



Bilkent University

Department of Computer Engineering

CS 353 - Database Systems

Design Report

Group No: 32

Online Flower Shopping System

Project Name: FlowerGarden

Assigned TA:

Duygu Durmuş

Team Members:

Munib Emre Sevilgen -3

Esra Nur Deniz -2

Meryem Banu Cavlak -2

Osman Burak İntişah -3

Table of Contents

Revised E/R Model	3
Modifications	3
Revised E/R Diagram	5
Relational Schemas	6
Account	6
Timeslot	6
Seller	6
Customer	7
Saved_addresses	7
Courier	8
Customer_service	8
Complaint	8
Order	9
District	10
Province	10
Flower	10
Flower_arrangement	11
Occasion	11
Comment	12
Notification	12
Seller_serves_to	12
Courier_serves_to	13
Flower_stock	13
Composed_of	13
Seller_working_time	14
Courier_working_time	14
Service_working_time	15
User Interface Design and SQL Statements	16
Authorization Pages	16
Login Page	16
Forgot Password Page	17
Sign Up Page	18
Customer and Customer Service	18
Courier	19
Seller	20
Customer Pages	21
Select District Page	21
Home Page	22
Flower Arrangement Page	23

Order Creation Page	24
Receiver Information Page	24
Payment Page	25
Order Tracking Page	26
Order Page	27
Create Complaint Page	28
Create Comment Page	29
Seller Pages	30
Select Districts and Working Hours Page	30
Arrangement List Page	31
Arrangement Page	32
Create Arrangement Page	33
Sale List Page	34
Sale Page	35
Assign Courier Page	37
Stock Update Page	38
Courier Pages	39
Select Districts Page	39
Delivery List Page	40
Delivery Page	41
Customer Service Pages	42
Complaints Page	42
Complaint Page	43
Common Pages	44
Notification List Page	44
Account Page	45
Customer	45
Seller	46
Courier	47
Customer Service	47
Implementation Plan	48
Website	48

1. Revised E/R Model

1.1. Modifications

- In the proposal, there were both order and suborder entities. When a customer makes an order, the order is divided into suborders by the system so that customers could order from different stores. Because of the relation between order and suborder, there was no relation between order and floral arrangement and seller, order and courier. In the design report, we have removed the suborder because it would be hard to implement and added attributes of the suborder to the order entity. In this design, if a customer wants to order from different stores, s/he must make the orders separately. We have added a relation between floral arrangement and order and a ternary relation between the seller, order, and courier.
- In the proposal, there was no total participation relation between flower arrangement and flower entity. However, we have realized that there should be a total participation relation between them. Flower entities are entered into the system by the administrator and when a seller adds an arrangement, s/he must pick the flowers that arrangement contains. Therefore, there should be total participation in the relation.
- In the proposal, there was no attribute or relation for the stock information of the flowers which sellers have. We have added a "flower_stock" relation between seller and flower entity to hold the stock information. Even though the sellers sell flower arrangements we decided to store the stock information of flowers because we assumed that sellers produce the flower arrangements as they are ordered. Then, we planned to calculate the maximum amount of flower arrangements that can be produced to decide on whether or not a seller can prepare the order before approving a customer's order.
- In the proposal, there was a "working hours" attribute belonging to the seller, courier and customer entities. However, we have removed this attribute from these entities, made a new entity for the working hours and added relations between this entity and seller, courier and customer service. In this design, the time slots are entered into the database by the administrator and seller, courier and customer service pick the time slots they work.
- Sellers and couriers also select the districts they are going to work in our system. It is an important feature we added to ease the jobs of customers, sellers, and couriers. Because in this way customers are going to search for sellers that serve in the corresponding district and the sellers will not need to reject orders that are not suitable for service for them and customers will not need to make several orders if a seller rejects the order because of the same reason. Similarly, sellers will be able to select couriers only from those ones that serve in the corresponding district.
- Customers can pick the delivery date and time of their orders. For this functionality, we have added attributes to the order entity which is one of the new functionalities we added to the system. Also, customers can display the status of their order, therefore we added relevant attributes to the order entity.
- As a new functionality for the system, we have added the comment entity and relation between comment and flower arrangement entity. A customer can comment on the arrangement s/he buys and rates the flower arrangement. Besides, customers can display flower arrangements with respect to the rates of the arrangements. We have added the rating functionality also to the customer service. Customers can rate the

customer service for the response of their complaint. There is an attribute of the customer service entity that holds the average of the ratings.

- As another functionality, we have added the notification entity to the system. The seller, customer, courier and customer service have a relation between the notification entity.

1.2. Revised E/R Diagram

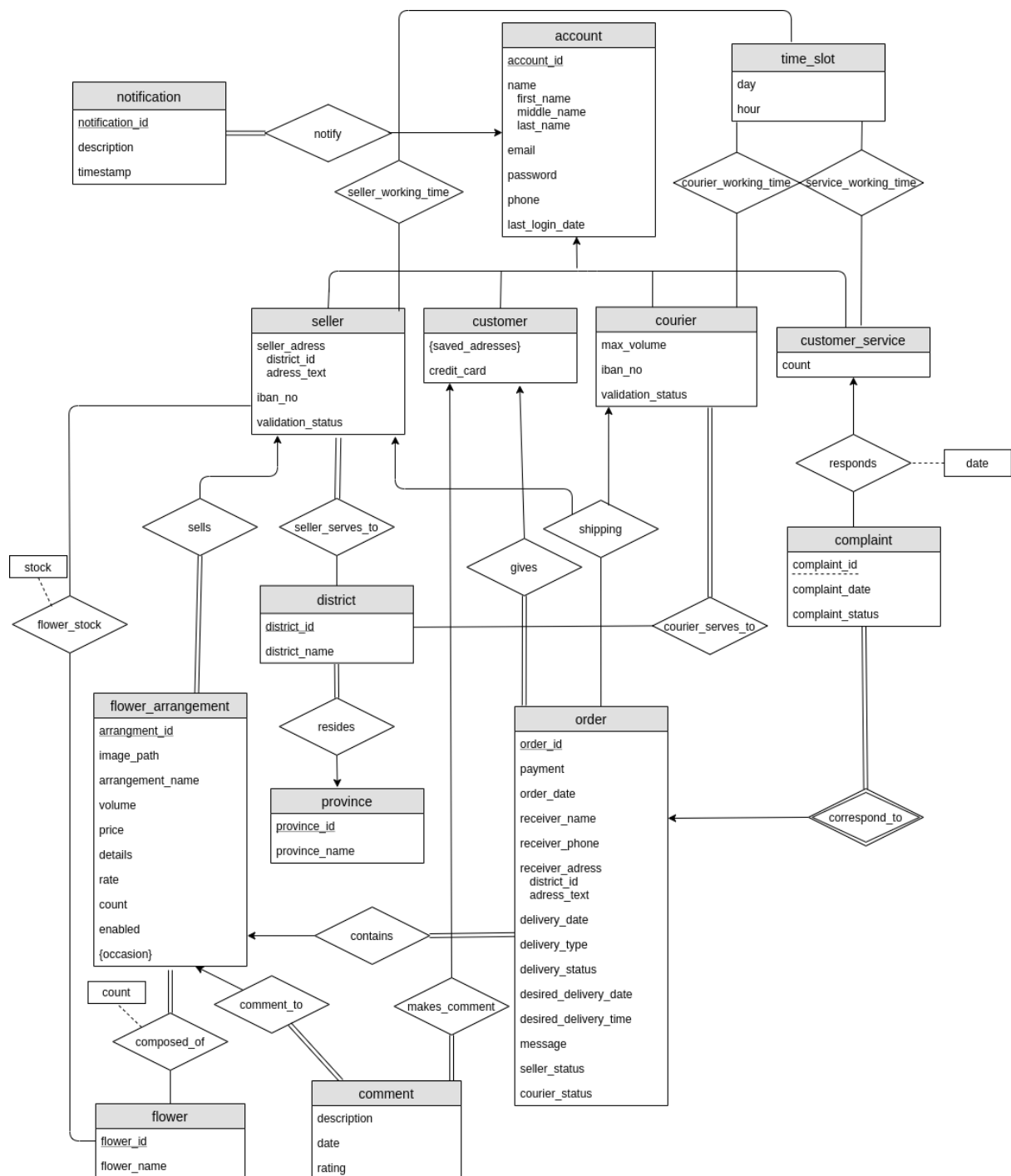


Figure 1: ER Diagram

2. Relational Schemas

2.1. Account

Relational Model: Account(account_id, first_name, middle_name, last_name, email, password, phone, last_login_date)

Functional Dependencies: {(account_id -> first_name, middle_name, last_name, email, password, phone, last_login_date), (email -> account_id, first_name, middle_name, last_name, password, phone, last_login_date)}

Candidate Keys: {{account_id} , {email}}

Primary Key: {account_id}

Foreign Keys: {}

Normal Form: BCNF

Table Definition:

```
create table account(  
    account_id int primary key auto_increment,  
    first_name varchar(20) not null,  
    middle_name varchar(20),  
    last_name varchar(20) not null,  
    email varchar(50) not null,  
    password varchar(20) not null,  
    phone int,  
    last_login_date timestamp,  
    primary key (account_id)  
);
```

2.2. Timeslot

Relational Model: Timeslot(hour,day)

Functional Dependencies: {(hour,day -> hour,day)}

Candidate Keys: {(hour,day)}

Primary Key: {(hour,day)}

Foreign Keys: {}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE Timeslot(  
    hour numeric(2,0) not null,  
    day enum(monday,tuesday,wednesday,thursday,friday,saturday,sunday),  
    primary key (hour,day)  
);
```

2.3. Seller

Relational Model: Seller(account_id, district_id, address_text, iban_no, validation_status)

Functional Dependencies: {(account_id -> district_id, address_text, iban_no, validation_status)}

Candidate Keys: {{account_id}}

Primary Key: {account_id}

Foreign Keys: {(account_id -> Account.account_id), (district_id -> District.district_id)}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE Seller(  
    account_id int,  
    district_id int,  
    address_text varchar(50),  
    iban_no char(26) not null,  
    validation_status boolean,  
    primary key (account_id),  
    foreign key (account_id) references Account,  
    foreign key (district_id) references District  
);
```

2.4. Customer

Relational Model: Customer(account_id, credit_card)

Functional Dependencies: {(account_id -> credit_card)}

Candidate Keys: {{account_id}}

Primary Key: {{account_id}}

Foreign Keys: {(account_id -> Account.account_id)}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE Customer(  
    account_id int,  
    credit_card char(16),  
    primary key (account_id),  
    foreign key (account_id) references Account  
);
```

2.5. Saved_addresses

Relational Model: Saved_addresses(customer_id, district_id, address)

Functional Dependencies: {(customer_id, district_id, address -> customer_id, district_id, address)}

Candidate Keys: {{customer_id, district_id, address}}

Primary Key: {{customer_id, district_id, address}}

Foreign Keys: {(customer_id -> Customer.customer_id), (district_id -> District.district_id)}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE Saved_addresses(  
    customer_id int,  
    district_id int,  
    address varchar(250),  
    primary key (customer_id, district_id, address),  
    foreign key (account_id) references Account,  
    foreign key (district_id) references District
```


);

2.6. Courier

Relational Model: Courier(account_id, max_volume, iban_no, validation_status)

Functional Dependencies: {(account_id -> max_volume, iban_no, validation_status)}

Candidate Keys: {{account_id}}

Primary Key: {{account_id}}

Foreign Keys: {(account_id -> Account.account.id)}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE Courier(  
    account_id int,  
    max_volume int not null,  
    iban_no char(26) not null,  
    validation_status boolean not null,  
    primary key (account_id),  
    foreign key (account_id) references Account  
);
```

2.7. Customer_service

Relational Model: Customer_service(account_id, count)

Functional Dependencies: {(account_id -> count)}

Candidate Keys: {{account_id}}

Primary Key: {{account_id}}

Foreign Keys: {(account_id -> Account.account.id)}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE Customer_service(  
    account_id int,  
    count int,  
    primary key (account_id),  
    foreign key (account_id) references Account  
);
```

