Bilkent University

Department of Computer Engineering

CS 353 - Database Systems

Final 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 Content

Table of Content **2**

System Description **4**

Final E/R **5**

Modifications 5

Revised E/R Diagram 6

Table Schemas **7**

Account 7

Timeslot 7

Seller 7

Customer 7

Saved_addresses 7

Courier 7

Customer_service 7

Complaint 7

Order 8

District 8

Province 8

Flower 8

Flower_arrangement 8

Occasion 8

Comment 8

Notification 8

Seller_serves_to 9

Courier_serves_to 9

Flower_stock 9

Composed_of 9

Seller_working_time 9

Courier_working_time 9

Implementation Details **10**

Environment, Framework, Languages 10

Problems 10

Contribution 10

Advanced DB Features **11**

Reports 11

Views 11

Trigger 12

Stored Procedure 12

Constraint 12

Secondary Indices 12

User Manual **13**

Sign Up & Login 13

Customer Manual 15

Seller Manual 19

Courier Manual 24

Customer Service Manual 25

1. System Description

In this project, we have proposed, designed, and implemented an online flower shopping system, FlowerGarden. We've provided a user-friendly interface for the users of our application: customers, sellers, couriers, and customer services. We tried to design a complete system that is suitable to adopt as a real system to be able to see the challenges of designing a system that considers the different needs of different types of users. We think that we achieved our goal because we provided all the essentials required by our users to use our application as they wish. The sellers of the application choose the districts they serve and when they serve in those districts. Then, they can add custom flower arrangements based on the flowers they select to add. The customers can buy these arrangements if they want to send a flower arrangement to a district where that seller serves. To make it more real we added several options for customers in the order creation page, they can customize their deliveries by selecting the desired delivery type & date and even message for the seller if they want something special. The sellers can assign couriers to their orders if the couriers work in the corresponding district and timeslot.

The customers are welcome to add comments and complaints about their orders. The rating of each flower arrangement is calculated by the rating part of the comments they add. Moreover, their complaints are answered by the customer service employees of our application. The customers of our system, along with the sellers, the couriers and the customer service are notified by the application for each of the actions made by other users of the application that interests them. For example, a customer is notified when a courier is assigned to his/her order.

Finally, we have put a lot of effort into designing a good looking UI for our users which includes a comprehensive feedback system. We believe that the most problematic part of a lot of websites is the lack of a response system when an error occurs. We tried to provide a clear response on each failing request to make the application easy to use. We also provided a report facility for our sellers to see their order statistics to ease their job in deciding which districts to serve to maximize their profits.

2. Final E/R

2.1. Modifications

- We have removed the relation service_working_time, since we realized that their working time is not custom in our application so the relation is not needed.
- Validation_status attribute of seller and courier entities are removed also because they didn't add functionality in our system.
- For the saved addresses of the customer we needed to store district_id because all the shipments and deliveries are conducted based on districts. If a customer wants to buy an arrangement of a seller, that seller should serve in the district the customer wants to send an arrangement. Since we needed a relation between customer and district entities to be able to add district_id into saved_addresses, we converted saved_addresses into a relation. Each saved address, however, contains also a description of the address. To be able to have the description also we added an attribute to the relation saved_addresses named as address_description.
- Similarly, we need the address information of the seller's store since couriers are going to take the order from the sellers' stores and deliver it to the receiver address. Similarly, we needed to store the address as a district_id and address_description tuple. We added a similar relation as explained above but this time we made the relation many to one since each seller has only 1 store address.
- Similar additions were made to the order entity to store receiver addresses.
- The primary keys of comment and timeslot entities were missing, they are added to correct the diagram.
- Based on feedback for the design report, the relationships between seller-timeslot and courier-timeslot are updated. In the updated version, the seller and courier make total participation in the relationships.
- Finally, the relationship shipping and the entity order is revised. The attributes added to the order entity seller_status and courier_status are removed and added as the attributes of the shipping relationship.

2.2. Revised E/R Diagram

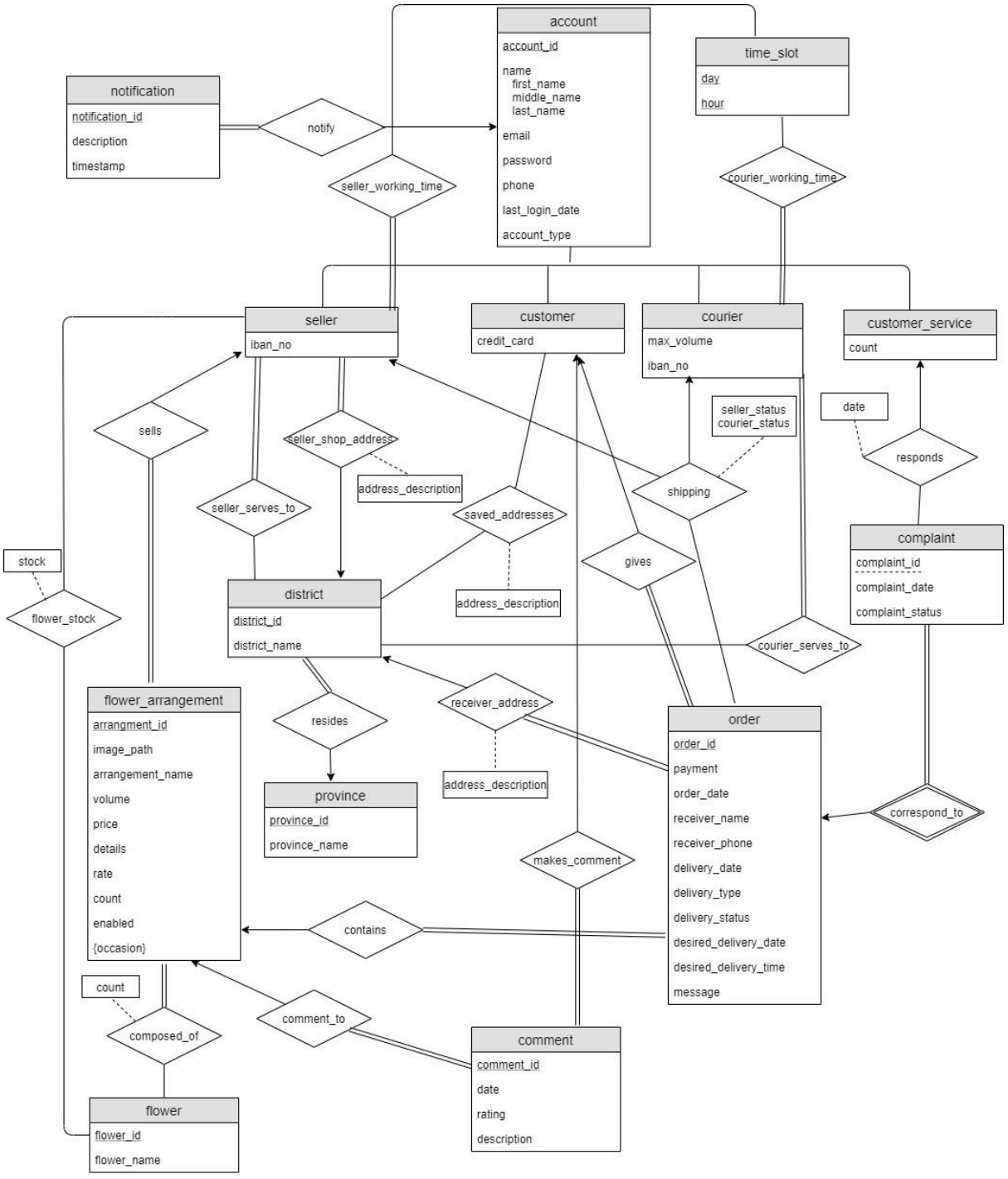


Figure 1: ER Diagram

3. Table Schemas

3.1. Account

Relational Model: account(account_id, first_name, middle_name, last_name, email, password, phone, last_login_date, account_type)
Unique(email)

3.2. Timeslot

Relational Model: timeslot(hour,day)

3.3. Seller

Relational Model: seller(account_id, district_id, address_text, iban_no)
Foreign Key account_id References account
Foreign Key district_id References district

3.4. Customer

Relational Model: customer(account_id, credit_card)
Foreign Key account_id References account

3.5. Saved_addresses

Relational Model: saved_addresses(customer_id, district_id, address)
Foreign Key customer_id References account(account_id)
Foreign Key district_id References district

3.6. Courier

Relational Model: courier(account_id, max_volume, iban_no)
Foreign Key account_id References account

3.7. Customer_service

Relational Model: customer_service(account_id, count)
Foreign Key account_id References account

3.8. Complaint

Relational Model: complaint(order_id, complaint_id, complaint_date, complaint_status, customer_service_id, response_date)
Foreign Key order_id References order
Foreign Key customer_service_id References customer_service(account_id)

3.9. 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)
Foreign Key seller_id References seller(account_id)
Foreign Key courier_id References courier(account_id)
Foreign Key customer_id References customer(account_id)
Foreign Key district_id References district
Foreign Key arrangement_id References flower_arrangement

3.10. District

Relational Model: district(district_id, district_name, province_id)
Foreign Key province_id References province

3.11. Province

Relational Model: province(province_id, province_name)

3.12. Flower

Relational Model: flower(flower_id, flower_name)

3.13. Flower_arrangement

Relational Model: flower_arrangement(arrangement_id, image_path, arrangement_name, volume, price, details, rate, count, seller_id, enabled)
Foreign Key seller_id References seller(account_id)

