DEPARTMENT OF COMPUTER ENGINEERING

CS 353 - Database Systems

Online Flower Shopping System

Project Design Report

Instructor: Özgür Ulusoy

TA: Duygu Durmuş

Group 31

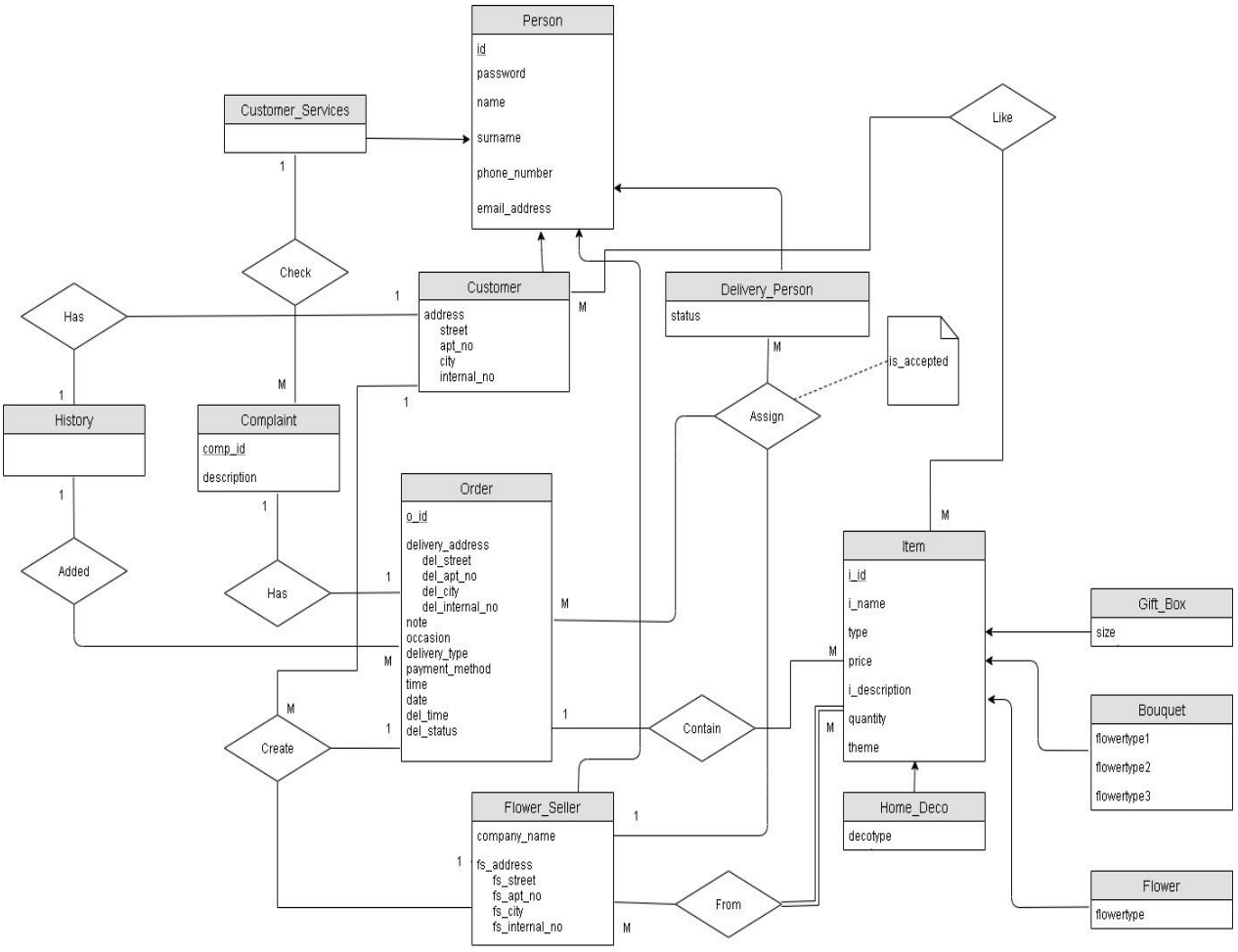
Doğacan Kaynak - 21400682

Burak Yeni - 21502761

Yiğit Gülben - 21101130

Deniz Doğanay - 21100658

1. Revised ER Model



According to our assistant's review and also during the design process we made the following changes in our E/R model in order to have a better structure for out database system:

- We have additional functionalities such that :
    - Customer can create their own gift box,
    - Customer can create their own bouquet,
    - Customer can order home decoration
    - Customer can pay without registering on our site.
- Primary key complications reduced.
- Limitations set to what customer can do on our system.
- Relationship complications reduced.
- Stock added as specifying quantity of items.

- Order history added to customer.
- Deliver relation deleted and its attributes added to order.
- Complaint functionality changed.
- In order you can make a description according to your demand.

2. Relation Schemas

2.1. Person

Relational Model:

Person(person_id, password, name, surname, phone_number(), email_address)

Functional Dependencies:

person_id -> password, name, surname, phone_number, email_address

email_address -> person_id ,password, name, surname, phone_number()

Candidate Keys:

{ (person_id),(email_address) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Person(
        person_id       INT PRIMARY KEY AUTO_INCREMENT,
        password        VARCHAR(32) NOT NULL,
        name            VARCHAR(32) NOT NULL,
        surname         VARCHAR(32) NOT NULL,
        phone_number    INT UNIQUE NOT NULL,
        email_address   VARCHAR(32) UNIQUE NOT NULL);
```

2.2. Customer_Services

Relational Model:

Customer_Services(cs_id)

Functional Dependencies:

None

Candidate Keys:

{ (cs_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Customer_Services(
        cs_id                   INT PRIMARY KEY,
        FOREIGN KEY (cs_id) REFERENCES Person(person_id);
```

2.3. Customer

Relational Model:

Customer(c_id)

Functional Dependencies:

c_id -> street, apt_no, street, city, internal_no

Candidate Keys:

