BİLKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

Final Report

Group 4

İlke Kaş 21803184

Zeynep Büşra Ziyagil 21802646

Bilgehan Akcan 21802901

Ömer Onat Postacı 21802349

Table of Contents

1. Introduction

This project presents a web-based shipping platform. The purpose of the project is to enable individual and corporate users to ship any product and follow the status of their packages easily. The name of the web application is Hermes, inspired from the Greek god Hermes who is the herald of gods. Hermes is a shipping company data management system that enables customers, couriers, shippers and employees in branches to keep track of the packages which are shipped. In Hermes, customers can sign up as a company or an individual, and both of them can send their packages to the desired branches of the company by either calling a courier to their own address or submitting the package to any branch in person. If they choose to call a courier to their own location, the system automatically assigns one of the couriers in the specified branch to receive that package. If they choose to submit a package in person to any branch, the package manager in that branch assigns a shipper to transfer that package to the destination branch which is specified while submitting the package by the customer. After the shipper accepts and transfers the package, the package manager in the destination branch assigns the courier to deliver the package to the recipient.

1.1. Contribution of Each Group Member

1.1.1. Bilgehan Akcan

Contributed to E/R diagram design and definition of relational schemas. Implemented half of the User Interface. Implemented data manipulation with SQL queries.

1.1.2. İlke Kaş

Contributed to E/R diagram design and definition of relational schemas. Created tables with SQL queries. Created database connection and implemented SQL queries in node.js for data manipulation of the front-end.

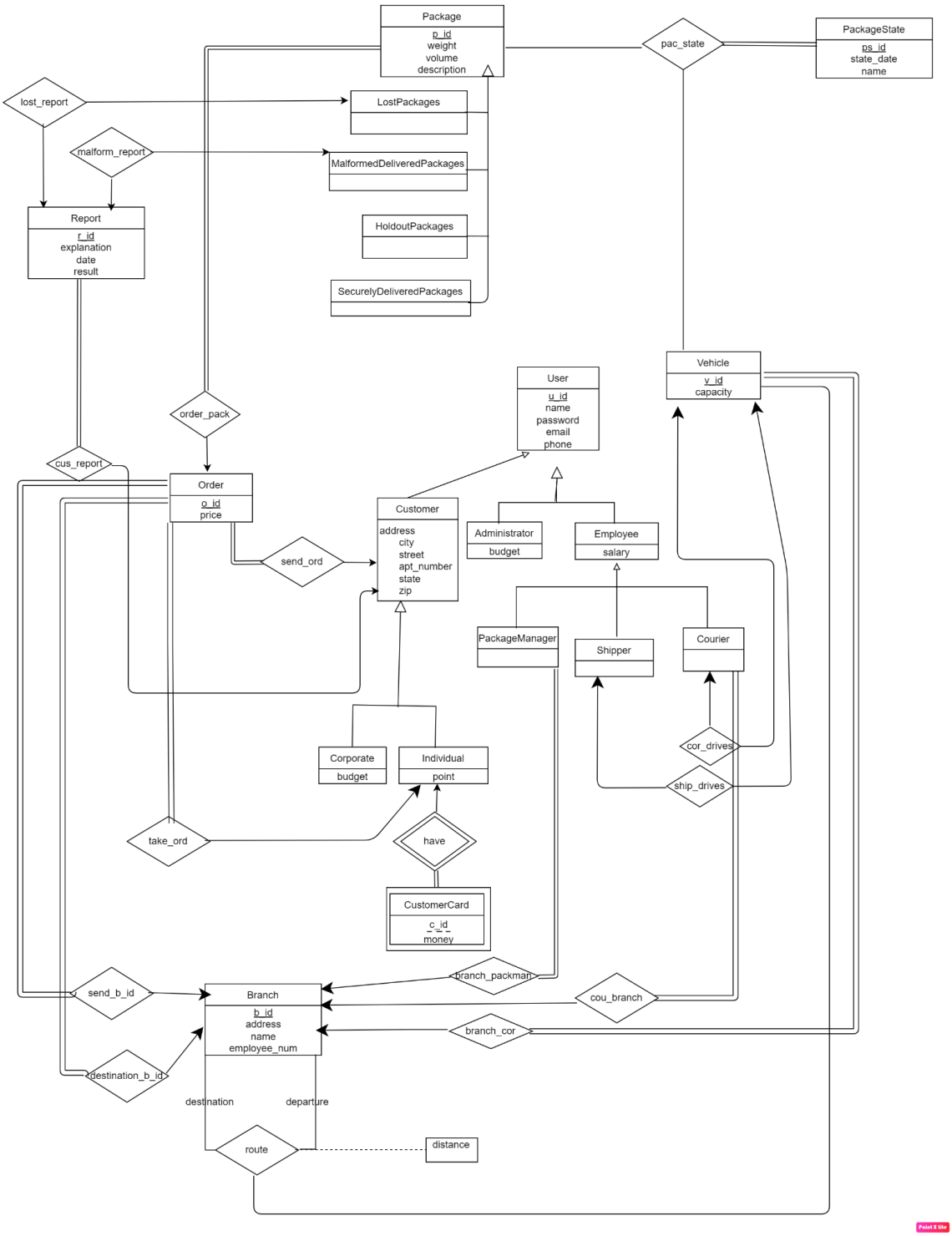
1.1.3. Ömer Onat Postacı

Contributed to E/R diagram design and definition of relational schemas. Implemented API and postgreSQL queries.

1.1.4. Zeynep Büşra Ziyagil

Contributed to E/R diagram design and definition of relational schemas. Implemented half of the User Interface. Implemented data manipulation with SQL queries.

2. Final E/R Diagram



3. Final Relational Schemas and SQL

3.1. User

Relational Model:

User(u_id, name, password, email, phone)

Primary Key: u_id

3.2. Administrator

Relational Model: Administrator(u_id, budget)

Primary Key: u_id

Foreign Key: u_id references User(u_id)

3.3. Employee

Relational Model: Employee(u_id, salary)

Primary Key: u_id

Foreign Key: u_id references User(u_id)

3.4. Package Manager

Relational Model: PackageManager(u_id, b_id)

Primary Key: u_id

Foreign Key: b_id references to Branch(b_id)

Foreign Key: u_id references Employee(u_id)

3.5. Shipper

Relational Model: Shipper(u_id, v_id)

Primary Key: u_id

Foreign Key: v_id references Vehicle(v_id)

Foreign Key: u_id references Employee(u_id)

3.6. Courier

Relational Model: Courier(u_id, v_id,b_id)

Primary Key: u_id

Foreign Key: u_id references Employee(u_id)

Foreign Key: v_id references Vehicle(v_id)

Foreign Key: b_id references Branch(b_id)

3.7. Customer

Relational Model: Customer(u_id, city, street, apt_number, state, zip)

Primary Key: u_id

Foreign Key: u_id references User(u_id)

3.8. Corporate

Relational Model: Corporate(u_id, budget)

Primary Key: u_id

Foreign Key: u_id references Customer(u_id)

3.9. Individual

Relational Model: Individual(u_id, point)

Primary Key: u_id

Foreign Key: u_id references Customer(u_id)

3.10. Customer Card

Relational Model: CustomerCard(u_id, c_id, money)

Primary Key: u_id, c_id

Foreign Key: u_id references Individual(u_id)

3.11. Order

Relational Model: Order(o_id, take_indv_id, send_corporate_id, send_individual_id, price, rate,destination_b_id,send_b_id)

Primary Key: o_id

Foreign Key: take_indv_id references Individual(u_id)

Foreign Key: send_individual_id references Individual(u_id)

Foreign Key: send_corporate_id references Corporate(u_id)

Foreign Key: send_b_id references Branch(b_id)

Foreign Key: destination_b_id references Branch(b_id)

3.12. Package

Relational Model: Package(p_id, o_id, weight, volume,item_dscrptn)

Primary Key: p_id

Foreign Key: o_id references Order(o_id)

3.13. Malformed Delivered Package

Relational Model: MalformedDeliveredPackage(p_id, report_id)

Primary Key: p_id

Foreign Key: p_id references Package(p_id)

Foreign Key: report_id references Report(r_id)

3.14. Lost Packages

Relational Model: LostPackages(p_id, report_id)

Primary Key: p_id

Foreign Key: p_id references Package(p_id)

Foreign Key: report_id references Report(r_id)

3.15. Securely Delivered Packages

Relational Model: SecurelyDeliveredPackages(p_id)

Primary Key: p_id

Foreign Key: p_id references Package(p_id)

3.16. Holdout Packages

Relational Model: HoldoutPackages(p_id)

Primary Key: p_id

Foreign Key: p_id references Package(p_id)

3.17. Package State

Relational Model: PackageState(ps_id, state_date, name)

Primary Key: ps_id

3.18. Report

Relational Model: Report(r_id, u_id, p_id, explanation,result, date)

Primary Key: r_id

Foreign Key: u_id references Individual(u_id)

Foreign Key: u_id references Package(p_id)

3.19. Vehicle

Relational Model: Vehicle(v_id, capacity)

Primary Key: v_id

3.20. Branch

Relational Model: Branch(b_id, address, name, employee_num)

Primary Key: b_id

3.21. Pac_State

Relational Model: Pac_State(ps_id,p_id,v_id)

Primary Key: ps_id, p_id, v_id

Foreign Key: v_id referencesVehicle(v_id)

Foreign Key: ps_id referencesPackageState(ps_id)

Foreign Key: p_id references Package(p_id)

3.22. Route

Relational Model: Route(destination_b_id, departure_b_id, v_id, distance)

Primary Key: destination_b_id, departure_b_id , v_id

Foreign Key: destination_b_id references Branch(b_id)

Foreign Key: departure_b_id references Branch(b_id)

Foreign Key: v_id referencesVehicle(v_id)

4. Implementation Details

4.1. Languages

In this project, four different languages have been used in order to implement the system efficiently. The first language that has been used is PostgreSQL to implement and modify database tables. The second language that has been used is Node.js. It is used to write queries and handle the database and server part of the project. The database connection is written in Node.js in JavaScript. The third language used is React.js which is a JavaScript framework for front-end and user interfaces. Moreover, to put out a well-done design, CSS, a style sheet language, has been used as the fourth language. By combining CSS and React.js, the front-end has been implemented. Most of the project is implemented in React.js and PostgreSQL. As implementation tools we have used VisualStudio Code IDE and IntelliJ IDE.

4.2. Database and Connection

PostgreSQL is the database management system that has been used for database definition and modification. The database is located in Google Cloud servers. By using Google Cloud, simultaneous access and modifications by various group members were enabled. The database connection was created in Node.js.

4.3. Implementation

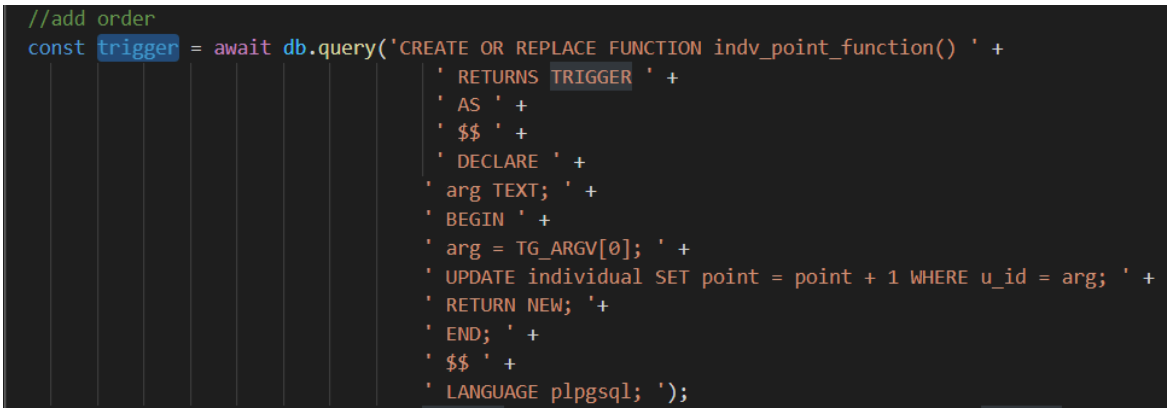
In the design and implementation stage, we have used several platforms. Firstly, the E/R diagram and schema of the database was designed. To design the E/R diagram, we have used the GitMind tool. Then, the database was created in Google Cloud by using PostgreSQL. The tables were created at this stage. User

interface pages of the project were designed based on database tables and different users' data needs. The front-end was designed by using React.js and CSS and the database manipulation parts for users were implemented. Having the UI implemented, database connection by user interactions were ready to implement. Then, the UI and database connections were written. It was the most challenging part of the project implementation. Each component of React.js that has database connections like text fields or buttons are connected at this stage. Each of these connections has their specific SQL queries written in server.js file. Each connection has been tested many times by all of the group members and debugging has taken a long time in the process.