3.14. Occasion

Relational Model: occasion(arrangement_id, occasion_name)
Foreign Key arrangement_id References flower_arrangement

3.15. Comment

Relational Model: comment(comment_id, description, customer_id, date, rating, arrangement_id)
Foreign Key customer_id References customer(account_id)
Foreign Key arrangement_id References flower_arrangement

3.16. Notification

Relational Model: notification(notification_id, account_id, description, timestamp)
Foreign Key account_id References account

3.17. Seller_serves_to

Relational Model: seller_serves_to(district_id, seller_id)
Foreign Key seller_id References seller(account_id)
Foreign Key district_id References district

3.18. Courier_serves_to

Relational Model: courier_serves_to(district_id, courier_id)
Foreign Key courier_id References courier(account_id)
Foreign Key district_id References district

3.19. Flower_stock

Relational Model: flower_stock(stock, flower_id, seller_id)
Foreign Key seller_id References seller(account_id)
Foreign Key flower_id References flower

3.20. Composed_of

Relational Model: composed_of(count, flower_id, arrangement_id)
Foreign Key flower_id References flower
Foreign Key arrangement_id References flower_arrangement

3.21. Seller_working_time

Relational Model: seller_working_time(seller_id, day, hour)
Foreign Key seller_id References seller(account_id)
Foreign Key day References timeslot
Foreign Key hour References timeslot

3.22. Courier_working_time

Relational Model: courier_working_time(courier_id, day, hour)
Foreign Key courier_id References courier(account_id)
Foreign Key day References timeslot
Foreign Key hour References timeslot

4. Implementation Details

4.1. Environment, Framework, Languages

In the FlowerGarden application, the MySQL database is chosen. The MySQL Workbench application is used to manipulate the database easily instead of the MySQL command line application.

The frontend of the application is built using React.js library with JavaScript. In order to initialize the project “npm create-react-app” is used. With this command needed libraries and files are created. For building beautiful forms and guaranteeing intuitive user experience we have used Bootstrap 4.

The backend of the application is built using JavaScript with the Node.js runtime environment. For accessing the MySQL Database, the MySQL2 driver is installed. The Express framework is also installed for handling the HTTP requests of the server. The Body Parser middleware module is used to handle the body of the POST requests in JSON format. Also, the CORS package is used to skip the same-origin policy of the browsers and get the requests correctly.

4.2. Problems

- While sending requests to the backend, the same-origin error is encountered at the frontend. The frontend of the application sends requests to the backend which is at the different IP or port. In order to skip this policy, the CORS package is used for managing the cross-origin resource sharing (CORS) requests.
- In the backend side of the application, the successive queries should be sent to the database and the current thread should wait for the response of the former query before sending later. To solve this problem, the MySQL2 driver that offers Promise Wrapper is chosen for asynchronous programming instead of MySQl driver that offers just callbacks.

4.3. Contribution

We have prepared the proposal and design documents together with equal contribution.

- Munib Emre Sevilgen: The backend of the application is built.
- Esra Nur Deniz: The GUI of the application is built with Burak İntişah.
- Meryem Banu Cavlak: I have updated the SQLs, created the database and wrote the final report.
- Osman Burak İntişah:The GUI of the application is built with Esra Deniz.

5. Advanced DB Features

5.1. Reports

1st Report: The report finds the order statistics of a seller based on districts of the orders. This information is useful for sellers (and displayed to them) as the sellers can choose the districts they serve based on this statistic.

```
SELECT district_id, district_name, min(order_date) as first_order, count(order_id) as
number_of_orders, sum(price) as total_revenue, avg(price) as average_revenue, avg(rate)
as average_rate
FROM flowergarden.order NATURAL JOIN district NATURAL JOIN flower_arrangement
WHERE seller_id = 1
GROUP BY district_id
ORDER BY average_revenue
```

district_id	district_name	first_order	number_of_orders	total_revenue	average_revenue	average_rate
13	Çukurova	2008-01-01 00:00:01.00	1	10.00	10.000000	5.00000
15	Kozan	2008-01-01 00:00:01.00	3	51.49	17.163333	5.00000
17	Beypazarı	2008-01-01 00:00:01.00	4	93.47	23.367500	5.00000
18	Çankaya	2008-01-01 00:00:01.00	5	119.46	23.892000	5.00000

2nd Report: The report finds the statistics of all the orders made by customers. It finds the number of arrangements, and their average price in each district for a specific occasion name. This information is useful for application managers to understand customer behaviour in different districts.

```
SELECT district_id, district_name, occasion_name, count(order_id) AS arrangement_count,
avg(price) AS average_price
FROM flowergarden.order NATURAL JOIN enabled_arrangements NATURAL JOIN
occasion NATURAL JOIN district
GROUP BY district_id, occasion_name
ORDER BY district_id
```

district_id	district_name	occasion_name	arrangement_count	average_price
11	Aladağ	Anniversary	4	17.995000
11	Aladağ	Congratulations	4	17.995000
15	Kozan	Birthday	1	15.500000
15	Kozan	Congratulations	1	15.500000
18	Çankaya	Anniversary	1	10.000000
18	Çankaya	Congratulations	1	10.000000
19	Yenimahalle	Birthday	1	15.500000
19	Yenimahalle	Anniversary	3	10.000000
19	Yenimahalle	Congratulations	4	11.375000

5.2. Views

1st View: The view is going to be used by all pages of seller, courier and customer except login, sign up and account settings pages. It will contain the account information except

password and last_login_date for security purposes and will be used mainly for finding names of people registered in the system.

```
CREATE VIEW secure_account AS
SELECT account_id, first_name, middle_name, last_name, email, account_type
FROM account
```

2nd View: To be able to give detailed information about past orders to customers we disallowed sellers to delete their flower arrangements but they can disable them. However, the disabled ones cannot be bought so they are not listed in the home page of the customer where the customer sees the flower arrangements he/she can buy. The view is going to be used to display flower_arrangements to the customers.

```
CREATE VIEW enabled_arrangements AS
SELECT *
FROM flower_arrangement
WHERE enabled <> 0
```

5.3. Trigger

This trigger is executed after an insertion to the order table and if the seller's stock is not sufficient to prepare the arrangement, deletes the order from the order table and inserts a notification to the notification table to notify the user about cancellation of his/her order.

5.4. Stored Procedure

In the proposal we add a limitation for the number of responses a customer service can make in a day. This stored procedure will return the number of responses each customer service will make in a day. It will use the attribute response_date of the complaint table to filter responses and then group the table by customer_service_id and count the number of complaints.

5.5. Constraint

To be able differentiate between user types consistently we added an attribute to the account table, account_type. This constraint is an assertion that checks whether each entry in seller, customer, courier and customer_service tables contains an entry in the account table with correct account_type value.

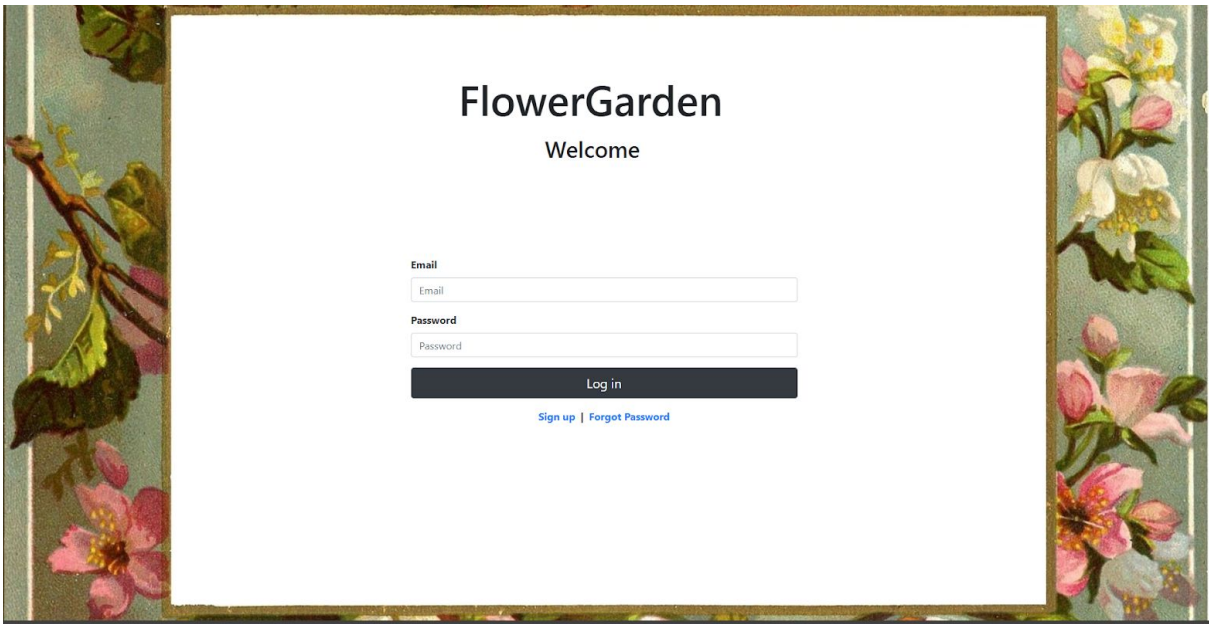
5.6. Secondary Indices

Three secondary indices are added to the system to increase the performance of the query that displays the flower arrangements to the customer on the main page. These attributes are selected because some of the filters are based on these attributes.

