():
        records = result.fetchall()
      self.data = records

    except:
      retmsg = ["1", "Error"]
    else :
      retmsg = ["1", "Not Found"]
      if records[0] != "" :
        retmsg = ["0", "Found"]
    finally:
      if (connection.is_connected()):
        connection.close()
        cursor.close()
      return retmsg
```

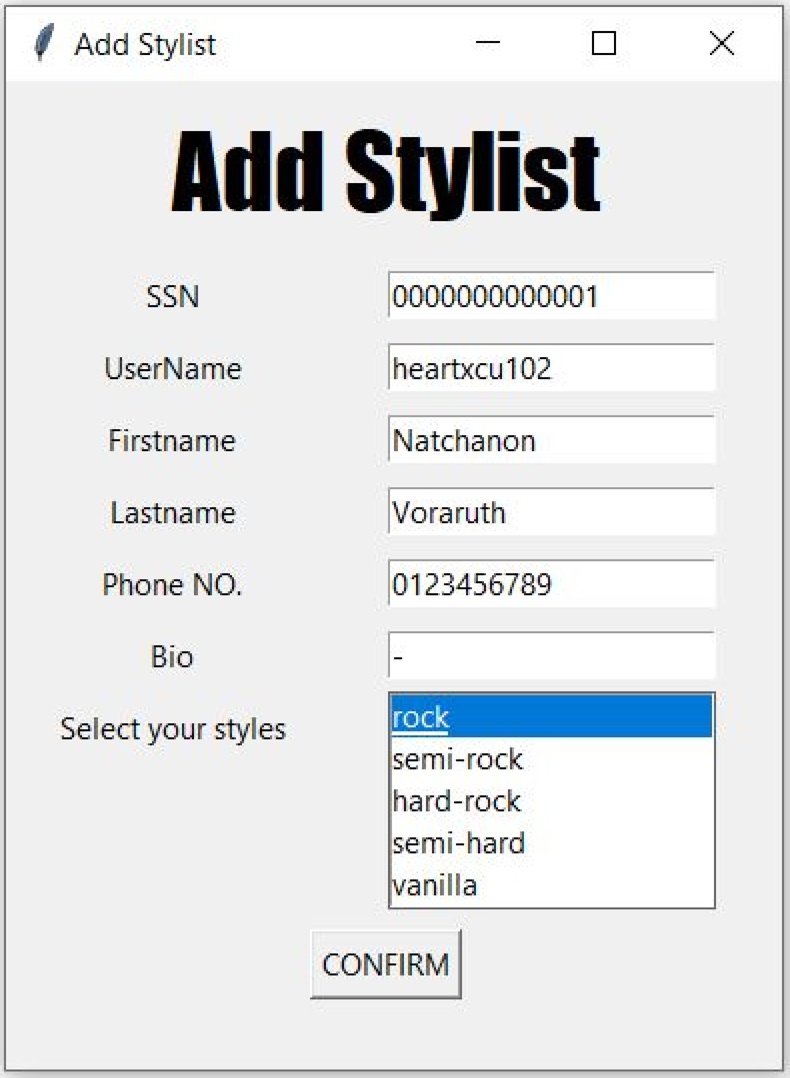


```
tle        semi-rock        2.33
```

Figure 16 : Request Stylist of GUI

## Add Stylist : QUERY AND INSERT

```
def addStylistDB(self, databasename, table) :
    wdata=self.data

    try:
        connection = mysql.connector.connect(host='localhost',
                                             database=databasename,
                                             user='root',
                                             password=pw)
        cursor = connection.cursor()