5. Advance Database Features

- **Trigger**

  When the user submits a package by either calling a courier or taking it to a branch in person, the user earns points, which is a kind of encouragement of using Hermes. Each time a package is inserted into the package table, the point attribute on the individual table needs to be increased by one. Therefore, using triggers is a wise choice in that case because a trigger is automatically invoked and executed in a particular table in response to certain events such as insert or update. As it can be seen from the code segments below, when an insert operation is done on the package table, a procedure which updates the point column of the individual table is called.



```
//add order
const trigger = await db.query('CREATE OR REPLACE FUNCTION indv_point_function() ' +
                               ' RETURNS TRIGGER ' +
                               ' AS ' +
                               ' $$ ' +
                               ' DECLARE ' +
                              ' arg TEXT; ' +
                              ' BEGIN ' +
                              ' arg = TG_ARGV[0]; ' +
                              ' UPDATE individual SET point = point + 1 WHERE u_id = arg; ' +
                              ' RETURN NEW; '+
                              ' END; ' +
                              ' $$ ' +
                              ' LANGUAGE plpgsql; ');
```

```
const trg = await db.query(' DROP TRIGGER IF EXISTS indv_point ON package; CREATE TRIGGER indv_point ' +
                                                                    ' AFTER INSERT ' +
                                                                    ' ON package ' +
                                                                    ' FOR EACH ROW ' +
                                                                    ' EXECUTE PROCEDURE indv_point_function("' + userid + '")');
```

- **View**

  In order to show all employees with their titles in the home page of administrator, views were needed. Since the titles of the employees are not stored as a column in the database, employee data and title was stored in a view by examining each employee type's database tables. Since views are virtual tables based on a set of SQL queries, adding a new title column and storing different types of employee data separately were achieved thanks to views. The query for the need explained above can be seen from the code segments below. Also, some other view queries were implemented directly in the database, not in the server.js. They can also be seen below.

```
('CREATE OR REPLACE VIEW couriertemp AS SELECT u_id, name, phone, salary, \'COURIER\' AS type FROM courier; ' +
'CREATE OR REPLACE VIEW shippertemp AS SELECT u_id, name, phone, salary, \'SHIPPER\' AS type FROM shipper; ' +
'CREATE OR REPLACE VIEW packageManagertemp AS SELECT u_id, name, phone, salary, \'PACKAGE_MANAGER\' AS type FROM packagemanager; ' +
'SELECT EMP.u_id, EMP.name, EMP.phone, EMP.salary, EMP.type FROM (' +
'SELECT u_id, name, phone, salary, type FROM couriertemp UNION ' +
'SELECT u_id, name, phone, salary, type FROM shippertemp UNION ' +
'SELECT u_id, name, phone, salary, type FROM packageManagertemp) AS EMP ORDER BY EMP.type');
```

View creation by using desired attributes with the specified join operation of specified tables.

```
CREATE VIEW table1_view AS
SELECT ps_id, pac_state.p_id AS p_id, v_id,o_id,weight,item_dscrptn,volume
FROM pac_state LEFT OUTER JOIN package ON package.p_id = pac_state.p_id
```

```
CREATE VIEW table2_view AS
SELECT table1_view.ps_id AS ps_id,p_id,v_id,o_id,weight,item_dscrptn,volume,"name",state_date
FROM table1_view LEFT OUTER JOIN packagestate ON  table1_view.ps_id = packagestate.ps_id ORDER BY ps_id
```

```
CREATE VIEW search_package_manager
SELECT ps_id,p_id,v_id,table2_view.o_id AS o_id,weight,item_dscrptn,volume,"name",state_date,take_indv_id,send_corporate_id,price,
      rate,send_individual_id,destination_b_id,send_b_id
FROM table2_view  LEFT OUTER JOIN "Order" ON  "Order".o_id = table2_view.o_id
```

6. User Manual

6.1. Sign In

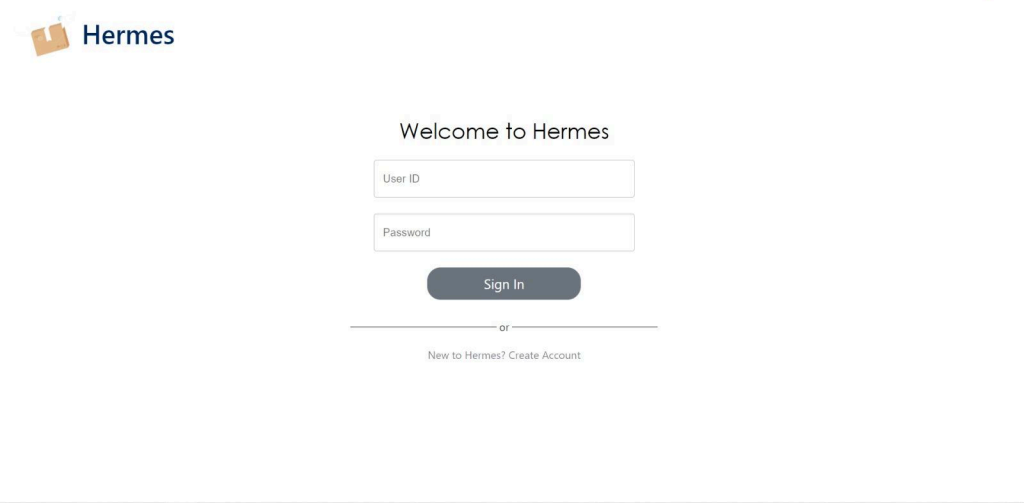


Figure 1

Explanation: In sign in page, users will sign in to Hermes by entering using their user ids and passwords. When a user hits the demonstrated sign in button, the system checks the entered information and directs the user to the homepage if the entered information exists in the system, and directs the user to this page again if not. If the data entered is not correct, an alert message is shown according to the wrong data.If the entered data is correct, the homepage will be displayed according to the type of the user.

6.2. Registration

6.2.1. Registration as Customer

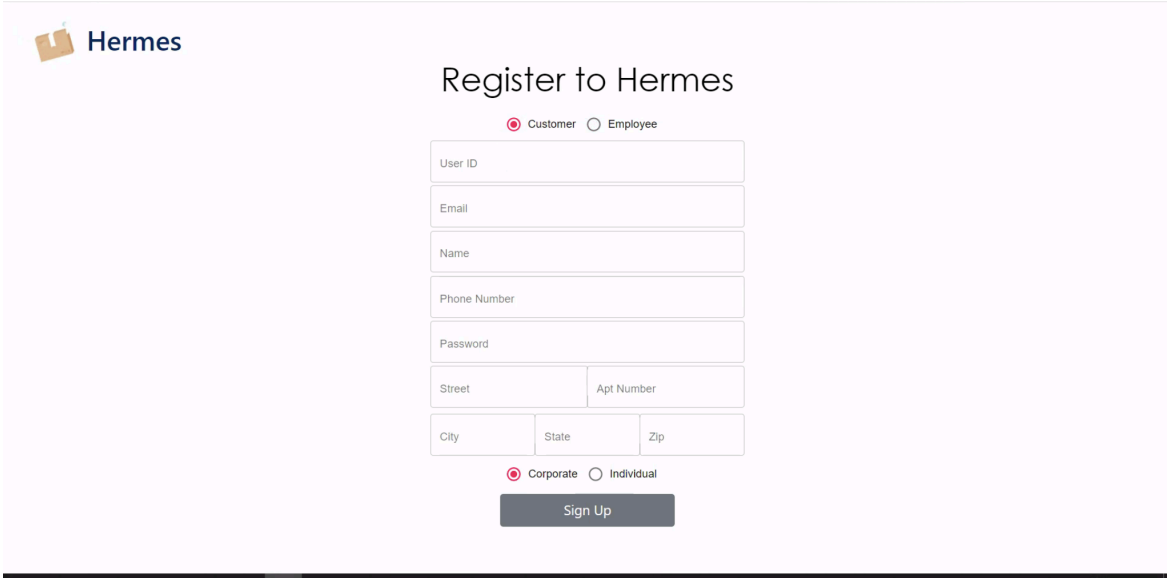


Figure 2

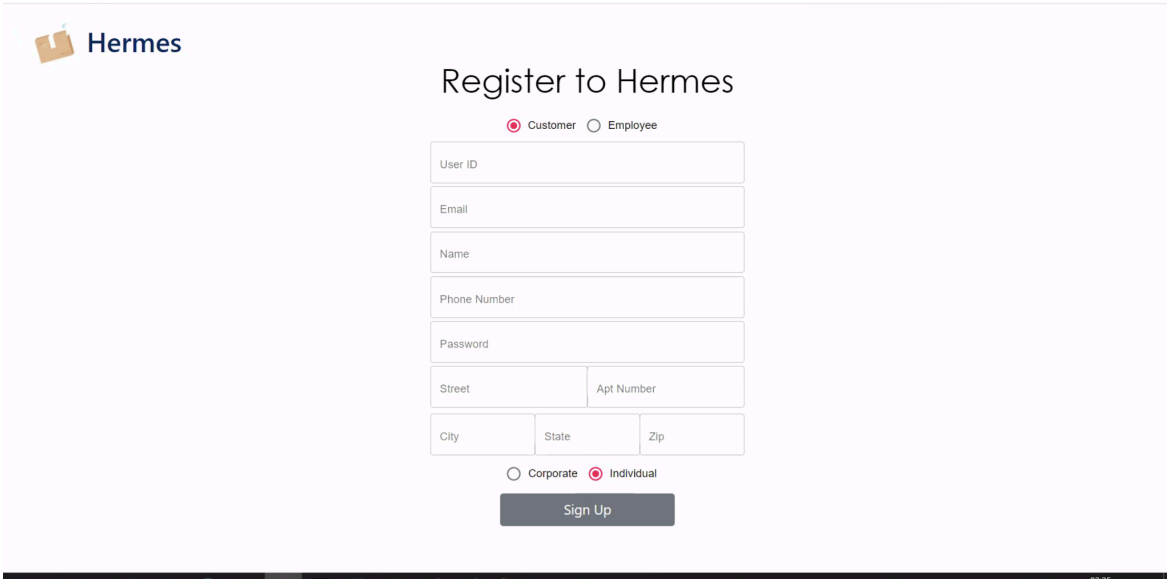


Figure 3

Explanation: Customers (both corporate and individual customers) will register to Hermes by entering their information. These information are respectively: user id that they will use when sign in to Hermes, email, name, password, phone number and their address information. When the Sign Up button is hitted, the entered data is inserted to the database. Besides that, they have to choose what kind of customer they are: corporate or individual since in the database, the data is divided accordingly. For each data, if the entered data is missing or an already existing data such as user id, an error alert is shown to the user.