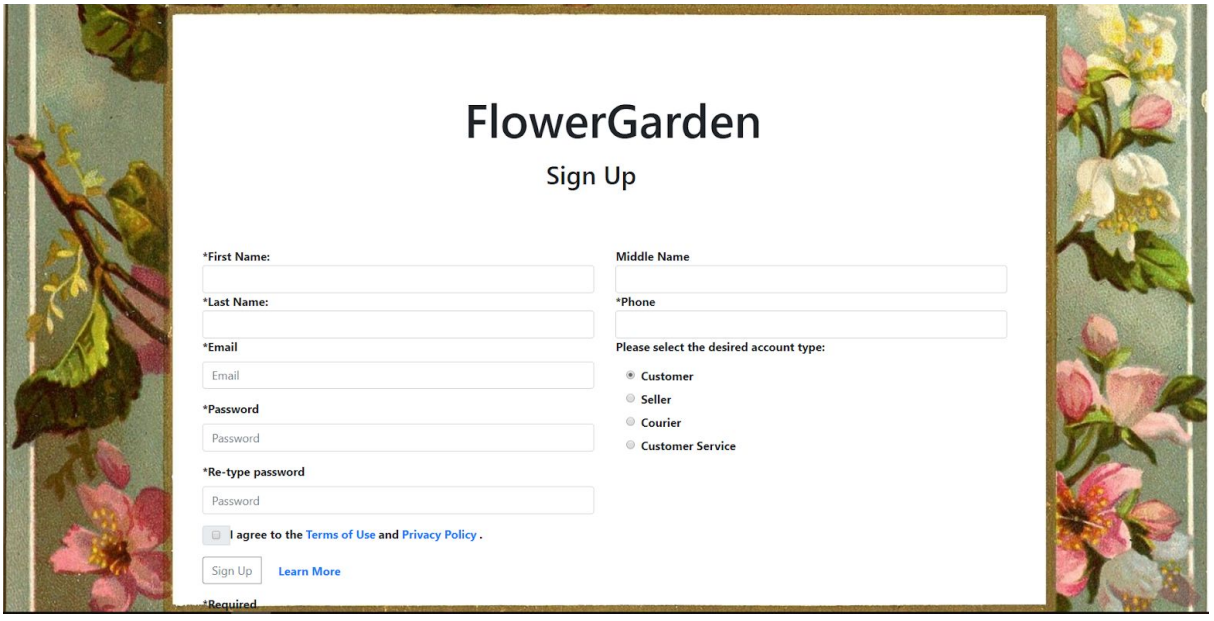
- CREATE INDEX filter_occasion USING BTREE ON occasion(occasion_name)
- CREATE INDEX filter_seller USING BTREE ON flower_arrangement(seller_id)
- CREATE INDEX filter_price USING BTREE ON flower_arrangement(price)

6. User Manual

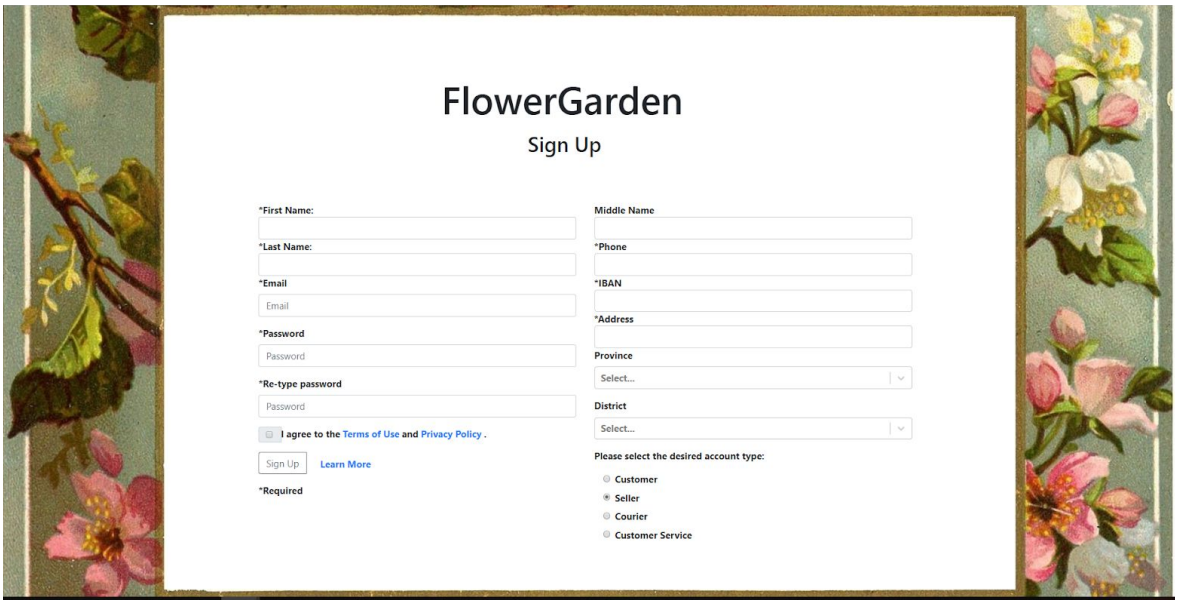
6.1. Sign Up & Login



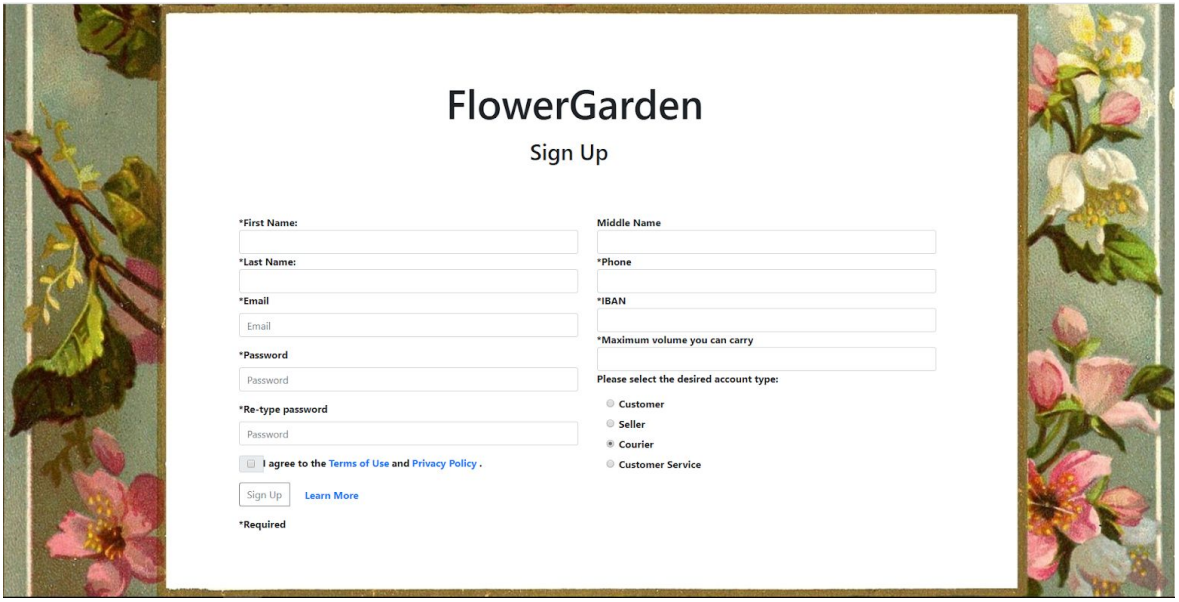
This is the login page for customers, sellers, couriers and customer service.



For customer and customer service this is the required information for sign up. They can register to the system after entering all the required information.

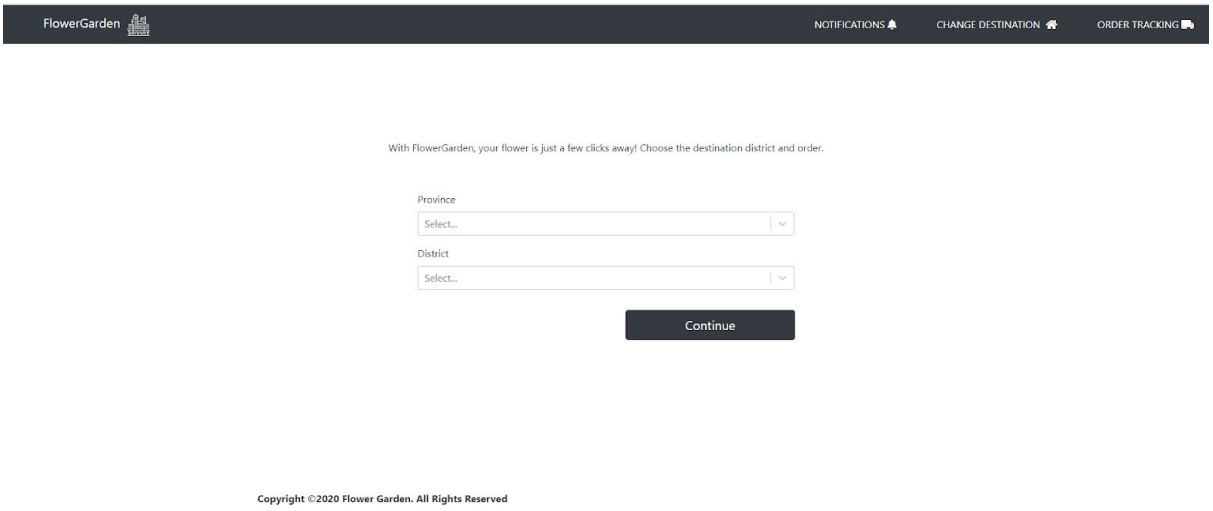


This is the sign up page for sellers. The sellers need to enter the address of their shops and their IBAN numbers to sign up, in addition to the information required for the customer and customer service.

