CS 353 - Database Systems

Project Design Report

Flover

Group 30

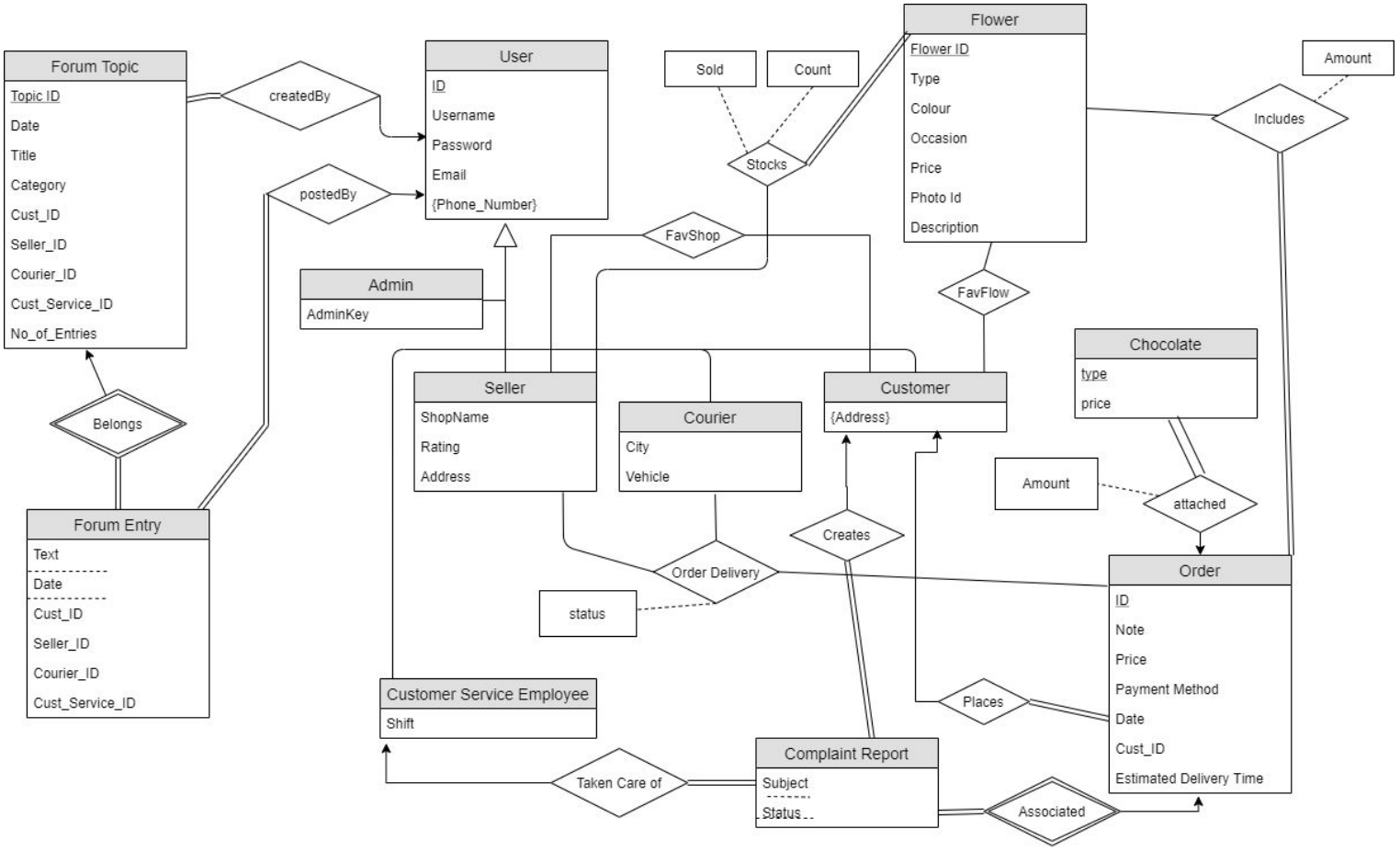
Muzaffer Yasin Köktürk - 21703552

Muhammed Berk KÖSE - 21704277

Alptekin Önder - 21602728

İsmail Şahal - 21703596

1.0 Revised E/R Model



What changed?

We have made some changes according to our new features and the feedback we have received:

- We have added a new functionality, forum. In this page people can share their opinions ask questions and socialize while learning new things about the flower related topics.

- We have added another user to the diagram, admin. Thus now admins can login the system.

- Deleted the favorite shop attribute in customer, we realized we did not need this.

- We added amount attribute to the Includes relation to keep track of amount of each flower type in an order.

- We made the flower totally participate with Seller through stocks, if there are no sellers there are no flowers.
- Stocks now has a new attribute, sold , which will yield us the amount of the flower sold in its lifetime.

- We have deleted the transports and prepares relations and brought a ternary relation named Order_delivery connecting seller, order and courier.

- Chocolate is now an entity for our convenience. It is related to Order entity.

2.0 **Relational Schemas**

2.1 **Customer**

- **Relational Model**
  - Customer(id, username, password, email)
  - Customer_phone_number(id,phone number)
  - Custome_address(id,address)
- **Candidate Keys**
  - id
  - email
- **Table Definition**

```
CREATE TABLE Customer (
        id INT PRIMARY KEY,
        username VARCHAR(64) NOT NULL ,
        password VARCHAR(64) NOT NULL,
        email VARCHAR(64) NOT NULL UNIQUE,
);
CREATE TABLE Customer_address(
        id INT,
        FOREIGN KEY id INT REFERENCES Customer(id),
        address VARCHAR(256) NOT NULL,
        PRIMARY KEY (id, address)
);
CREATE TABLE Customer_phone_number(
        id INT,
        FOREIGN KEY id REFERENCES Customer(id),
        phone_number VARCHAR(64) NOT NULL,
        PRIMARY KEY (id, phone_number)
);
```

2.2 Seller

- **Relational Model**
  - Seller(id, username, password, email, shop_name, rating, address)
  - Seller_phone_number(id,phone number)
- **Candidate Keys**

- id
- shop_name
- email
- address

- **Table Definition**

```
CREATE TABLE Seller (
        id INT PRIMARY KEY,
        username VARCHAR(64) NOT NULL ,
        password VARCHAR(64) NOT NULL,
        email VARCHAR(64) NOT NULL UNIQUE,
        shop_name VARCHAR(64) NOT NULL UNIQUE,
        rating VARCHAR(64),
        address VARCHAR(64) NOT NULL UNIQUE,
);
CREATE TABLE Seller_phone_number(
        id INT,
        FOREIGN KEY (id) REFERENCES Seller(id),
        phone_number VARCHAR(64) NOT NULL,
        PRIMARY KEY (id, phone_number)
);
```

2.3 Courier

- **Relational Model**
  - Courier(id, username, password, email, city, vehicle)
  - Courier_phone_number(id, phone_number)
- **Candidate Keys**
  - id
  - email
- **Table Definition**

```
CREATE TABLE Courier (
        id INT PRIMARY KEY,
        username VARCHAR(64) NOT NULL ,
        password VARCHAR(64) NOT NULL,
        email VARCHAR(64) NOT NULL UNIQUE,
        city VARCHAR(64) NOT NULL,
        vehicle VARCHAR(64),
);

CREATE TABLE Courier_phone_number(
        id INT,
        FOREIGN KEY (id) REFERENCES Courier(id),
        phone_number VARCHAR(64) NOT NULL,
        PRIMARY KEY (id, phone_number)
);
```

2.4 Flower

- **Relational Model**
  - Flower(flower_id, type, color, occasion, price, photo_id, description)
- **Candidate Keys**
  - Flower_id
- **Table Definition**

```
CREATE TABLE Flower (
        flower_id INT PRIMARY KEY,
        type VARCHAR(64) NOT NULL ,
        color VARCHAR(64) NOT NULL,
        occasion VARCHAR(64) NOT NULL UNIQUE,
        price DECIMAL(10, 2) NOT NULL,
        photo_id INT NOT NULL
        description VARCHAR(256)
);
```

2.5 Order

- **Relational Model**
  - Order(id, note, price, payment_method, date, est_delivery_time, seller_id, cust_id)
- **Candidate Keys**
  - id
- **Table Definition**

```
CREATE TABLE Order (
        id INT PRIMARY KEY,
        note VARCHAR(64) NOT NULL ,
        price DECIMAL(10, 2) NOT NULL,
        payment_method VARCHAR(64) NOT NULL,
        date VARCHAR(64) NOT NULL,
        est_delivery_time VARCHAR(64) NOT NULL,
        FOREIGN KEY (cus_id) REFERENCES Customer(id)
);
```

2.6 Complaint Report

- **Relational Model**
  - Complaint_report(order_id, subject, status, cust_id)
- **Candidate Keys**
  - None
- **Table Definition**

```
CREATE TABLE Complaint_report (
        order_id INT NOT NULL,
        cust_id INT NOT NULL,
        subject TEXT(1023) NOT NULL,
        status VARCHAR(64) NOT NULL,
        FOREIGN KEY(order_id) REFERENCES Order(id)
        FOREIGN KEY(cust_id) REFERENCES Customer(id)
);
```

2.7 Customer Service Employee

- **Relational Model**
  - Customer_service_employee(id, username, password, email)
  - Customer_service_employee_phone_number(id, phone_number)
- **Candidate Keys**
  - id
  - username
  - email
- **Table Definition**

```
CREATE TABLE Customer_service_employee (
        id INT PRIMARY KEY,
        username VARCHAR(64) NOT NULL ,
        password VARCHAR(64) NOT NULL,
        email VARCHAR(64) NOT NULL UNIQUE,
);
CREATE TABLE Customer_service_employee_phone_number(
        FOREIGN KEY id INT REFERENCES Customer_service_employee(id),
        phone_number VARCHAR(64) NOT NULL,
        PRIMARY KEY (id, phone_number)
);
```

2.8 Admin

- **Relational Model**
  - Admin(id, username, password, email, admin_key)
  - Admin_phone(id, phone_number)
- **Candidate Keys**
  - id
  - email
  - admin_key
- **Table Definition**

```
CREATE TABLE Admin (
        id INT PRIMARY KEY,
        username VARCHAR(64) NOT NULL ,
```

```
        password VARCHAR(64) NOT NULL,
        email VARCHAR(64) NOT NULL,
        admin_key VARCHAR(64) NOT NULL UNIQUE,
);
CREATE TABLE Admin_phone_number(
        id INT,
        FOREIGN KEY id INT REFERENCES Admin(id),
        phone_number VARCHAR(64) NOT NULL,
        PRIMARY KEY (id, phone_number)
);
```