{ (c_id)}

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Customer(
        c_id            INT PRIMARY KEY,
        street          VARCHAR(32) NOT NULL,
        apt_no          INT NOT NULL,
        city            VARCHAR(32) NOT NULL,
```

```
internal_no    VARCHAR(32) NOT NULL);
FOREIGN KEY (c_id) REFERENCES Person(person_id);
```

2.4. Flower_Seller

Relational Model:

Flower_Seller(fs_id)

Functional Dependencies:

fs_id -> company_name, street, apt_no, city, internal_no

Candidate Keys:

{ (fs_id)}

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Flower_Seller(
    fs_id            INT PRIMARY KEY,
    company_name     VARCHAR(32) NOT NULL,
    fs_street        VARCHAR(32) NOT NULL,
    fs_apt_no        INT NOT NULL,
    fs_city          VARCHAR(32) NOT NULL,
    fs_internal_no   VARCHAR(32) NOT NULL);
    FOREIGN KEY (fs_id) REFERENCES Person(person_id);
```

2.5. Delivery_Person

Relational Model:

Delivery_Person(dp_id)

Functional Dependencies:

dp_id -> person_id

Candidate Keys:

{ (dp_id)}

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Delivery_Person(
        dp_id               INT PRIMARY KEY,
        status              VARCHAR(32) NOT NULL,
        FOREIGN KEY (dp_id) REFERENCES Person(person_id);
```

2.6. Order

Relational Model:

Order(o_id, fs_id , street, apt_no, city, internal_no , note, occasion, delivery_type, payment_method, time, date, del_time, del_status)

Functional Dependencies:

o_id -> street, apt_no, city, internal_no , note, occasion, delivery_type, payment_method, time, date, del_time, del_status, fs_id

fs_id -> o_id , street, apt_no, city, internal_no , note, occasion, delivery_type, payment_method, time, date, del_time, del_status

Candidate Keys:

{ (o_id),(fs_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Order(
```

```
        o_id                INT PRIMARY KEY AUTO_INCREMENT,
        del_street          VARCHAR(32) NOT NULL,
        del_apt_no          INT NOT NULL,
        del_city            VARCHAR(32) NOT NULL,
        del_internal_no     VARCHAR(32) NOT NULL,
        note                VARCHAR(32),
        occasion            VARCHAR(32),
        delivery_type       VARCHAR(32) NOT NULL,
        payment_method      VARCHAR(32) NOT NULL,
        time                VARCHAR(32) NOT NULL,
        date                VARCHAR(32) NOT NULL,
        del_time            VARCHAR(32) NOT NULL,
        del_status          VARCHAR(32) NOT NULL,
        fs_id               INT NOT NULL);
        PRIMARY KEY (o_id, fs_id),
        FOREIGN KEY (fs_id) REFERENCES Flower_Seller);
```

2.6. Complaint

Relational Model:

Complaint(comp_id)

Functional Dependencies:

None

Candidate Keys:

{ (comp_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Complaint(
        comp_id             INT PRIMARY KEY AUTO_INCREMENT,
        description         VARCHAR(32),
        o_id                INT NOT NULL);
```

```
PRIMARY KEY (comp_id, o_id),
FOREIGN KEY (o_id) REFERENCES Order);
```

2.7. History

Relational Model:

History(c_id, o_id, del_street, del_apt_no, del_city, del_internal_no , del_time, del_status)

Functional Dependencies:

c_id, o_id -> del_street, del_apt_no, del_city, del_internal_no , del_time, del_status

Candidate Keys:

{ (comp_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE History(
        c_id                INT NOT NULL,
        o_id                INT NOT NULL,
        del_street          VARCHAR(32) NOT NULL,
        del_apt_no          INT NOT NULL,
        del_city            VARCHAR(32) NOT NULL,
        del_internal_no     VARCHAR(32) NOT NULL,
        del_time            VARCHAR(32) NOT NULL,
        del_status          VARCHAR(32) NOT NULL,
        PRIMARY KEY (c_id, o_id),
        FOREIGN KEY (c_id) REFERENCES Customer),
        FOREIGN KEY (o_id) REFERENCES Order),
        FOREIGN KEY (del_street) REFERENCES Order),
```

```
FOREIGN KEY (del_apt_no) REFERENCES Order),
FOREIGN KEY (del_city) REFERENCES Order),
FOREIGN KEY (del_internal_no) REFERENCES Order),
FOREIGN KEY (del_time) REFERENCES Order),
FOREIGN KEY (del_status) REFERENCES Order);
```

2.8. Assignment

Relational Model:

Assignment (dp_id, c_id, o_id, fs_street, fs_apt_no, fs_city, fs_internal_no , del_street, del_apt_no, del_city, del_internal_no , is_accepted)

Functional Dependencies:

dp_id -> c_id, o_id, fs_street, fs_apt_no, fs_city, fs_internal_no , del_street, del_apt_no, del_city, del_internal_no , is_accepted

Candidate Keys:

{ (dp_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Assignment(
    dp_id            INT NOT NULL,
    c_id             INT NOT NULL,
    o_id             INT NOT NULL,
    fs_street        VARCHAR(32) NOT NULL,
    fs_apt_no        INT NOT NULL,
    fs_city          VARCHAR(32) NOT NULL,
    fs_internal_no   VARCHAR(32) NOT NULL,
    del_street       VARCHAR(32) NOT NULL,
    del_apt_no       INT NOT NULL,
    del_city         VARCHAR(32) NOT NULL,
    del_internal_no  VARCHAR(32) NOT NULL,
    is_accepted      VARCHAR(32) NOT NULL,
```