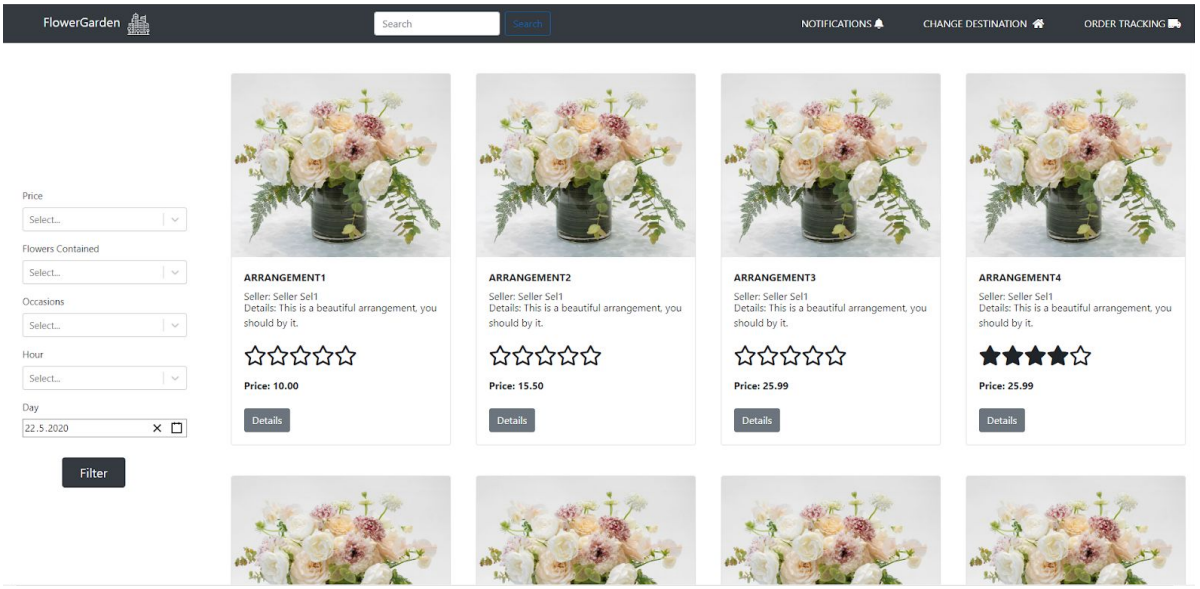


This is the sign up page for the couriers, they need to enter their IBAN numbers and the maximum volume they can carry in the sign up additionally.

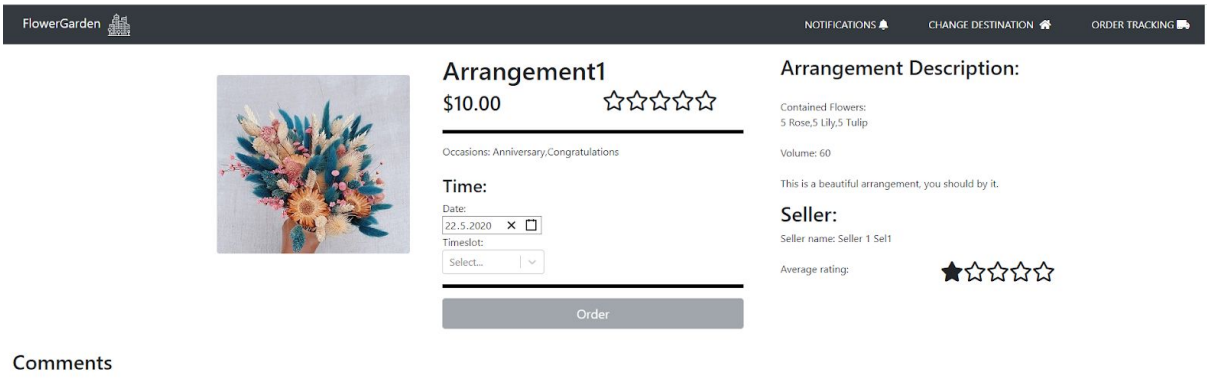
6.2. Customer Manual



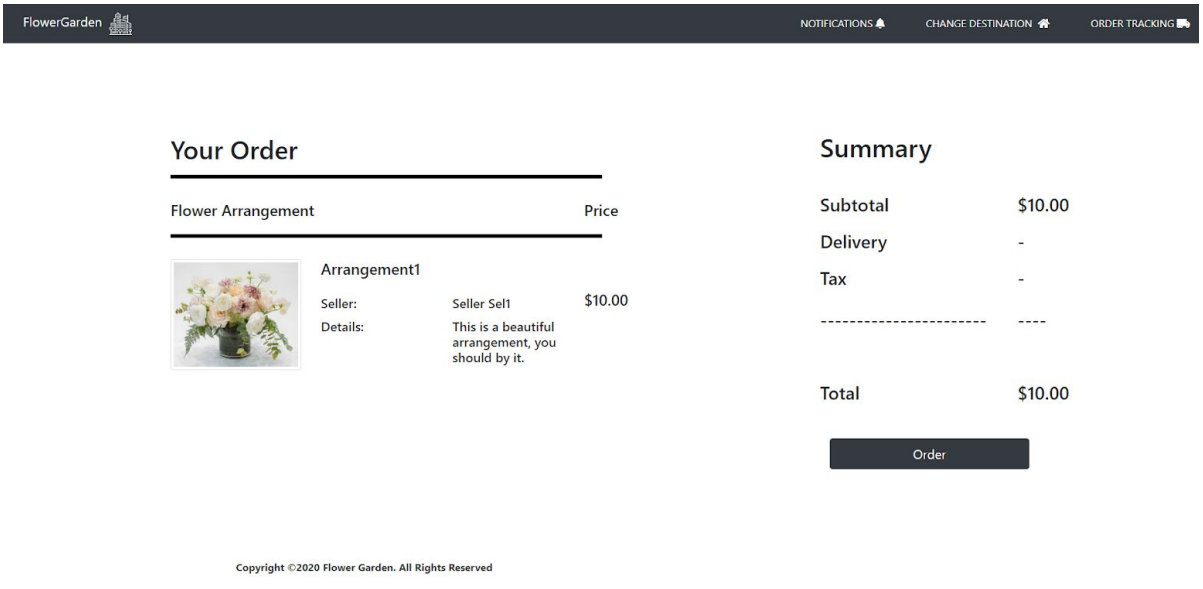
This is the first page a customer sees when he/she has logged in to the FlowerGarden. Here the customer is required to select the province and district information for the receiver address they are going to send a flower arrangement.



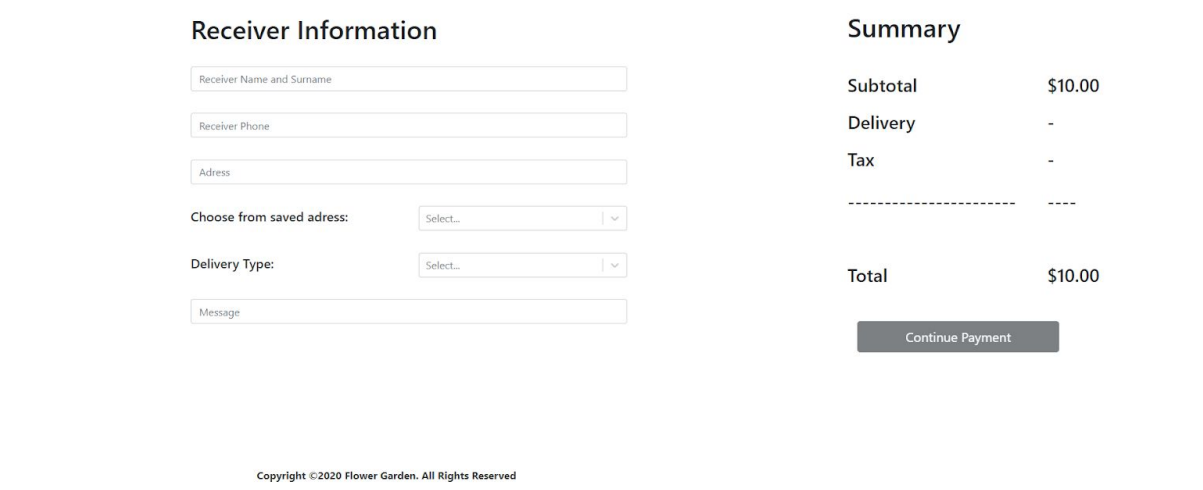
This page is the next page that displays the flower arrangements based on the district selected by the customer in the previous page. Here only the arrangements of the sellers that serve in the district chosen are displayed. The search button and the filter options can be used to filter the displayed flower arrangements further.