2.9 Forum Entry

- **Relational Model**
  - forum_entry(topic_id, date, text, cust_id, seller_id, courier_id, cust_service_id)
- **Candidate Keys**
  - None
- **Table Definition**

```
CREATE TABLE Forum_entry(
        date VARCHAR(64) NOT NULL ,
        text MEDIUMTEXT NOT NULL,
        FOREIGN KEY topic_id REFERENCES forum_topic(topic_id),
        FOREIGN KEY (cust_id) REFERENCES Customer(id),
        FOREIGN KEY (seller_id) REFERENCES Seller(id),
        FOREIGN KEY (courier_id) REFERENCES Courier(id),
        FOREIGN KEY (cust_service_id) REFERENCES
Customer_service_employee(id)
);
```

2.10 Chocolate

- **Relational Model**
  - Chocolate(type, price)
- **Candidate Keys**
  - type
- **Table Definition**

```
CREATE TABLE chocolate(
        type VARCHAR(64) PRIMARY KEY,
        price DECIMAL(10, 2) NOT NULL,
);
```

2.11 Forum Topic

- **Relational Model**
  - Forum_topic(topic id, date, title, category, cust_id, seller_id, courier_id, cust_service_id, no_of_entries)
- **Candidate Keys**
  - type
- **Table Definition**

```
CREATE TABLE Forum_topic(
        topic_id VARCHAR(255) PRIMARY KEY,
        date VARCHAR(64)  NOT NULL,
        title VARCHAR(64) NOT NULL,
        category VARCHAR(64) NOT NULL,
        no_of_entries INT NOT NULL,
        FOREIGN KEY (cust_id) REFERENCES Customer(id),
        FOREIGN KEY (seller_id) REFERENCES Seller(id),
        FOREIGN KEY (courier_id) REFERENCES Courier(id),
        FOREIGN KEY (cust_service_id) REFERENCES
Customer_service_employee(id)
);
```

2.12 Attached

- **Relational Model**
  - Attached(type, id, amount)
- **Candidate Keys**
  - {(type, id)}
- **Table Definition**

```
CREATE TABLE Attached(
        type VARCHAR(64),
        id INT,
        PRIMARY KEY (type , id),
        FOREIGN KEY (type) REFERENCES Chocolate(type),
        FOREIGN KEY (id) REFERENCES Order(id)
);
```

2.13 Fav_shop

- **Relational Model**
  - Fav_shop(customer_id, seller_id)
- **Candidate Keys**
  - {(customer_id, seller_id)}
- **Table Definition**

```
CREATE TABLE Fav_shop (
        customer_id INT,
```

```
        seller_id INT,
        PRIMARY KEY (customer_id, seller_id),
        FOREIGN KEY (customer_id) REFERENCES Customer(id),
        FOREIGN KEY (seller_id) REFERENCES Seller(id)
);
```

2.14 Fav_flow

- **Relational Model**
  - Fav_flow(customer_id, flower_id)
- **Candidate Keys**
  - {(customer_id, flower_id)}
- **Table Definition**

```
CREATE TABLE Fav_shop (
        customer_id INT,
        flower_id INT,
        PRIMARY KEY (customer_id, flower_id),
        FOREIGN KEY (customer_id) REFERENCES Customer(id),
        FOREIGN KEY (seller_id) REFERENCES Flower(flower_id)
);
```

2.15 Includes

- **Relational Model**
  - Includes(flower_id, id, amount)
- **Candidate Keys**
  - {(flower_id, id)}
- **Table Definition**

```
CREATE TABLE Includes(
        flower_id INT,
        id INT,
        amount INT,
        PRIMARY KEY (flower_id, id),
        FOREIGN KEY (flower_id) REFERENCES flower(flower_id),
        FOREIGN KEY (id) REFERENCES order(id)
);
```

2.16 Stocks

- **Relational Model**
  - Stocks(flower_id, id, sold, count)
- **Candidate Keys**
  - {(flower_id, id)}
- **Table Definition**

```
CREATE TABLE Stocks(
        flower_id INT,
        id INT,
```

```
        sold INT NOT NULL,
        count INT NOT NULL,
        PRIMARY KEY (flower_id, id),
        FOREIGN KEY (flower_id) REFERENCES flower(flower_id),
        FOREIGN KEY (id) REFERENCES seller(id)
);
```

2.17 Order Delivery

- **Relational Model**
  - Order_delivery(order_id, courier_id, seller_id, status)
- **Candidate Keys**
  - {(order_id, courier_id, seller_id)}
- **Table Definition**

```
CREATE TABLE Order_delivery(
        order_id INT,
        courier_id INT,
        seller_id INT,
        status VARCHAR(64)
        PRIMARY KEY (order_id, courier_id, seller_id),
        FOREIGN KEY courier_id REFERENCES Courier(id),
        FOREIGN KEY seller_id REFERENCES Seller(id),
        FOREIGN KEY order_id REFERENCES Order(id)
);
```

3.0 User Interface Design and Corresponding SQL Statements

3.1 Main Page

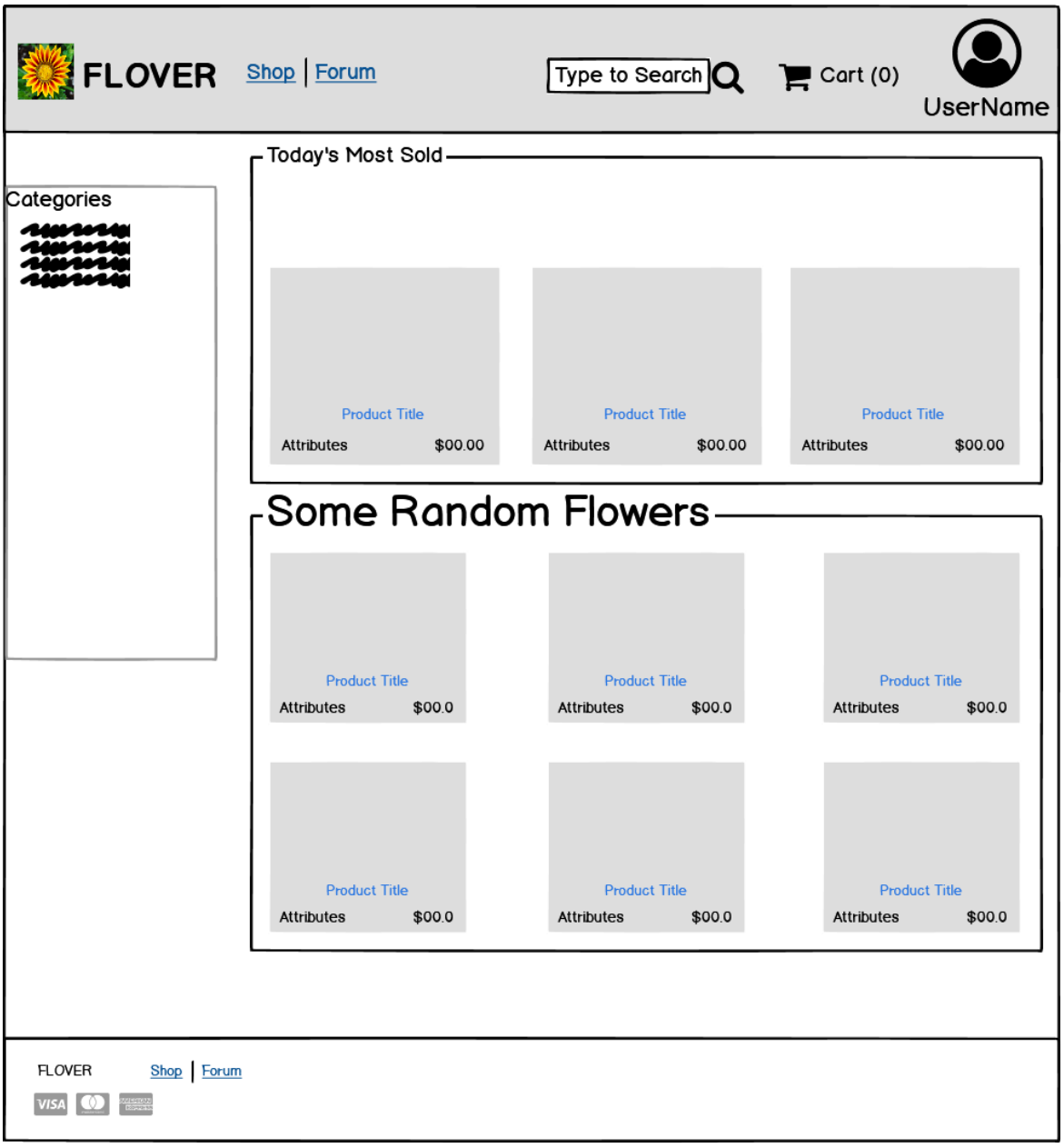


Figure 1: Main Page

In this page, we will be displaying the categories in the left bar and the most sold flowers of that day with some random flowers below.
Here is the SQL statement for Today's Top Sellers:

```
SELECT F.photo_id, F.price, F.type, F.flower_id
FROM flowers F
WHERE F.flower_id IN
(SELECT top (3) S.flower_id
FROM Stocks S
ORDER BY sold DESC, flower_id)
```

Here is the SQL statement for selecting random flowers to display:

```
SELECT F.photo_id, F.price, F.type, F.flower_id
FROM Flowers F
ORDER BY RAND()
```