2.8. Complaint

Relational Model: Complaint(order_id, complaint_id, complaint_date, complaint_status, customer_service_id, response_date)

Functional Dependencies: {(order_id, complaint_id -> complaint_date, complaint_status, customer_service_id, response_date)}

Candidate Keys: {{order_id, complaint_id}}

Primary Key: {order_id, complaint_id }

Foreign Keys: {order_id -> Order.order_id, customer_service_id -> Customer_service.customer_service_id}

Normal Form: BCNF

Table Definition: CREATE TABLE Complaint(

```

complaint_id int not null auto increment,
order_id int,
complaint_date date not null,
complaint_status enum ('Waiting', 'Replied', 'Solved') not null,
customer_service_id int,
response_date date not null,
primary key (order_id, complaint_id ),
foreign key (order_id) references Order,
foreign key (customer_service_id) references Customer_service,
);

```

2.9. Order

Relational Model: Order(order_id, payment, order_date, receiver_name, receiver_phone, district_id, address_text, delivery_date, delivery_type, delivery_status, desired_delivery_date, desired_delivery_time, message, seller_status, courier_status, seller_id, courier_id, customer_id, arrangement_id)

Functional Dependencies: {(order_id → payment, order_date, receiver_name, receiver_phone, district_id, address_text, delivery_date, delivery_type, delivery_status, desired_delivery_date, desired_delivery_time, message, seller_status, courier_status, seller_id, courier_id, customer_id, arrangement_id)}

Candidate Keys: {{order_id}}

Primary Key: {order_id}

Foreign Keys: {seller_id → Seller.account_id, courier_id → Courier.account_id, customer_id → Customer.account_id, district_id → District.district_id, arrangement_id → Flower_arrangement.arrangement_id}

Normal Form: BCNF

Table Definition: CREATE TABLE Order(
order_id int primary key auto_increment,
payment enum(cash, card, havale) not null,
order_date date not null,
receiver_name varchar(30) not null,
receiver_phone numeric(10,0) not null,
district_id int,
address_text varchar(50) not null,
delivery_date date,
delivery_type enum(hand, ring_the_bell, call_seller) not null,
delivery_status enum('Preparing', 'On Delivery', 'Delivered') not null,
desired_delivery_date date not null,
desired_delivery_time time not null,
message varchar(250),
seller_status enum('Not Assigned', 'Accepted', 'Rejected', 'Pending', 'Assigned to Courier'),
courier_status enum('Not Assigned', 'Accepted', 'Rejected', 'Pending'),
seller_id int,
courier_id int,
customer_id int,

```

arrangement_id int,
primary key (order_id),
foreign key (distric_id) references District,
foreign key (seller_id) references Seller,
foreign key (courier_id ) references Courier,
foreign key (customer_id ) references Customer,
foreign key (arrangement_id ) references Flower_arrangement
);

```

2.10. District

Relational Model: District(district_id, district_name, province_id)

Functional Dependencies: {(district_id -> district_name, province_id)}

Candidate Keys: {{district_id}}

Primary Key: {district_id}

Foreign Keys: {(province_id -> Province.province_id)}

Normal Form: BCNF

Table Definition: CREATE TABLE District(
 district_id int primary key auto_increment,
 district_name varchar(20),
 province_id int,
 primary key (district_id),
 foreign key (province_id) references Province
);

2.11. Province

Relational Model: Province(province_id, province_name)

Functional Dependencies: {(province_id -> province_name)}

Candidate Keys: {{province_id}}

Primary Key: {province_id}

Foreign Keys: {}

Normal Form: BCNF

Table Definition: CREATE TABLE Province(
 province_id int primary key auto_increment,
 province_name varchar(20),
 primary key (province_id)
);

2.12. Flower

Relational Model: Flower(flower_id, flower_name)

Functional Dependencies: {(flower_id -> flower_name)}

Candidate Keys: {{flower_id}}

Primary Key: {flower_id}

Foreign Keys: {}

Normal Form: BCNF

Table Definition: CREATE TABLE Flower(

```

        flower_id int primary key auto_increment,
        flower_name varchar(20),
        primary key (flower_id)
    );

```

2.13. Flower_arrangement

Relational Model: Flower_arrangement(arrangement_id, image_path, arrangement_name, volume, price, details, rate, count, seller_id, enabled)

Functional Dependencies: {(arrangement_id -> image_path, arrangement_name, volume, price, details, rate, count, seller_id, enabled)}

Candidate Keys: {{arrangement_id}}

Primary Key: {arrangement_id}

Foreign Keys: {seller_id -> Seller.account_id}

Normal Form: BCNF

Table Definition: CREATE TABLE Flower_arrangement(
 arrangement_id int primary key auto_increment,
 image_path longblob,
 arrangement_name varchar(20),
 volume int,
 price numeric (6, 2),
 details varchar(100),
 rate numeric (2,1),
 count int,
 seller_id int,
 enabled boolean not null,
 primary key (arrangement_id),
 foreign key (seller_id) references Seller
);

2.14. Occasion

Relational Model: Occasion(arrangement_id, occasion_name)

Functional Dependencies: {(arrangement_id, occasion_name -> arrangement_id, occasion_name)}

Candidate Keys: {{arrangement_id , occasion_name}}

Primary Key: {arrangement_id, occasion_name }

Foreign Keys: {(arrangement_id -> Flower_arrangement.arrangement_id)}

Normal Form: BCNF

Table Definition: CREATE TABLE Customer(
 arrangement_id int,
 occasion_name enum('Birthday', 'Anniversary', 'Congratulations', 'Just Because', 'I'm Sorry', 'Love & Romance', 'Thank You', 'Corporate Gifting'),
 primary key (arrangement_id, occasion_name),
 foreign key (arrangement_id) references Flower_arrangement,
);

2.15. Comment

Relational Model: Comment(description, customer_id, date, rating, arrangement_id)

Functional Dependencies: {(customer_id, date, arrangement_id -> description, rating)}

Candidate Keys: {{customer_id, date, arrangement_id}}

Primary Key: {(customer_id, date, arrangement_id)}

Foreign Keys: {(customer_id -> Customer.account_id), (arrangement_id -> Flower_arrangement.arrangement_id)}

Normal Form: BCNF

Table Definition: CREATE TABLE Comment(
description varchar(250),
account_id int,
date timestamp not null,
rating numeric(1,0),
arrangement_id int,
primary key (customer_id, date, arrangement_id),
foreign key (customer_id) references Customer,
foreign key (arrangement_id) references Flower_arrangement
);

2.16. Notification

Relational Model: Notification(notification_id, account_id, description, timestamp)

Functional Dependencies: {(notification_id -> account_id, description, timestamp)}

Candidate Keys: {{notification_id}}

Primary Key: {(notification_id)}

Foreign Keys: {(account_id -> Account.account_id)}

Normal Form: BCNF

Table Definition: CREATE TABLE Notification(
notification_id int primary key auto_increment not null,
account_id int,
description varchar(250) not null,
timestamp timestamp,
primary key (notification_id),
foreign key (account_id) references Account
);

2.17. Seller_serves_to

Relational Model: Seller_serves_to(district_id, seller_id)

Functional Dependencies: {(district_id, seller_id -> district_id, seller_id)}

Candidate Keys: {{district_id, seller_id}}

Primary Key: {(district_id, seller_id)}

Foreign Keys: {(district_id -> District.district_id, seller_id -> Seller.account_id)}

Normal Form: BCNF

Table Definition: CREATE TABLE Seller_serves_to(
district_id int,

```

        seller_id int,
        primary key (district_id, seller_id),
        foreign key (district_id ) references District,
        foreign key (seller_id) references Seller
    );

```

2.18. Courier_serves_to

Relational Model: Courier_serves_to(district_id, courier_id)

Functional Dependencies: {(district_id, courier_id -> district_id, courier_id)}

Candidate Keys: {{district_id, courier_id}}

Primary Key: {(district_id, courier_id)}

Foreign Keys: {(district_id -> District.district_id, courier_id -> Courier.account_id)}

Normal Form: BCNF

Table Definition: CREATE TABLE Courier_serves_to(
 district_id int,
 courier_id int,
 primary key (district_id, courier_id),
 foreign key (district_id) references District,
 foreign key (courier_id) references Courier
);

2.19. Flower_stock

Relational Model: Flower_stock(stock, flower_id, seller_id)

Functional Dependencies: {(flower_id, seller_id -> stock)}

Candidate Keys: {{flower_id, seller_id}}

Primary Key: {(flower_id, seller_id)}

Foreign Keys: {flower_id -> Flower.flower_id, seller_id -> Seller.account_id}

Normal Form: BCNF

Table Definition: CREATE TABLE Flower_stock(
 flower_id int,
 seller_id int,
 stock int,
 primary key (flower_id, seller_id),
 foreign key (flower_id) references Flower,
 foreign key (seller_id) references Seller
);

2.20. Composed_of

Relational Model: Composed_of(count, flower_id, arrangement_id)

Functional Dependencies: {(flower_id, arrangement_id -> count)}

Candidate Keys: {(flower_id, arrangement_id)}

Primary Key: {(flower_id, arrangement_id)}

Foreign Keys: {flower_id -> Flower.flower_id, arrangement_id -> Flower_arrangement.arrangement_id}

Normal Form: BCNF

Table Definition: CREATE TABLE Composed_of(
 flower_id int,
 arrangement_id int,
 count int,
 primary key (flower_id, arrangement_id),
 foreign key (flower_id) references Flower,
 foreign key (arrangement_id) references Flower_arrangement
);

2.21. Seller_working_time

Relational Model: Seller_working_time(seller_id, day, hour)

Functional Dependencies: {(seller_id, day, hour -> seller_id, day, hour)}

Candidate Keys: {{seller_id, day, hour}}

Primary Key: {(seller_id, day, hour)}

Foreign Keys: {seller_id -> Seller.account_id, day -> Timeslot.day, hour -> Timeslot.hour}

Normal Form: BCNF

Table Definition: CREATE TABLE Seller_working_time(
 seller_id int,
 hour numeric(2,0),
 day enum('monday', 'tuesday', 'wednesday', 'thursday', 'friday',
 'saturday', 'sunday'),
 primary key ((seller_id, day, hour)),
 foreign key (seller_id) references Seller,
 foreign key (day, hour) references Timeslot
);

2.22. Courier_working_time

Relational Model: Courier_working_time(courier_id, day, hour)

Functional Dependencies: {(courier_id, day, hour -> courier_id, day, hour)}

Candidate Keys: {{courier_id, day, hour}}

Primary Key: {(courier_id, day, hour)}

Foreign Keys: {courier_id -> Courier.account_id, day -> Timeslot.day, hour -> Timeslot.hour}

Normal Form: BCNF

Table Definition: CREATE TABLE Courier_working_time(
 courier_id int,
 hour numeric(2,0),
 day enum('monday', 'tuesday', 'wednesday', 'thursday', 'friday',
 'saturday', 'sunday'),
 primary key ((courier_id, day, hour)),
 foreign key (courier_id) references Courier,
 foreign key (day, hour) references Timeslot
);

2.23. Service_working_time

Relational Model: Service_working_time(customer_service_id, day, hour)

Functional Dependencies: {(customer_service_id, day, hour -> customer_service_id, day, hour)}

Candidate Keys: {(customer_service_id, day, hour)}

Primary Key: {(customer_service_id, day, hour)}

Foreign Keys: {customer_service_id -> Customer_service.account_id, day -> Timeslot.day, hour -> Timeslot.hour}