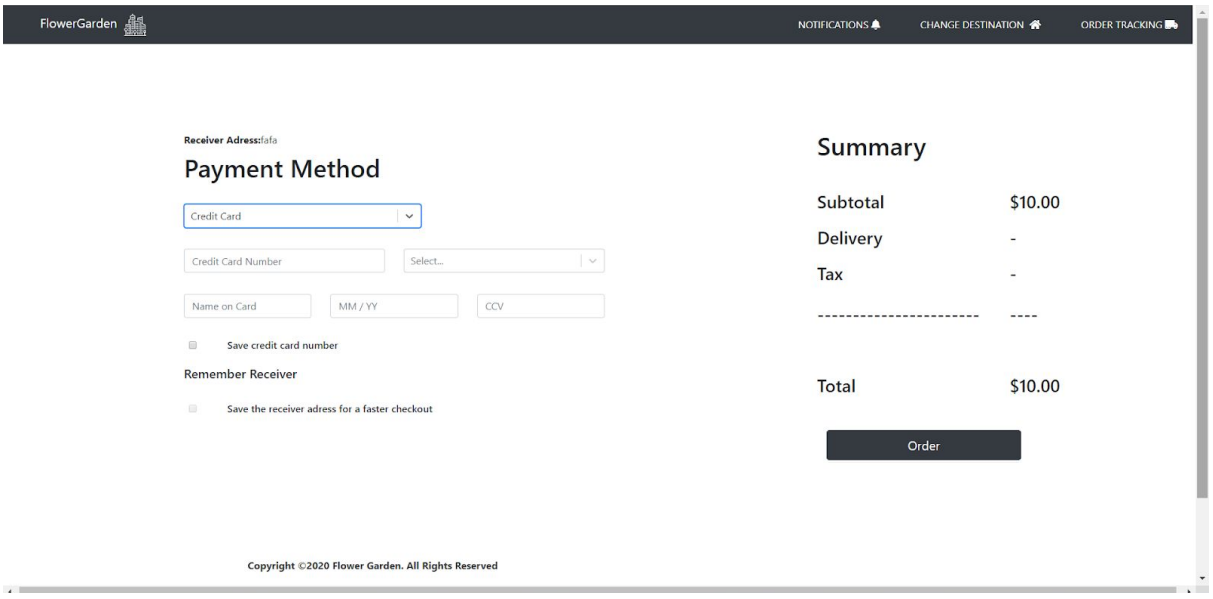
This is the page displayed if the customer presses the details button in the previous page. Here, the customer can choose date and time information to give an order. Also, the customer can see the details of the arrangement before deciding on whether or not to give the order.



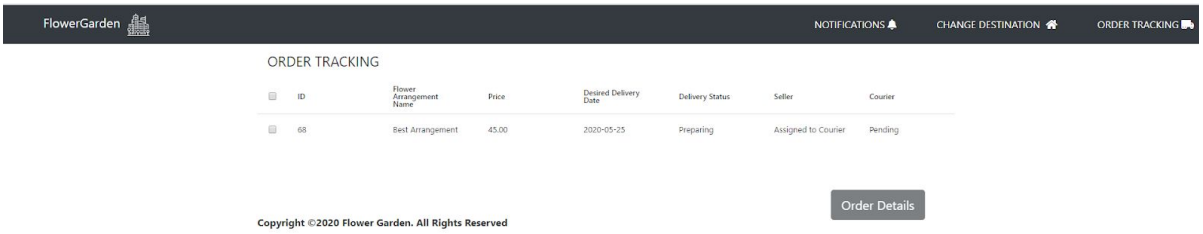
In this page the customer sees the preview of the order he/she will make.



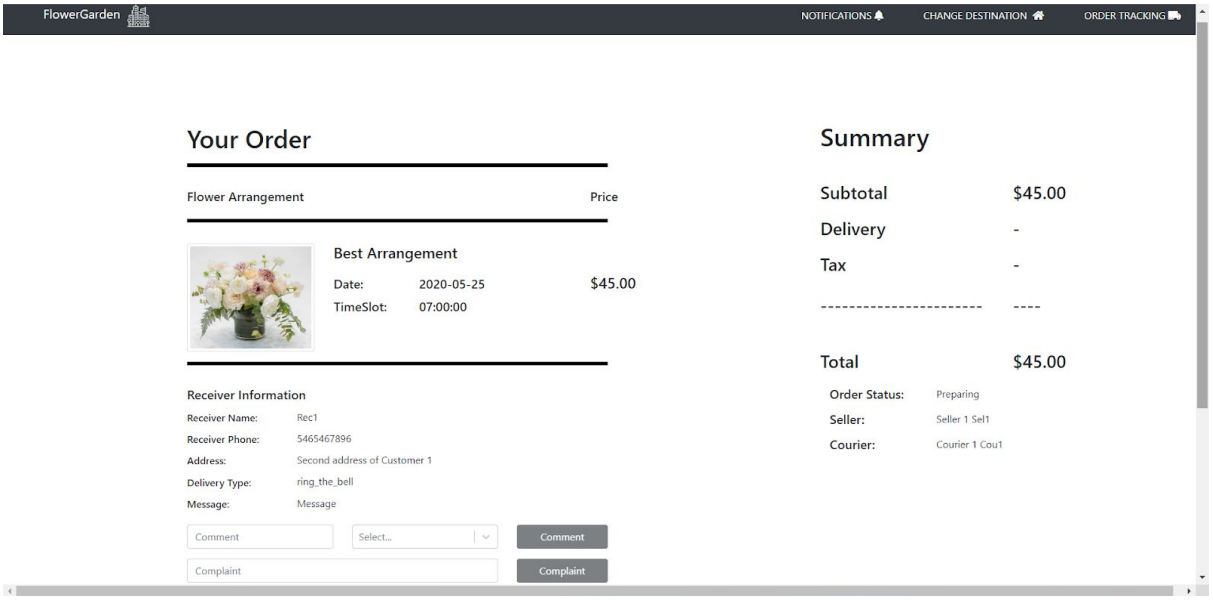
In this page, the customer enters the necessary information to create an order.



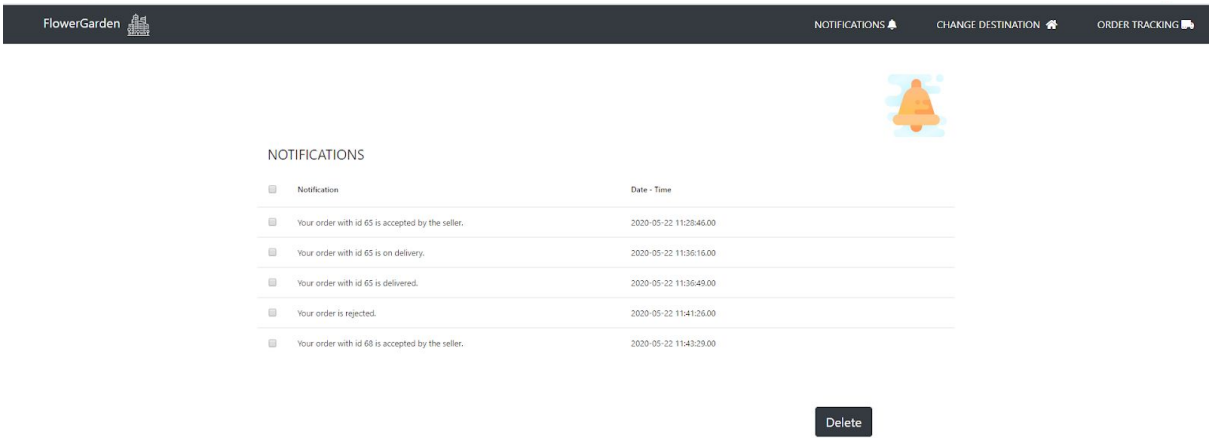
This is the final page before giving the order. The customer should enter the payment method and information in here.



Here the customer can see complete list of his/her orders.

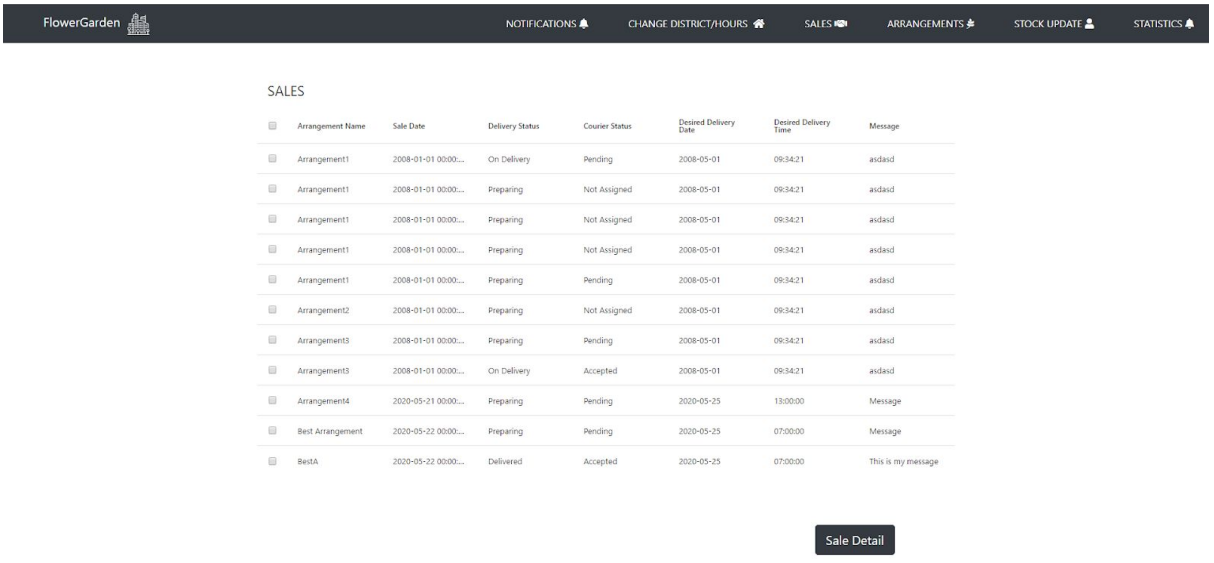


This page displays the detailed information of one of the customer's orders. Here the customer can add a comment or complaint for the order.