```
PRIMARY KEY (dp_id, c_id, o_id),
FOREIGN KEY (c_id) REFERENCES Customer),
FOREIGN KEY (o_id) REFERENCES Order);
FOREIGN KEY (fs_street) REFERENCES Flower_Seller),
FOREIGN KEY (fs_apt_no) REFERENCES Flower_Seller),
FOREIGN KEY (fs_city) REFERENCES Flower_Seller),
FOREIGN KEY (fs_internal_no ) REFERENCES Flower_Seller),
FOREIGN KEY (del_street) REFERENCES Order),
FOREIGN KEY (del_apt_no) REFERENCES Order),
FOREIGN KEY (del_city) REFERENCES Order),
FOREIGN KEY (del_internal_no ) REFERENCES Order);
```

2.9. Item

Relational Model:

Item(i_id, i_name, type, price, i_description, quantity, theme)

Functional Dependencies:

i_id-> i_name, type, price, i_description, quantity, theme

Candidate Keys:

{ (i_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Item(
    i_id            INT PRIMARY KEY AUTO_INCREMENT,
    i_name          VARCHAR(32) NOT NULL,
    type            VARCHAR(32) NOT NULL,
    price           INT NOT NULL,
    i_description   VARCHAR(32) NOT NULL,
    quantity        INT,
```

```
        theme                   VARCHAR(32) NOT NULL),
```

2.10. Gift_Box

Relational Model:

Gift_Box(gb_id,size)

Functional Dependencies:

None

Candidate Keys:

{ (i_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Gift_Box(
        gb_id                   INT PRIMARY KEY,
        size                    VARCHAR(32) NOT NULL,
        FOREIGN KEY (gb_id) REFERENCES Item(i_id);
```

2.11. Bouquet

Relational Model:

Bouquet(bq_id,flowertype1, flowertype2, flowertype3)

Functional Dependencies:

None

Candidate Keys:

{ (bq_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Bouquet(
        bq_id           INT PRIMARY KEY,
        flowertype1     VARCHAR(32) NOT NULL,
        flowertype2     VARCHAR(32) NOT NULL,
        flowertype3     VARCHAR(32) NOT NULL,
        FOREIGN KEY (bq_id) REFERENCES Item(i_id);
```

2.12. Flower

Relational Model:

Flower(fl_id, flowertype)

Functional Dependencies:

None

Candidate Keys:

{ (fl_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Flower(
        fl_id           INT PRIMARY KEY,
        flowertype      VARCHAR(32) NOT NULL,
        FOREIGN KEY (fl_id) REFERENCES Item(i_id);
```

2.13. Home_Deco

Relational Model:

Home_Deco(hd_id)

Functional Dependencies:

None

Candidate Keys:

{ (hd_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