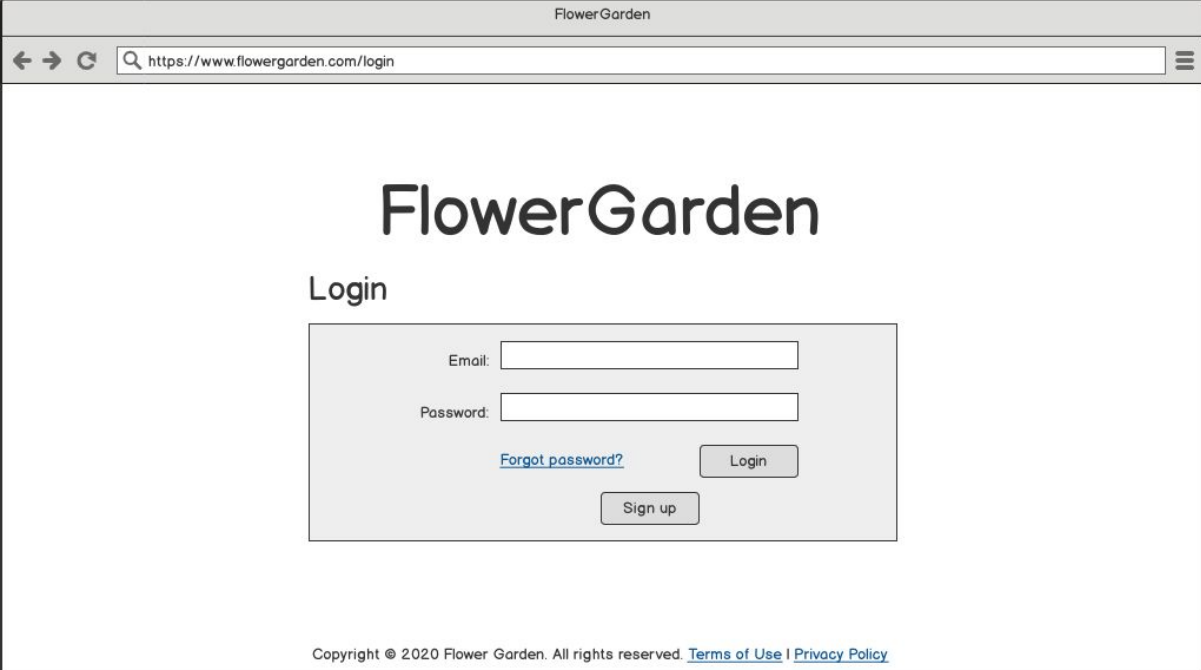
Normal Form: BCNF

Table Definition: CREATE TABLE Service_working_time(
customer_service_id int,
hour numeric(2,0),
day enum('monday', 'tuesday', 'wednesday', 'thursday', 'friday',
'saturday', 'sunday'),
primary key ((customer_service_id, day, hour)),
foreign key (customer_service_id) references Customer_service,
foreign key (day, hour) references Timeslot
);

3. User Interface Design and SQL Statements

3.1. Authorization Pages

3.1.1. Login Page



The screenshot shows a web browser window with the title "Flower Garden". The address bar displays "https://www.flowergarden.com/login". The main content area features the "FlowerGarden" logo at the top, followed by the word "Login". Below this is a light gray rectangular box containing the login form. The form has two input fields: "Email:" and "Password:". To the right of the "Password:" field is a "Login" button. Below the "Email:" field is a blue link labeled "Forgot password?". At the bottom of the form box is a "Sign up" button. At the very bottom of the page, there is a copyright notice: "Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)".

Figure 2: Login Page

Sign up button directs the user to the page shown in part 3.1.3.

SQL query for the login page:

```
SELECT email, password
FROM Account WHERE email= 'aaa@b.com' and password='12345';
```

3.1.2. Forgot Password Page

FlowerGarden

https://www.flowergarden.com/forgotpassword

FlowerGarden

Forgot Password

Email:

Get new password

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 3: Forgot Password Page

Users will be able to update their passwords from this page. We will update their password with a random password and send this as an email to the user.

SQL query for changing the password of the given email:

UPDATE User

SET password = 'sEm4?!*3Fd1' WHERE email = 'aaa@b.com';

3.1.3. Sign Up Page

3.1.3.1. Customer and Customer Service

FlowerGarden

Sign Up

* First Name: Middle Name:

* Last Name: * Phone:

* Email:

* Password: ?

* Re-type password:

Please select the desired account type:

☒ Customer

☐ Seller

☐ Courier

☐ Customer Service

☐ I agree to the [Terms of Use](#) and [Privacy Policy](#).

[Learn more](#)

* Required

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 4: Sign Up Page for Customer and Customer Service Accounts

According to the given information, the customer account will be created. Since our account id is incremented automatically we do not specify any account id while inserting it into the 'Account' table. Then we will take the id from the account table with the `LAST_INSERT_ID()` function and use it when adding it to the 'Customer' page.

SQL query for registering an account as Customer:

```
INSERT INTO Account (first_name, middle_name, last_name, email, password, phone, last_login_data)
VALUES ('First', 'Middle', 'Last', 'aaa@b.com', 'sEm4?!*3Fd1', '05426789586', '2008-01-01 00:00:01');
```

SQL query for taking the account id which is created by the database itself:

```
SELECT LAST_INSERT_ID();
```

```
INSERT INTO Customer (account_id, credit_card)
VALUES (1, NULL);
```

3.1.3.2. Courier

The screenshot shows a web browser window with the address bar displaying 'https://www.flowergarden.com/signup'. The page title is 'FlowerGarden'. The main heading is 'Sign Up'. The form contains the following fields and options:

- * First Name: [text input]
- * Last Name: [text input]
- * Email: [text input]
- * Password: [text input] with a help icon (?)
- * Re-type password: [text input]
- Middle Name: [text input]
- * Phone: [text input]
- * IBAN: [text input]
- * Maximum volume you can carry: [dropdown menu showing 125]
- Please select the desired account type:
 - ☐ Customer
 - ☐ Seller
 - ☒ Courier
 - ☐ Customer Service
- ☐ I agree to the [Terms of Use](#) and [Privacy Policy](#).
- [Learn more](#)
- * Required

At the bottom, there is a copyright notice: 'Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)'.

Figure 5: Sign Up Page for Courier Account

According to the given information, the courier account will be created. Then we will take the `account_id` from the `account` table with the `LAST_INSERT_ID()` function and use it when adding it to the 'Courier' page. The information with the star sign is required information.

SQL query for registering an account as Courier:

```
INSERT INTO Account (first_name, middle_name, last_name, email, password, phone, last_login_data)
VALUES ( 'First', 'Middle', Last, 'aaa@b.com', 'sEm4?!*3Fd1', '05426789586', '2008-01-01 00:00:01');
```

SQL query for taking the account id which is created by the database itself:

```
SELECT LAST_INSERT_ID();
```

```
INSERT INTO Courier (account_id, max_volume, iban_no, validation_status)
VALUES (2, 125, 'TR123412341234123412341234', false);
```

3.1.3.3. Seller

The screenshot shows a web browser window with the URL <https://www.flowergarden.com/signup>. The page title is "FlowerGarden" and the main heading is "Sign Up". The form is divided into two columns. The left column contains fields for: * First Name, * Last Name, * Email, * Password (with a help icon), * Re-type password, and a checkbox for "I agree to the Terms of Use and Privacy Policy". The right column contains fields for: Middle Name, * Phone, * IBAN, * Address, and dropdown menus for District and Province. Below the address field, there is a section titled "Please select the desired account type:" with radio buttons for Customer, Seller (selected), Courier, and Customer Service. A "Sign up" button and a "Learn more" link are located at the bottom left of the form. A footer at the bottom of the page reads: "Copyright © 2020 Flower Garden. All rights reserved. Terms of Use | Privacy Policy".

Figure 6: Sign Up Page for Seller Account

According to the given information, the seller account will be created. Then we will take the `account_id` from the account table with the `LAST_INSERT_ID()` function and use it when adding it to the 'Seller' page. Seller will enter his/her flower shop's address while registering. For that the province information will be selected first from the list created with the result of the first query. After that, based on the selected province corresponding districts will be listed to the seller for the completion of address specification.

SQL query for showing province:

```
SELECT * FROM Province;
```

After the user selects the province we will show the districts according to that province.

SQL query for showing districts:

```
SELECT district_id, district_name  
FROM District WHERE = province_id = 34;
```

SQL query for registering an account as Seller:

```
INSERT INTO Account (first_name, middle_name, last_name, email, password, phone,  
last_login_data)  
VALUES ('First', 'Middle', 'Last', 'aaa@b.com', 'sEm4?!*3Fd1', '05426789586', '2008-01-01  
00:00:01');
```

SQL query for taking the account id which is created by the database itself:

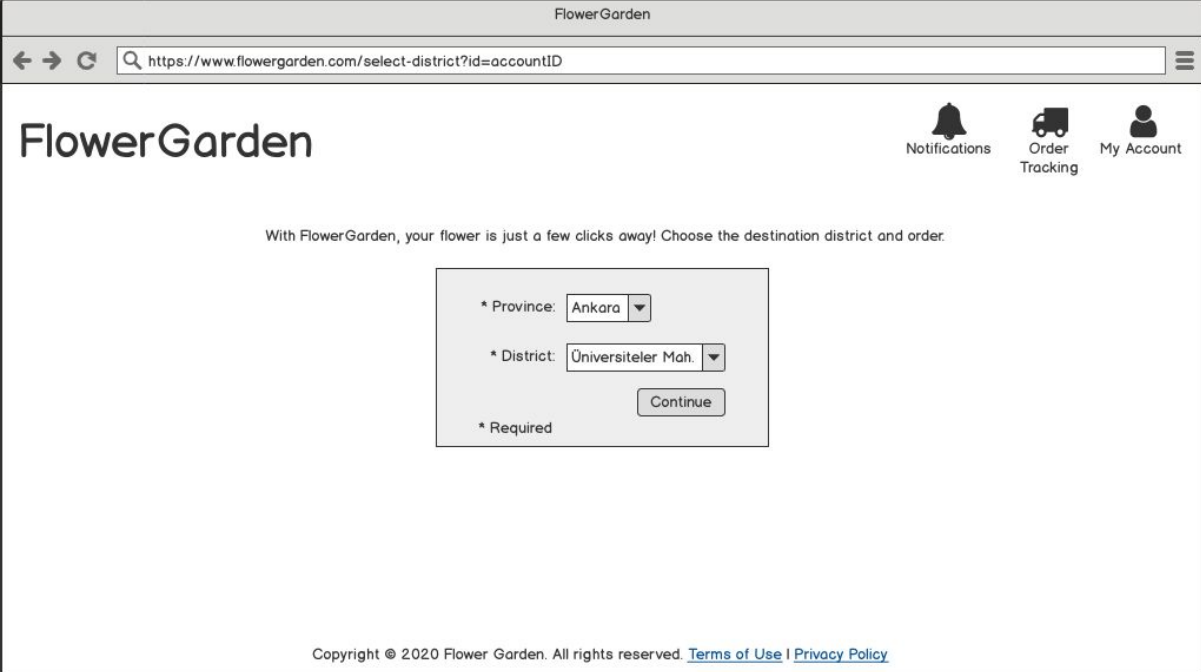
```
SELECT LAST_INSERT_ID();
```

SQL query for adding seller to the seller table:

```
INSERT INTO Seller (account_id, district_id, address_text, iban_no, validation_status)
VALUES (3, 06, NULL , 'TR12123412341234', false) ;
```

3.2. Customer Pages

3.2.1. Select District Page



The screenshot shows a web browser window with the URL `https://www.flowergarden.com/select-district?id=accountID`. The page header includes the "FlowerGarden" logo and navigation links for "Notifications", "Order Tracking", and "My Account". The main content area features a message: "With FlowerGarden, your flower is just a few clicks away! Choose the destination district and order." Below this is a form with two dropdown menus: "* Province:" with "Ankara" selected and "* District:" with "Üniversiteler Mah." selected. A "Continue" button is positioned to the right of the district dropdown. A "* Required" label is at the bottom left of the form. The footer contains the copyright notice "Copyright © 2020 Flower Garden. All rights reserved." and links to "Terms of Use" and "Privacy Policy".