3.2 Product Page

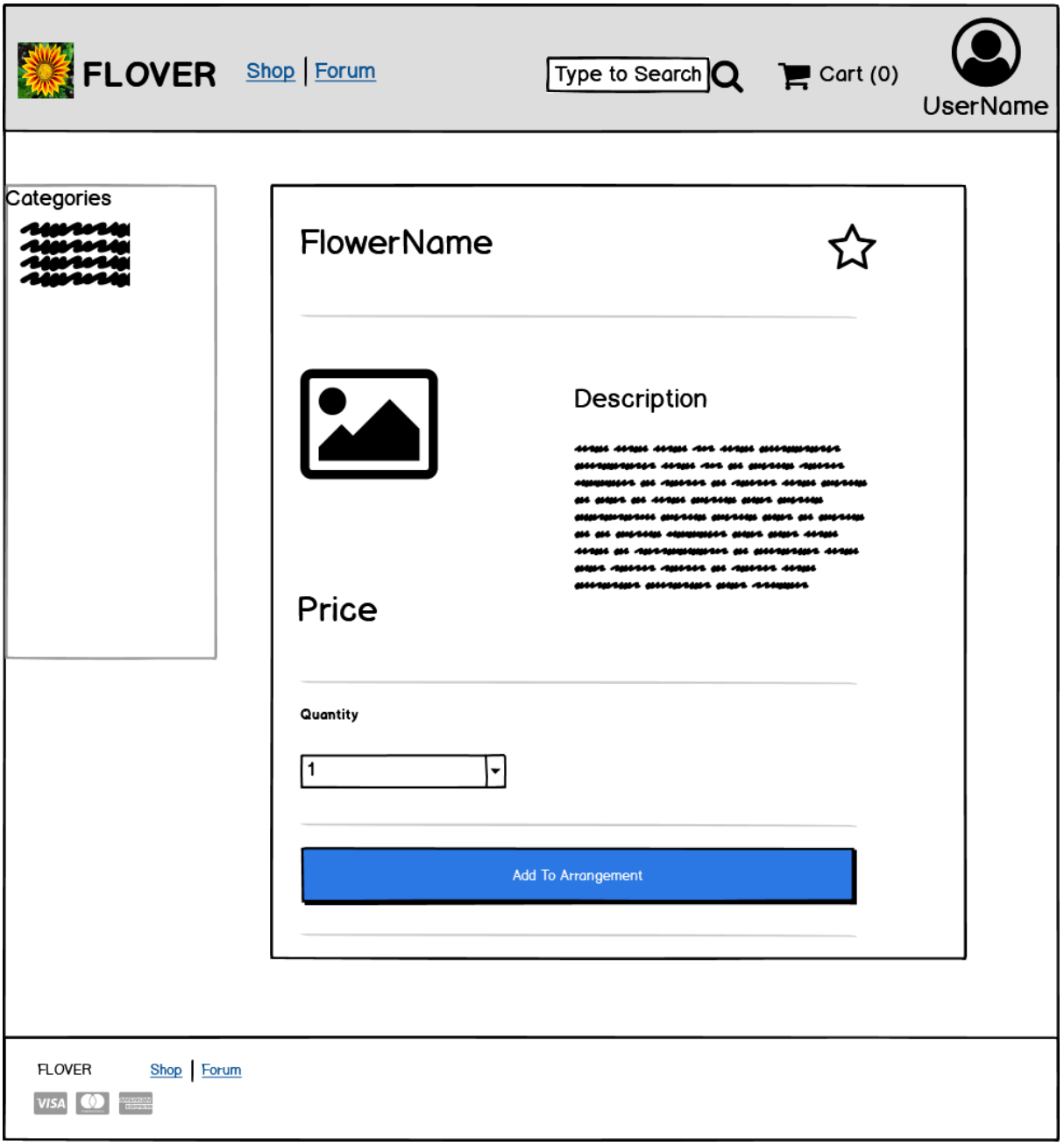


Figure 2: Product Page Mockup

In this page, we will be displaying the information of a selected product.

```
SELECT type, price, photo_id, description
FROM Flower
WHERE flower_id = @flower_id
SELECT count
```

```
FROM Stocks
WHERE flower_id = @flower_id
```

Here is the statement for adding flower to the favorite flowers:

```
INSERT INTO Fav_flow (customer_id, flower_id)
VALUES (@id, @flower_id)
```

3.3 Profile / Profile Information

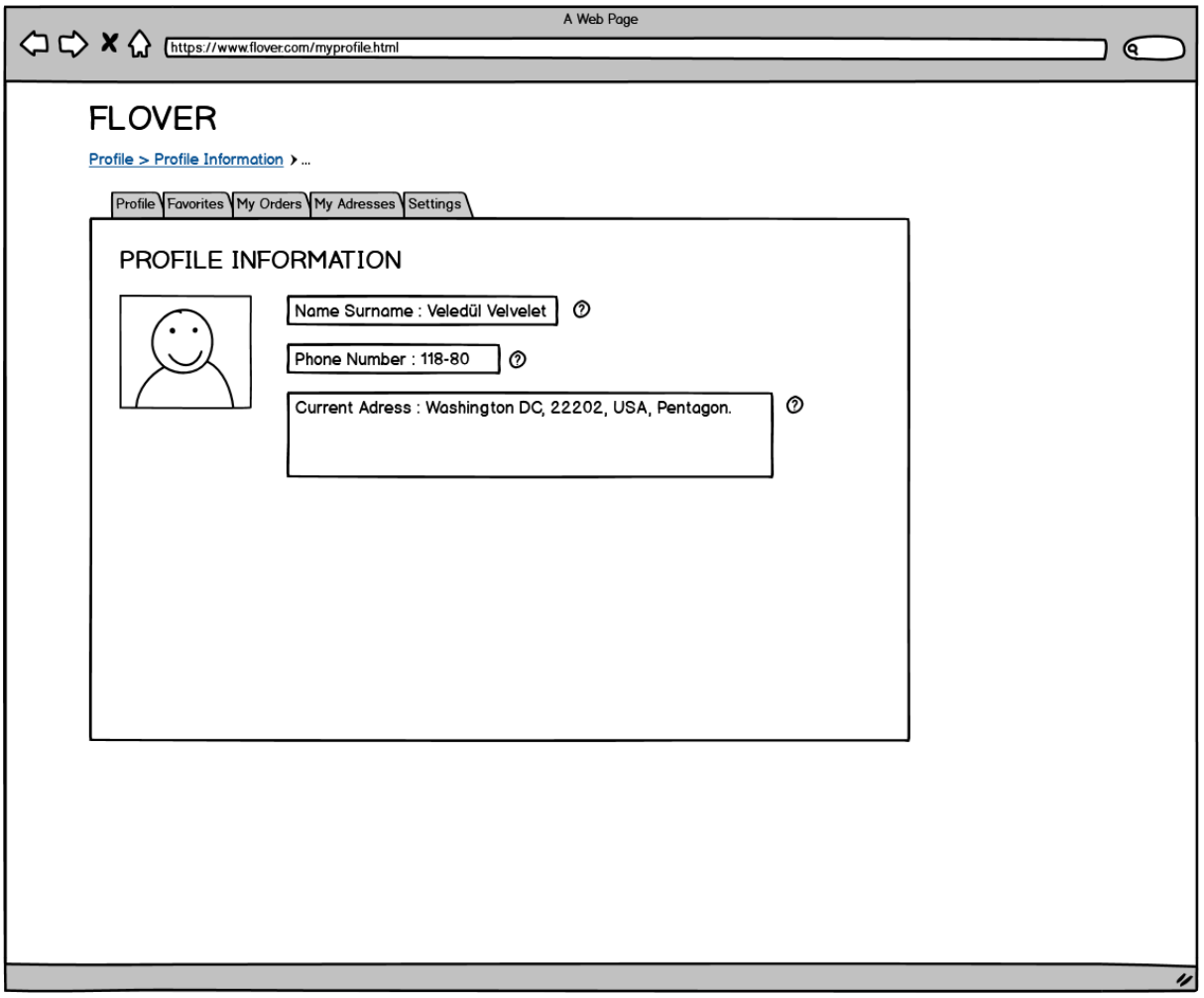


Figure 3: Profile / Profile Information

Here, the user will be able to view and edit their personal information.
SQL Statement if the user is customer:

```
SELECT username
FROM Customer
WHERE id = @id
```

```
SELECT phone_number
FROM Customer_phone_number
WHERE id = @id

SELECT address
FROM Customer_address
WHERE id = @id
```

These statements will vary depending on the user logged in the system.

3.4 Profile / Favorites

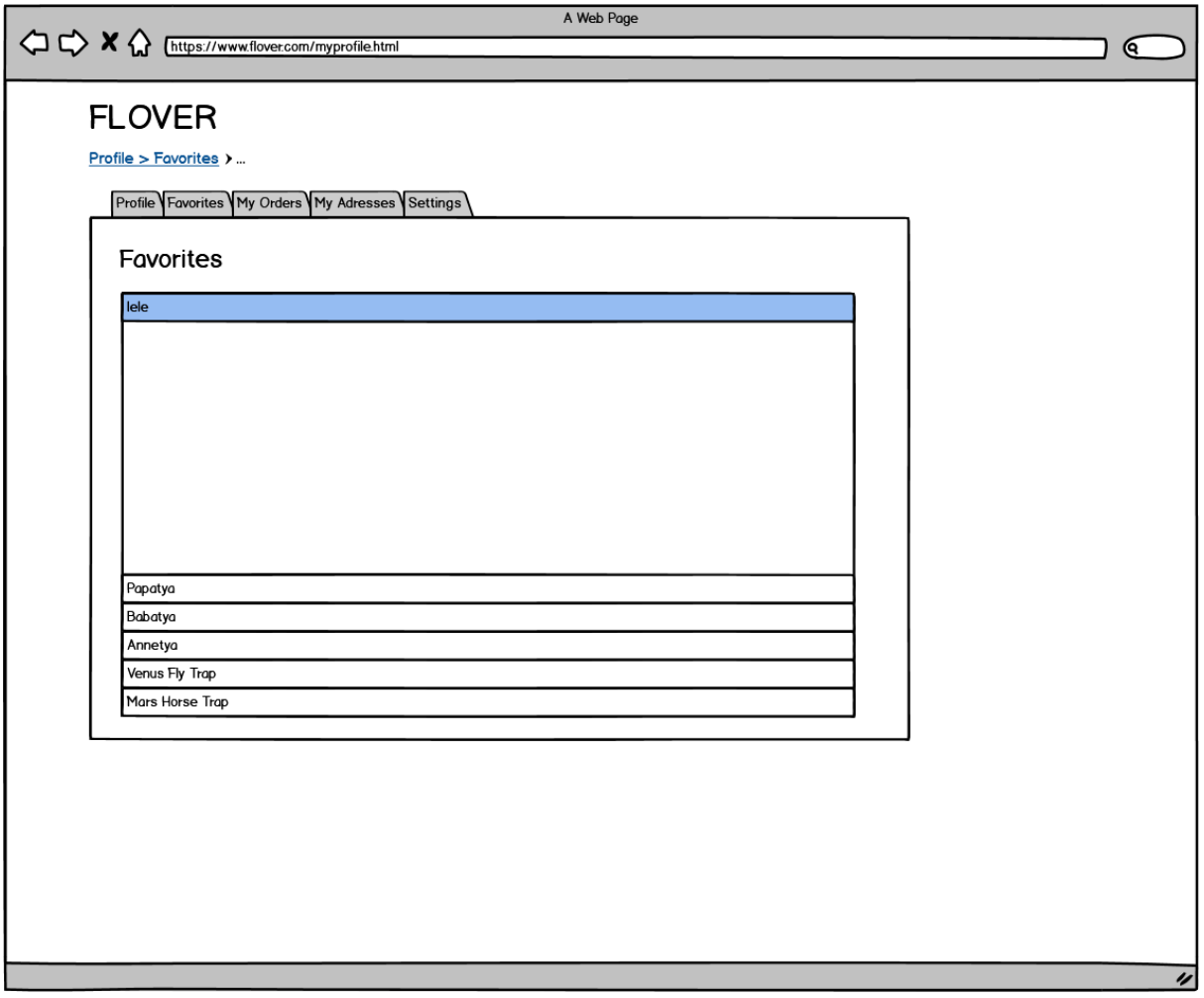


Figure 4: Profile / Favorites

In this page, we will display the favorite flowers of a user.
Here is the SQL statement for this:

```
SELECT F.type
FROM Flower F
WHERE F.flower_id IN (
SELECT F.flower_id
FROM FavFlow
WHERE id = @id)
```