```
CREATE TABLE Flower(
        hd_id           INT PRIMARY KEY,
        decotype        VARCHAR(32) NOT NULL,
        FOREIGN KEY (hd_id) REFERENCES Item(i_id);
```

2.14. Favorites

Relational Model:

Favorites(f_id)

Functional Dependencies:

None

Candidate Keys:

{ (f_id) }

Normal Form:

Because it's already in BCNF it is also 3NF

Table Definition:

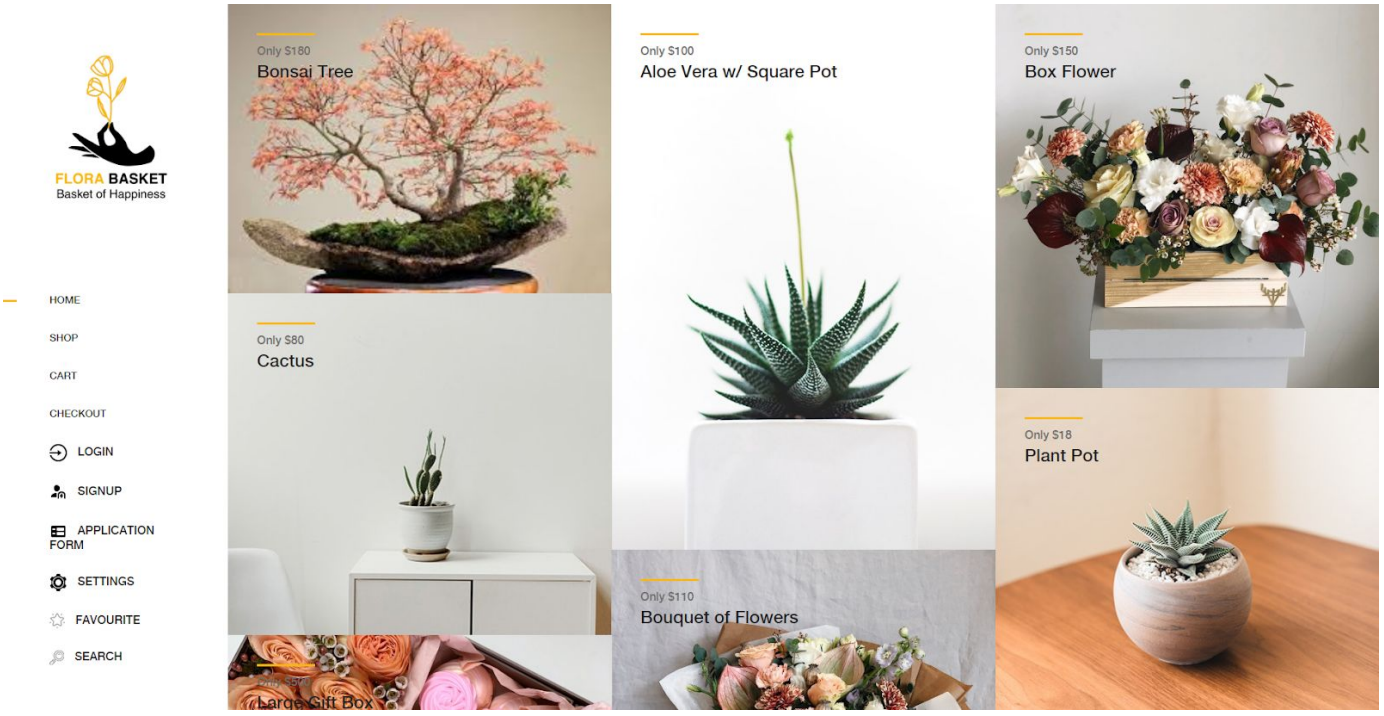
```
CREATE TABLE Favorites(
        c_id            INT NOT NULL,
        i_id            INT NOT NULL,
        FOREIGN KEY (c_id) REFERENCES Customer,
```

FOREIGN KEY (i_id) REFERENCES Item;

3. User Interface Design & Corresponding SQL Statements

3.1. Customer Interface Design

3.1.1. Customer Home Screen



Inputs: @searchquery

Process: The homepage for the Online Flower Shopping System is displayed above. Non-registered users are welcomed with random items. There is a navigation menu on the left side of the main page. Users can access many pages through this left menu.

SQL Statements:

Displaying random items

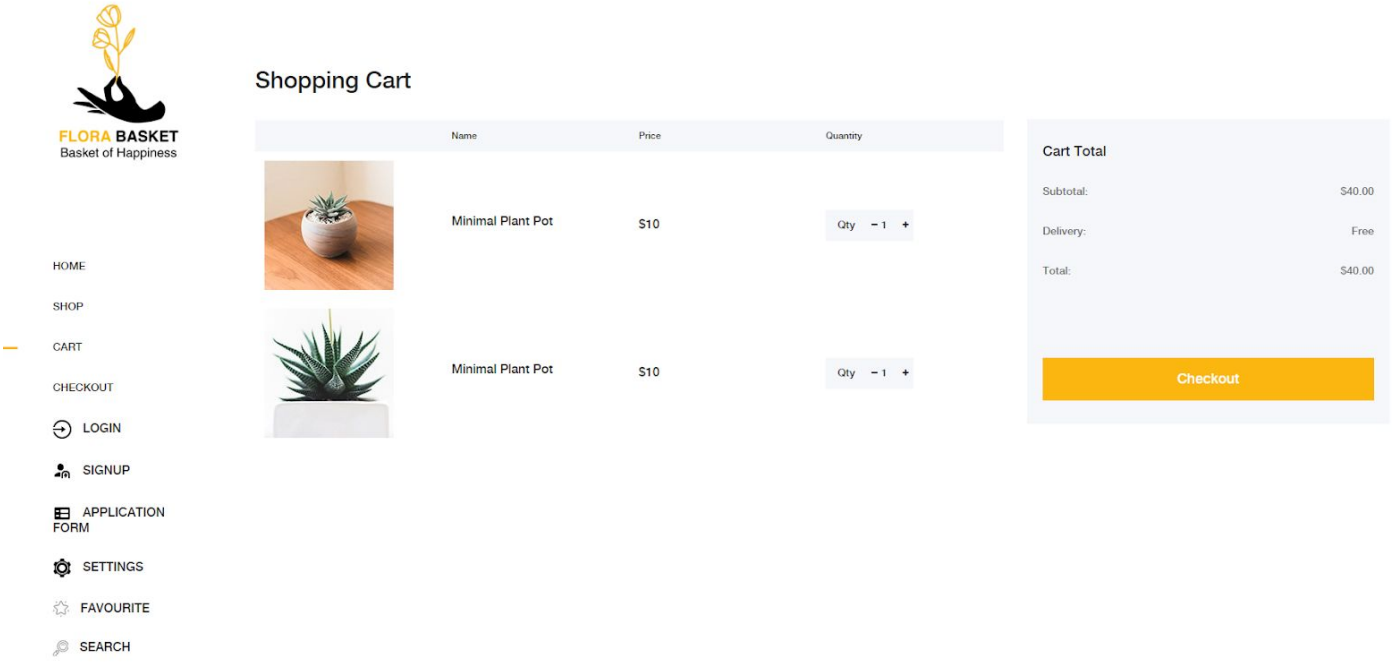
CREATE VIEW displayed_items (name, price) AS