6.2.2. Registration as Employee

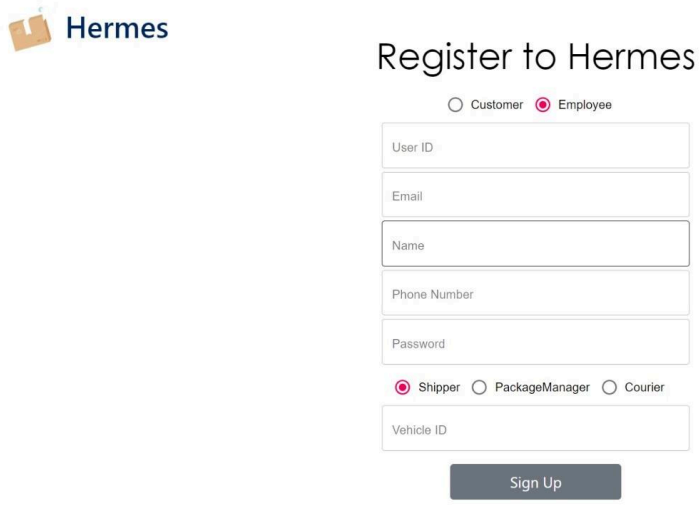


Figure 4

Explanation: Employees will register to Hermes by entering their information. These are: user id that they will use when sign in to Hermes, email, name, password and phone number. Besides that they have to choose what kind of employee they are: shipper, package manager or courier .

6.2.2.1. Register as PackageManager

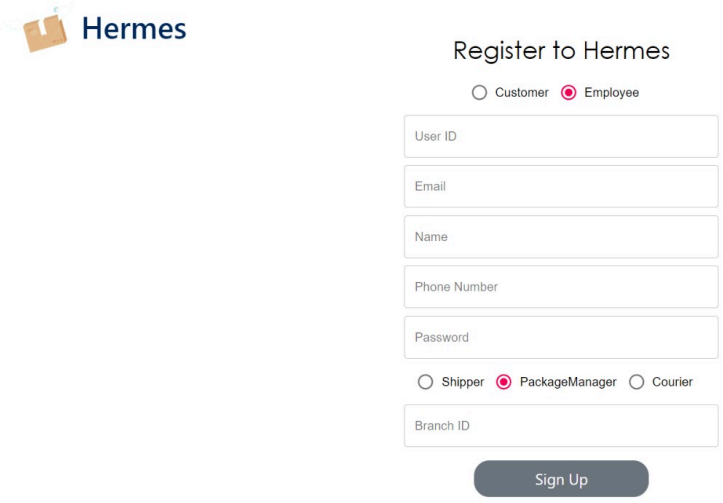


Figure 5

Explanation: According to their employee type they will enter dependent information. If the employee is a package manager, then branch id data will also be taken from the user.

6.2.2.2. Register as Shipper or Courier

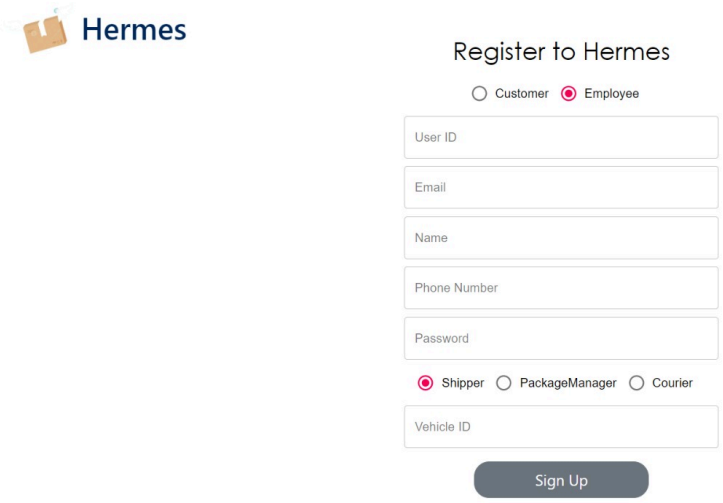


Figure 6

Explanation: According to their employee type they will enter dependent information. If the employee is a courier or a shipper then branch id data will also be taken from the user.

6.3. Individual Customer

6.3.1. Individual Customer Home Page

Figure 7

Explanation: At the main page of the customer, the submission type of package will be selected. There are two options: Customers can either call a courier to her/his/its own location or submit a package in person. According to selection, if customers choose to call the courier option, they will be directed to the page shown in ***Figure 8 .*** If customers choose to submit the package in person option, they will be directed to the page shown in ***Figure 9.***

6.3.2. Individual Customer Call Courier

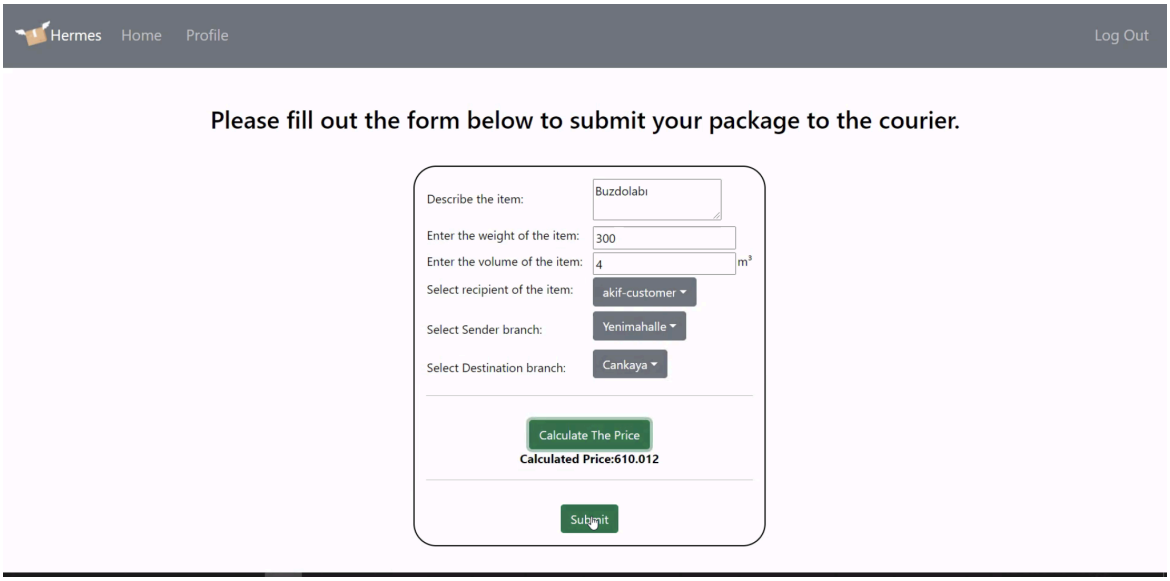


Figure 8

Explanation: Customers can submit packages to couriers by using this page. They will enter the necessary information of the package such as description of the package, weight, volume. They can select a recipient among a list of possible customers from the menu. They will also choose a sender and destination branch from the list of branches that are shown in menus. According to this information, they will be able to see the calculated price for the package by pressing the calculate the price button. Then they will submit it to the courier by the submit button. An available courier from the selected branch will be automatically assigned by the system to take the package from the customer.

6.3.3. Individual Customer Submit In Branch

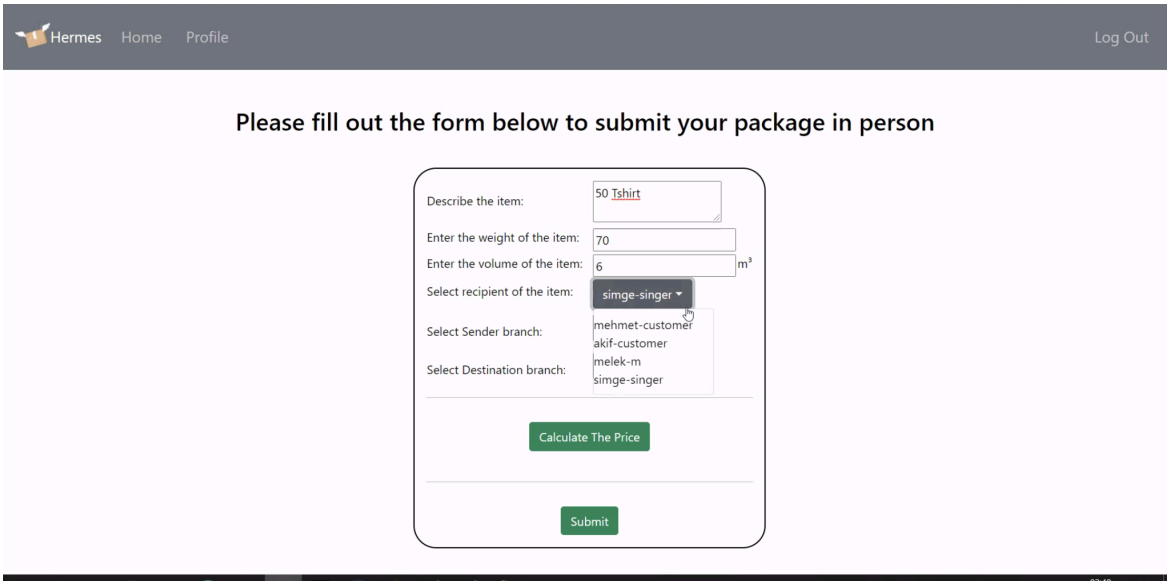


Figure 9

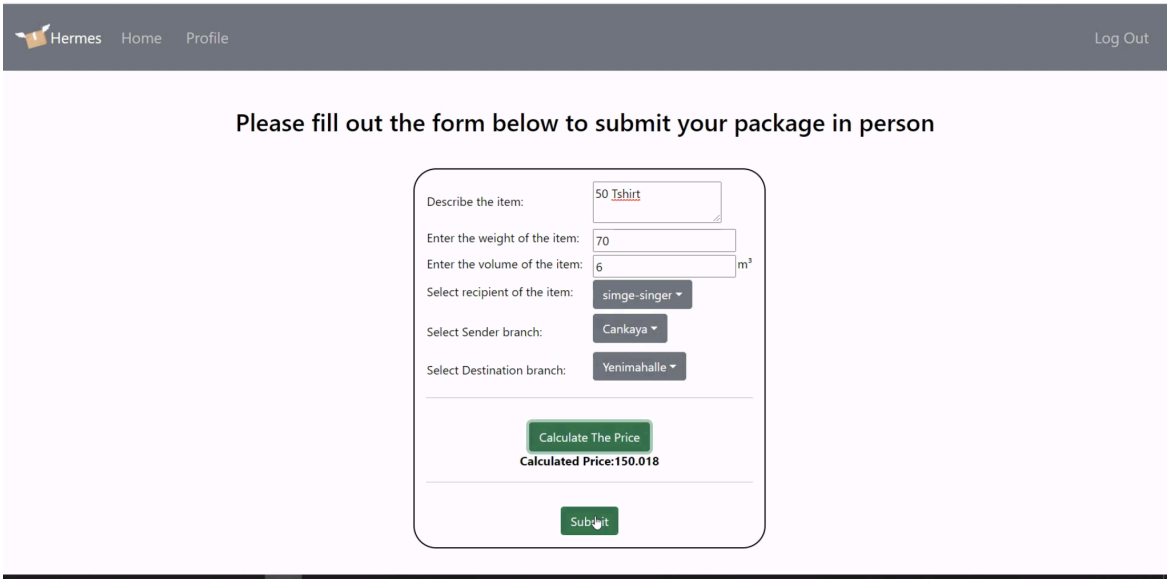


Figure 10

Explanation: Customers can submit packages in person by using this page. They will enter the necessary information of the package such as description of the package, weight, volume. They can select a recipient among a list of possible customers. They also need to select the closest

branch to them and the destination branch. They will select it from the menu next to it which lists all the branches. Then they are able to calculate the price of the package by the calculate price button and submit it to the package manager by the submit button. ***Figure 9*** demonstrates that an individual customer can select a recipient to send the package, while ***Figure 10*** demonstrates that by the same process an individual customer can select the sender and destination branch and when the user hits the submit button the package is submitted and waits for the approval of the package manager.

