

**BİLKENT UNIVERSITY**  
**ENGINEERING FACULTY**  
**DEPARTMENT OF COMPUTER ENGINEERING**



**Design Report**

**Group 4**

**İlke Kaş 21803184**

**Zeynep Büşra Ziyagil 21802646**

**Bilgehan Akcan 21802901**

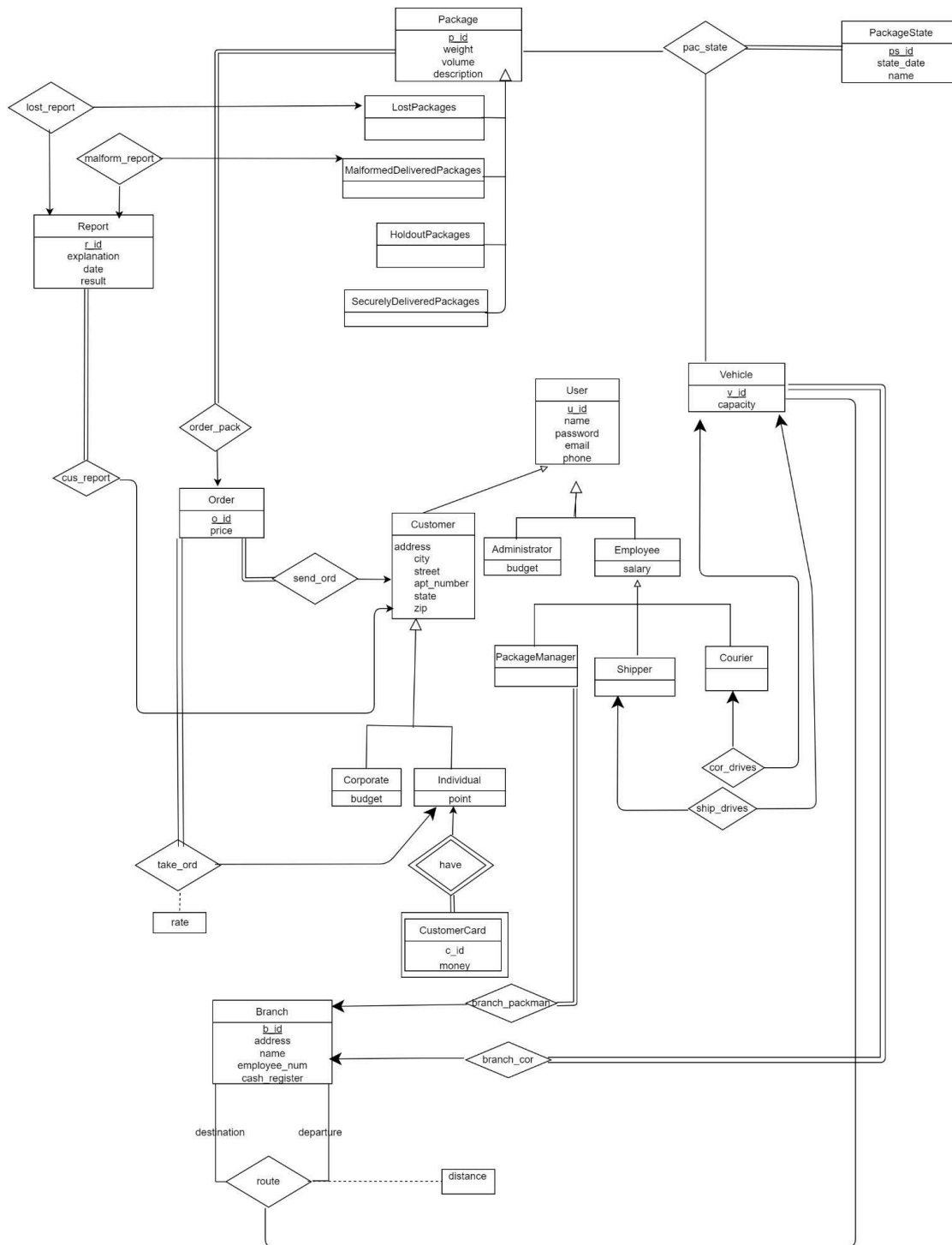
**Ömer Onat Postacı 21802349**

## **Table of Contents**

<b>Revised E/R Diagram</b>	<b>4</b>
<b>Relational Schemas and SQL</b>	<b>5</b>
<b>User</b>	<b>5</b>
<b>Administrator</b>	<b>5</b>
<b>Employee</b>	<b>5</b>
<b>Package Manager</b>	<b>6</b>
<b>Shipper</b>	<b>6</b>
<b>Courier</b>	<b>7</b>
<b>Customer</b>	<b>7</b>
<b>Corporate</b>	<b>7</b>
<b>Individual</b>	<b>8</b>
<b>Customer Card</b>	<b>8</b>
<b>Order</b>	<b>9</b>
<b>Package</b>	<b>9</b>
<b>Malformed Delivered Package</b>	<b>10</b>
<b>Lost Packages</b>	<b>10</b>
<b>Securely Delivered Packages</b>	<b>11</b>
<b>Holdout Packages</b>	<b>11</b>
<b>Transaction</b>	<b>11</b>
<b>Package State</b>	<b>12</b>
<b>Report</b>	<b>12</b>
<b>Vehicle</b>	<b>13</b>
<b>Courier Vehicle</b>	<b>13</b>
<b>Shipper Vehicle</b>	<b>14</b>
<b>Branch</b>	<b>14</b>
<b>Pac_State</b>	<b>15</b>
<b>Route</b>	<b>15</b>
<b>User Interface Design and Needed SQL Statements</b>	<b>16</b>
<b>Sign in</b>	<b>16</b>