SELECT name, price

FROM item

LIMIT 15; -- number of items shown in main screen

3.1.2. Shopping Cart Screen



Inputs: @quantity, @i_id

Process: Users can increase or decrease the number of flowers then go to the checkout page by clicking Checkout.

SQL Statements:

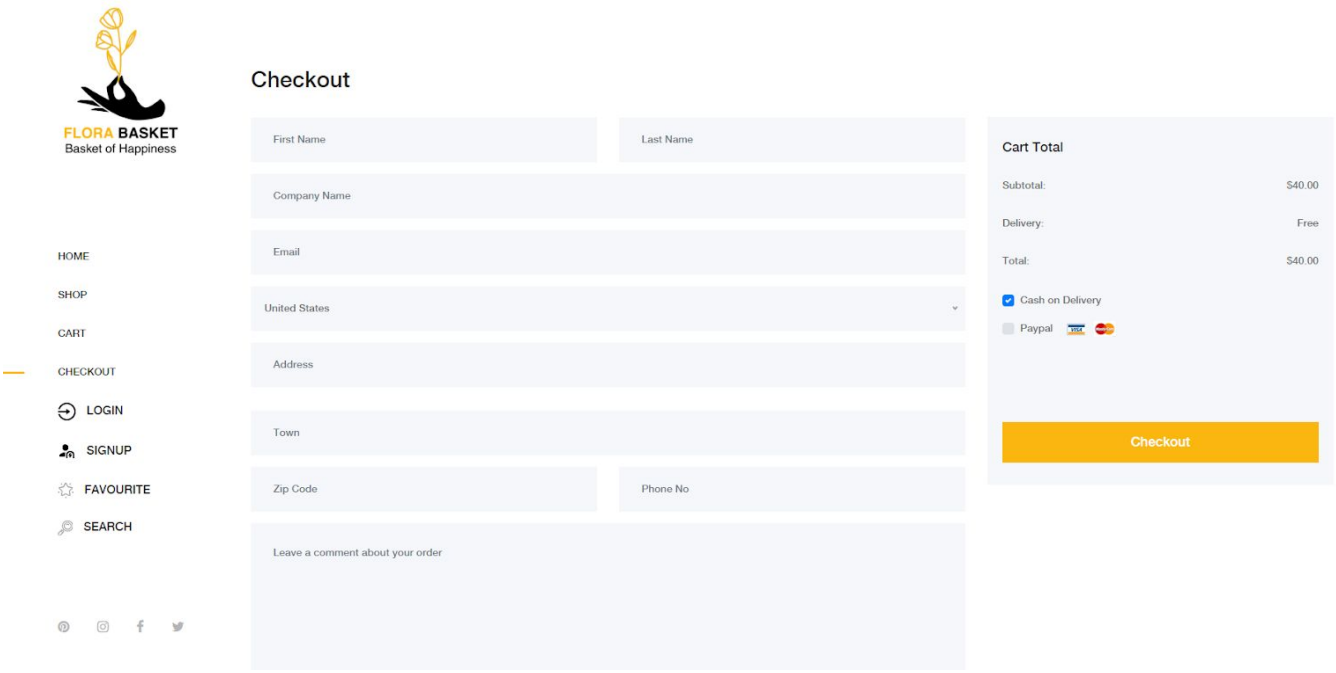
Shopping Cart

UPDATE item

SET quantity = @quantity

WHERE i_id = @i_id

3.1.3. Checkout Screen



Inputs: @del_street, @del_apt_no, @del_city, @del_internal_no, @note, @occasion, @delivery_type, @payment_method

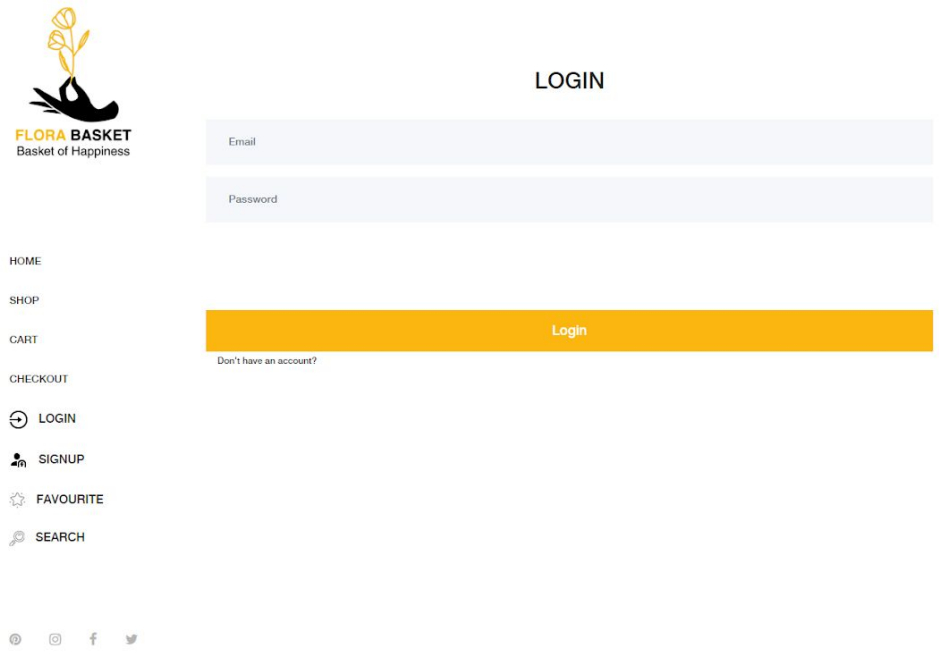
Process: The user can checkout items in his cart after he enters the necessary information.

SQL Statements:

Checkout Information

INSERT INTO order (street, apt_no, city, internal_no, note, occasion, delivery_type, payment_method) -- time, date and flower_seller will be determined by the system
VALUES (@del_street, @del_apt_no, @del_city, @del_internal_no, @note, @occasion, @delivery_type, @payment_method)

3.1.4. Login Page



Inputs: @email, @password