6.3.4. Individual Customer Profile

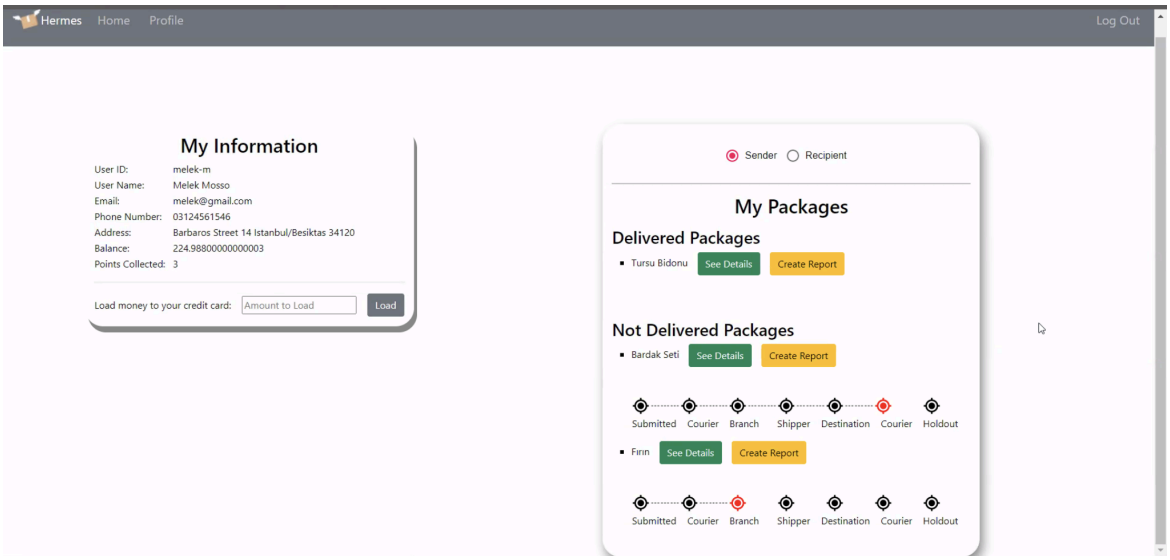


Figure 11

Explanation: As demonstrated in ***Figure 11***, individual customers can see their personal information in their profile pages, and they are able to load money to their card by using the load button. With every package submission in ***Figure 10*****,** users collect one point and the collected points for each individual customer are demonstrated in the "My Information" part in ***Figure 11***. They can see the list of delivered and not delivered packages in the case that they are the senders or recipients of the packages as demonstrated in ***Figure 11*** and ***Figure 12***. Besides that, they can see the status of not delivered packages (again for both of the cases which are sender and recipient) such as, on branch, on courier, etc in ***Figure 11*** and ***Figure12***.

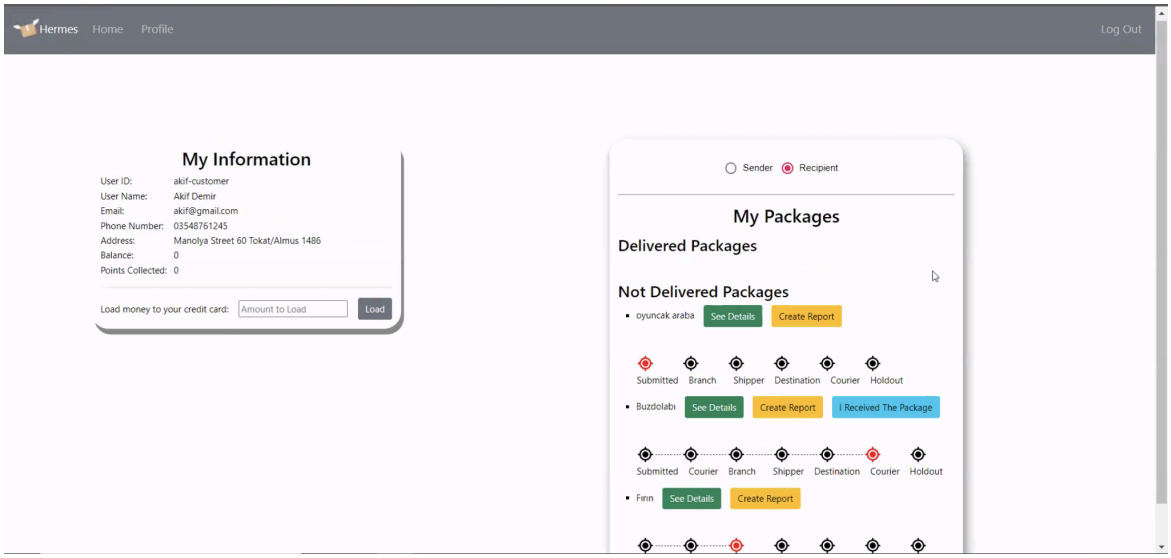


Figure 12

They will use the button “I Received The Package” as demonstrated in ***Figure 12***, if they successfully take the package. This button will not be displayed until the courier takes the package in order to send it to the customer.

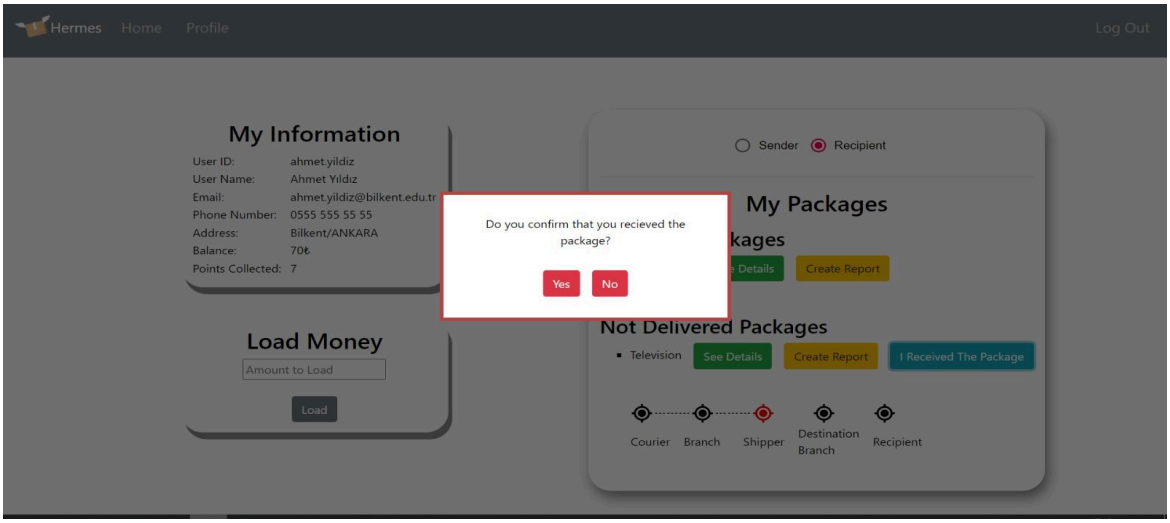


Figure 13

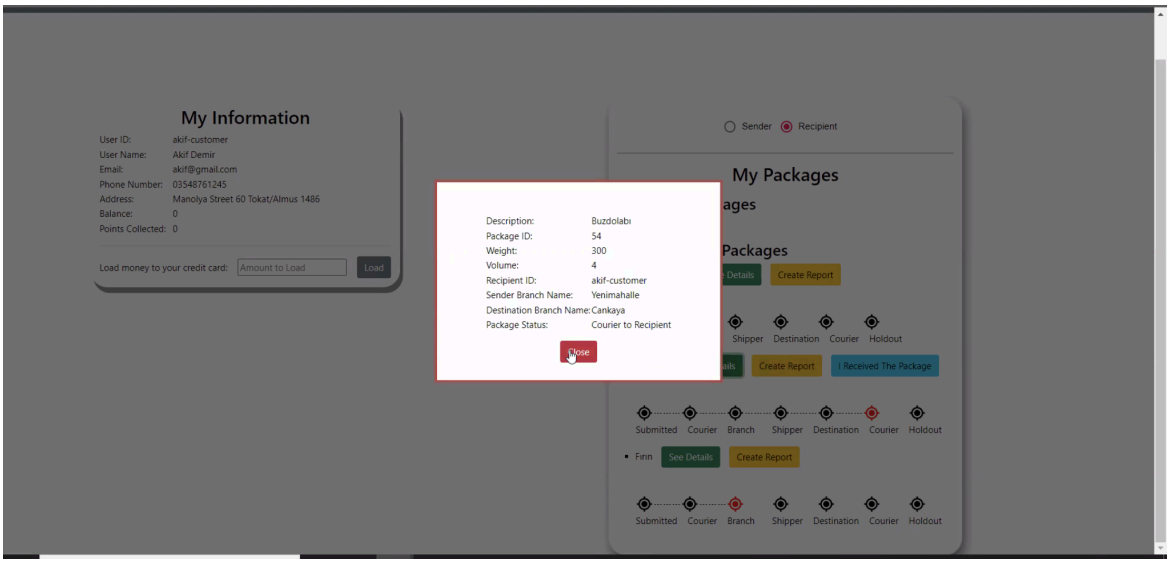


Figure 14

Also, they can see details (the data of the package) of the packages by clicking the "See Details" button demonstrated in ***Figure 11***. When they hit the "See Details" button, a pop up with the data of the desired order comes up as indicated in ***Figure 14***.

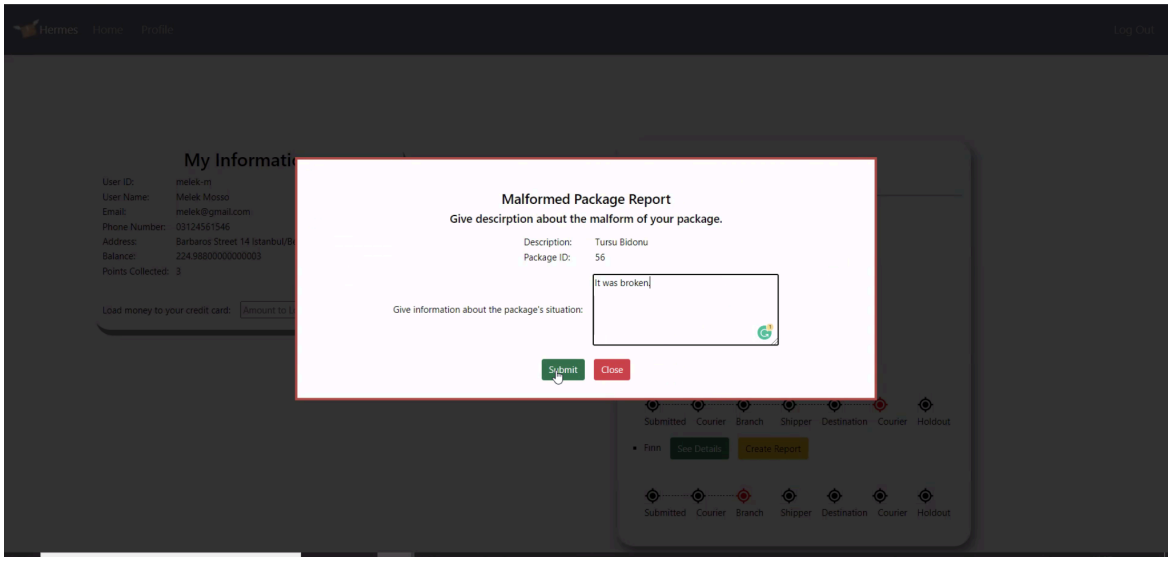


Figure 15

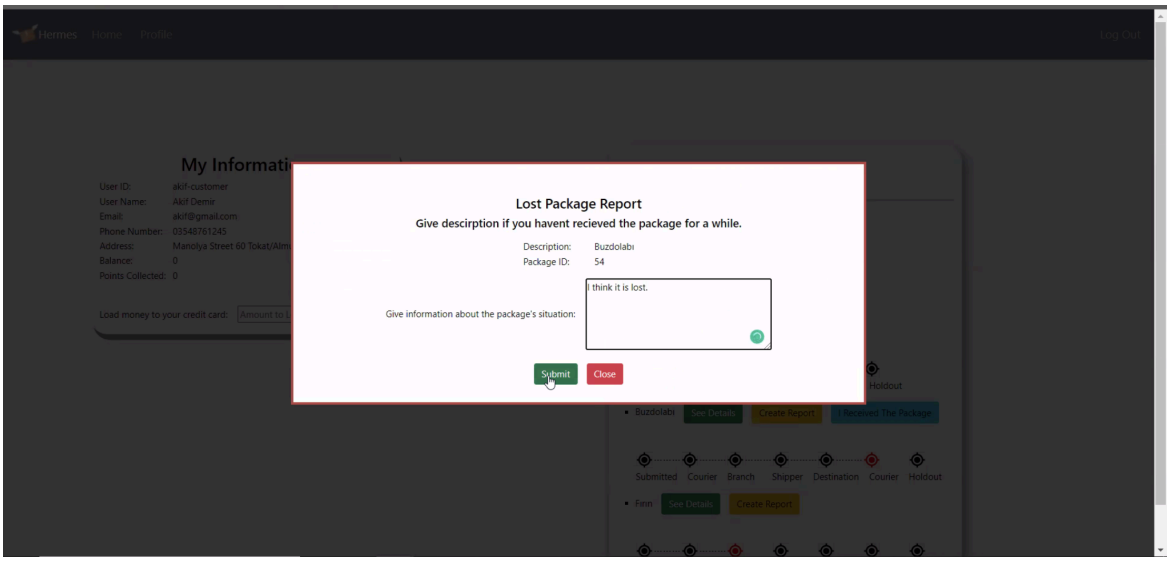


Figure 16

They also can create malformed reports for delivered packages as in ***Figure 15.*** In addition to the malformed reports, they can also create lost reports for undelivered packages as in ***Figure 16.*** When these reports are submitted to the system, the new status of the package will be a malformed report or lost report accordingly, and if these reports are accepted by the package manager (this process will be mentioned), the status of these packages change as lost or malformed. As it can be observed in ***Figure 17*** and ***Figure 18*** the status of the package changes accordingly.