<b>Registration</b>	<b>17</b>
<b>Registration as Customer</b>	<b>17</b>
<b>Registration as Employee</b>	<b>18</b>
<b>Register as PackageManager</b>	<b>19</b>
<b>Register as Shipper or Courier</b>	<b>20</b>

<b>Home Page</b>	<b>21</b>
<b>Customer Home Page</b>	<b>21</b>
<b>Employee Home Page</b>	<b>22</b>
<b>Package Manager Home Page</b>	<b>22</b>
<b>Courier Home Page</b>	<b>24</b>
<b>Shipper Home Page</b>	<b>25</b>
<b>Submitting Packages For Customers</b>	<b>26</b>
<b>Submit Package to Courier</b>	<b>26</b>
<b>Submit Package In Person</b>	<b>27</b>
<b>Profile Page</b>	<b>28</b>
<b>Customer Profile Page</b>	<b>28</b>
<b>Individual Profile Page</b>	<b>28</b>
<b>Corporate Profile Page</b>	<b>33</b>
<b>Employee Profile Page</b>	<b>34</b>
<b>Courier Profile Page</b>	<b>34</b>
<b>Package Manager Profile Page</b>	<b>35</b>
<b>Shipper Profile Page</b>	<b>37</b>
<b>Reports</b>	<b>37</b>
<b>File a Report Customer</b>	<b>37</b>
<b>Malformed Report</b>	<b>37</b>
<b>Lost Report</b>	<b>38</b>
<b>Implementation Plan</b>	<b>39</b>

# 1. Final E/R Diagram



## **2. Relational Schemas and SQL**

### **2.1. User**

**Relational Model:**

User(u\_id, name, password, surname, email, phone)