Figure 7: Select District Page for Customer Account

According to the given information, the flower arrangements from the sellers who serve that province and district will be selected and shown to the user on the home page.

Firstly, the user will be able to select the province. This is for showing all the provinces in the database.

SQL query for showing provinces:

```
SELECT * FROM Province;
```

After the user selects the province we will show the districts according to that province.

SQL query for showing districts according to selected province:

```
SELECT district_id, district_name
FROM District WHERE province_id = 34;
```

3.2.2. Home Page

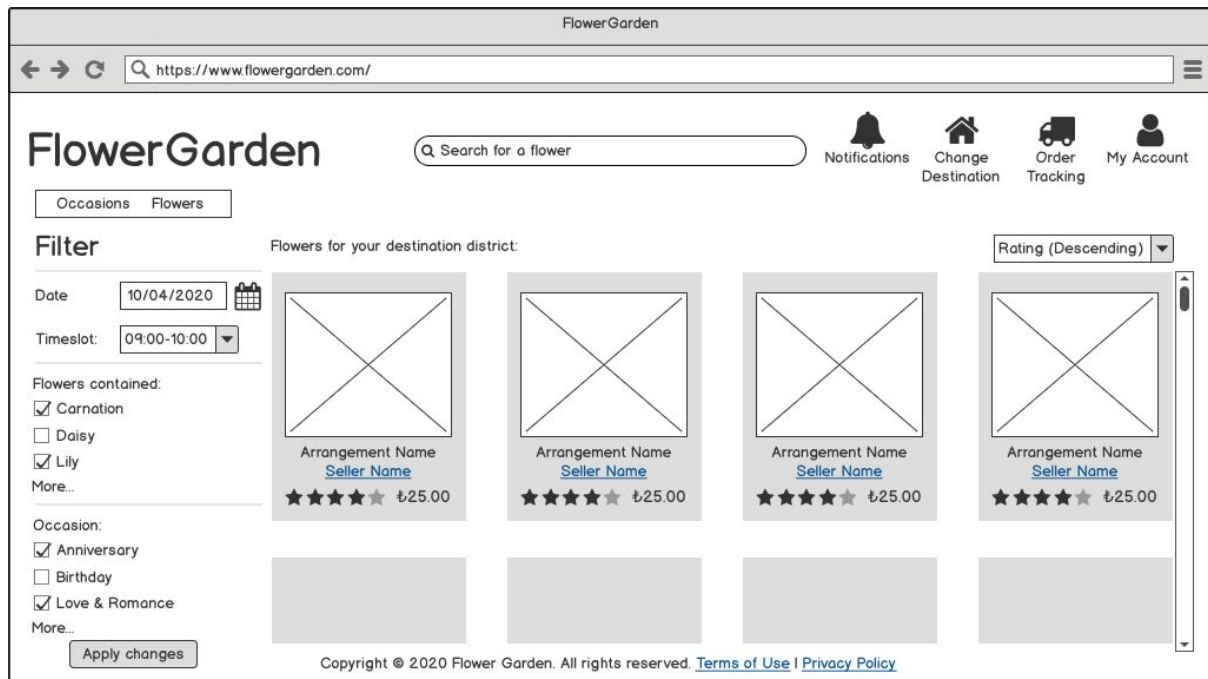


Figure 8: Home Page for Customer Account

All of the selected flower arrangements will be displayed on this page. From the left panel, the user can filter the flower arrangements according to their needs. The flowers will be sent on the date and time slot that is entered. Users will only be able to order their flowers at least 3 hours before. Also sorting can be done from the right top.

SQL query for showing the flower arrangements to customers:

```
SELECT arrangement_id, image_path, arrangement_name, volume, price, seller_id, details,
rate, count
FROM Flower_arrangement
WHERE seller_id IN ( SELECT seller_id
                     FROM Seller_serves_to
                     WHERE district_id = 34; )
```

SQL query for showing the flower arrangements according to occasion, flowers contained and date filtering:

```
SELECT TD.arrangement_id, TD.image_path, O. arrangement_name, TD.price,
A.first_name, A.last_name, TD.details, TD.rate, TD.count
FROM to_display as TD, Occasion as O, Seller_working_time as SWT, Composed_of C,
Account as A
WHERE TD.arrangement_id = O.arrangement_id AND O.occasion_name = 'Anniversary'
AND SWT.day='monday' AND SWT.hour=2 AND C.count=2 AND C.flower_id=1 AND
A.account_id = TD.seller_id;
```

3.2.3. Flower Arrangement Page

The screenshot shows the 'FlowerGarden' website interface. At the top, there's a navigation bar with a search bar labeled 'Search for a flower' and links for 'Notifications', 'Change Destination', 'Order Tracking', and 'My Account'. Below the navigation bar, the main content area is divided into sections. On the left, there's a 'Comments' section with a placeholder for a comment. The central section displays the 'Arrangement Name' as 'Arrangement Name' with a price of '₹25.00'. It includes a 'Time' section with a date picker set to '10/04/2020' and a time slot of '09:00-10:00'. Below this is a blue 'Order' button. To the right, the 'Arrangement Description' section shows 'Contained Flowers: Flower, Flower' and 'Volume: 25'. Below the description, there's a 'Seller' section with 'Seller Name: Name' and an 'Average Rating' of five stars. At the bottom of the page, there's a copyright notice: 'Copyright © 2020 Flower Garden. All rights reserved. Terms of Use | Privacy Policy'.

Figure 9: Flower Arrangement Page for Customer Account

Only one of the flower arrangements that is selected is displayed. The arrangement id will be taken when it is selected from the list in the part shown 3.2.2. The corresponding comments for this flower arrangement will be shown.

SQL query for getting all the information about flower arrangement:

```
SELECT *  
FROM Flower_arrangement WHERE arrangement_id = 81;
```

From the above query, we will have all the information needed to get the seller name, rating, etc.

SQL query for displaying comments:

```
SELECT description, account_id, date, rating, arrangement_id  
FROM Comment WHERE arrangement_id = 123;
```

SQL query for displaying the seller of the flower arrangement:

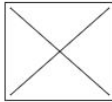
```
SELECT A.first_name, A.middle_name, A.last_name  
FROM Account as A WHERE A.account_id = 55;
```

SQL query for displaying seller's average rating:

```
SELECT AVG(F.rate)  
FROM Seller as S, Flower_arrangement as F  
WHERE S.seller_id = F.seller_id AND S.seller_id = 55;
```


3.2.4. Order Creation Page

The screenshot shows the 'Order Creation' page for a customer account. The browser address bar displays 'https://www.flowergarden.com/order-creation?id=arrangementID'. The page header includes the 'FlowerGarden' logo and navigation links for Notifications, Change Destination, Order Tracking, and My Account. The main content area is titled 'Your Order' and features a table with the following details:

Flower Arrangement	Price
 <div>Arrangement Name Date: 10/04/2020 Timeslot: 09:00-10:00</div>	₺25.00

To the right, a 'Summary' box displays the following information:

Subtotal	₺25.00
Delivery	-
Tax	-
Total	₺25.00

A blue button labeled 'Go to Receiver Information' is located below the summary. The footer contains the copyright notice: 'Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)'.

Figure 10: Order Creation Page for Customer Account

After selecting all of the flower arrangements, the user can check all of the items selected on this window.

3.2.5. Receiver Information Page

The screenshot shows the 'Receiver Information' page for a customer account. The browser address bar displays 'https://www.flowergarden.com/receiver-information?id=orderID'. The page header includes the 'FlowerGarden' logo and navigation links for Notifications, Change Destination, Order Tracking, and My Account. The main content area is titled 'Order > Receiver Information' and features a 'Customer Information' section with the following form fields:

- Receiver Name and Surname
- Receiver Phone
- Address
- District: District Province: District Choose from saved addresses
- Delivery Type
- Message

To the right, a 'Summary' box displays the following information:

Subtotal	₺25.00
Delivery	-
Tax	-
Total	₺25.00

A blue button labeled 'Go to Payment' is located below the summary. A link '< Return to Order' is located at the bottom left. The footer contains the copyright notice: 'Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)'.

Figure 11: Receiver Information Page of Order for Customer Account

The receiver information will be given through this page.

3.2.6. Payment Page

The screenshot shows a web browser window with the URL `https://www.flowergarden.com/payment-method?id=orderID`. The page title is "FlowerGarden". The breadcrumb trail is "Order > Receiver Information > Payment Method".

Receiver Address: Üniversiteler Mah. Bilkent Üniversitesi Çankaya/Ankara

Payment Method: Credit Card (selected)

Card Details:

- Card Number: 12 **** *34
- Name on card: [input field]
- MM/YY: [input field]
- CVV: [input field]

☒ Save credit card number

Remember Receiver:

☐ Save the receiver address for a faster checkout

[< Return to Receiver Information](#)

Summary:

Subtotal	₺25.00
Delivery	-
Tax	-
Total	₺25.00

[Complete Order](#)

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 12: Payment Page of Order for Customer Account

According to given information on sections 3.2.3, 3.2.5 and 3.2.6, an order will be created in the order table and if the save options are selected they will be stored in the database as well.

SQL query for showing saved credit card and address information:

```
SELECT *  
FROM Customer WHERE account_id = 3;
```

```
SELECT *  
FROM Saved_addresses WHERE customer_id = 3;
```

SQL query for customer credit card information update:

```
UPDATE Customer  
SET credit_card= '1234123412341234' WHERE customer_id=55;
```

SQL query for inserting new address:

```
INSERT INTO Saved_addresses  
VALUES address= 'Address' , district_id=12, customer_id=55;
```

SQL query for creating order:

```
INSERT INTO Order (payment, order_date, receiver_name, receiver_phone, district_id,  
address_text, delivery_date, delivery_type, delivery_status, desired_delivery_date,  
desired_delivery_time, message, seller_status, courier_status, seller_id, courier_id,  
customer_id, arrangement_id)
```

```
VALUES ('cash', '2008-01-01 00:00:01', 'r_name', '5061231233', 11, 'address', '2008-01-01',
'2008-11-11', 'hand', 'Preparing', '2008-05-01', "09:34:21.000001", 'message', 'Pending', 'Not
Assigned', 12, NULL, 55, 12);
```

SQL query for getting the order id when recently added:

```
SELECT LAST_INSERT_ID();
```

SQL query for checking stock:

```
SELECT count, flower_id, arrangement_id
FROM Composed_of WHERE arrangement_id = 5;
```

```
SELECT flower_id, stock
FROM Flower_stock WHERE seller_id = 12 AND flower_id = 1;
```