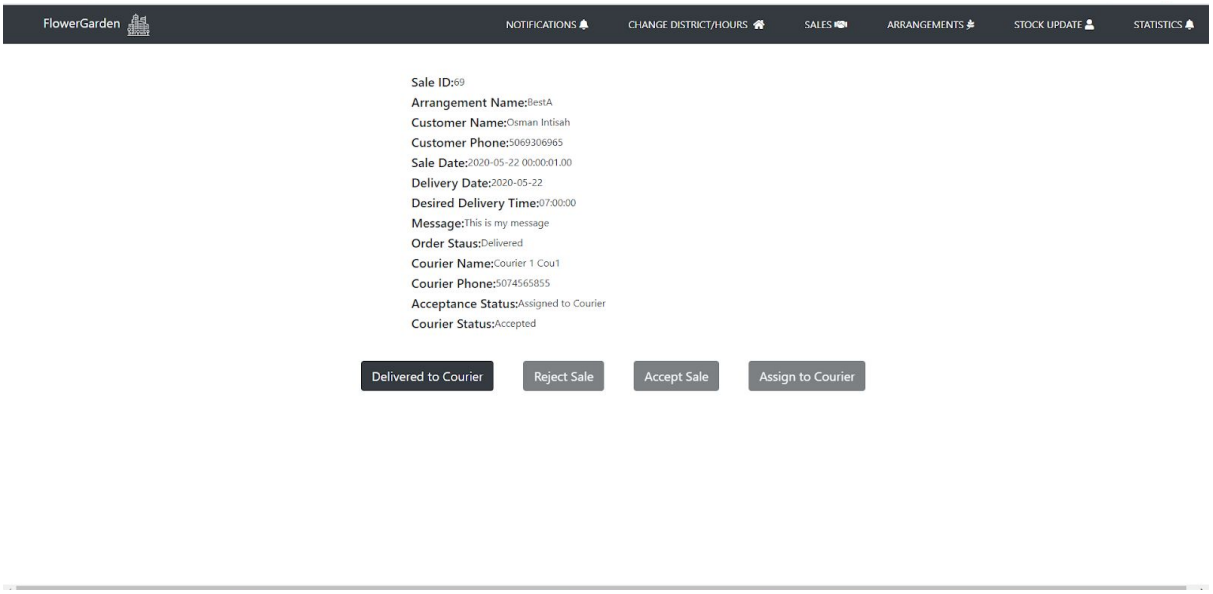


This is the notification page of the customer. Here the customer is informed on the actions made on his/her orders. A similar notification page exists for all user types but excluded in the report for clarity.

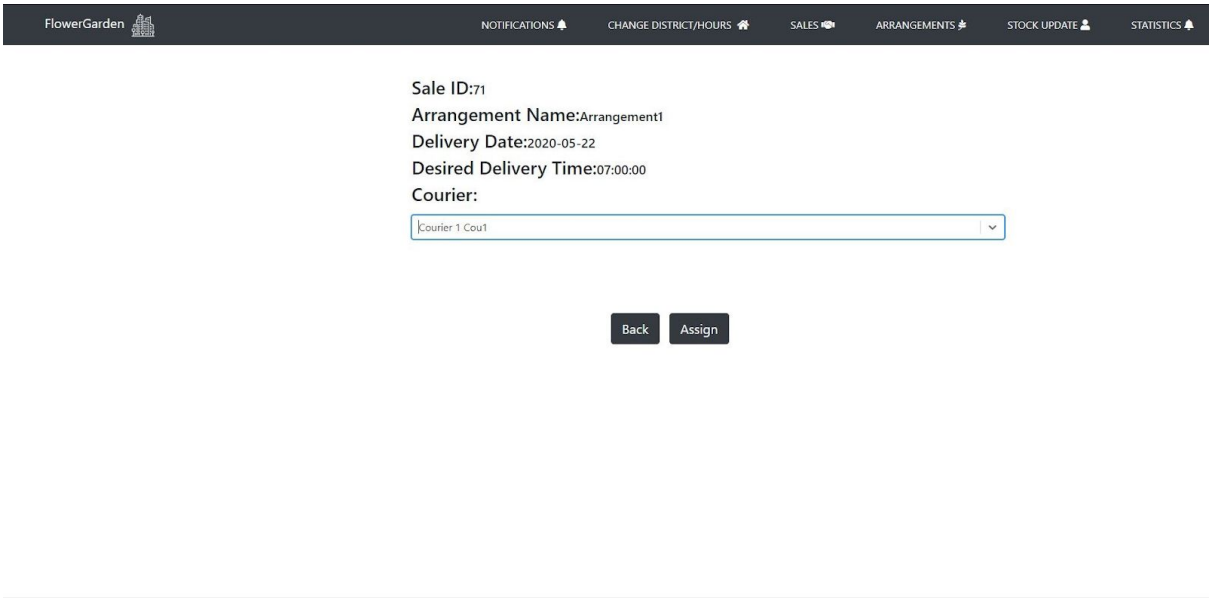
6.3. Seller Manual



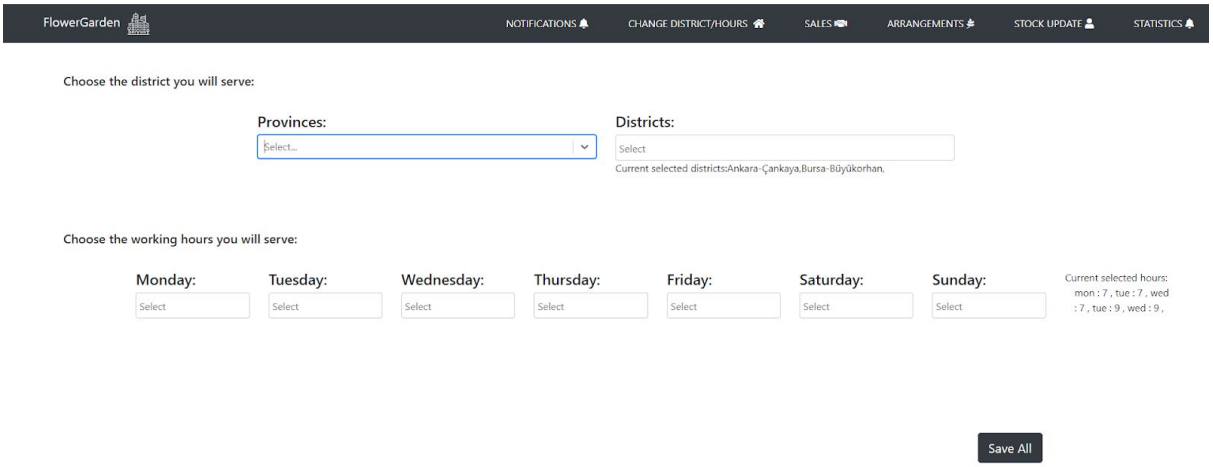
Here is the main page of the seller. Here the seller can see the complete list of his/her sales.



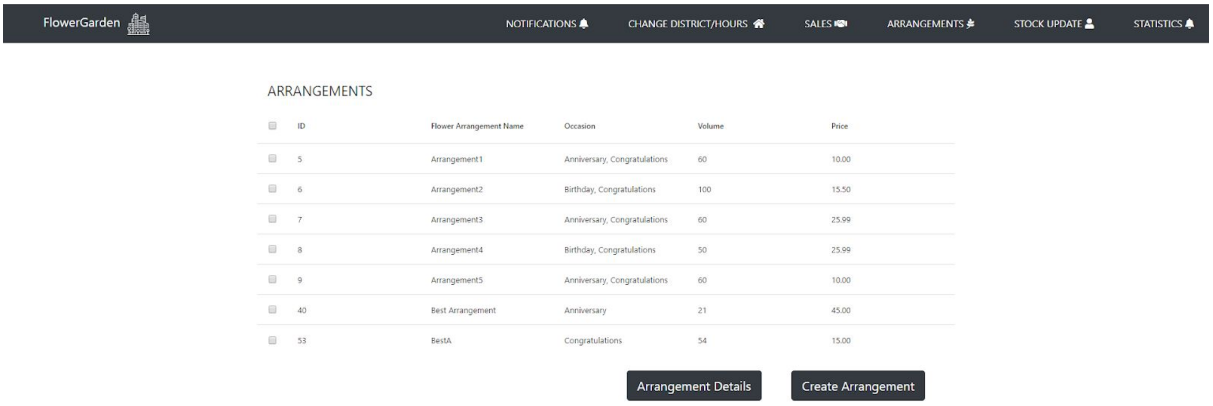
This page explains one sale of the seller. In this page the seller can perform multiple operations on the sale such as accepting or rejecting it.