3.5 Profile / My Addresses

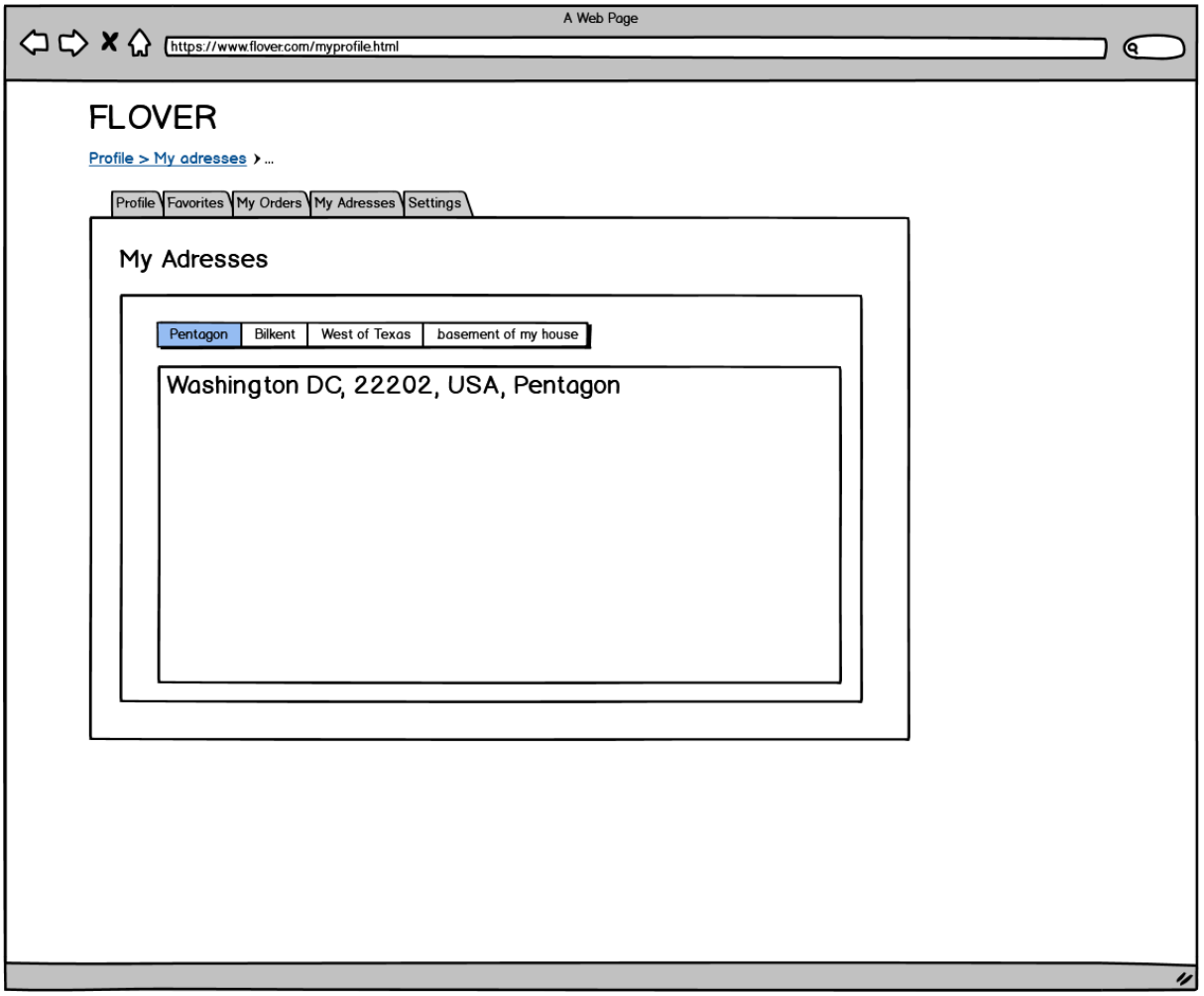


Figure 5: Profile / My Addresses

In this page, we will display different addresses of user.
Here is the SQL statement for this page:

```
SELECT address
FROM Customer_address
WHERE id = @id
```

3.6 Profile / Settings

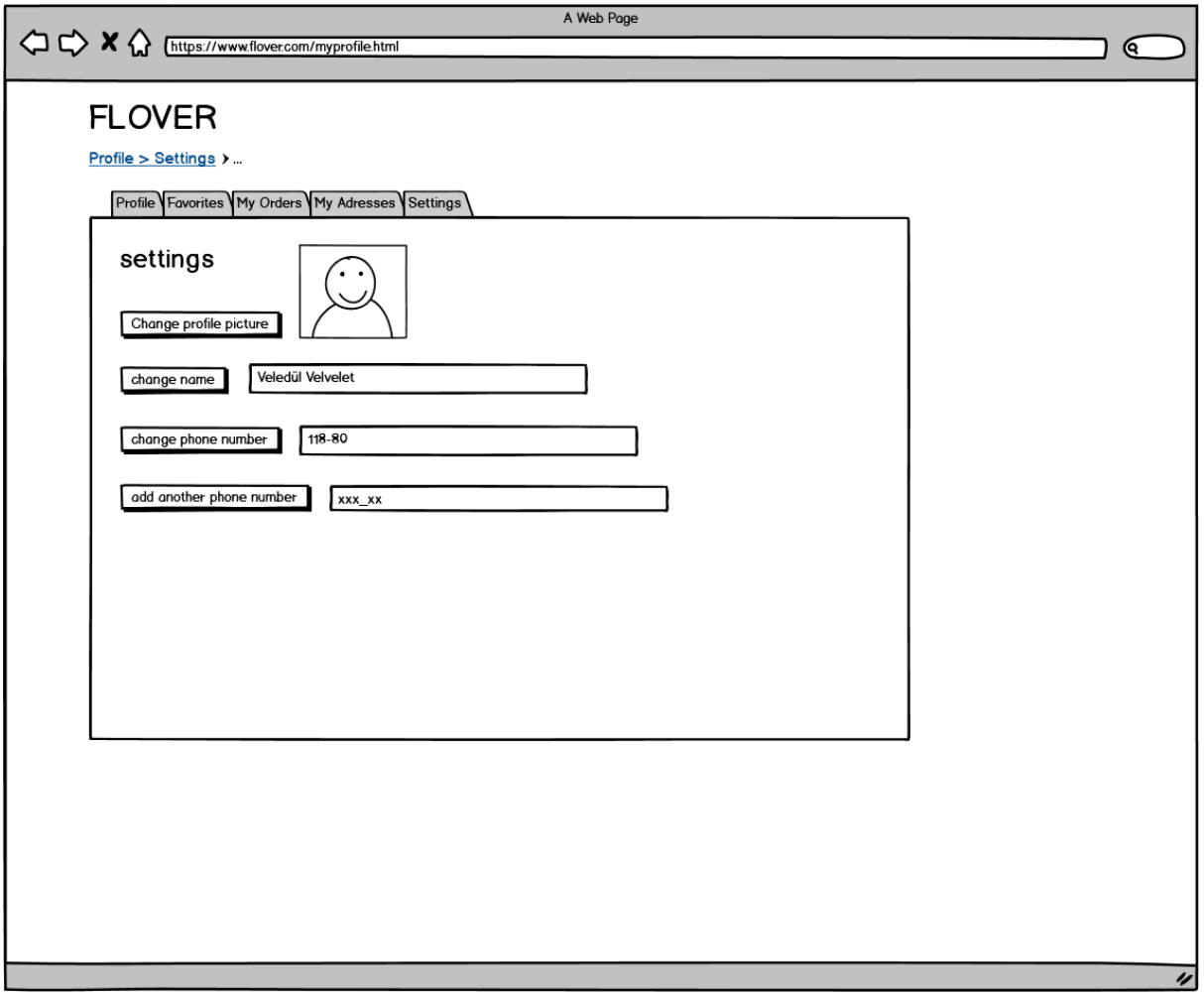


Figure 6: Profile / Settings

In this page, we will display profile information that you can change.
Displaying Current Profile Info:
SQL Statement for retrieving user info
if user is Customer

```
SELECT username
```

```
FROM Customer
WHERE id = @id

SELECT phone_number
FROM Customer_phone_number
WHERE id = @id

SELECT address
FROM Customer_address
WHERE id = @id
```

These statements will vary depending on the user logged in the system.

Updating Profile Info:
SQL Statement for retrieving user info
if user is Customer:

```
UPDATE Customer_phone_number
SET phone_number = @phone_number
WHERE id = @id

UPDATE Customer_address
SET address = @address
WHERE id = @id

UPDATE Customer
SET username = @username
WHERE id = @id
```

These statements will vary depending on the user logged in the system.

3.7 Profile / My Orders

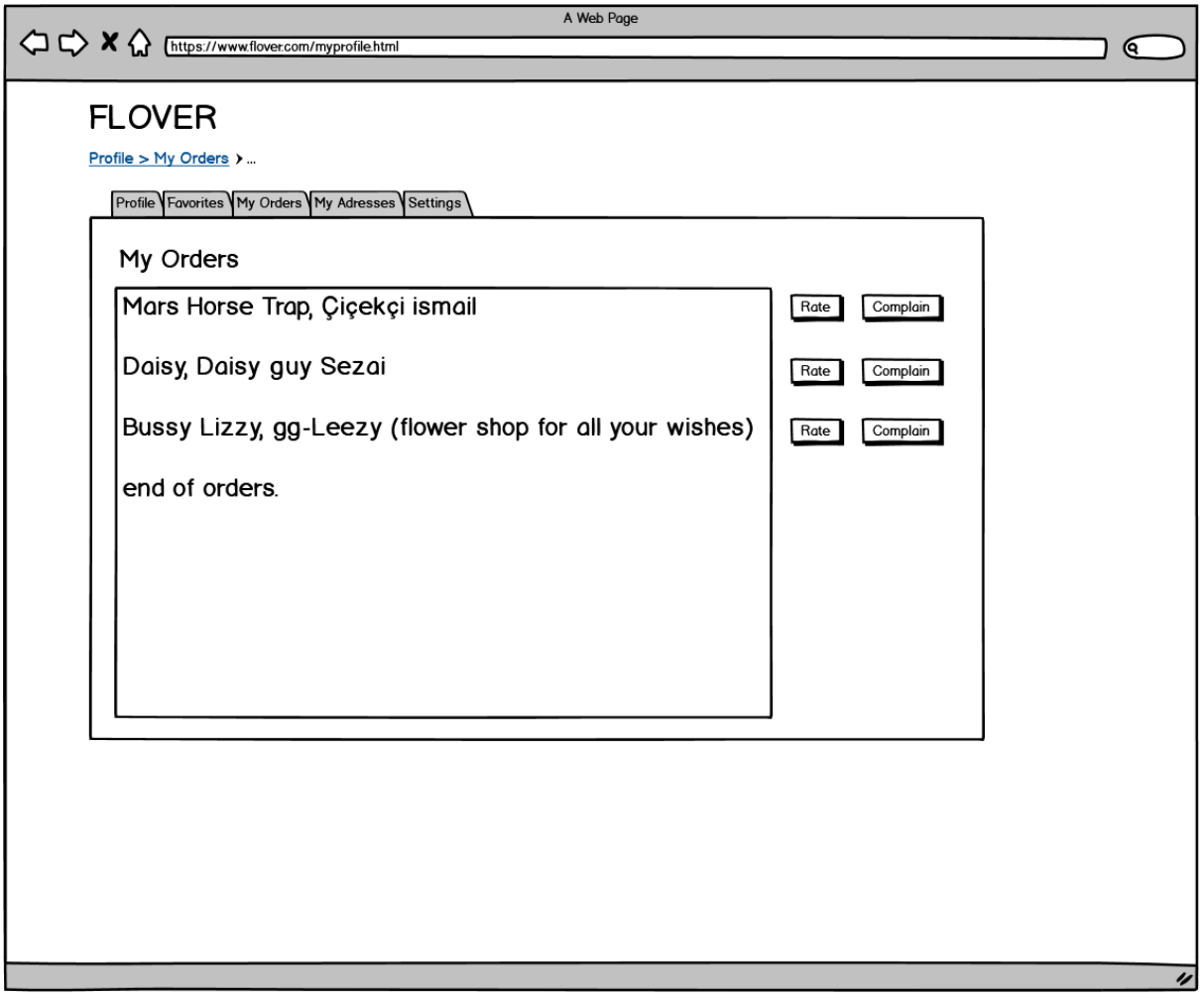


Figure 7: Profile / My Orders

In this page, the user will be able to view their past orders so that they can rate the shop or file a complaint.