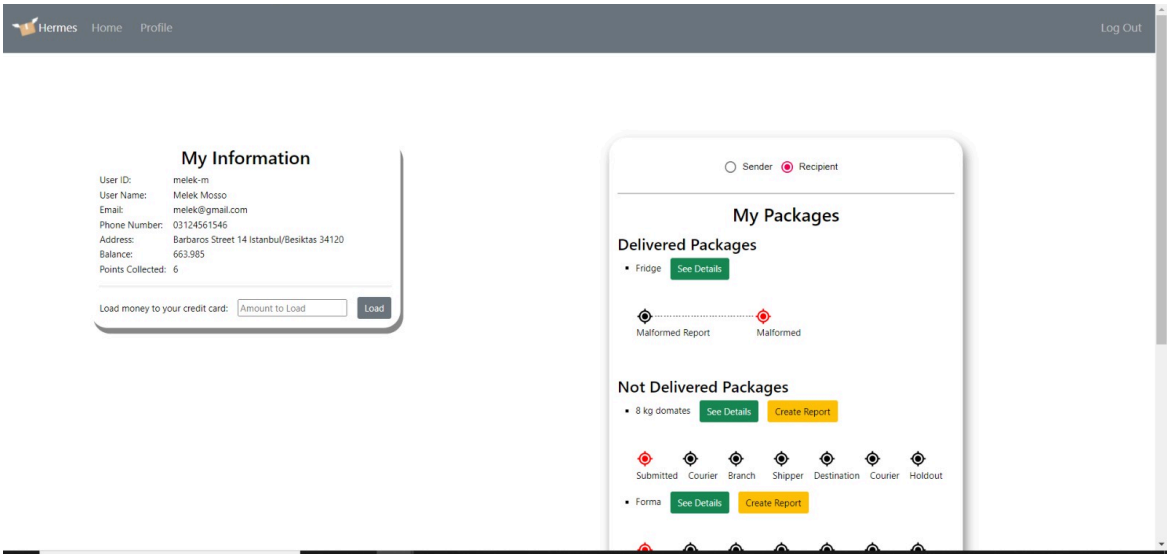


Figure 17

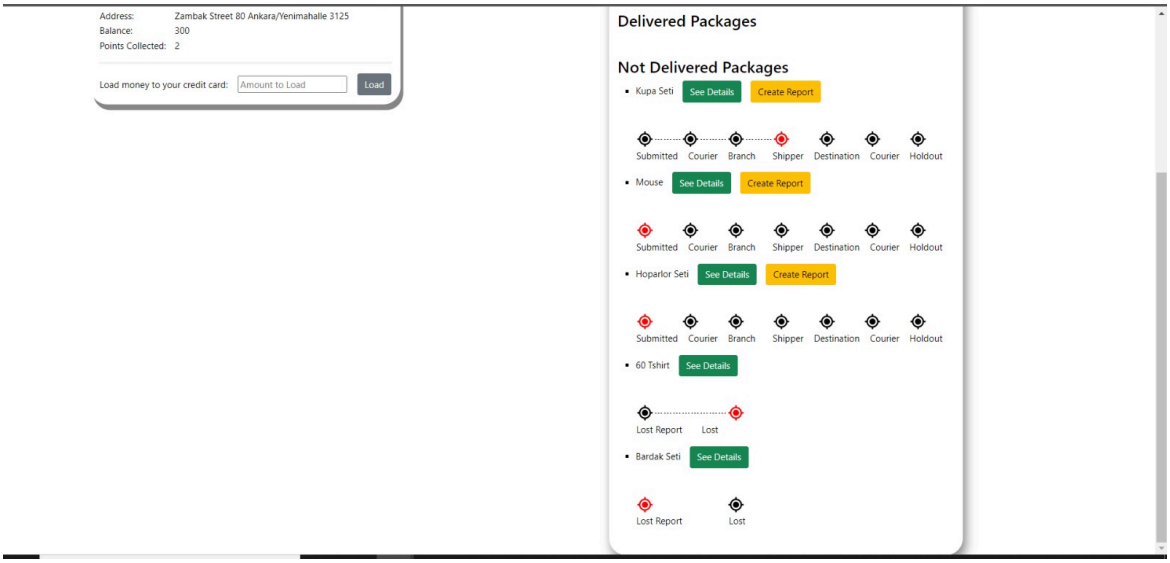


Figure 18

6.4. Corporate Customer

6.4.1. Corporate Customer Home Page

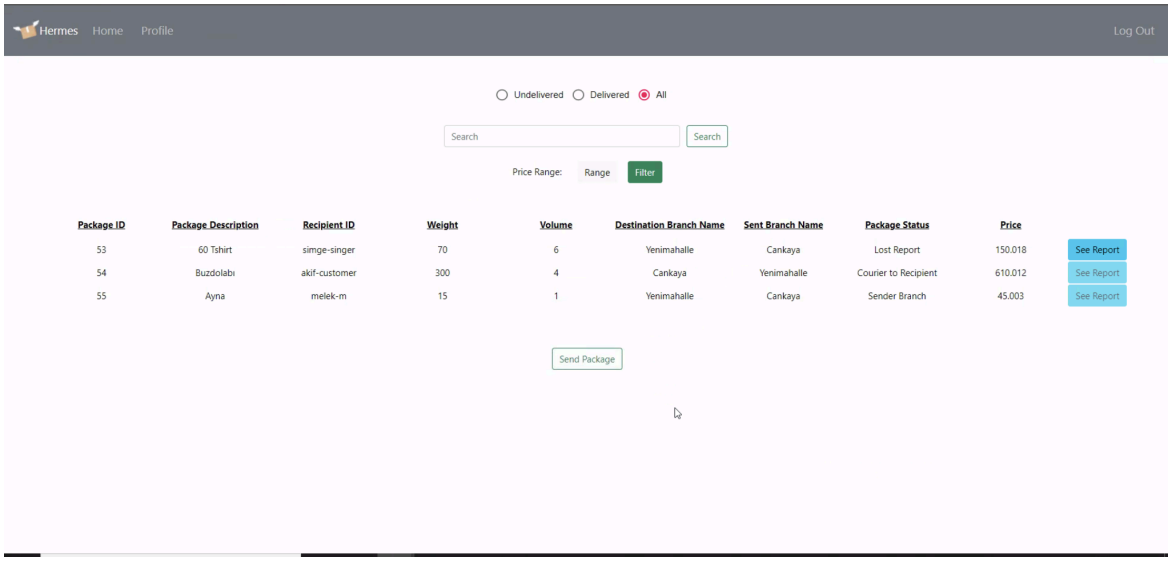


Figure 19

Explanation: In addition to individual customers, corporate customers can also see the packages that they send in their homepage as indicated in ***Figure 19***. If a report exists for a package that a corporate customer sent, corporate customers can see a button to see the details of the report as shown in ***Figure 19***. However, if a report does not exist for the indicated package, then the "See Report" button is disabled, and thus does not have any functionality.

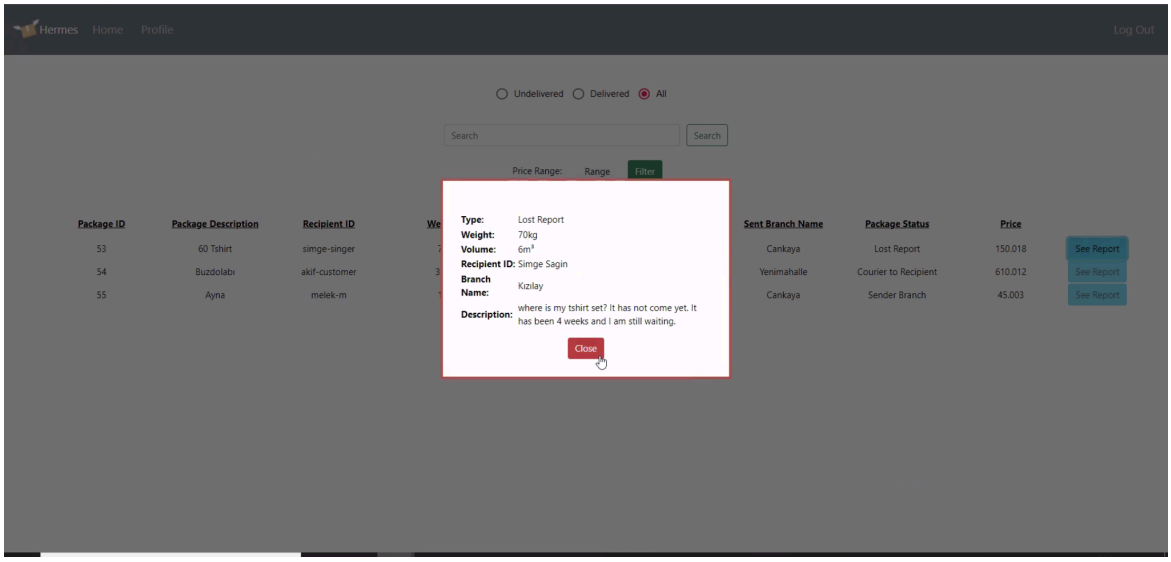


Figure 20

When the “See Report” button is clicked the data of the report is shown as in ***Figure 20***. Also, corporate customers can see the packages according to the status of the packages by selecting a status with the indicated radio buttons in ***Figure 21***.