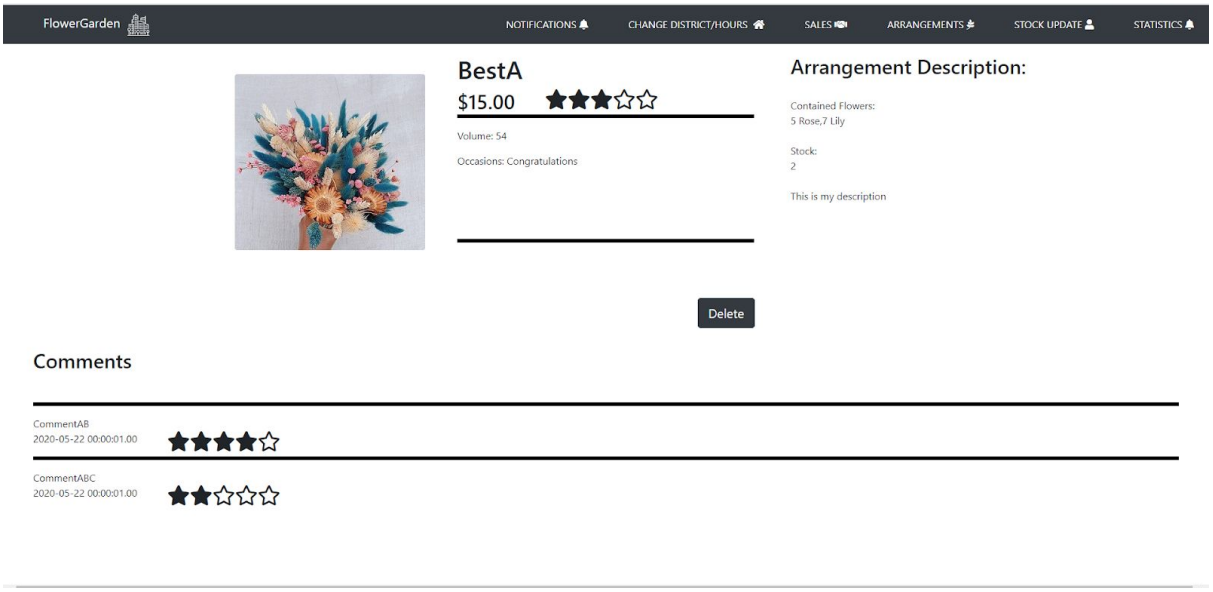
This page comes if the seller clicks assign to the courier button in the previous page. Here the seller can choose one of the couriers to assign the order.



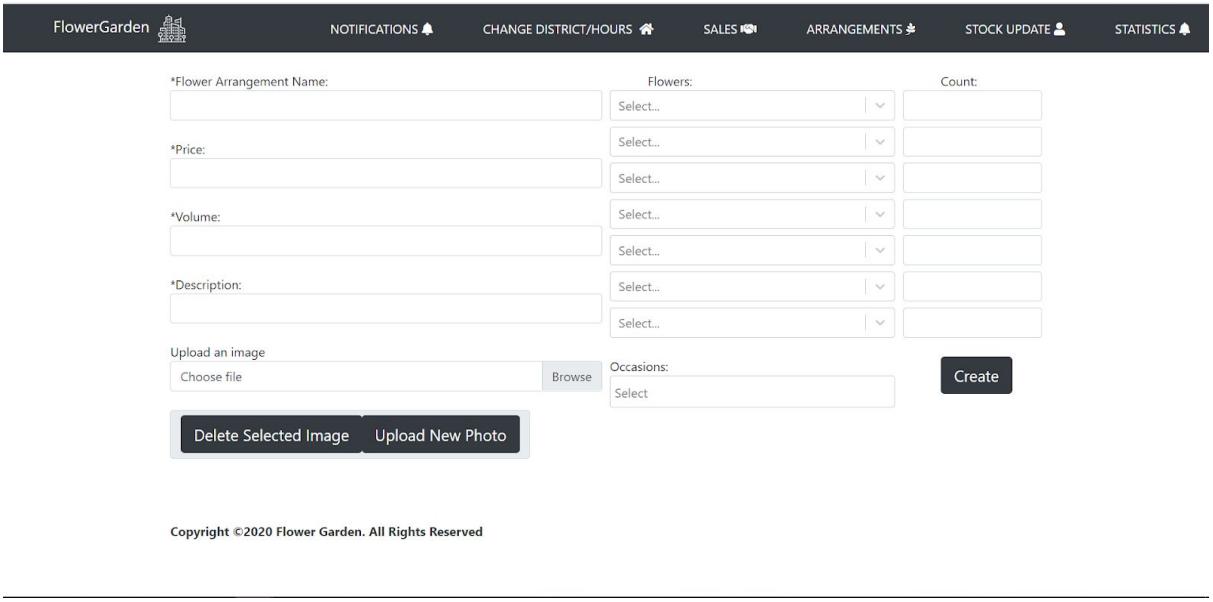
The orders can be given if the seller is serving in the district of the order and in the time the order is going to be delivered. Here, the seller can arrange the districts he/she will serve and the timeslot.



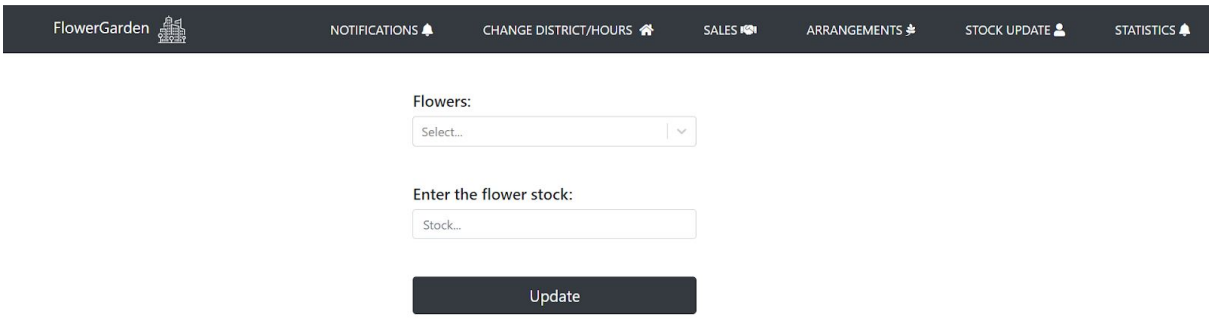
This page lists the flower arrangements the seller has.



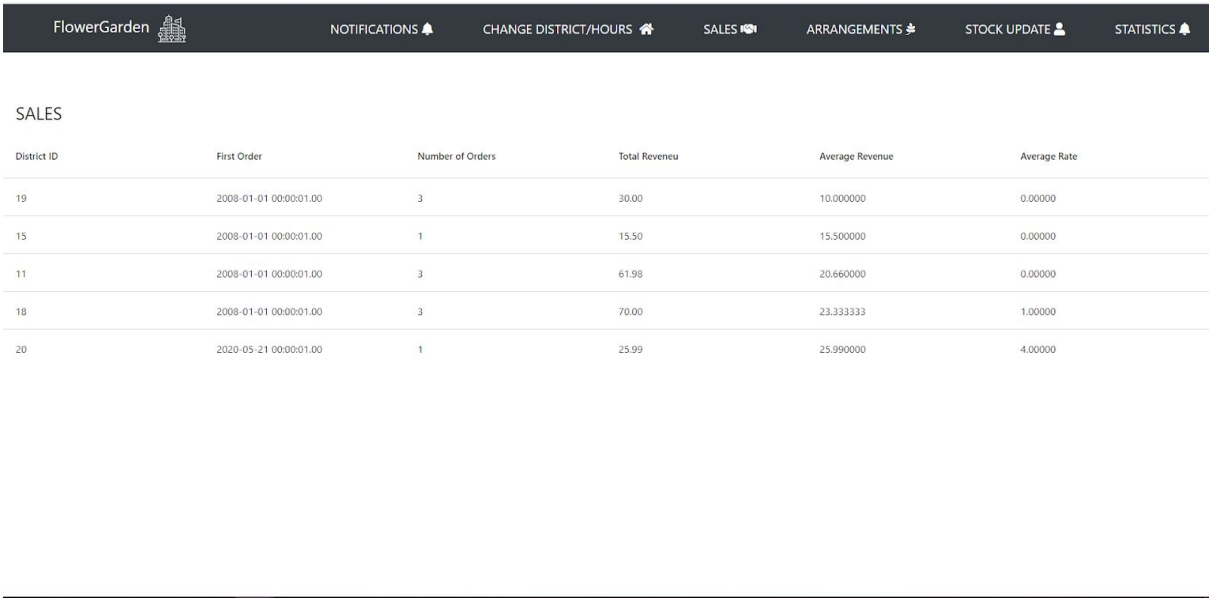
The arrangement details button navigates the seller to here and in here the seller can examine his/her arrangement.



This is the create arrangement page of the seller.

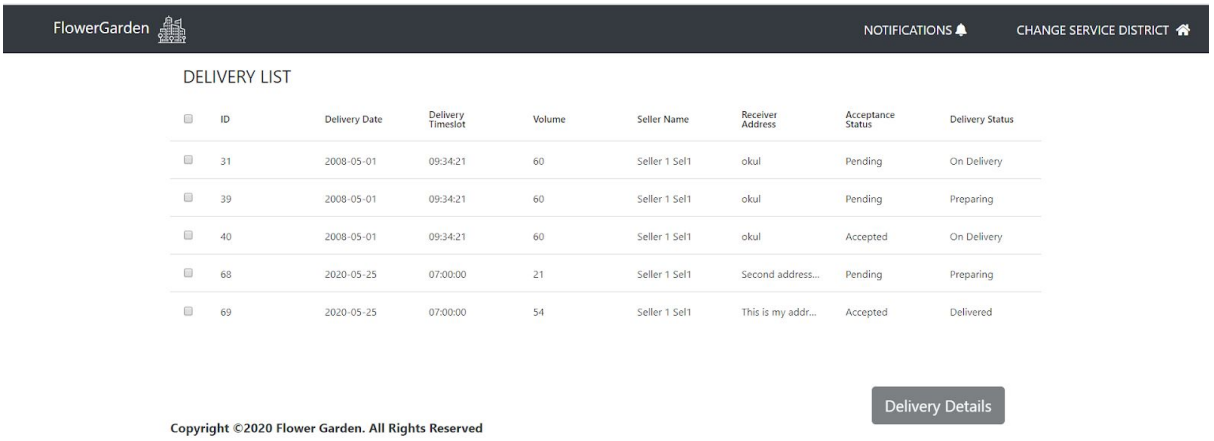


Here the seller can update his/her stock after choosing the flower's name and the new stock of that flower.