Here is the SQL statements to display these:

```
SELECT id, note, price, date
FROM Order
WHERE cust_id = @id

SELECT shopname
FROM Seller S
WHERE S.id IN (
SELECT seller_id
```

```
FROM Order_delivery
WHERE id IN (
SELECT id
FROM Order
WHERE cust_id = @id))

SELECT type
FROM Flower F
WHERE F.id IN (
SELECT id
FROM Places
WHERE id IN (
SELECT id
FROM Order
WHERE cust_id = @id))

SELECT amount
FROM Includes
WHERE id IN (
SELECT id
FROM Order
WHERE cust_id = @id)
```

3.8 Profile / Contact

Figure 8: Profile / Contact Us

No need for database.

3.9 Profile / About Us

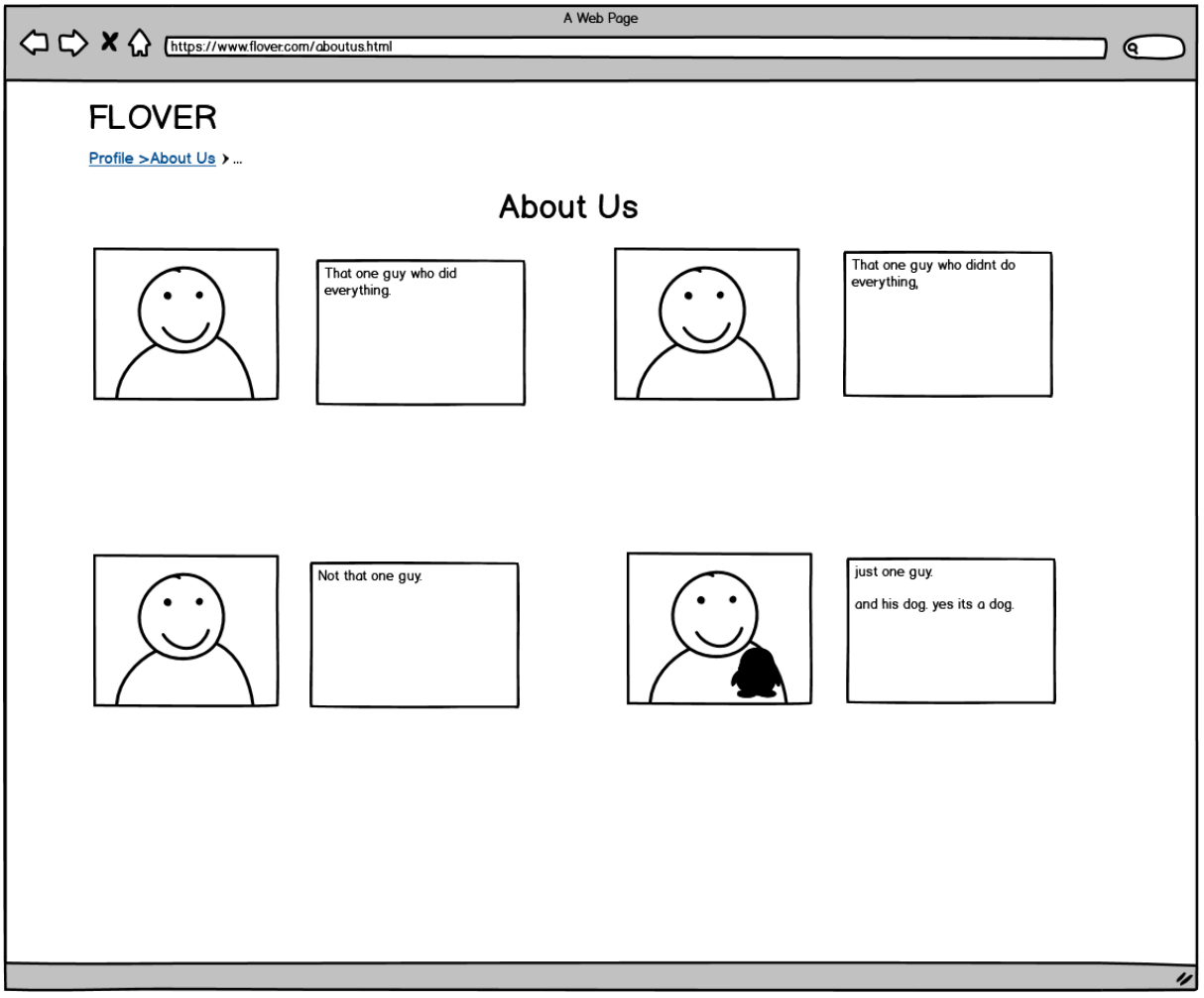


Figure 9: Profile / About Us

No need for database.

3.10 Flower Shop Page

Figure 10: Flower Shop Page

In this page, we will display information about a selected flower shop.
Here is SQL statement for information of Shop:

```
SELECT shopname, rating, address, phone_number, email
FROM Seller
WHERE id = @sellerid
```

SQL statement for information of Shop:

```
SELECT F.photoid, F.price, F.type, F.flower_id
FROM Flower
WHERE F.flower_id IN
(SELECT flower_id
FROM Stocks
WHERE seller_id = @sellerid)
```

SQL Statement for inserting a favorite shop to a user

```
INSERT INTO Fav_shop( customer_id , seller_id)
VALUES (@cust_id, @seller_id)
```

3.11 Order Arrangement Page

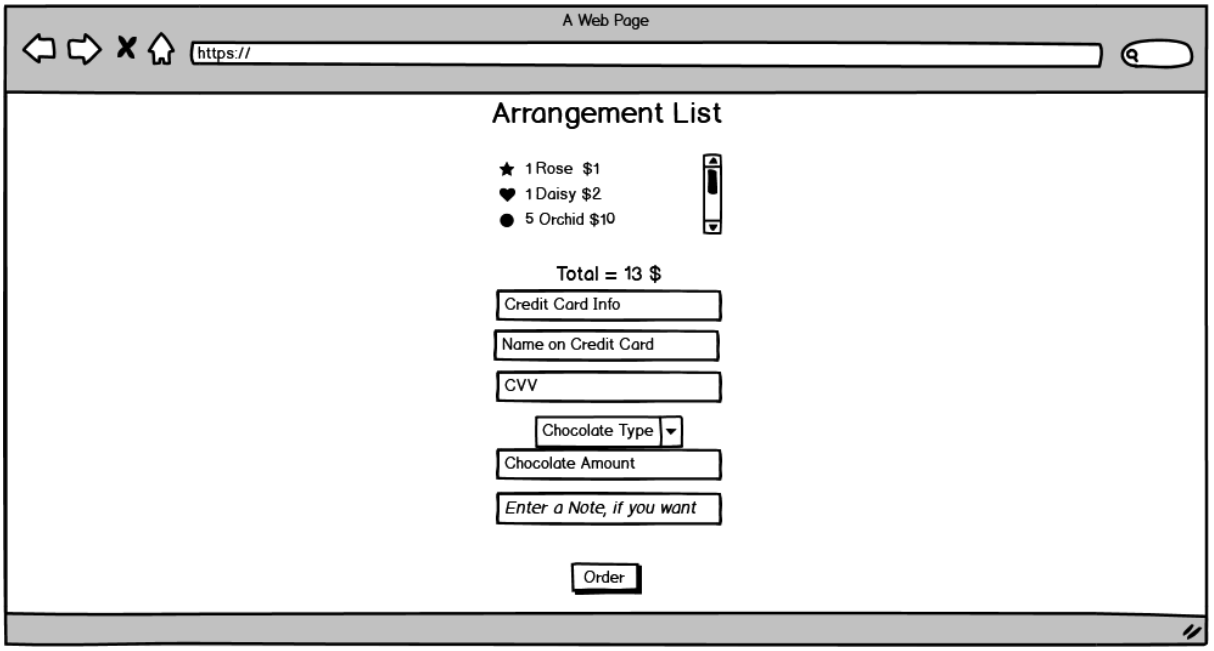


Figure 11: Order Arrangement Page

In this page, we will display your arrangement of flowers from chosen shop
Here are SQL Statements:

```
INSERT INTO Order(id, note, price, payment_method, date,
estimated_delivery_time, seller_id, cust_id)
VALUES (@id, @note, @price, @payment_method, @date,
NULL, @seller_id, @cust_id)

INSERT INTO Order_delivery(order_id, courier_id, seller_id,
status)
VALUES(@order_id, NULL, “waiting”)

INSERT INTO Includes (id, flower_id, amount)
VALUES (@order_id, @flower_id, @amount)
```

This statement will be repeated for each flower type in the arrangement list.
If the user has selected to attach chocolates to their order:

```
INSERT INTO Attached (type, id, amount)
VALUES (@type, @order_id, amount)
```