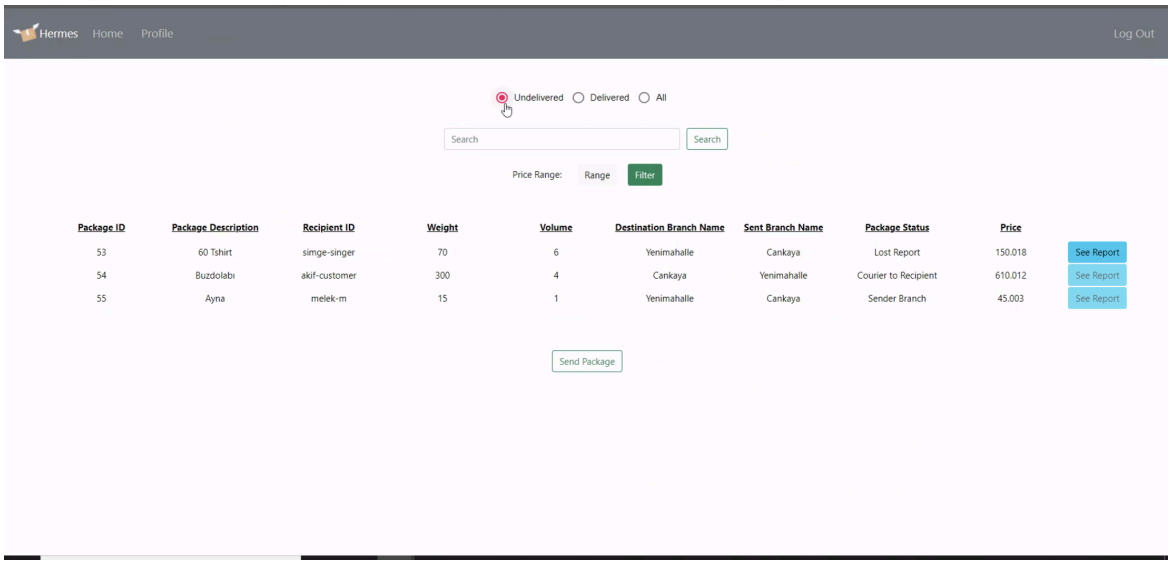


Figure 21

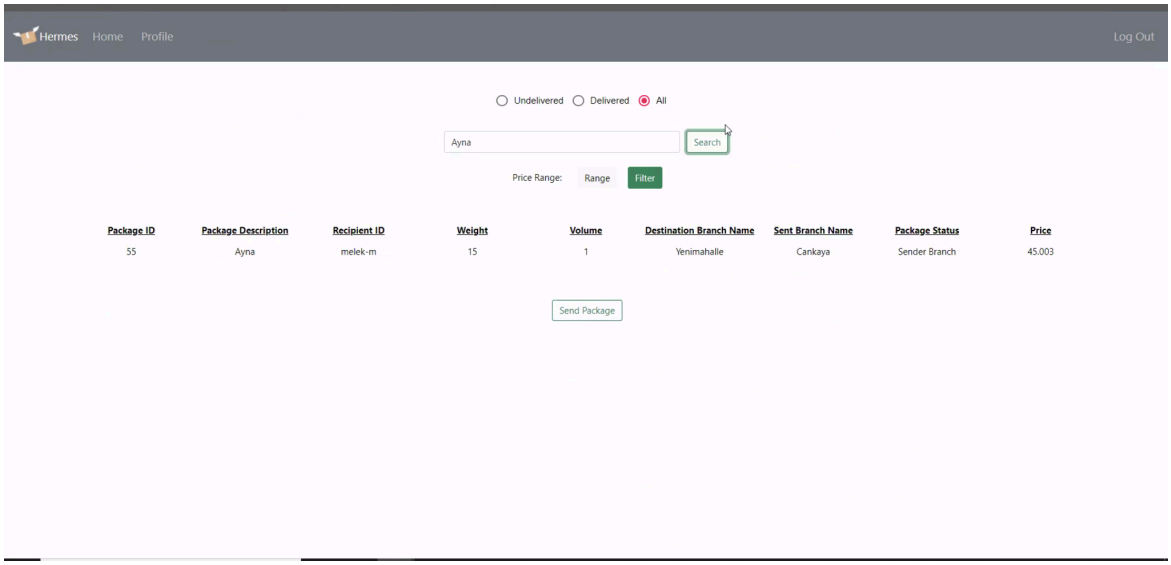


Figure 22

Lastly, corporate customers can search through the packages that they sent by using the entered keyword from the users, and they also can specify a price range to search through the packages by filtering as in ***Figure 22***.

6.4.2. Corporate Customer Profile

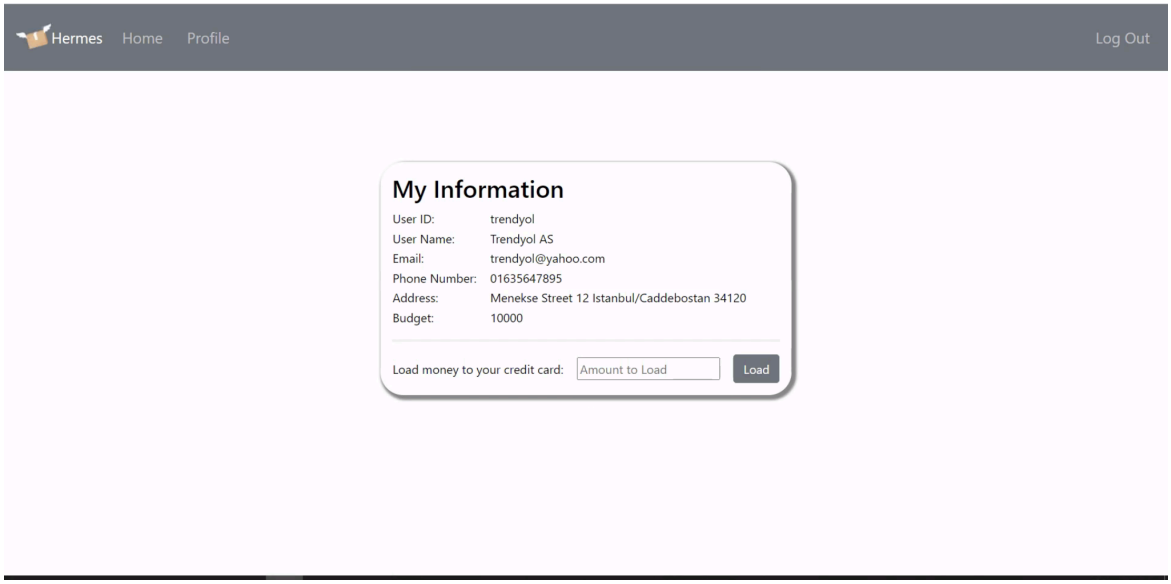


Figure 23

Explanation: ***Figure 23*** indicates the profiles of the corporate customers. In this page, there is only the data of the corporate customers along with the load money feature. In this page, corporate customers can load any amount of money to their accounts.

6.5. Shipper

6.5.1. Shipper Home Page

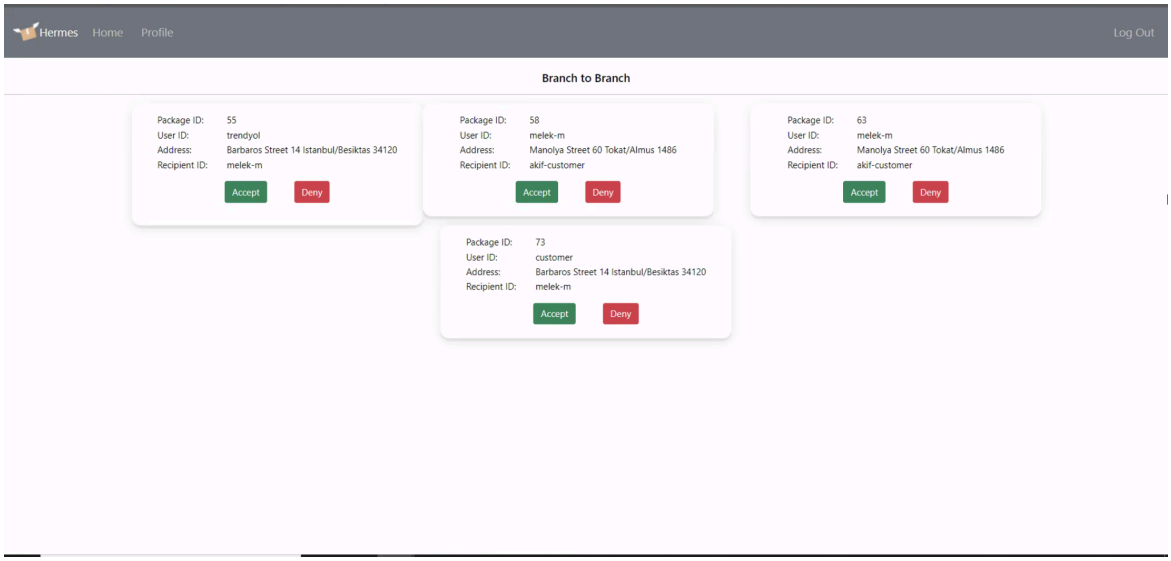


Figure 24

Explanation: ***Figure 24*** indicates the home page of the shippers. In this page, there are requests done by package managers. Shippers transports the packages from branches to branches; therefore, only "Branch to Branch" requests are visible in Shipper Home Page. As shown in ***Figure 24***, shippers can both accept or deny the requests. When they accept the requests, the package status in ***Figure 10*** is updated as "Shipper", whereas if they deny the request, then the money that is taken from the customer is deposited again to the account of that customer.

6.5.2. Shipper Profile

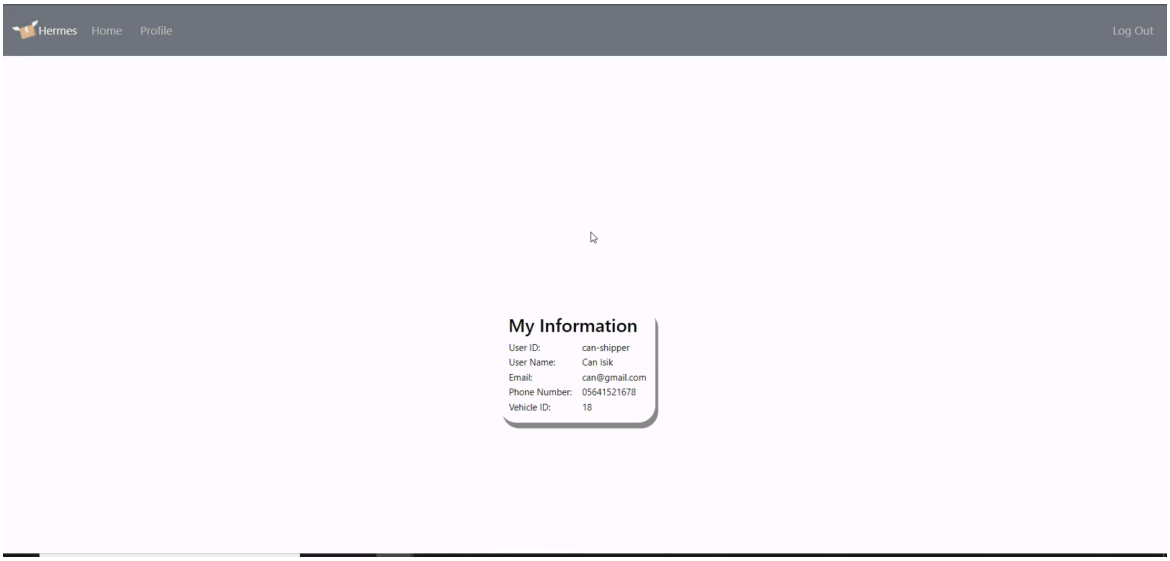


Figure 25

Explanation: ***Figure 25*** indicates the profile of the shippers. In this page, the employee information is shown including vehicle id of the user which implies the branch.