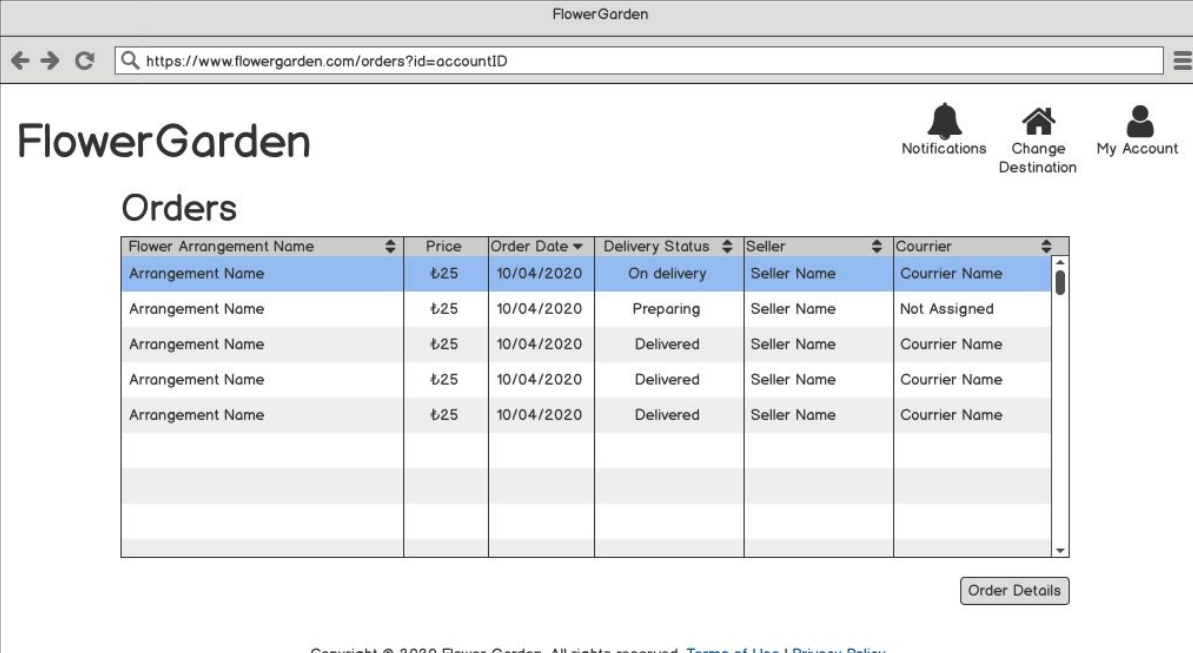
SQL query for updating stock of the seller if s/he has enough stock:

```
UPDATE Flower_stock(stock, flower_id, seller_id)
SET stock = stock - 4 WHERE seller_id = 12 AND flower_id = 1;
```

SQL query for creating a notification for the seller when order created (seller_id is given as a parameter):

```
INSERT INTO Notification(account_id, description, timestamp)
VALUES (12,'description','2008-01-01 00:00:01');
```

3.2.7. Order Tracking Page



FlowerGarden

Notifications Change Destination My Account

Orders

Flower Arrangement Name	Price	Order Date	Delivery Status	Seller	Courier
Arrangement Name	₺25	10/04/2020	On delivery	Seller Name	Courier Name
Arrangement Name	₺25	10/04/2020	Preparing	Seller Name	Not Assigned
Arrangement Name	₺25	10/04/2020	Delivered	Seller Name	Courier Name
Arrangement Name	₺25	10/04/2020	Delivered	Seller Name	Courier Name
Arrangement Name	₺25	10/04/2020	Delivered	Seller Name	Courier Name

Order Details

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 13: Order Tracking Page for Customer Account

If a user selects one order and clicks the order details button, it will direct the user to the page shown in section 3.2.8. The delivery status can be accepted, rejected, preparing or on delivery.

SQL query for showing orders for the corresponding customer:

```
SELECT order_id, payment, order_date, receiver_name, receiver_phone, district_id,
address_text, delivery_date, delivery_type, delivery_status, desired_delivery_date,
desired_delivery_time, message, seller_status, courier_status, seller_id, courier_id,
customer_id, F.arrangement_name, F.price
FROM Order natural join Flower_arrangement as F
WHERE customer_id=55 ;
```

Corresponding seller's id will be taken from the above query.

SQL query for getting seller name:

```
SELECT first_name, middle_name, last_name
FROM Account WHERE account_id = 123;
```

SQL query for getting courier name:

```
SELECT first_name, middle_name, last_name
FROM Account WHERE account_id = 43;
```

3.2.8. Order Page

The screenshot shows a web browser window with the URL `https://www.flowergarden.com/order?orderID=orderID&accountID=accountID`. The page title is "FlowerGarden". The main content area is titled "Your Order". It displays a placeholder for a flower arrangement image, the arrangement name, date (10/04/2020), timeslot (09:00-10:00), and price (₺25.00). Below this, it shows receiver information: Name (Receiver Name), Phone (+123456789), Address (redacted), District (Üniversiteler Mah.), and Province (Ankara). It also shows Delivery Type (Type) and a Message (redacted). On the right, a "Summary" table shows Subtotal (₺25.00), Delivery (-), Tax (-), Total (₺25.00), and Card Number (12** ***** 34). Below the summary, it shows Order Status (On Delivery), Seller (Seller Name), and Courier (Courier Name). At the bottom, there are "Comment" and "Complaint" buttons. The footer contains copyright information and links to Terms of Use and Privacy Policy.

Figure 14: Order Page for Customer Account

SQL query for showing order for the corresponding customer:

```
SELECT order_id, payment, order_date, receiver_name, receiver_phone, district_id,
address_text, delivery_date, delivery_type, delivery_status, desired_delivery_date,
```

```
desired_delivery_time, message, seller_status, courier_status, seller_id, courier_id,
customer_id, F.arrangement_name, F.price
FROM Order natural join Flower_arrangement as F
WHERE customer_id=55 AND order_id = 1;
```

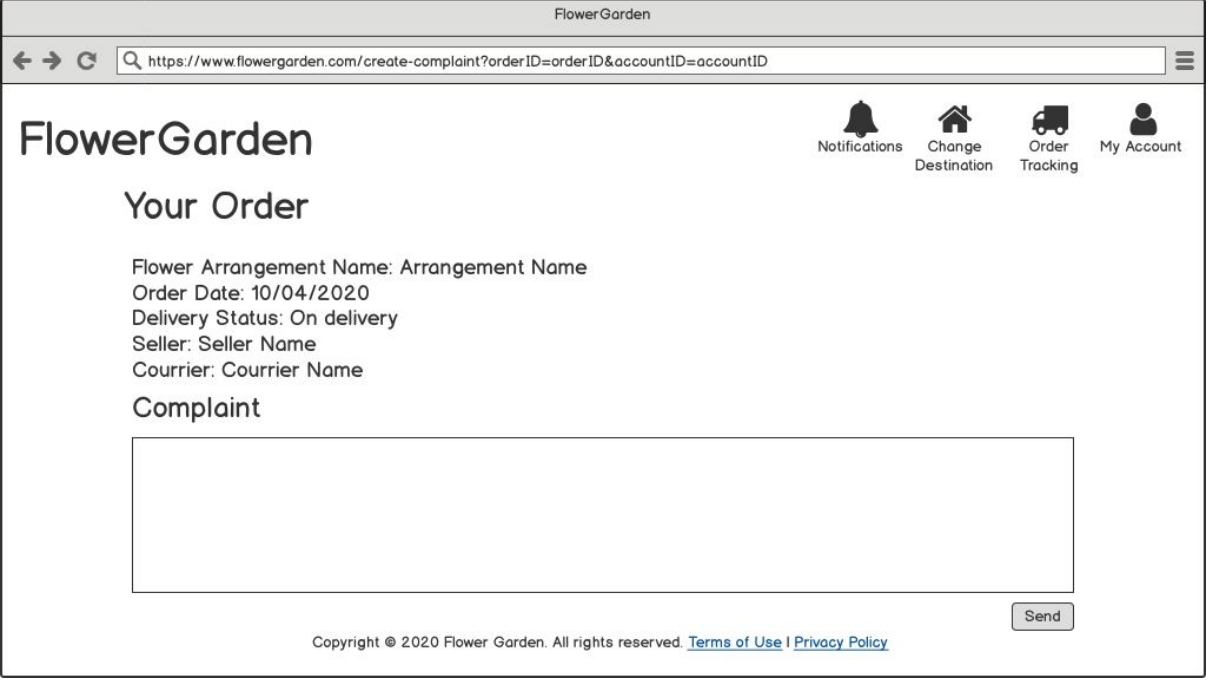
SQL query for getting seller name:

```
SELECT first_name, middle_name, last_name
FROM Account WHERE account_id = 123;
```

SQL query for getting courier name:

```
SELECT first_name, middle_name, last_name
FROM Account WHERE account_id = 43;
```

3.2.9. Create Complaint Page



The screenshot shows a web browser window with the URL `https://www.flowergarden.com/create-complaint?orderId=orderId&accountID=accountID`. The page header includes the 'FlowerGarden' logo and navigation links: Notifications, Change Destination, Order Tracking, and My Account. The main content area is titled 'Your Order' and displays the following details:

- Flower Arrangement Name: Arrangement Name
- Order Date: 10/04/2020
- Delivery Status: On delivery
- Seller: Seller Name
- Courrier: Courrier Name

Below the order details is a section titled 'Complaint' with a large text input area. At the bottom right of the input area is a 'Send' button. The footer contains the copyright notice: 'Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)'.

Figure 15: Create Complaint Page for Customer Account

SQL query for showing orders for the corresponding customer:

```
SELECT order_id, order_date, delivery_date, message, seller_id, courier_id, customer_id,
F.arrangement_name, F.price
FROM Order natural join Flower_arrangement as F
WHERE customer_id=55 AND order_id = 1;
```

SQL query for getting seller name:

```
SELECT first_name, middle_name, last_name
FROM Account WHERE account_id = 123;
```

SQL query for getting courier name:

```
SELECT first_name, middle_name, last_name
```

```
FROM Account WHERE account_id = 43;
```

SQL query for creating complaints:

```
INSERT INTO Complaint (order_id, complaint_id, complaint_date, customer_service_id,  
response_date, complaint_status)  
VALUES ( 123234, 789, '2008-01-01 00:00:01', 1111, NULL, false);
```

3.2.10. Create Comment Page

The screenshot displays a web browser window titled 'FlowerGarden'. The address bar shows the URL: `https://www.flowergarden.com/comment?orderID=orderID&accountID=accountID`. The page header includes the 'FlowerGarden' logo and navigation links: 'Notifications', 'Change Destination', 'Order Tracking', and 'My Account'. The main content area is titled 'Your Order' and displays the following details: 'Flower Arrangement Name: Arrangement Name', 'Order Date: 10/04/2020', 'Delivery Status: On delivery', and 'Seller: Seller Name'. Below this is a 'Comment' section with a large text input field. Under the input field is a 'Rating' section with five stars, where the first four are filled. A 'Send' button is located to the right of the rating stars. At the bottom of the page, there is a copyright notice: 'Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)'.

Figure 16: Create Comment Page for Customer Account

SQL query for showing orders for the corresponding customer:

```
SELECT order_id, order_date, delivery_date, message, seller_id, courier_id, customer_id,  
F.arrangement_name, F.price, F.rate  
FROM Order natural join Flower_arrangement as F  
WHERE customer_id=55 AND order_id = 1;
```

SQL query for getting seller name:

```
SELECT first_name, middle_name, last_name  
FROM Account WHERE account_id = 123;
```

SQL query for creating comment:

```
INSERT INTO Comment(description, account_id, date,rating, arrangement_id)  
VALUES ( "Actual Comment", 55, '2008-01-01 00:00:01', 4, 17);
```

SQL query for updating the flower arrangement rating according to comment:

```
UPDATE Flower_arrangement  
SET rate = 4, count=count+1  
WHERE arrangement_id = 12 and seller_id;
```

3.3. Seller Pages

3.3.1. Select Districts and Working Hours Page

FlowerGarden

Choose the districts you will serve

Provinces:

Districts:

Working Hours:

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
<input type="text" value="07:00 - 08:00"/>	<input type="text" value="07:00 - 08:00"/>	<input type="text" value="07:00 - 08:00"/>	<input type="text" value="07:00 - 08:00"/>	<input type="text" value="07:00 - 08:00"/>	<input type="text" value="07:00 - 08:00"/>	<input type="text" value="07:00 - 08:00"/>
<input type="text" value="08:00 - 09:00"/>	<input type="text" value="08:00 - 09:00"/>	<input type="text" value="08:00 - 09:00"/>	<input type="text" value="08:00 - 09:00"/>	<input type="text" value="08:00 - 09:00"/>	<input type="text" value="08:00 - 09:00"/>	<input type="text" value="08:00 - 09:00"/>
<input type="text" value="09:00 - 10:00"/>	<input type="text" value="09:00 - 10:00"/>	<input type="text" value="09:00 - 10:00"/>	<input type="text" value="09:00 - 10:00"/>	<input type="text" value="09:00 - 10:00"/>	<input type="text" value="09:00 - 10:00"/>	<input type="text" value="09:00 - 10:00"/>
<input type="text" value="10:00 - 11:00"/>	<input type="text" value="10:00 - 11:00"/>	<input type="text" value="10:00 - 11:00"/>	<input type="text" value="10:00 - 11:00"/>	<input type="text" value="10:00 - 11:00"/>	<input type="text" value="10:00 - 11:00"/>	<input type="text" value="10:00 - 11:00"/>
<input type="text" value="11:00 - 12:00"/>	<input type="text" value="11:00 - 12:00"/>	<input type="text" value="11:00 - 12:00"/>	<input type="text" value="11:00 - 12:00"/>	<input type="text" value="11:00 - 12:00"/>	<input type="text" value="11:00 - 12:00"/>	<input type="text" value="11:00 - 12:00"/>
<input type="text" value="12:00 - 13:00"/>	<input type="text" value="12:00 - 13:00"/>	<input type="text" value="12:00 - 13:00"/>	<input type="text" value="12:00 - 13:00"/>	<input type="text" value="12:00 - 13:00"/>	<input type="text" value="12:00 - 13:00"/>	<input type="text" value="12:00 - 13:00"/>

Save

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 17: Select Districts and Working Hours Page for Seller Account

After creating the account seller directed to this page to select the province, districts they are serving and serving hours for each day. For each selected row these SQL queries will be called.

```
INSERT INTO Seller_serves_to (district_id, seller_id)
VALUES ( 81, 55) ;
```

```
INSERT INTO Seller_working_time (seller_id, day, hour)
VALUES ( 55, 'monday', 8) ;
```

Firstly the user will be able to select the province. This is for showing all the provinces in the database.

SQL query for showing provinces:

```
SELECT *
FROM Province;
```

After the user selects the province we will show the districts according to that province.

SQL query for showing districts according to selected province:

```
SELECT district_id, district_name
FROM District
WHERE province_id = 34;
```

3.3.2. Arrangement List Page

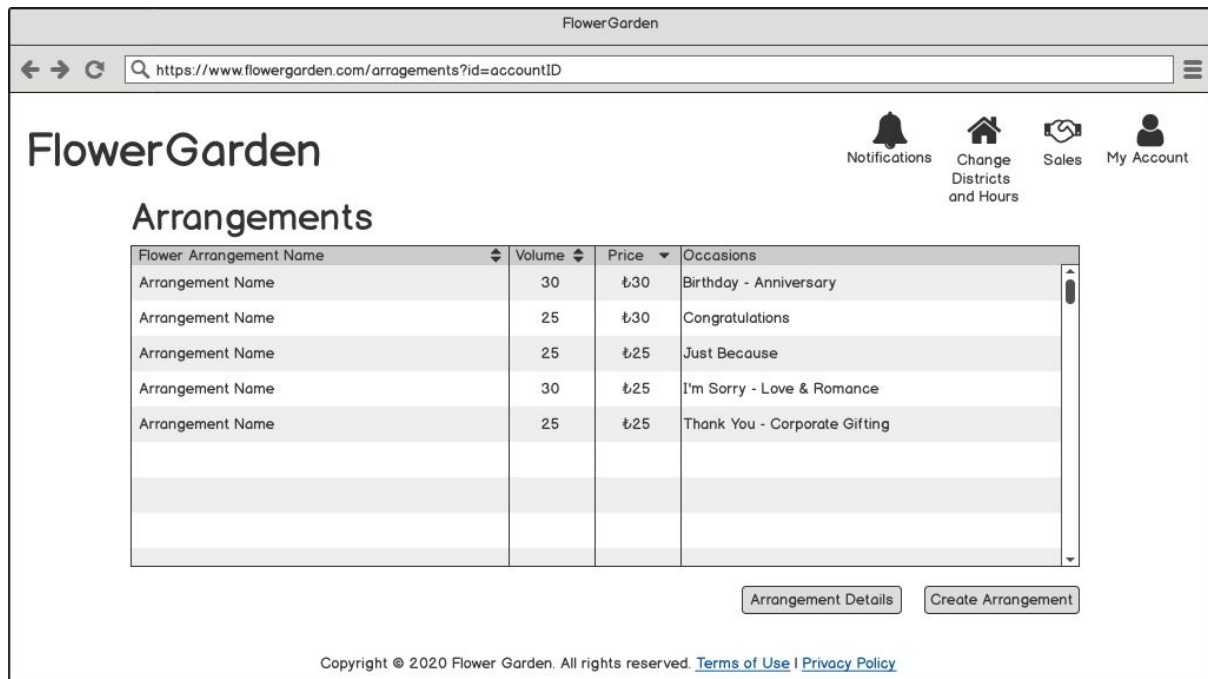


Figure 18: Arrangement List Page for Seller Account

On this page, sellers will be able to see their flower arrangements that are in the market. By clicking the arrangement details button users can see the details of each arrangement and by clicking the create arrangement button the seller will be able to create an arrangement and be directed to the page shown in part 3.3.4.

SQL query for displaying seller's arrangements:

```
SELECT arrangement_id, arrangement_name, volume, price, occasion_name
FROM Flower_arrangement natural join Occasion
WHERE seller_id = 55;
```


3.3.3. Arrangement Page

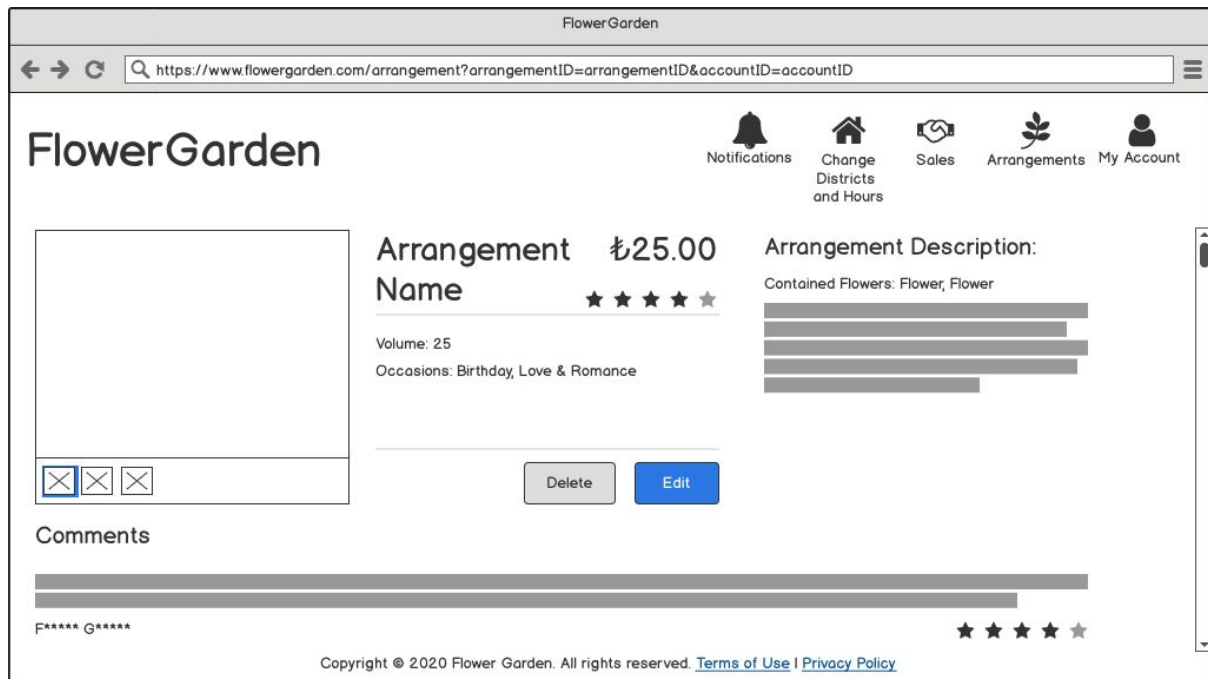


Figure 19: Arrangement Page for Seller Account

When the user selects an arrangement in section 3.3.2 we will use its arrangement id to get more information about that arrangement.

SQL query for displaying all of the information about selected arrangement:

```
SELECT *  
FROM Flower_arrangement natural join Occasion  
WHERE seller_id = 55 AND arrangement_id= 6;
```

SQL query for displaying comments:

```
SELECT (description, account_id, date, rating, arrangement_id)  
FROM Comment  
WHERE arrangement_id = 6;
```

For the flower arrangements we will be saving a variable enabled which shows us that the seller is still selling the corresponding arrangement or not. With this variable even though the arrangement is deleted from the seller, customers who purchased the item before, will be able to see the information about the arrangement.

SQL query for deleting flower arrangement:

```
UPDATE Flower_arrangement(arrangement_id, image_path, arrangement_name, volume,  
price, details, rate, count, seller_id, enabled)  
SET enabled= false  
WHERE seller_id = 12 AND arrangement_id = 6;
```

3.3.4. Create Arrangement Page

FlowerGarden

Notifications Change Districts and Hours Sales Arrangements My Account

* Flower Arrangement Name:

* Price:

* Volume:

* Description:

Upload an image:

Flowers:

- Flower 1
- Flower 2
- Flower 3
- Flower 4

Flower 1:

Flower 4:

Occasions:

- Birthday
- Congratulations
- Thank You
- Love & Romance

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 20: Create Arrangement Page for Seller Account

We will use this window also when the user selects the 'EDIT' button on page shown in part 3.3.3. For that case we will use update sql queries.

SQL query for updating flower arrangement:

```
UPDATE Flower_arrangement (arrangement_id, image_path, arrangement_name, volume, price, seller_id, details, rate, count)
SET arrangement_name = 'FlowerName'
WHEN arrangement_id = 7 AND seller_id = 55;
```

SQL query for updating flower counts for each arrangement:

```
UPDATE Composed_of (count, flower_id, arrangement_id)
SET count = 3
WHEN arrangement_id = 7 AND flower_id=3;
```

SQL query for updating occasions table:

```
UPDATE Occasion(arrangement_id, occasion_name)
SET occasion_name= 3
WHEN arrangement_id=7;
```

SQL query for adding flower arrangement:

```
INSERT INTO Flower_arrangement (arrangement_id, image_path, arrangement_name, volume, price, seller_id, details, rate, count)
VALUES ( 7, 'C:/Images/flower.jpg'.$arrangement_id, 'Arrangement Name', 40, 30, 55, 'Details', 4, 25);
```

For each selected row these SQL queries will be called.

SQL query for adding flower counts for each arrangement:

```
INSERT INTO Composed_of (count, flower_id, arrangement_id)
VALUES (3,1,7);
```

SQL query for adding occasions table:

```
INSERT INTO Occasion(arrangement_id, occasion_name)
VALUES (7, 'Anniversary');
```

3.3.5. Sale List Page

ID	Flower Arrangement Name	Sale Date	Delivery Date	Delivery Time	Courier Name	Acceptance Status	Message
ID1	Arrangement Name	10/04/2020	10/04/2020	09:00 - 10:00	Courier Name	Pending	Yes
ID2	Arrangement Name	10/04/2020	10/04/2020	09:00 - 10:00	Courier Name	Rejected	No
ID3	Arrangement Name	10/04/2020	10/04/2020	09:00 - 10:00	Courier Name	Accepted	Yes
ID4	Arrangement Name	10/04/2020	10/04/2020	09:00 - 10:00	Courier Name	Accepted	Yes
ID5	Arrangement Name	10/04/2020	10/04/2020	09:00 - 10:00	-	Not Assigned	Yes

Figure 21: Sale List Page for Seller Account

Sellers will be able to see their sales through this screen.