3.12. Login and Sign Up Page

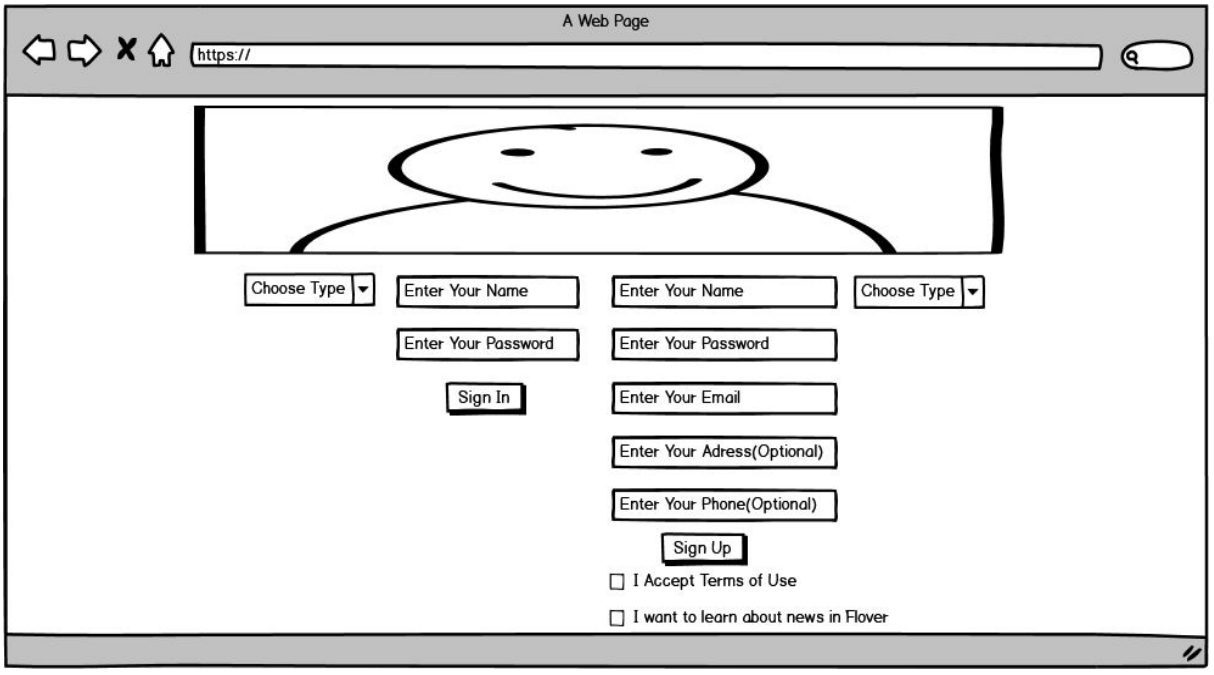


Figure 12: Sign in/Sign Up Page

In this Page, users can sign in or sign up.

SQL Statements:

If a customer is logging in or signing in:

1. **Login:**

```
SELECT id , password
FROM Customer
WHERE email = @email
```

2. **Sign up:**

   For Checking whether there is another account uses the same email

```
SELECT email
FROM Customer
WHERE email = @email
```

   If the email does not already exist:

```
INSERT INTO Customer(id, username, password, phone_number, email)
VALUES (@id, @username, @password)
```

If a seller is logging in or signing in:

1. **Login:**

```
SELECT id, password
FROM Seller
WHERE email = @email
```

2. **Sign up:**

```
SELECT email
FROM Seller
WHERE email = @email
```

If email does not already exist:

```
INSERT INTO Seller(id, username, password, phone_number,
Email)
VALUES (@id, @username, @password)
```

If a courier is logging in or signing in:

1. **Login:**

```
SELECT id, password
FROM Courier
WHERE email = @email
```

2. **Sign up:**

```
SELECT email
FROM Courier
WHERE email = @email
```

If email does not already exist:

```
INSERT INTO Courier(id, username, password, phone_number,
email)
VALUES (@id, @username, @password)
```

If a customer service employee is signing up or logging in:

1. **Login:**

```
SELECT id, password
FROM customer_service_employee
WHERE email = @email
```

2. **Sign up:**

```
SELECT email
FROM customer_service_employee
WHERE email = @email
```

If email does not already exist:

```
INSERT INTO customer_service_employee(id, username,
password, phone_number, email)
VALUES (@id, @username, @password)
```

3.13 Change Password

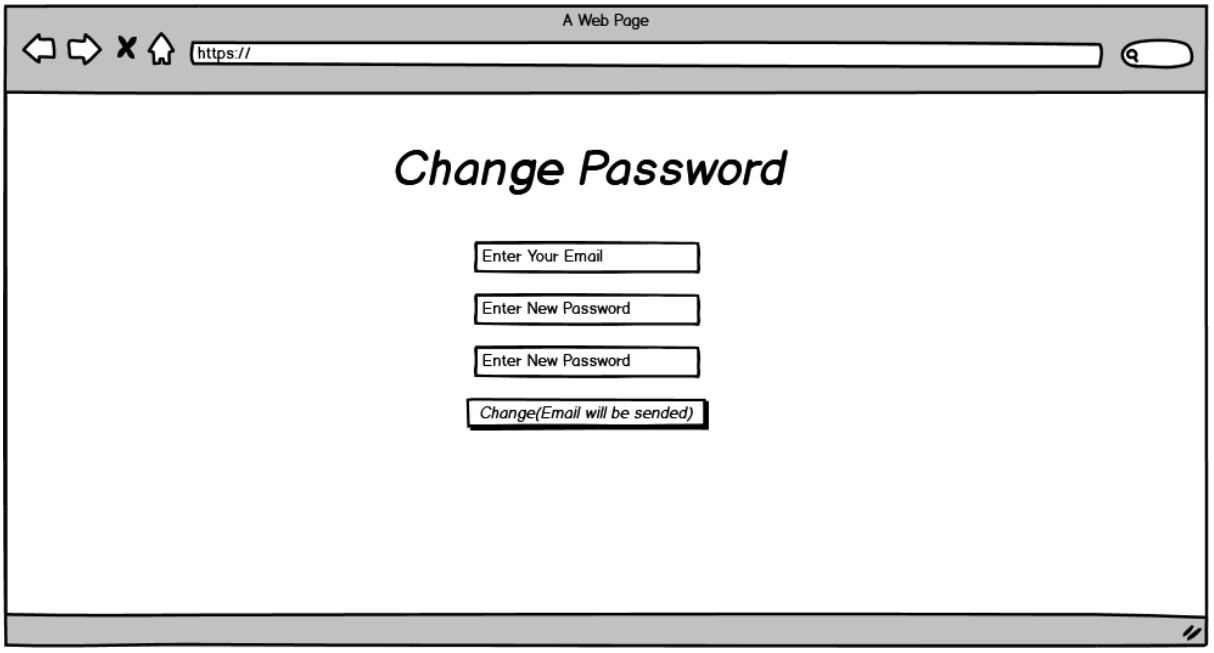


Figure 13: Change password

In this page, user can change password. Statements changes depends on your account type, account type will be chosen when you sign in.
Here is SQL Statement:

If user is a Customer:

```
UPDATE Customer
SET password= @password
```

```
WHERE id = @id
```

If user is a seller:

```
UPDATE seller
SET password= @password
WHERE id = @id
```

If user is a courier:

```
UPDATE courier
SET password= @password
WHERE id = @id
```

If user is a customer service employee:

```
UPDATE Customer_service_employee
SET password= @password
WHERE id = @id
```

3.14 Profile / Complaints

Figure 14: Profile / Complaints

In this page, customers will be able to view their complaints.
Here is the associated SQL statement:

```
SELECT cust_id, status, subject
FROM Complaint_report
WHERE cust_id = @id
```

3.15 Forum Page

Figure 15: Forum page

In this page, we will display new topics in fLover Forum.
SQL statements:
For New Topics:

```
SELECT F.category , F.title, F.date, F.id
FROM Forum_topic F
WHERE F.topic_id IN
(SELECT top (3) T.topic_id
FROM Forum_topic T
ORDER BY date DESC, topic_id)
```

For Hot Topics:

```
SELECT F.title, F.category, F.date, F.id
FROM Forum_topic F
WHERE F.topic_id IN
(SELECT top (3) T.topic_id
FROM Forum_topic T
ORDER BY no_of_entries DESC, topic_id)
```

3.16 Forum Entry

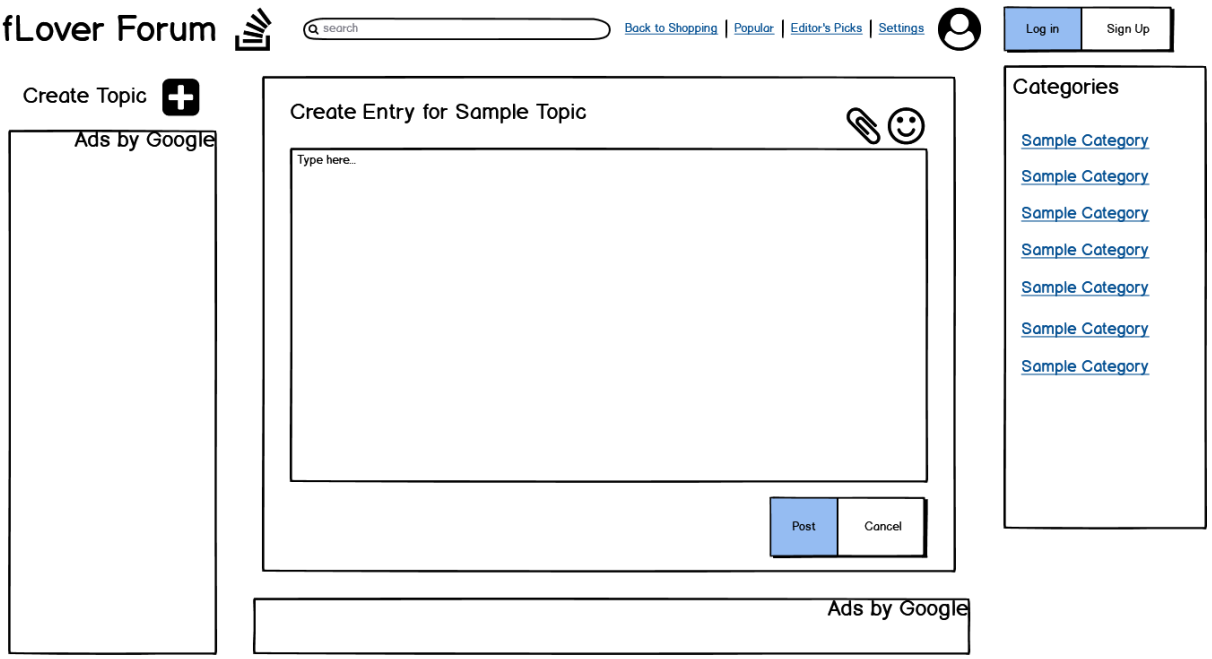


Figure 16: Forum Entry

In this page, we will display creating Forum_entry.
SQL Statement for inserting a new Entry to forum:

```
INSERT INTO Forum_entry(topic_id, date, text, cust_id, seller_id,
courier_id, cust_service_id)
VALUES (@topic_id, @date, @text, @id, NULL, NULL, NULL,
NULL)
```

This statement will vary depending on the user who created the entry. Depending on this, 4 of 5 values will be NULL.