        objdata = (wdata[0],wdata[5])
        objdata1 = (wdata[0],wdata[6])
        objdata2 = (wdata[0],wdata[4])
        objdata3 = (wdata[0],wdata[1],wdata[2],wdata[3])
        sqlQuery3 = "insert into usert (SSN,username,first_name,last_name) values
(%s,%s,%s,%s)"
        cursor.execute(sqlQuery3, objdata3)
        sqlQuery2 = "insert into user_phone_no (SSN,phone_no) values (%s,%s)"
        cursor.execute(sqlQuery2, objdata2)
        sqlQuery = "insert into stylist(SSN,profile_pic,bio,bio_image,avg_rating) values
(%s,null,%s,null,3.00)"
        cursor.execute(sqlQuery, objdata)
        sqlQuery1 = "insert into stylist_style (SSN,style) values (%s,%s)"
        cursor.execute(sqlQuery1, objdata1)
        connection.commit()
    except:
        retmsg = ["1", "Add error"]
    else :
        retmsg = ["0", "Add done"]
    finally:
        if (connection.is_connected()):
            connection.close()
            cursor.close()
        return retmsg
```


Add Stylist

# Add Stylist

| | |
|---|---|
| SSN | 0000000000001 |
| UserName | heartxcu102 |
| Firstname | Natchanon |
| Lastname | Voraruth |
| Phone NO. | 0123456789 |
| Bio | - |
| Select your styles | rock |

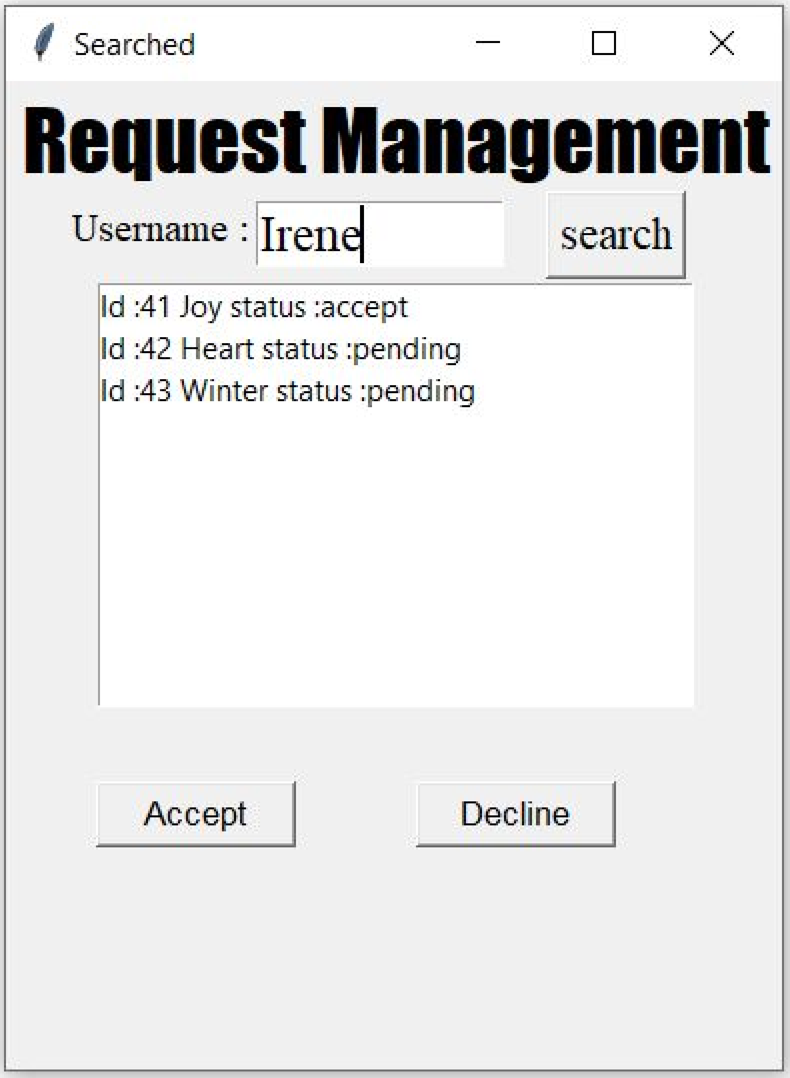
Figure 17 : Add Stylist of GUI

## Request Management : QUERY AND UPDATE

```
def searchReDB(self, databasename, table) :
    wkey = str(self.data[0]) #correct here
    try:
        connection = mysql.connector.connect(host='localhost',
                                             database=databasename,
                                             user='root',
                                             password=pw)

        cursor = connection.cursor()
        cursor.callproc('searchRe', (self.data[0],))
        for result in cursor.stored_results():
            records = result.fetchall()
        self.data = records

    except:
        retmsg = ["1", "Error"]
    else :
        retmsg = ["1", "Not Found"]
        if records[0] != "" :
            retmsg = ["0", "Found"]
    finally:
        if (connection.is_connected()):
            connection.close()
            cursor.close()
        return retmsg
```

```
def acceptReDB(self, databasename, table) :
    wkey = str(self.data[0]) #correct here

    try:
        connection = mysql.connector.connect(host='localhost',
                                             database=databasename,
                                             user='root',
                                             password=pw)
        objdata = (wkey,)
        sqlQuery = "call acceptRequest(%s)"
        cursor = connection.cursor()
        cursor.execute(sqlQuery, objdata)
        connection.commit()

    except:
        retmsg = ["1", "Error"]
    else :
        retmsg = ["1", "Done"]
    finally:
        if (connection.is_connected()):
            connection.close()
            cursor.close()
        return retmsg
```