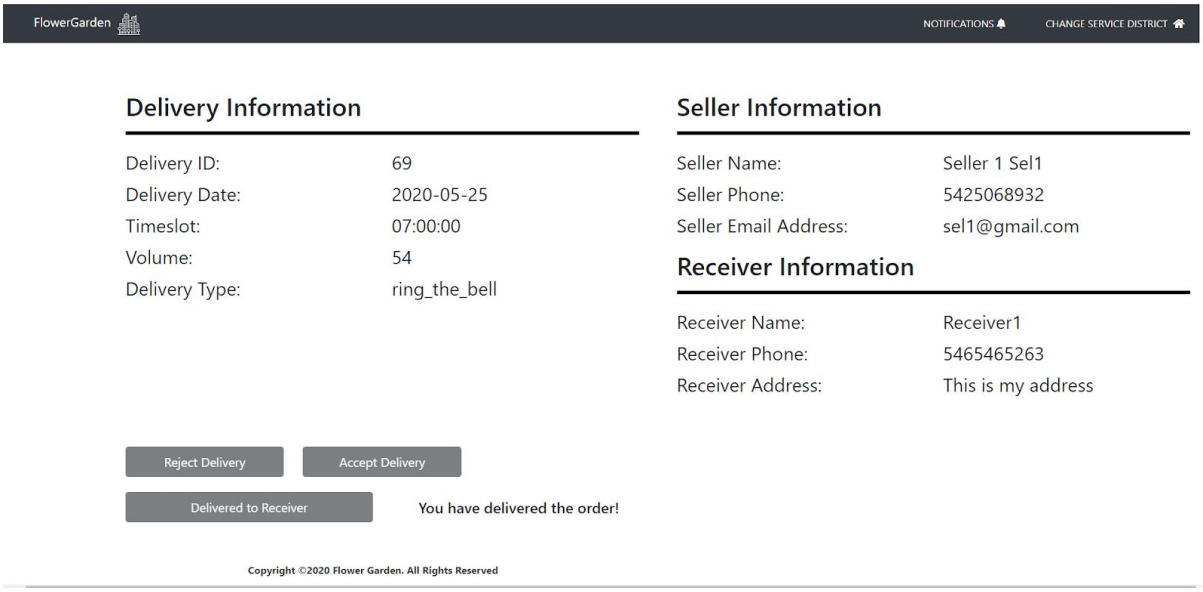


Here the seller can see the statistics about his/her sales based on the districts of the sales.

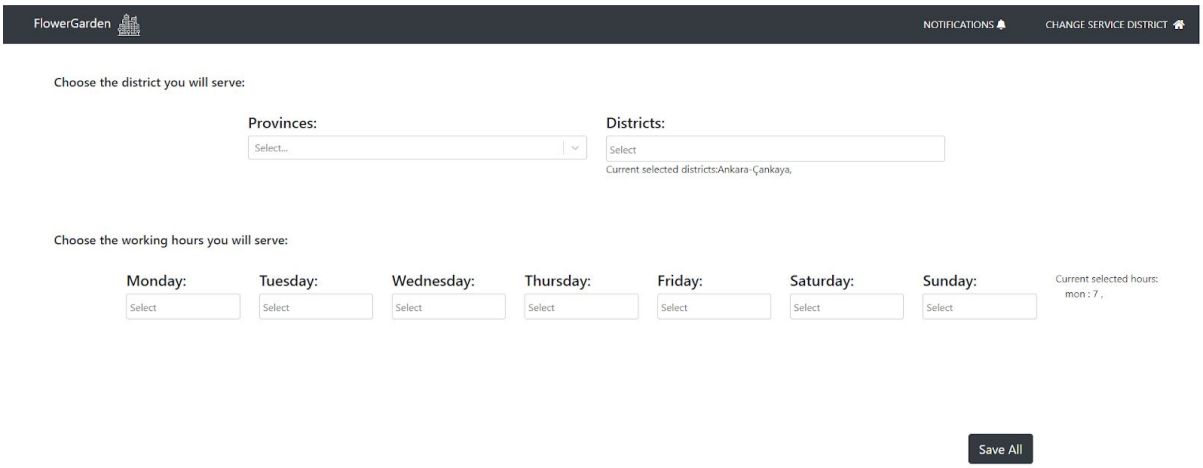
6.4. Courier Manual



This is the main page of the courier. This page gives the complete list of the deliveries the courier has.

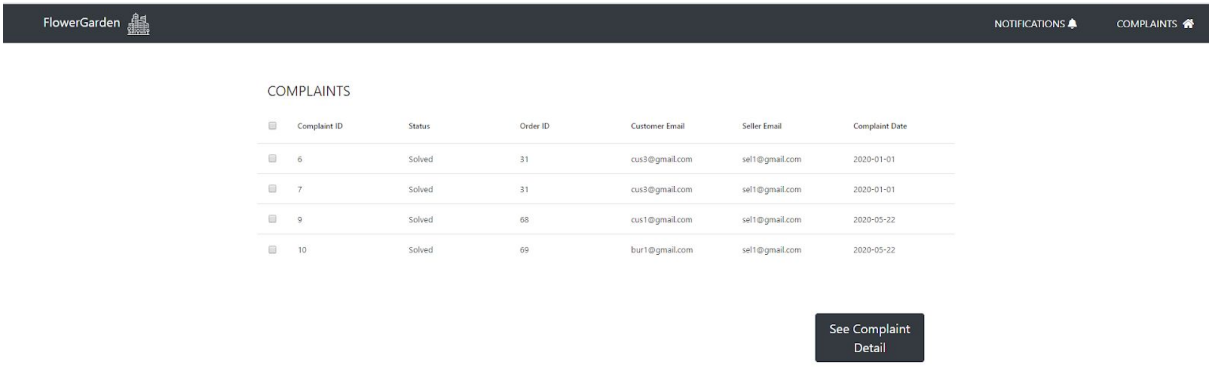


This page gives the detailed information of one of the deliveries selected in the previous page.

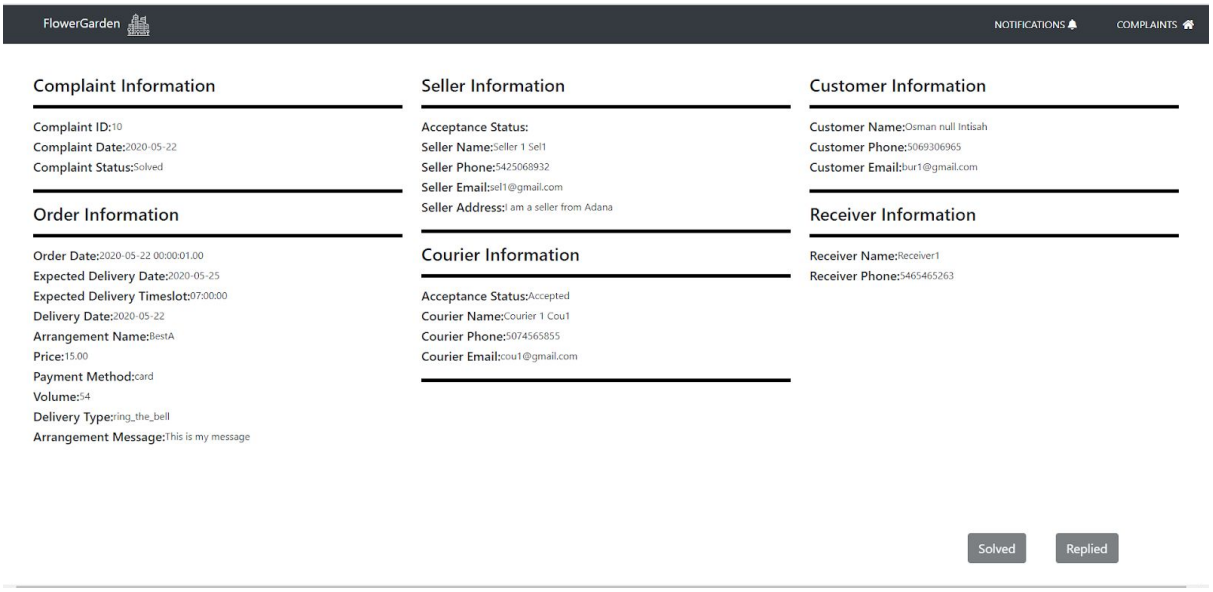


This page is equivalent to the one given in the seller pages. In this page the courier can select the districts and the timeslots he/she is going to serve.

6.5. Customer Service Manual



This is the main page of the customer service. Here the customer service can see the complete list of the complaints he/she should reply and solve.



This is the description page of a complaint selected in the previous page. Here the customer service can see the detailed information and change the status of a complaint as replied or solved.