3.17 Forum Topic

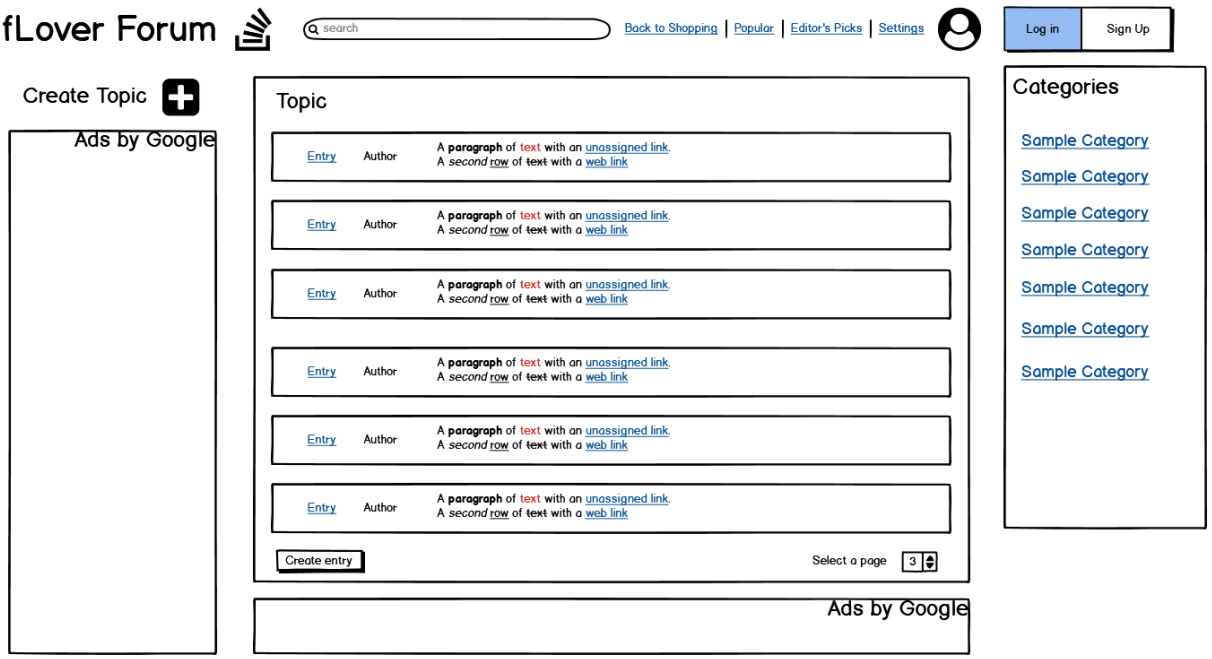


Figure 17: Forum Topic

In this page, we will display entries for the topic.
SQL Statement for retrieving Usernames and ids:

This statement will be repeated for each user type once.

```
SELECT username
FROM Customer
WHERE id IN (
SELECT cust_id
FROM Forum_entry F
WHERE F.topic_id, F.date, F.text IN (
SELECT topic_id, date, text
FROM Forum_entry E
WHERE E.topic_id = @topic_id))
```

SQL Statement for Forum_Entry properties:

```
SELECT text, date, id
FROM Forum_entry E
WHERE topic_id = @topic_id
```

3.18 Create Forum Topic

Figure 18: Create Forum Topic

In this page, we will display creating Forum_Topic.
SQL Statement for inserting a new Topic to forum:

INSERT INTO Forum_topic(topic_id, date, title, category, cust_id, seller_id, courier_id, cust_service_id, no_of_entries)
VALUES (@topic_id, @date, @title, @category, @id, NULL, NULL, NULL, NULL, 0)

This statement will change depending on the user creating the forum topic. According to the user type, 4 of the 5 id values will be NULL.

3.19 Add Flower to Shop

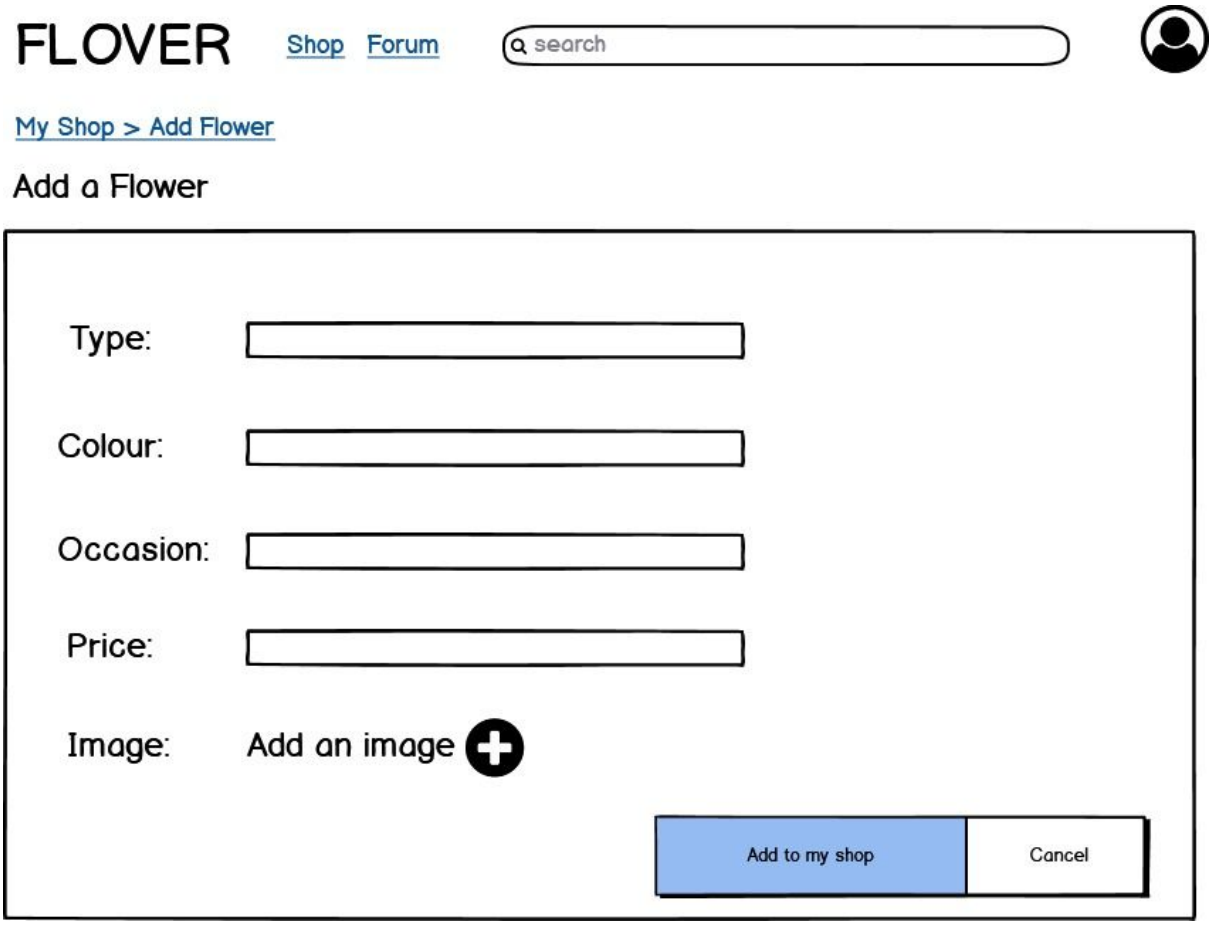


Figure 19: Add Flower to Shop

In this page, sellers can add new flowers to their shops.

SQL Statement for adding flower.

```
INSERT INTO Flowers(flower_id, type, color, occasion, price,
photo_id)
VALUES (@flower_id, @type, @color, @occasion, @price,
@photo_id)
INSERT INTO Stocks (flower_id, id, sold, count)
VALUES (@flower_id, @id, NULL, @count)
```

3.20 Received Orders

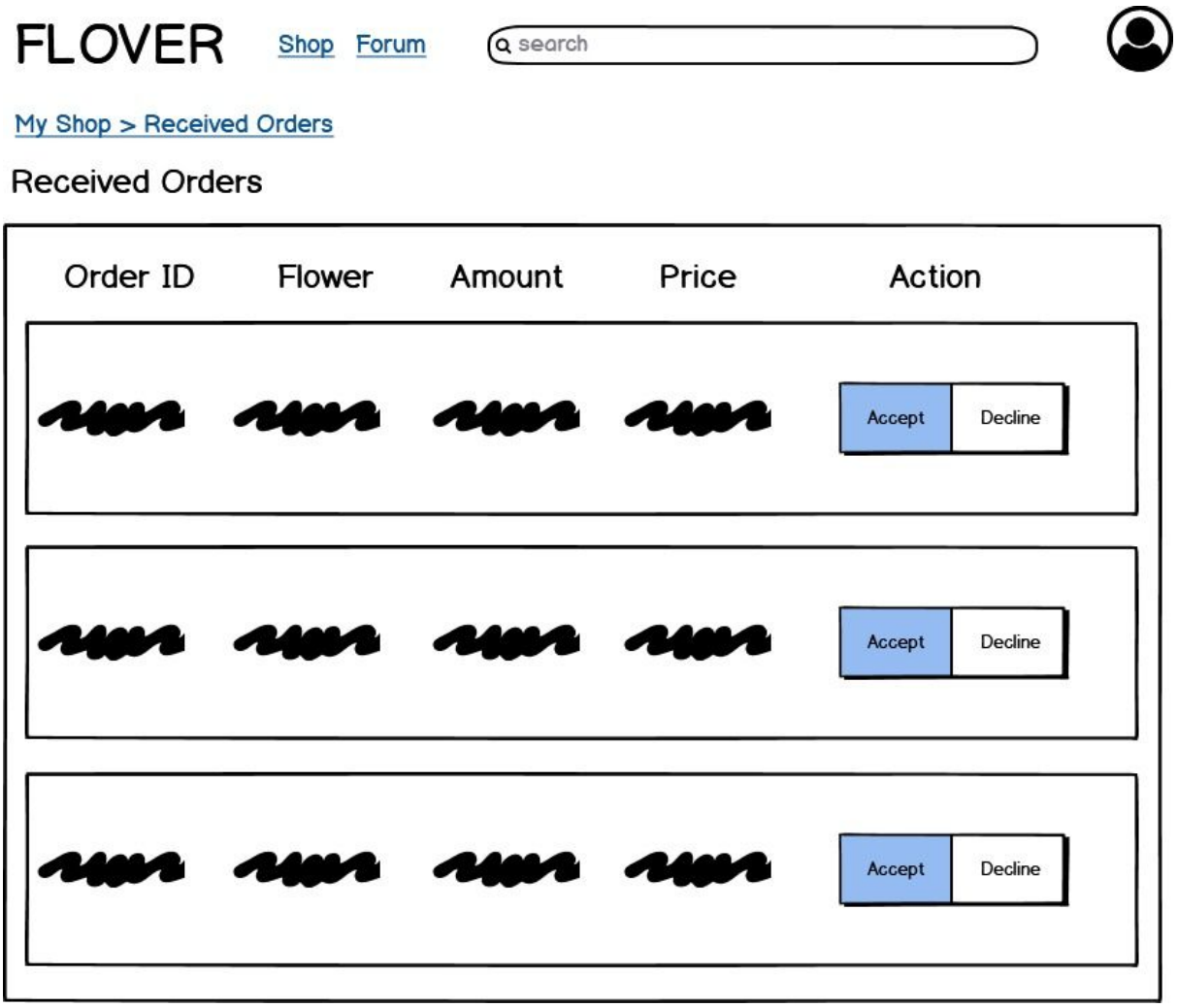


Figure 20: Received Orders

1. For seller
   SQL Statement for accepting order:

```
UPDATE Order_delivery,
SET status = @status
WHERE order_id = @order_id
```