Process: Users can login by entering their emails and passwords. If a user does not have any account, he can go to the sign-up page by clicking Don't have an account.

SQL Statements:

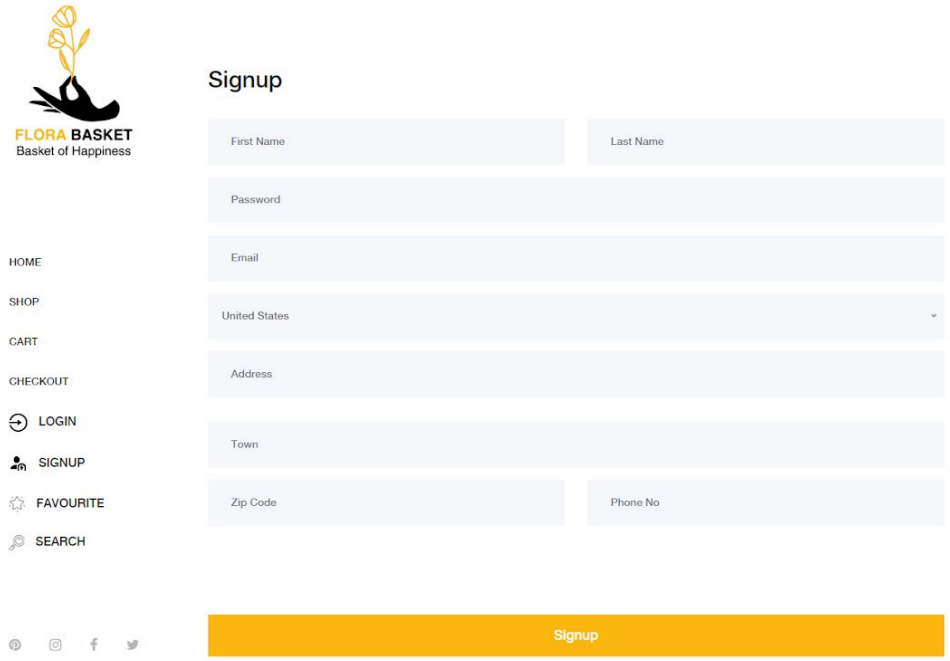
Login

SELECT *

FROM person

WHERE email = @email AND password = @password

3.1.5. Signup Page



Inputs: @name, @surname, @phone_number, @email_address, @password, @street, @apt_no, @city, @internal_no

Process: The user enters necessary information then clicks Sign-up to create a new account.

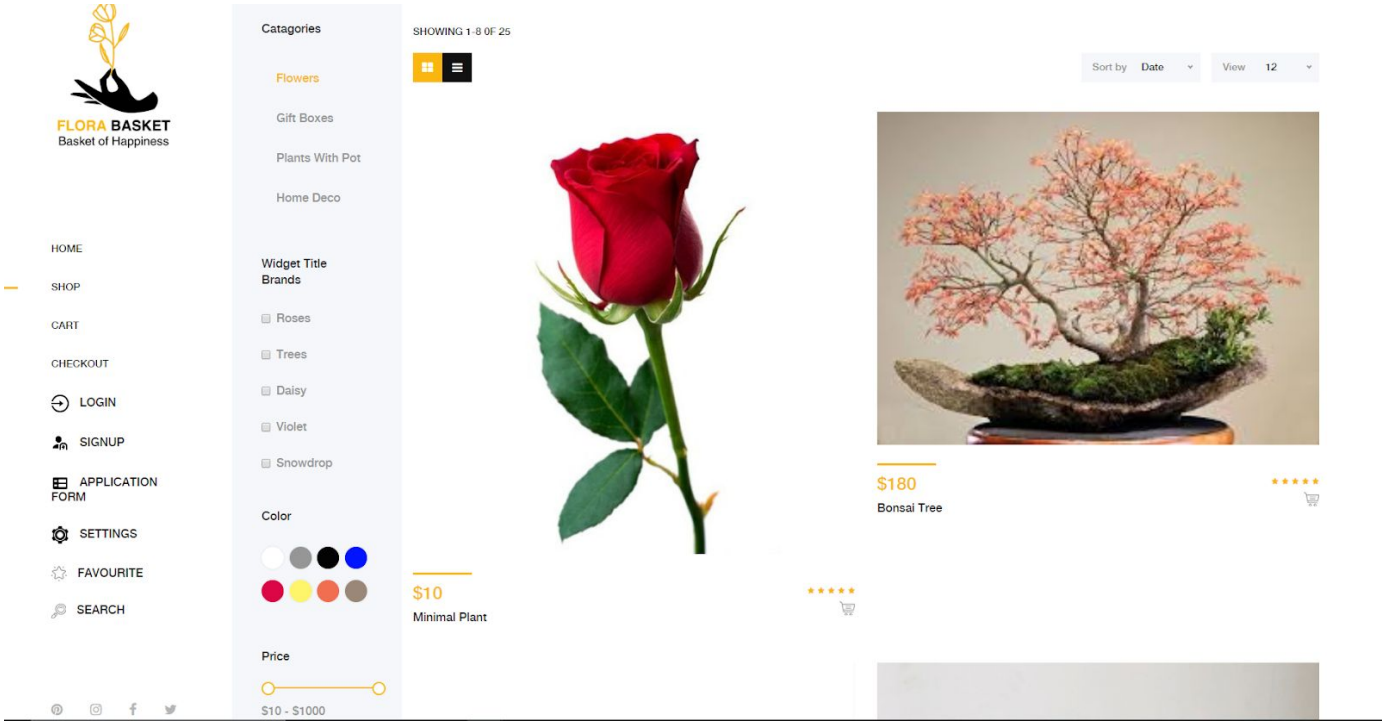
SQL Statements:

Registration

INSERT INTO person

VALUES (@name, @surname, @phone_number, @email_address, @password, @street, @apt_no, @city, @internal_no)

3.1.6. Categories Screen



Inputs: @search_query, @type, @min_price, @max_price, @order_type, @view_limit

Process: In this screen, users can see items. Grid and list options are available. Users can also sort items and adjust how many items can be seen on one page. Users can access different types of item categories or select some filters to eliminate unnecessary items.

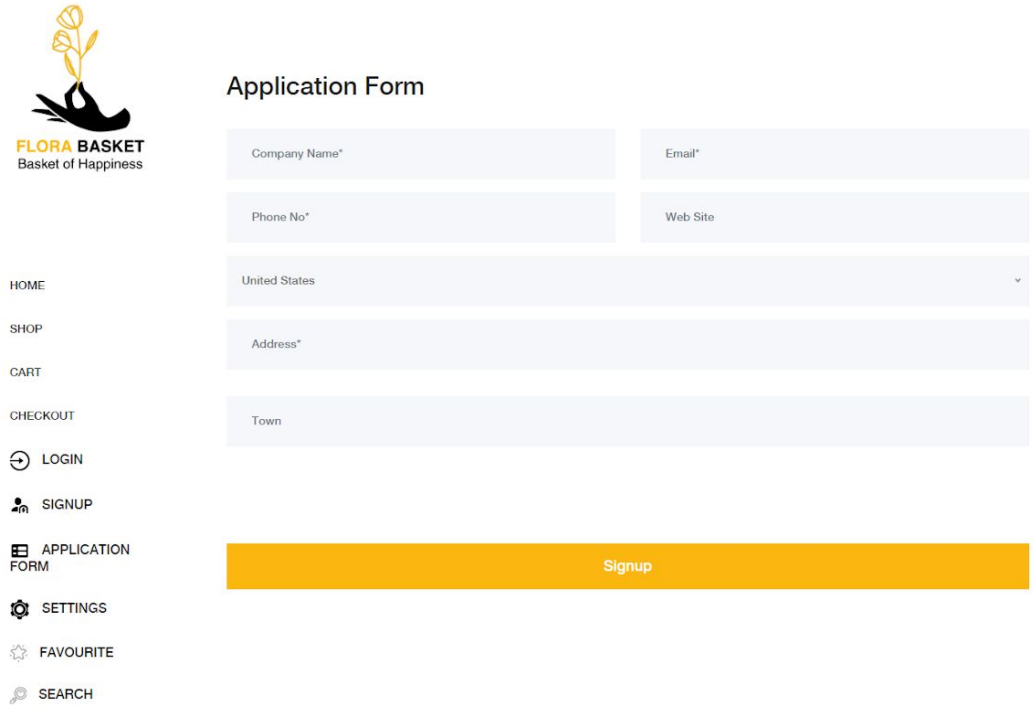
SQL Statements:

Listing items