```
def declineReDB(self, databasename, table) :
    wkey = str(self.data[0]) #correct here

    try:
        connection = mysql.connector.connect(host='localhost',
                                             database=databasename,
                                             user='root',
                                             password=pw)
        objdata = (wkey,)
        sqlQuery = "call rejectRequest(%s)"
        cursor = connection.cursor()
        cursor.execute(sqlQuery, objdata)
        connection.commit()
    except:
        retmsg = ["1", "Error"]
    else :
        retmsg = ["1", "Done"]
    finally:
        if (connection.is_connected()):
            connection.close()
            cursor.close()
        return retmsg
```



Id :41 Joy status :accept
Id :42 Heart status :pending
Id :43 Winter status :pending

Figure 18 : Request Management of GUI

## Delete Stylist : DELETE

```
 def deleteDB(self, databasename, table):
     wdata = self.data[0]
     try:
        connection = mysql.connector.connect(host='localhost',
                                 database=databasename,
                                 user='root',
                                 password=pw)

        objdata = (wdata,)
        cursor = connection.cursor()

        checkQuery = "SELECT * FROM usert U, stylist S WHERE U.username = %s AND
S.SSN = U.SSN"
        cursor.execute(checkQuery, objdata)
        myresult = cursor.fetchall()
        if myresult == []:
           retmsg = ["1","Stylist is not found"]
           return
        sqlQuery = "delete from usert where username = %s"
        cursor.execute(sqlQuery, objdata)
        connection.commit()

     except:
        retmsg = ["1", "Delete Error"]
     else:
        retmsg = ["0", "Delete Complete"]
     finally:
        if (connection.is_connected()):
           connection.close()
           cursor.close()
        return retmsg
```

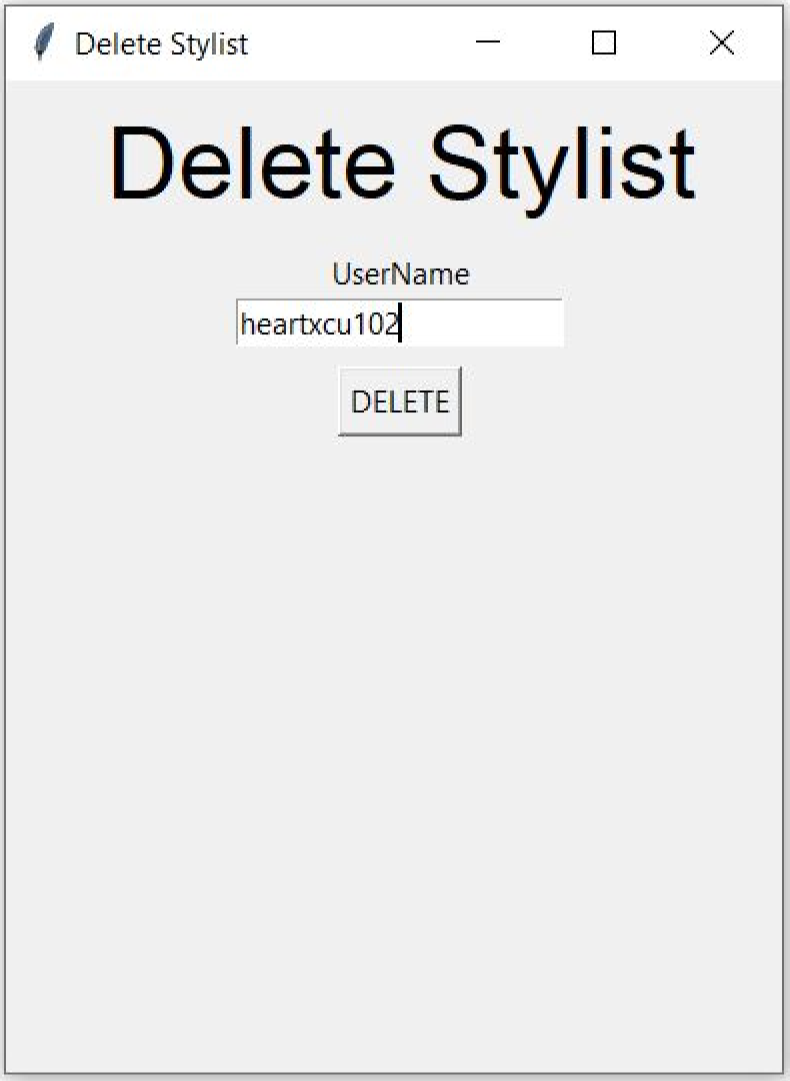


Figure 19 : Delete Stylist of GUI

Source code link: https://github.com/NatthawutThongsai/TailorGuide-DB.git