6.6. Package Manager

6.6.1. Package Manager Home Page

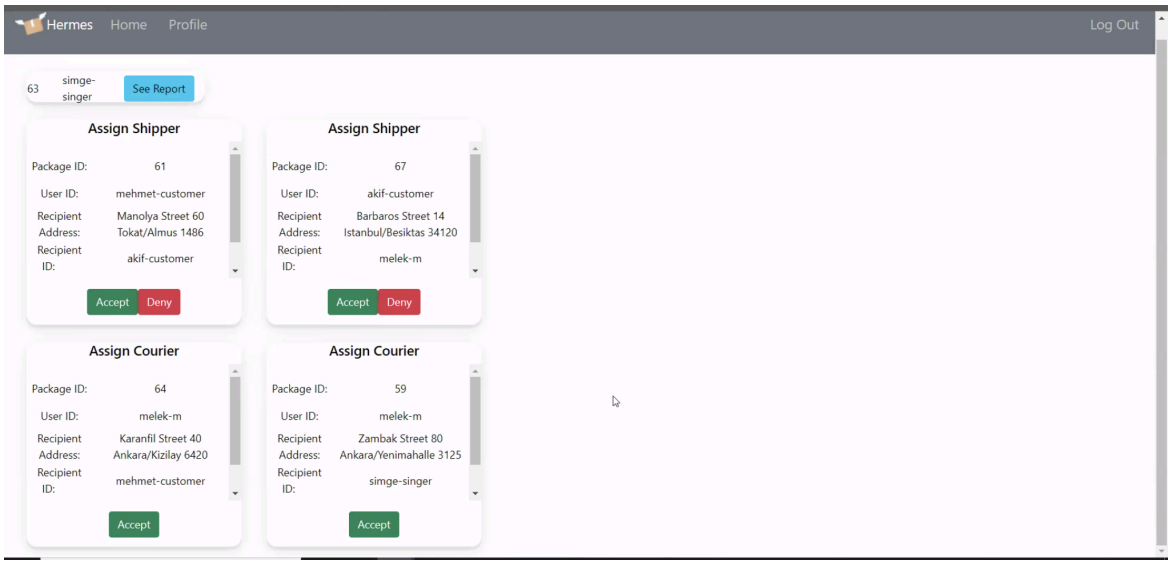


Figure 26

Explanation: Package managers can assign shippers to the packages that come to the branch by a courier from a menu where couriers of the branch are listed as indicated in ***Figure 27***. They should also choose the shipper who is currently in this branch and press accept as demonstrated in ***Figure 28***. Then, the shipper will be assigned. If they deny one of the assigned shipper sections, the denied package will be shown in the assigned courier section of the package manager's home page in order to send the package back to the customer via courier.

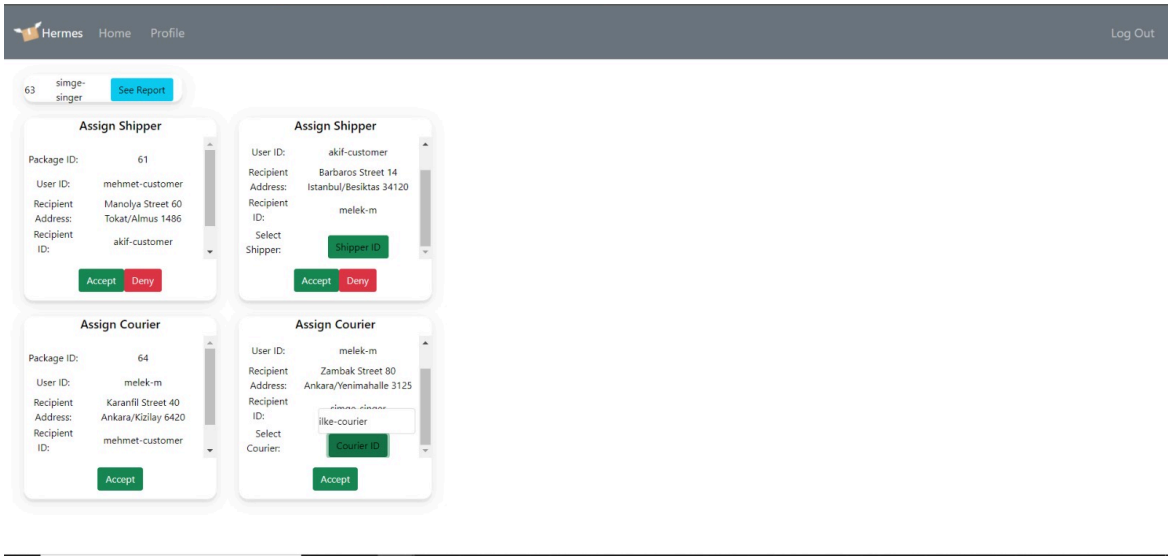


Figure 27

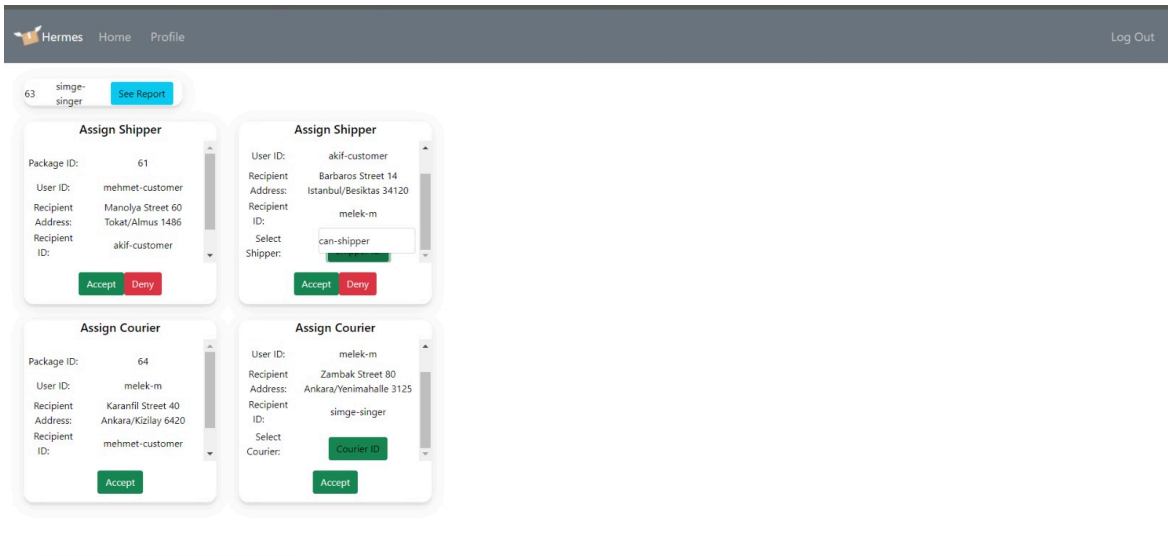


Figure 28

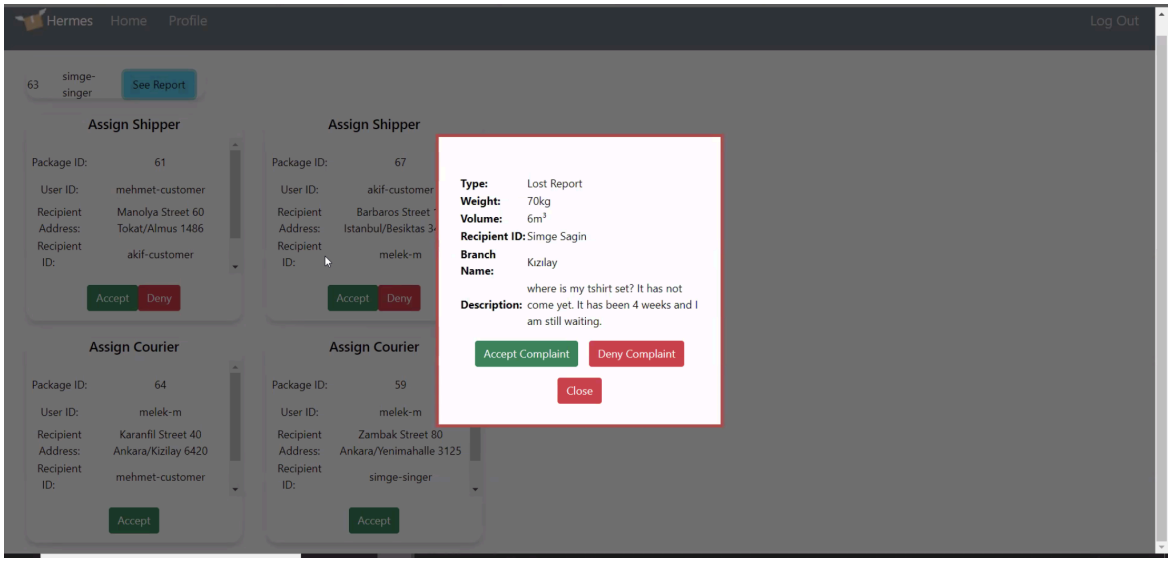


Figure 29

Besides these, package managers can see the list of reports filled by customers, and when they hit the "See Report" button in their homepages as indicated in ***Figure 27.*** They can see the data of the corresponding reports, and at the coming pop up as shown in ***Figure 29*** and ***Figure 30***. They can both accept or deny the complaints in the reports. If the reports are accepted the package status demonstrated in ***Figure 10*** are updated as "Lost" or "Malformed" accordingly.

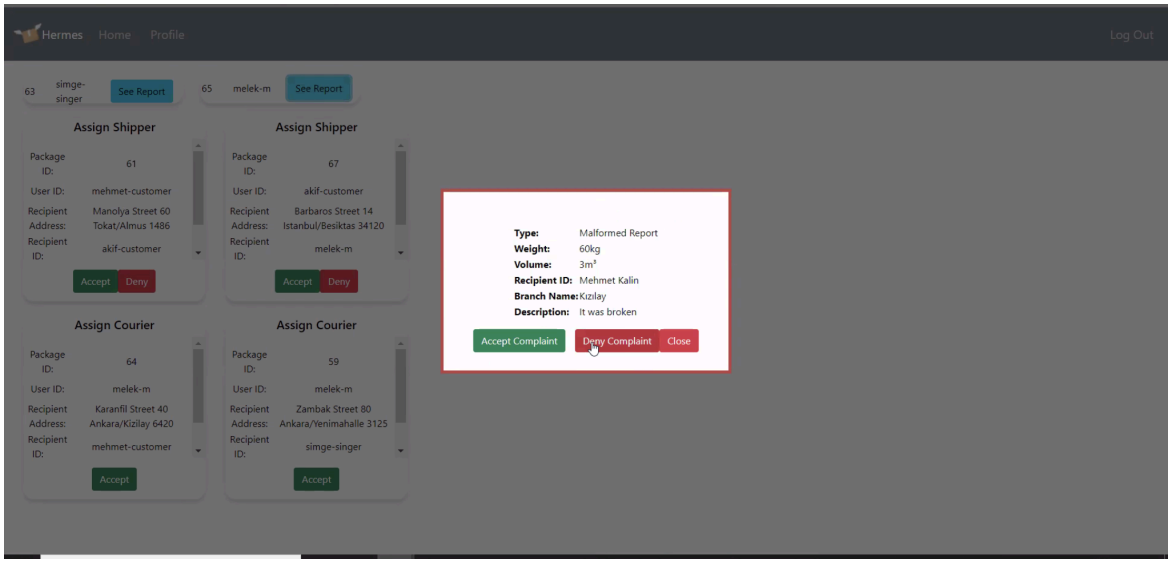


Figure 30

6.6.2. Package Manager Profile

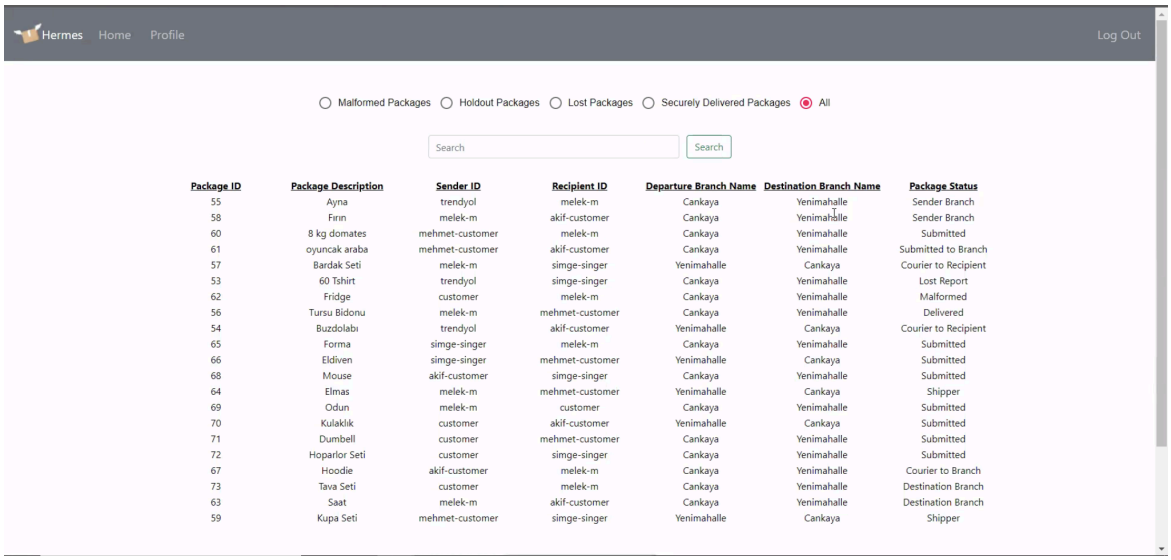


Figure 31

Explanation: Package managers can see package id, package description, sender id, recipient id, departure branch name, destination branch name, and package status of all of the packages that are associated with the branch that they work in their profile page indicated in ***Figure 31***. Also, as indicated in ***Figure 31,*** package managers can specify the package status by using radio buttons to see just the packages that are corresponding to the package status in the radio buttons.