```
SELECT name, price
FROM item
WHERE type = @type AND price >= @min_price AND price <= max_price
ORDER BY @order_type
LIMIT @view_limit
```

3.2. Flower-seller Interface Design

3.2.1. Application Form Page for Flower Sellers



Inputs: @name, @surname, @phone_number, @email_address, @password, @street, @apt_no, @city, @internal_no

Process: The flower-sellers enter necessary information then clicks Sign-up to create a new account for their companies.

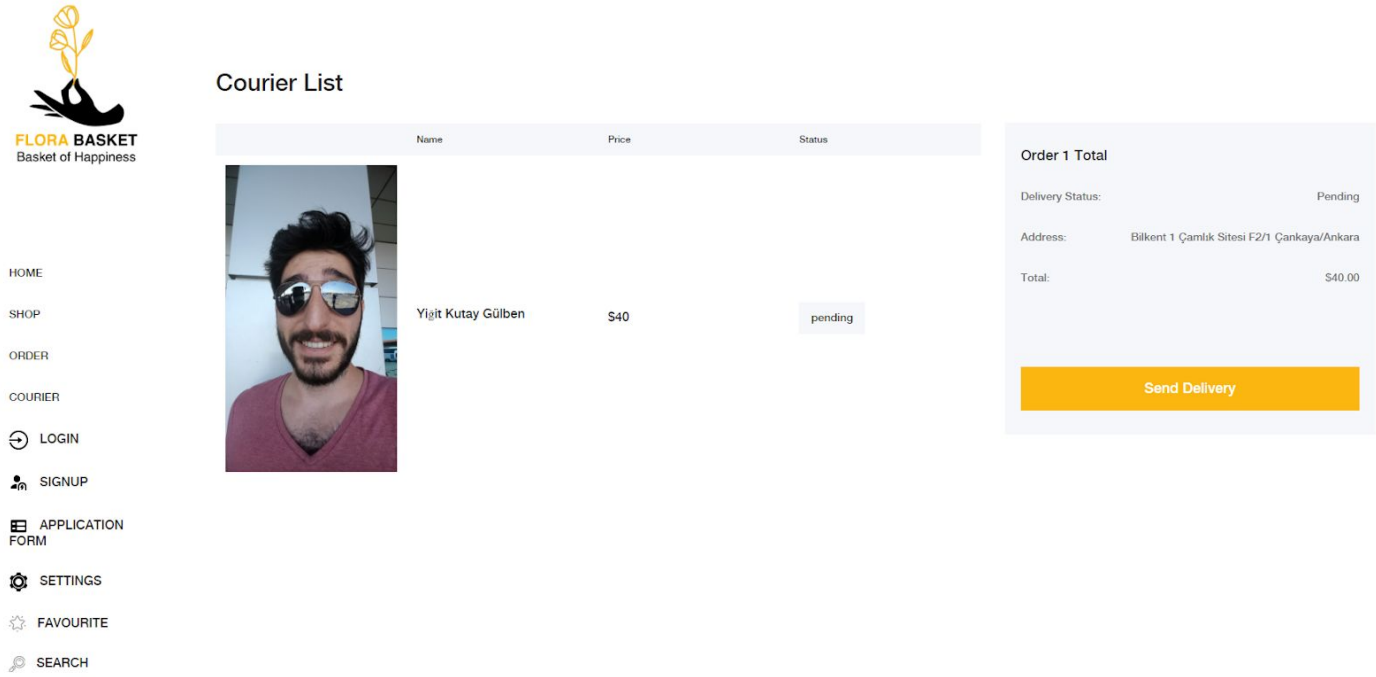
SQL Statements:

Application

INSERT INTO flower_seller

VALUES (@name, @surname, @phone_number, @email_address, @password, @fs_street, @fs_apt_no, @fs_city, @fs_internal_no)

3.2.2. Courier List Page For Flower Sellers



Inputs: @searchquery

Process: In this page, the available couriers for flower sellers are listed and flower sellers choose one of them to deliver the order.

SQL Statements:

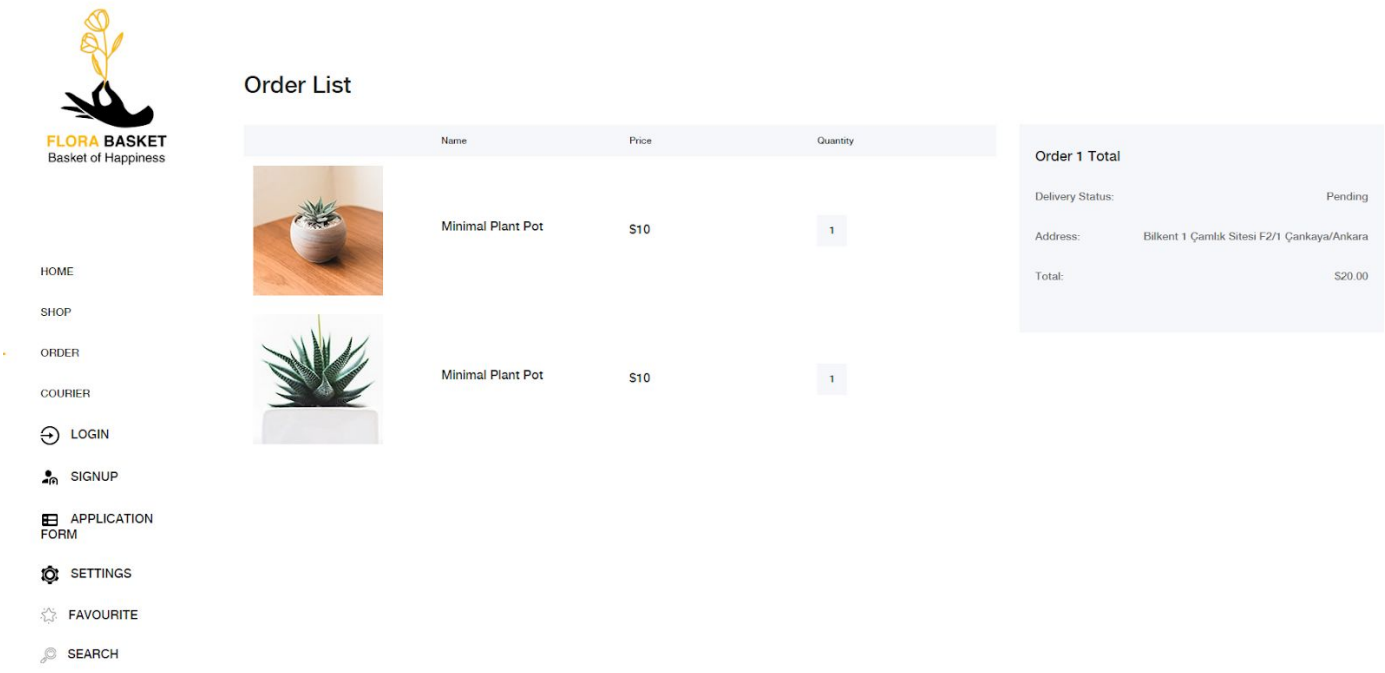
Courier List

SELECT name, surname, price, availability

FROM delivery_person

WHERE availability = ‘available’

3.2.3. Order Page for flower seller



Inputs: @searchquery, @selected_o_id

Process: In this page, flower sellers can see the specifications of chosen order.

SQL Statements:

Items in Order

SELECT name, price, quantity

FROM item

Order Specs

SELECT del_status, del_street, del_apt_no, del_city, del_internal_no

FROM order

WHERE o_id = @selected_o_id

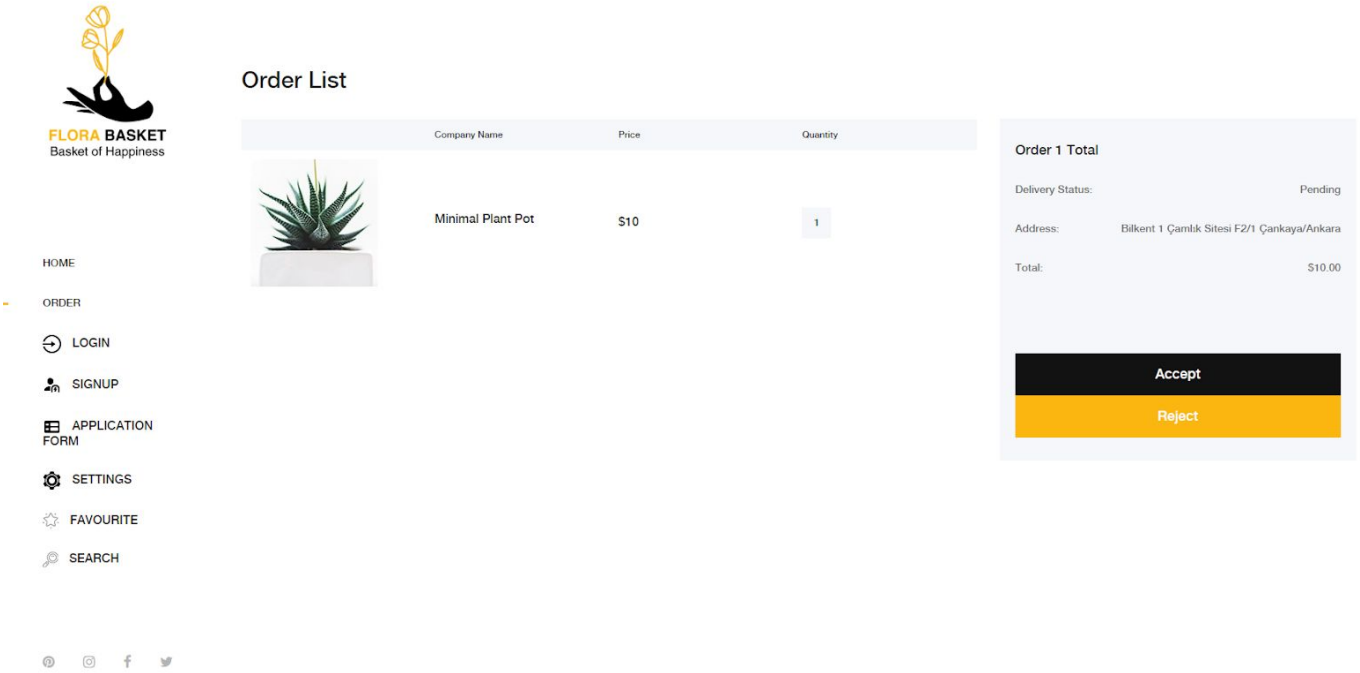
Total Price

SELECT SUM(price)

FROM item

3.3. Courier Interface Design

3.3.1. Available Orders For Courier



Inputs: @searchquery, @selected_o_id

Process: In this page, courier can accept or reject available order by clicking buttons

SQL Statements:

Items in Order

SELECT name, price, quantity

FROM item

Order Specs

SELECT del_status, del_street, del_apt_no, del_city, del_internal_no

FROM order

WHERE o_id = @selected_o_id

Total Price

```
SELECT SUM(price)
FROM item
```

4. Implementation Plan

For our system functionalities and user interface in our flower shopping system, we have finished the UI with Bootstrap, HTML, CSS, PHP and Javascript, within Javascript we think to use JQuery. In order to manage the flow of data in our project, we are planning to use MySQL Server.

5. Website

- Our project website link is on the below:
  https://dogacankaynak.github.io/OnlineFlowerShopingSystem2/
- Our project information link is on the below:
  https://dogacankaynak.github.io/onlineFlowerShoppingSystem/