SQL query for showing orders of seller:

```
SELECT O.order_id, F.arrangement_name, O.order_date, O.delivery_status,
O.desired_delivery_date, O.desired_delivery_time, O.message, O.courier_id
FROM Order as O natural join Flower_arrangement as F;
WHERE O.seller_id = 55;
```

SQL query for getting courier name:

```
SELECT first_name, middle_name, last_name
FROM Account
WHERE account_id = 43;
```

3.3.6. Sale Page

The screenshot shows a web browser window with the URL `https://www.flowergarden.com/sale?id=orderID`. The page header includes the 'FlowerGarden' logo and navigation links: Notifications, Change Districts and Hours, Sales, Arrangements, and My Account. The main content area displays the following order details:

- Sale ID: ID
- Flower Arrangement Name: Arrangement Name
- Customer Name: Name
- Customer Phone: +123456789
- Sale Date: 10/04/2020
- Delivery Date: 10/04/2020
- Timeslot: 09:00-10:00
- Message: [Redacted]

Below the details, the order status is 'On Delivery'. Further information includes:

- Order Status: On Delivery
- Courier Name: Name
- Courier Phone: +123456789
- Acceptance Status: Accepted

At the bottom, there are four buttons: 'Delivered to Courier', 'Reject Sale', 'Accept Sale', and 'Assign to a Courier'. The footer contains the copyright notice: 'Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)'.

Figure 22: Sale Page for Seller Account

This page shows every information about an order that has been selected. Sellers have a chance to reject and accept the orders. After accepting the order sellers will be able to assign a courier to that order. And after giving the orders to the courier seller will be able to click the 'delivered to courier' button.

SQL query for showing details of orders:

```
SELECT O.order_id, F.arrangement_name, O.order_date, O.receiver_phone,
O.address_text, O.delivery_type, O.delivery_status, O.desired_delivery_date,
O.desired_delivery_time, O.message, O.courier_status, O.seller_id, O.courier_id,
O.arrangement_id
FROM Order as O natural join Flower_arrangement as F
WHERE O.seller_id = 55;
```

```
SELECT first_name, middle_name, last_name, phone
FROM account_id = 55 ;
```

SQL query for getting courier name:

```
SELECT first_name, middle_name, last_name, phone
FROM Account
WHERE account_id = 43;
```

SQL query for accepting and rejecting the order & changing the delivery status:

```
UPDATE Order
SET seller_status = 'Accepted'
```

```
WHERE order_id=1234;
```

```
UPDATE Order  
SET courier_status = 'Pending'  
WHERE order_id = 1234;
```

```
UPDATE Order  
SET delivery_status = 'On Delivery'  
WHERE order_id = 1234;
```

SQL query for updating stock for each flower in the arrangement if the sale is rejected:

```
SELECT count, flower_id, arrangement_id  
FROM Composed_of WHERE arrangement_id = 5;
```

```
SELECT flower_id, stock  
FROM Flower_stock WHERE seller_id = 12 AND flower_id = 1;
```

```
UPDATE Flower_stock  
SET stock = stock + 3  
WHERE flower_id = 1 AND seller_id= 43
```

SQL query for updating seller status information as “Rejected” or “Accepted” of the order:

```
UPDATE Order  
SET seller_status = 'Accepted'  
WHERE order_id = 12
```

SQL query for updating delivery status of the order:

```
UPDATE Order  
SET delivery_status = 'On Delivery'  
WHERE order_id = 12
```

3.3.7. Assign Courier Page

The screenshot shows a web browser window with the URL `https://www.flowergarden.com/assign-courier?id=orderID`. The page header includes the "FlowerGarden" logo and navigation links: Notifications, Change Districts and Hours, Sales, Arrangements, and My Account. The main content area displays the following information:

- Sale ID: ID
- Flower Arrangement Name: Arrangement Name
- Delivery Date: 10/04/2020
- Timeslot: 09:00-10:00
- Courrier:

Below this information are two buttons: "Back" and "Assign". At the bottom of the page, there is a copyright notice: "Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)".

Figure 23: Assign Courier Page for Seller Account

From this page, the sellers will be able to assign their orders to the couriers that will be shown according to how much volume they can carry, which dates and time they are working and where they are serving.

SQL query for selecting the couriers that are suitable:

```
SELECT C.courier_id,A.first_name, A.middle_name, A.last_name,
FROM Courier as C,Courier_serves_to as CST, Courier_working_time as CWT,Account as
A
WHERE CWT.day='monday' AND CWT.hour=1 AND CST.district_id=1 AND 25 <
C.max_volume AND C.courier_id = A.account_id;
```

SQL query for assigning the courier:

```
UPDATE Order
SET courier_status = 'Pending'
WHERE order_id = 1234;
```

3.3.8. Stock Update Page

The screenshot shows a web browser window with the URL `https://www.flowergarden.com/stock?id=accountID`. The page header includes the "FlowerGarden" logo and navigation links: Notifications, Change Districts and Hours, Sales, Arrangements, and My Account. The main content area features a "Flowers:" section with a list of four flowers (Flower 1, Flower 2, Flower 3, Flower 4) and a corresponding stock input field for each. The stock for Flower 1 and Flower 4 is currently set to 100. Below the list is a blue "Update" button. The footer contains the copyright notice: "Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)".

Figure 24: Stock Update Page for Seller Account

Sellers can update their stock information through this screen.

The queries will be executed for each flower.

SQL query for updating the stock of the flower:

```
UPDATE Flower_stock
SET stock=100
WHERE flower_id=11 AND seller_id=12323;
```

SQL query for the stock of a new flower:

```
INSERT INTO Flower_stock (flower_id, seller_id, stock)
VALUES (11, 12323, 100) ;
```

SQL query for deleting the stock of the flower:

```
DELETE FROM Flower_stock
WHERE flower_id=11 AND seller_id=12323;
```

3.4. Courier Pages

3.4.1. Select Districts Page

FlowerGarden

Choose the districts you will serve

Provinces:

Districts:

Working Hours:

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
07:00 - 08:00	07:00 - 08:00	07:00 - 08:00	07:00 - 08:00	07:00 - 08:00	07:00 - 08:00	07:00 - 08:00
08:00 - 09:00	08:00 - 09:00	08:00 - 09:00	08:00 - 09:00	08:00 - 09:00	08:00 - 09:00	08:00 - 09:00
09:00 - 10:00	09:00 - 10:00	09:00 - 10:00	09:00 - 10:00	09:00 - 10:00	09:00 - 10:00	09:00 - 10:00
10:00 - 11:00	10:00 - 11:00	10:00 - 11:00	10:00 - 11:00	10:00 - 11:00	10:00 - 11:00	10:00 - 11:00
11:00 - 12:00	11:00 - 12:00	11:00 - 12:00	11:00 - 12:00	11:00 - 12:00	11:00 - 12:00	11:00 - 12:00
12:00 - 13:00	12:00 - 13:00	12:00 - 13:00	12:00 - 13:00	12:00 - 13:00	12:00 - 13:00	12:00 - 13:00

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Save

Figure 25: Select Districts Page for Courier Account

From this page, couriers will be able to select their working time, dates and which places they can serve.

SQL query for working hours and places for couriers:

```
INSERT INTO Courier_serves_to (district_id, courier_id)
VALUES ( 81, 55) ;
```

```
INSERT INTO Courier_working_time (courier_id, day, hour)
VALUES ( 55, 'monday', 8) ;
```

Firstly, the user will be able to select the province. This is for showing all the provinces in the database.

SQL query for showing provinces:

```
SELECT * FROM Province;
```

After the user selects the province we will show the districts according to that province.

SQL query for showing districts according to selected province:

```
SELECT district_id, district_name
FROM District WHERE province_id = 34;
```


3.4.2. Delivery List Page

FlowerGarden

Deliveries

ID	Delivery Date ▼	Delivery Timeslot ⬆	Volume ⬆	Seller Name ⬆	Receiver Address	Acceptance Status	Delivery Status
ID1	10/04/2020	09:00 - 10:00	25	Name	Address	Pending	Not Get From Seller
ID2	10/04/2020	09:00 - 10:00	25	Name	Address	Accepted	Get From Seller
ID3	10/04/2020	09:00 - 10:00	25	Name	Address	Accepted	Delivered to Receiver
ID4	10/04/2020	09:00 - 10:00	30	Name	Address	Accepted	Delivered to Receiver
ID5	10/04/2020	09:00 - 10:00	30	Name	Address	Accepted	Delivered to Receiver

Delivery Details

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 26: Delivery List Page for Courier Account

From this page, couriers will be able to see all of the orders that have been assigned on themselves with given details. After selecting one of the orders users will be directed to the page that is shown in part 3.4.3.

SQL query for displaying orders assigned to courier:

```
SELECT order_id, desired_delivery_date, desired_delivery_time, volume, A.first_name,
A.last_name, address_text, delivery_status, courier_status
FROM Order natural joins Account as A
WHERE courier_id = 55 and seller_id = A.account_id;
```

3.4.3. Delivery Page

The screenshot shows a web browser window with the URL `https://www.flowergarden.com/delivery?id=orderID`. The page title is "FlowerGarden". In the top right corner, there are four icons: a bell for "Notifications", a house for "Change Districts and Hours", a truck for "Deliveries", and a person for "My Account".

The main content area is divided into four sections:

- Delivery Information:**
 - Delivery ID: ID
 - Delivery Date: 10/04/2020
 - Timeslot: 09:00-10:00
 - Volume: 35
 - Delivery Type: A Type
- Customer Information:**
 - Customer Name: Name
 - Customer Phone: +123456789
 - Buttons: "Cancel Delivery" and "Accept Delivery"
 - Button: "Delivered to Receiver"
- Seller Information:**
 - Seller Name: Name
 - Seller Phone: +123456789
 - Seller Address: [Redacted]
- Receiver Information:**
 - Receiver Name: Name
 - Receiver Phone: +123456789
 - Receiver Address: [Redacted]

At the bottom, there is a copyright notice: "Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)".

Figure 27: Delivery Page for Courier Account

Couriers will be able to accept or cancel the assigned deliveries on themselves. They can also see the other details about the order which are shown above.

SQL query for displaying the details of the delivery (Delivery and Receiver information will be obtained):

```
SELECT (order_id, order_date, receiver_name, receiver_phone, district_id, address_text,
delivery_date, delivery_type, delivery_status, desired_delivery_date, desired_delivery_time,
seller_status, courier_status, seller_id, courier_id, customer_id)
FROM Order as O
WHERE O.courier_id = 55 AND order_id = 12
```

SQL query for displaying the details of seller for this delivery:

```
SELECT A.first_name, A.middle_name, A.last_name, A.phone, A.Address
FROM Account as A
WHERE A.account_id = 22
```

SQL query for displaying the details of customer for this delivery

```
SELECT A.first_name, A.middle_name, A.last_name, A.phone, A.Address
FROM Account as A
WHERE A.account_id = 56
```

SQL query for updating courier status information as "Rejected" or "Accepted" of the order:

```
UPDATE Order
```

```
SET courier_status = 'Accepted'
WHERE order_id = 12
```

SQL query for updating delivery status of the order:

```
UPDATE Order
SET delivery_status = 'Delivered'
WHERE order_id = 12
```

3.5. Customer Service Pages

3.5.1. Complaints Page

Order ID	Complaint	Customer Email	Seller Email	Complaint Status
ID1	10/04/2020	account@flowergarden.com	account@flowergarden.com	Waiting for Service
ID2	10/04/2020	account@flowergarden.com	account@flowergarden.com	Replied
ID3	10/04/2020	account@flowergarden.com	account@flowergarden.com	Solved
ID4	10/04/2020	account@flowergarden.com	account@flowergarden.com	Solved
ID5	10/04/2020	account@flowergarden.com	account@flowergarden.com	Solved

Complaint Details

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 28: Complaints Page for Customer Service Account

From this page, customer services will be able to see the orders that are assigned to themselves with the given information that is shown above.