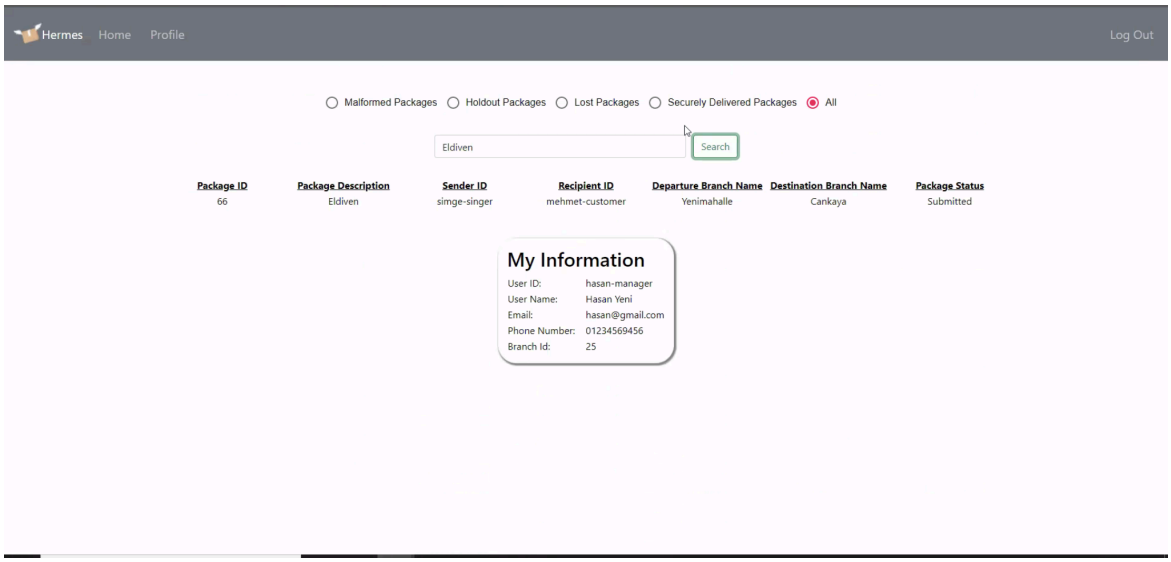


Figure 32

As can be observed from ***Figure 32***, package managers can search packages with the desired keyword by using the search button. Lastly, as can be seen

from Figure 32, each package manager has a part in their profile page that demonstrates account information of the package manager.

6.7. Courier

6.7.1. Courier Home Page

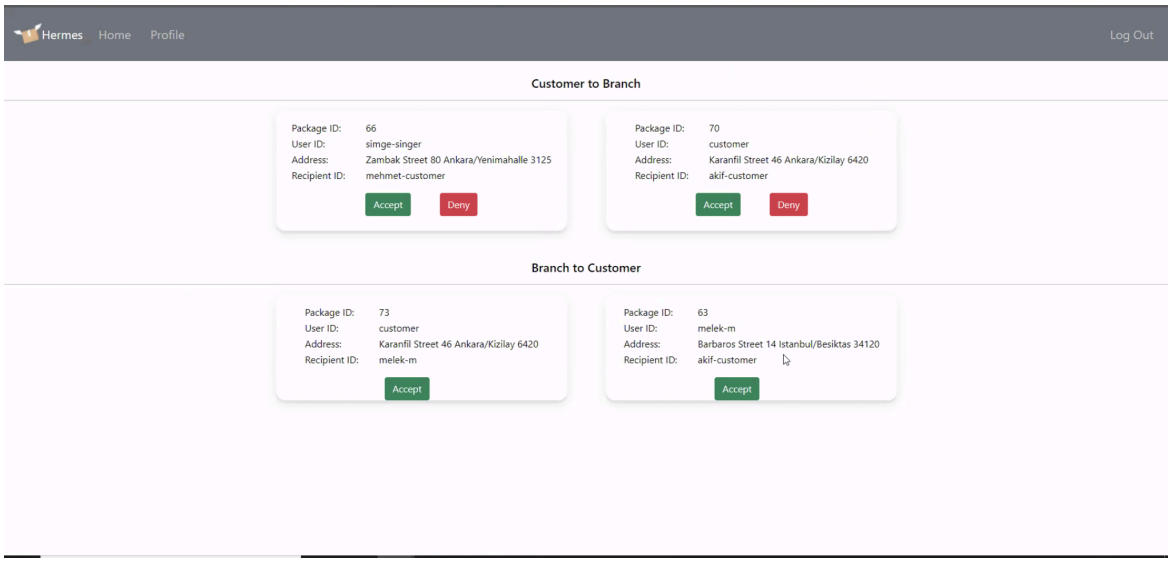


Figure 33

Explanation: Couriers can see the assigned packages that will be delivered from customer address to branch at the top of their home page under the label "Customer to Branch", while couriers also can see the assigned packages that will be delivered from branch to customer address unde the label "Branch to Customer" as indicated in **Figure 33**. They can accept or deny the packages that are under the label "Customer to Branch". If they deny those packages in the customer to branch section, the package state of the package does not change and remains submitted in the customer profile page, and a new random courrier that works for the same branch will be assigned for the package. If they accept the packages in the customer to branch section, this package will be placed in package managers' home page as a shipper assignment as in ***Figure 27***. They have to accept the package to deliver it from the branch to the customer and with each accept, the package status in the customer profile page is updated accordingly .

6.7.2. Courier Profile

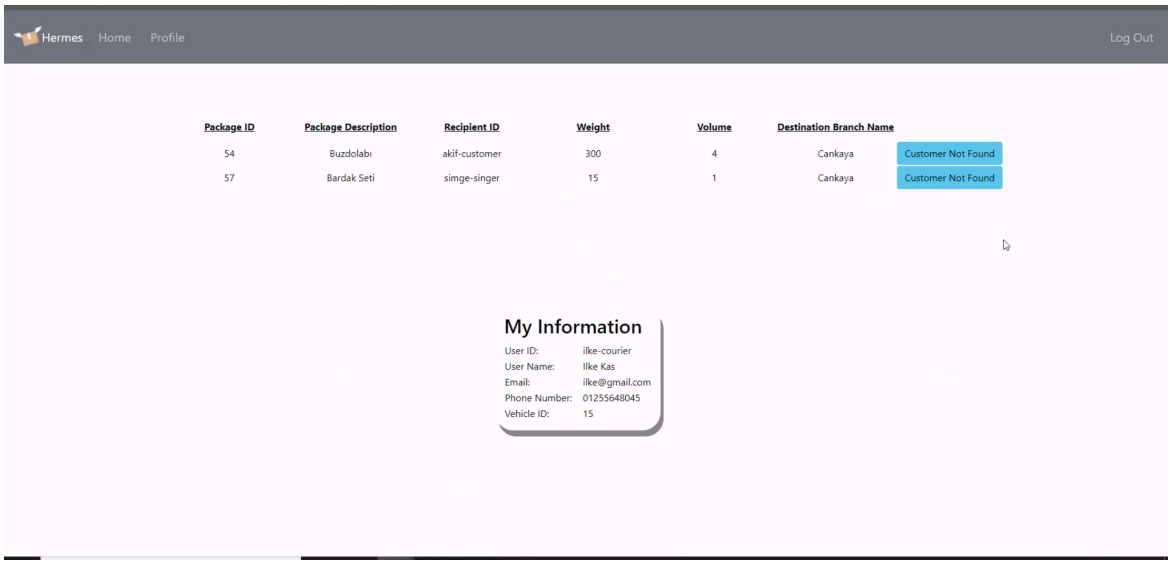


Figure 34

Explanation: Couriers can see the assigned packages that will be delivered to the customer address listed at the top of the page. If they cannot find the customer, they will press the "Customer Not Found" button and the package status will be changed to holdout. At the bottom part the courier information is shown.

6.8. Admin

6.8.1. Admin Home Page

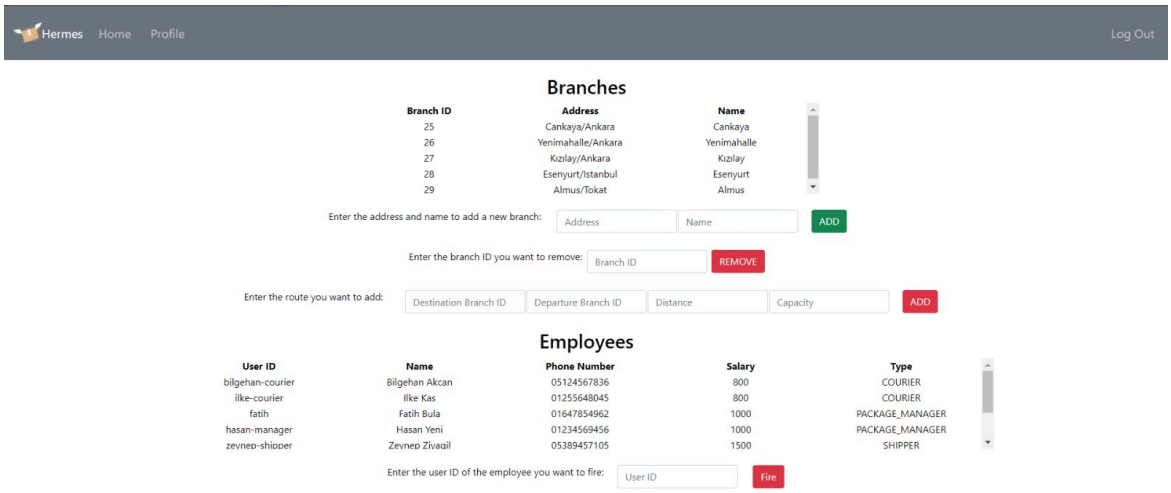


Figure 35

Explanation: Admin can see the branches and the employees of every branch as indicated in **Figure 35**. As shown in Figure 36, the admin can create a new branch, close a branch, create a route and also fire an employee from the system.

6.8.2. Admin Profile

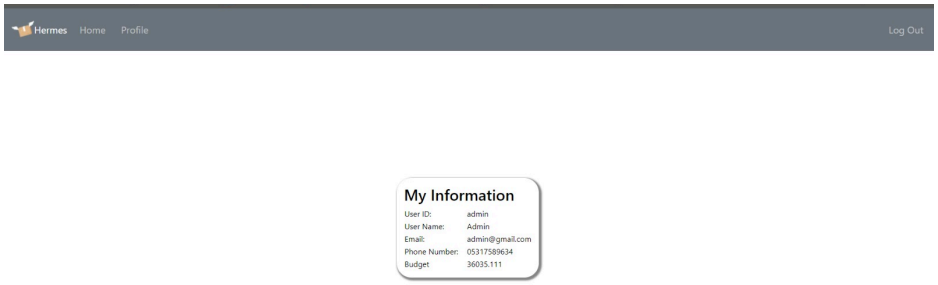


Figure 36

Explanation: Also, in the profile page of the admin, there is the account information of the admin as indicated in **Figure 36.**

7. Websites

- The link of GitHub page of the project is the following: https://github.com/zeynepziyagil/db

- The link of Reports Website is the following: https://ilke-kas.github.io/