SQL Statement for listing the orders:

```
SELECT flower_id, type, price
FROM Flower
WHERE flower_id IN (
SELECT flower_id
```

```
FROM Includes
WHERE order_id IN (
SELECT order_id
FROM Order_delivery
WHERE seller_id = @id
))

SELECT flower_id, amount
FROM Includes
WHERE order_id IN (
SELECT id
FROM Order
WHERE seller_id = @seller_id)
```

2. For courier:
   System selects a courier for the accepted order:

```
UPDATE Order_delivery,
SET courier_id = @courier_id
WHERE order_id = @order_id
```

SQL Statement for accepting order:

```
UPDATE Order_delivery,
SET status = @status
WHERE order_id = @order_id
```

SQL Statement for listing the orderst:

```
SELECT flower_id, type, price
FROM Flower
WHERE flower_id IN (
SELECT flower_id
FROM Includes
WHERE order_id IN (
SELECT id
FROM Order
WHERE id IN (
```

```
SELECT order_id
FROM Order_delivery
WHERE courier_id = @courier_id)))

SELECT flower_id, amount
FROM Includes
WHERE order_id IN (
SELECT id
FROM Order
WHERE id IN (
SELECT order_id
FROM Order_delivery
WHERE courier_id = @courier_id))
```

3.21 Customer Service Page

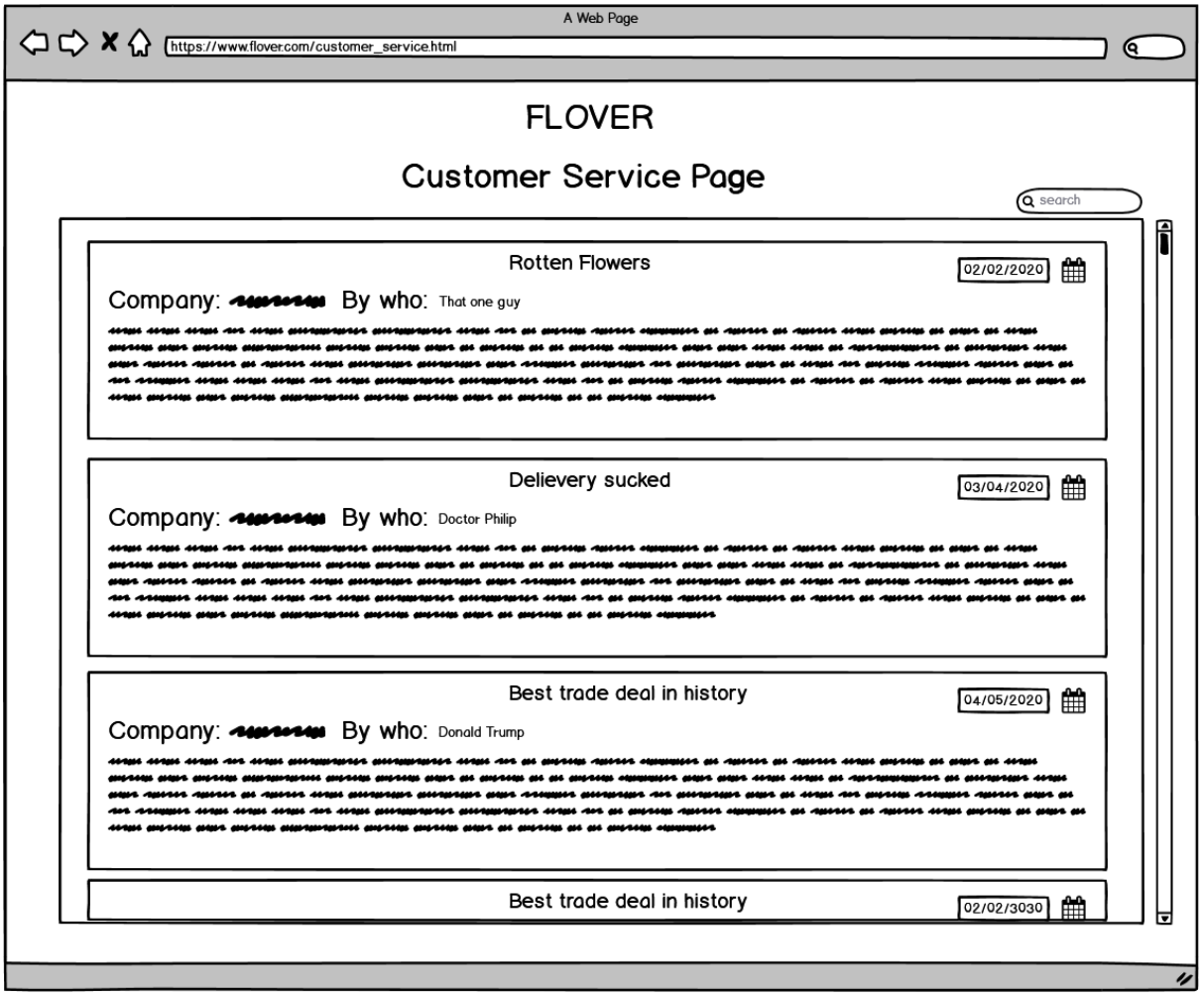


Figure 21: Customer Service Page

In this page customer service can access the complaint reports.
SQL Statement for retrieving all complaints:

```
SELECT order_id, subject, status, cust_id
FROM Complaint_Report
```

SQL Statement for retrieving all Usernames:

```
SELECT Username
FROM Customer
WHERE id IN
(SELECT cust_id
FROM Complaint_Report)
```

3.22 Customer Service Report Page

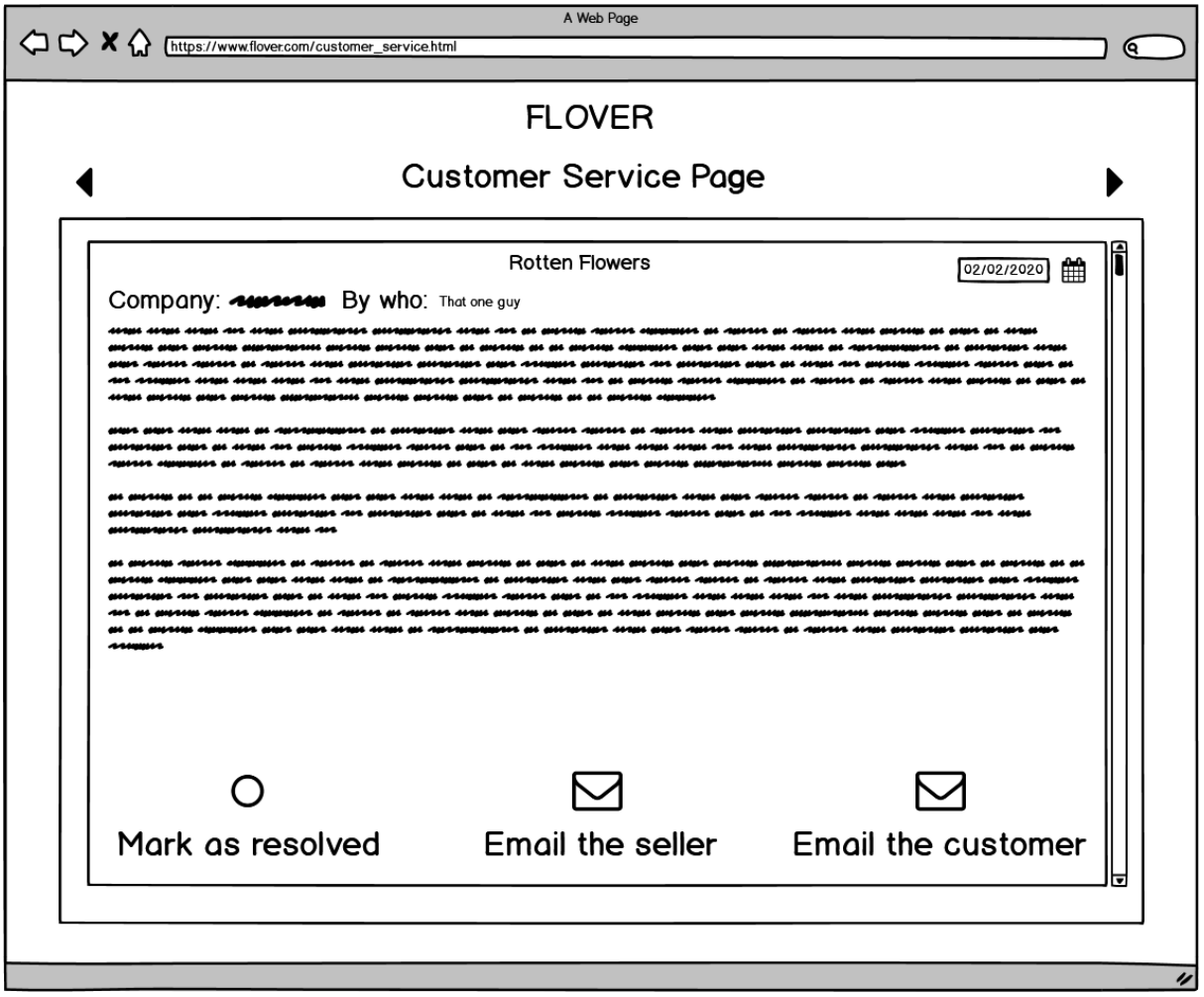


Figure 22: Customer Service Specific Report Page

SQL Statement for retrieving seller email:

```
SELECT email
FROM Seller S
WHERE S.id IN (
SELECT seller_id
FROM Order_delivery
WHERE order_id IN (
SELECT id
FROM Order
WHERE id IN (
SELECT order_id
FROM Complaint_report
WHERE order_id = @order_id)))
```

SQL Statement for retrieving Customer email:

```
SELECT email
FROM Customer C
WHERE C.id IN (
SELECT cust_id
FROM Order
WHERE id IN (
SELECT order_id
FROM Complaint_report
WHERE order_id = @order_id))
```

SQL Statement for updating resolved complaint:

```
UPDATE Complaint_Report
SET status = @status
WHERE order_id = @order_id, subject = @subject, status =
@status
```