SQL query for displaying the complaints:

```
SELECT complaint_id, order_id, complaint_date, complaint_status
FROM Complaint, Order as O
WHERE customer_service_id = 55 AND order_id=O.order_id );
```

SQL query to get customer email:

```
SELECT C.complaint_id, A.email
FROM Complaint as C, Order as O, Account as A
WHERE C.order_id=O.order_id and O.customer_id=A.account_id and customer_service_id
= 55 ;
```

SQL query to get seller email:

```
SELECT C.complaint_id, A.email
```

FROM Complaint as C, Order as O , Account as A
WHERE C.order_id=O.order_id and O.seller_id=A.account_id and customer_service_id = 55;

3.5.2. Complaint Page

The screenshot shows a web browser window with the URL <https://www.flowergarden.com/delivery?id=orderID>. The page title is "FlowerGarden". The navigation bar includes links for Notifications, Change Working Hours, Complaints, and My Account. The main content area is divided into six sections:

- Complaint Information:** Complaint ID: ID, Complaint Date: 10/04/2020, Complaint Status: Waiting for Service.
- Order Information:** Order Date: 10/04/2020, Expected Delivery Date: 10/04/2020, Excepted Delivery Timeslot: 09:00-10:00, Delivery Date: 10/04/2020, Arrangement Name: Arrangement Name, Price: ¥25, Payment Method: Credit Card, Volume: 35, Delivery Type: A Type, Arrangement Message: Message.
- Seller Information:** Acceptance Status: Accepted, Seller Name: Name, Seller Phone: +123456789, Seller Email: mail@host.com, Seller Address: [Redacted].
- Courier Information:** Acceptance Status: Accepted, Courier Name: Name, Courier Phone: +123456789, Courier Email: mail@host.com.
- Customer Information:** Customer Name: Name, Customer Phone: +123456789, Customer Email: mail@host.com.
- Receiver Information:** Receiver Name: Name, Receiver Phone: +123456789, Receiver Address: [Redacted].

At the bottom right, there are buttons for "Solved" and "Replied". The footer contains the copyright notice: "Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)".

Figure 29: Complaint Page for Customer Service Account

The customer service accounts can see the complaint, order, seller, courier, customer, and receiver information through this screen. They can also change the status of the complaint to "Replied" and "Solved".

SQL query for complaint information:

```
SELECT order_id, complaint_date, complaint_status, customer_service_id, response_date
FROM Complaint
WHERE complaint_id=123234 AND order_id=123234;
```

SQL query for order information:

```
SELECT payment, order_date, receiver_name, receiver_phone, district_id, address_text,
delivery_date, delivery_type, delivery_status, desired_delivery_date, desired_delivery_time,
message, seller_status, courier_status, seller_id, courier_id, customer_id
FROM Order WHERE order_id=123234;
```

SQL query for seller information:

```
SELECT first_name, middle_name, last_name, email, phone
FROM Account
WHERE account_id=123234
```

```
SELECT address_text FROM Seller WHERE account_id=123234;
```

SQL query for courier information:

```
SELECT first_name, middle_name, last_name, email, phone FROM Account WHERE account_id=123234 ;
```

SQL query for customer information:

```
SELECT first_name, middle_name, last_name, email, phone FROM Account WHERE account_id=123234 ;
```

SQL query for updating complaint status as “Replied” or “Solved”:

```
UPDATE Complaint  
SET complaint_status='Replied'  
WHERE complaint_id=123234 AND order_id=123234;
```

3.6. Common Pages

3.6.1. Notification List Page

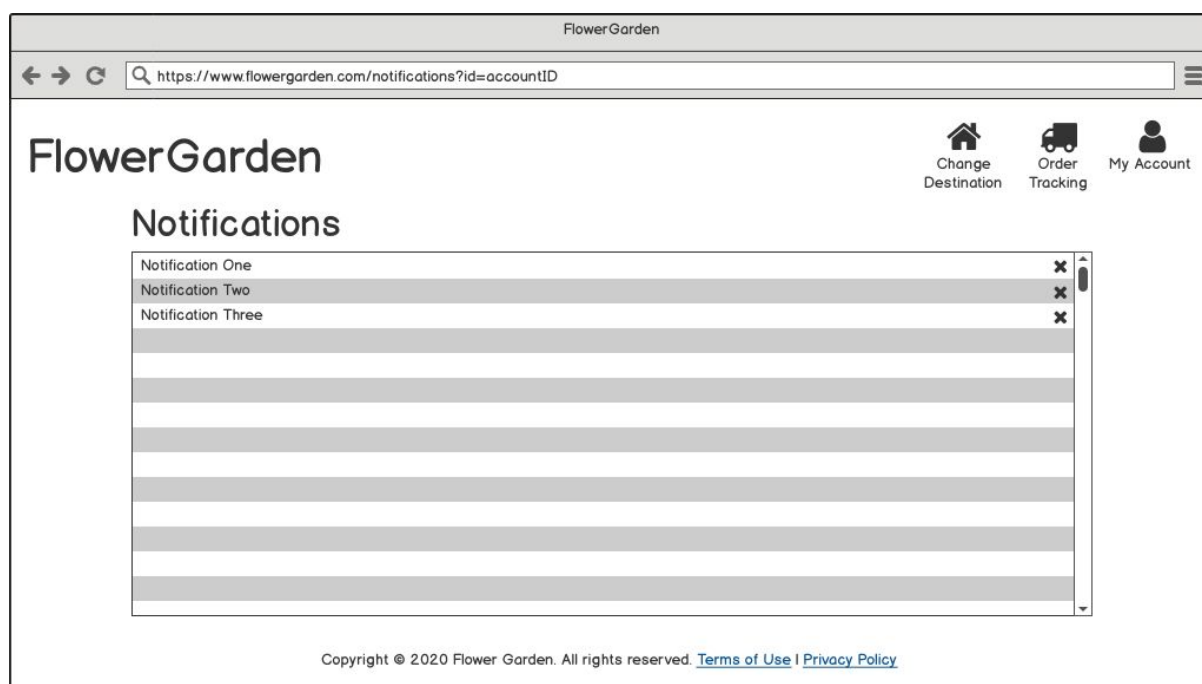


Figure 30: Notification List Page for All Accounts

According to the account, the notifications will be shown. The notifications can be deleted by pressing the cross button at the right of the corresponding notification.

SQL query for getting the notifications of the corresponding accounts:

```
SELECT notifiacion_id, description, timestamp  
FROM Notification  
WHERE account_id=123234;
```

SQL query for deleting the notification:

DELETE FROM Notification
WHERE notification_id=123234;

3.6.2. Account Page

3.6.2.1. Customer

The screenshot shows the 'FlowerGarden' website's account page. The browser address bar shows 'https://www.flowergarden.com/account?id=accountID'. The page title is 'FlowerGarden'. In the top right corner, there are three icons: a bell for 'Notifications', a house for 'Change Destination', and a truck for 'Order Tracking'. The main heading is 'Account Information'. Below this, there are input fields for account details: * First Name (Name), Middle Name (Middle Name), * Last Name (Last Name), * Email (account@flowergarden.com), * Password (masked with asterisks), * Phone (+123456789), and * Re-type password. A 'Save Information' button is located to the right of the password fields. Below the form is a table of saved addresses:

No	Address	District	Province
1	Address One	District One	Province One
2	Address Two	District Two	Province Two
3	Address Three	District Three	Province Three

Below the table, it says 'Last Card Number: 12** **** **34' and a 'Delete Address' button. At the bottom, there is a copyright notice: 'Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)'.

Figure 31: Account Page for Customer Account

According to the given information, the seller account will be updated. The information with the star sign is required information.

SQL query for getting the account information:

```
SELECT first_name, middle_name, last_name, email, password, phone FROM Account
WHERE account_id=123234;
```

SQL query for getting the customer specific information:

```
SELECT credit_card FROM Customer
WHERE account_id=123234;
```

SQL query for getting the saved addresses of the customer:

```
SELECT district_id, address FROM Saved_addresses
WHERE customer_id=123234;
```

SQL query for updating the account information:

```
UPDATE Account
SET first_name='Name', middle_name='Middle Name', last_name='Last Name',
email='account@flowergarden.com', password='sEm4?l*3Fd1', phone='123456789'
WHERE account_id=123234;
```

SQL query for deleting the saved addresses:

```
DELETE FROM Saved_addresses  
WHERE customer_id=123234 AND district_id=123234 AND address='Adress One';
```

3.6.2.2. Seller

FlowerGarden

Account Information

* First Name:

* Last Name:

* Email:

* Password:

* Re-type password:

Middle Name:

* Phone:

* IBAN:

* Adress:

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 32: Account Page for Seller Account

According to the given information, the seller account will be updated. The information with the star sign is required information. The sellers also change their stock information by clicking the “Update Stock Information”, then they will be navigated to the screen stated in section 3.3.8.

SQL query for getting the account information:

```
SELECT first_name, middle_name, last_name, email, password, phone FROM Account  
WHERE account_id=123234;
```

SQL query for getting the seller specific information:

```
SELECT district_id, adress_text, iban_no FROM Seller  
WHERE account_id=123234;
```

SQL query for updating the account information:

```
UPDATE Account  
SET first_name='Name', middle_name='Middle Name', last_name='Last Name',  
email='account@flowergarden.com', password='sEm4?l*3Fd1', phone='123456789'  
WHERE account_id=123234;
```

SQL query for updating the seller specific information:

```
UPDATE Seller
```

```
SET district_id=123234, adress_text='Address', iban_no='TR12345678901234567890234'
WHERE account_id=123234;
```

3.6.2.3. Courier

The screenshot shows a web browser window with the URL `https://www.flowergarden.com/account?id=accountID`. The page title is "FlowerGarden" and the main heading is "Account Information". There are three navigation icons in the top right: "Notifications", "Change Districts and Hours", and "Deliveries". The form contains the following fields:

- * First Name:
- * Last Name:
- * Email:
- * Password:
- * Re-type password:
- Middle Name:
- * Phone:
- * IBAN:
- * Maximum volume you can carry: (with a dropdown arrow)

A "Save Information" button is located at the bottom right of the form. At the bottom of the page, there is a copyright notice: "Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)".

Figure 33: Account Page for Courier Account

According to the given information, the courier account will be updated. The information with the star sign is required information.

SQL query for getting the account information:

```
SELECT first_name, middle_name, last_name, email, password, phone FROM Account
WHERE account_id=123234
```

SQL query for getting the courier specific information:

```
SELECT max_volume, iban_no FROM Courier
WHERE account_id=123234;
```

SQL query for updating the account information:

```
UPDATE Account
SET first_name='Name', middle_name='Middle Name', last_name='Last Name',
email='account@flowergarden.com', password='sEm4?l*3Fd1', phone='123456789'
WHERE account_id=123234;
```

SQL query for updating the courier specific information:

```
UPDATE Courier
SET max_volume=125, iban_no='TR12345678901234567890234'
WHERE account_id=123234;
```


3.6.2.4. Customer Service

FlowerGarden

Account Information

* First Name:

Middle Name:

* Last Name:

* Email:

* Password:

* Phone:

* Re-type password:

Copyright © 2020 Flower Garden. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Figure 34: Account Page for Customer Service Account

According to the given information, the customer service account will be updated. The information with the star sign is required information.

SQL query for getting the account information:

```
SELECT first_name, middle_name, last_name, email, password, phone FROM Account
WHERE account_id=123234;
```

SQL query for updating the account information:

```
UPDATE Account
SET first_name='Name', middle_name='Middle Name', last_name='Last Name',
email='account@flowergarden.com', password='sEm4?l*3Fd1', phone='123456789'
WHERE account_id=123234;
```

4. Implementation Plan

We will be utilizing PHP for the backend and MySQL as the database. For the web application interface, we will use React and JavaScript.

5. Website

<https://burakintisah.github.io/FlowerGarden/>