**Primary Key:** u\_id

### **2.2. Administrator**

**Relational Model:** Administrator(u\_id, budget)

**Primary Key:** u\_id

**Foreign Key:** u\_id references User(u\_id)

### **2.3. Employee**

**Relational Model:** Employee(u\_id, salary)

**Primary Key:** u\_id

**Foreign Key:** u\_id references User(u\_id)

### **2.4. 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 User(u\_id)

### **2.5. 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 User(u\_id)

### **2.6. Courier**

**Relational Model:** Courier(u\_id, v\_id)

**Primary Key:** u\_id

**Foreign Key:** u\_id references User(u\_id)

**Foreign Key:** v\_id references Vehicle(v\_id)

## **2.7. Customer**

**Relational Model:** Customer(u\_id, city, street, apt\_number, state, zip)

**Primary Key:** u\_id

**Foreign Key:** u\_id references User(u\_id)

## **2.8. Corporate**

**Relational Model:** Corporate(u\_id, budget)

**Primary Key:** u\_id

**Foreign Key:** u\_id references User(u\_id)

## **2.9. Individual**

**Relational Model:** Individual(u\_id, point)

**Primary Key:** u\_id

**Foreign Key:** u\_id references User(u\_id)

## **2.10. Customer Card**

**Relational Model:** CustomerCard(u\_id, c\_id, money)

**Primary Key:** u\_id, c\_id

**Foreign Key:** u\_id references User(u\_id)

**SQL Definition:**

```
CREATE TABLE CustomerCard(  
  u_id          VARCHAR(30) NOT NULL,  
  c_id          VARCHAR(20) NOT NULL,  
  "money"      NUMERIC DEFAULT 0 ,  
  PRIMARY KEY(u_id, c_id),  
  FOREIGN KEY(u_id) REFERENCES "User"(u_id)
```

ON DELETE CASCADE

);

## 2.11. Order

**Relational Model:** Order(o\_id, take\_indv\_id, send\_customer\_id, price, rate)

**Primary Key:** o\_id

**Foreign Key:** take\_indv\_id references Individual(u\_id)

**Foreign Key:** send\_customer\_id references Customer(u\_id)

**SQL Definition:**

```
CREATE TABLE "Order"(  
  o_id                SERIAL NOT NULL,  
  take_indv_id        VARCHAR(30) NOT NULL,  
  send_customer_id    VARCHAR(30) NOT NULL,  
  price               NUMERIC NOT NULL,  
  rate                NUMERIC,  
  PRIMARY KEY(o_id),  
  FOREIGN KEY(take_indv_id) REFERENCES "User"(u_id)  
  ON DELETE CASCADE,  
  FOREIGN KEY(send_customer_id) REFERENCES "User"(u_id)  
  ON DELETE CASCADE  
);
```

## 2.12. Package

**Relational Model:** Package(p\_id, o\_id, weight, volume)

**Primary Key:** p\_id

**Foreign Key:** o\_id references Order(o\_id)

**SQL Definition:**

```
CREATE TABLE Package(  
  p_id          SERIAL NOT NULL,  
  o_id          INTEGER NOT NULL,  
  weight        NUMERIC NOT NULL,  
  item_dscrptn  VARCHAR not null,  
  volume        NUMERIC NOT NULL,  
  PRIMARY KEY(p_id),  
  FOREIGN KEY(o_id) REFERENCES "Order"(o_id)  
  ON DELETE CASCADE  
);
```

**2.13. 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)

**SQL Definition:**

```
CREATE TABLE MalformedDeliveredPackage(  
  report_id     INTEGER,  
  FOREIGN KEY(report_id) REFERENCES Report(r_id)  
  ) INHERITS(Package);
```

**2.14. 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)

**SQL Definition:**

```
CREATE TABLE LostPackages(  
    report_id    INTEGER,  
    FOREIGN KEY(report_id) REFERENCES Report(r_id)  
) INHERITS(Package);
```

## 2.15. Securely Delivered Packages

**Relational Model:** SecurelyDeliveredPackages(p\_id)

**Primary Key:** p\_id

**Foreign Key:** p\_id references Package(p\_id)

**SQL Definition:**

```
CREATE TABLE SecurelyDeliveredPackages(  
) INHERITS(Package);
```

## 2.16. Holdout Packages

**Relational Model:** HoldoutPackages(p\_id)

**Primary Key:** p\_id

**Foreign Key:** p\_id references Package(p\_id)

**SQL Definition:**

```
CREATE TABLE HoldoutPackages(  
) INHERITS(Package);
```

## 2.17. Transaction

**Relational Model:** Transaction(t\_id, o\_id, tot\_price)

**Primary Key:** t\_id

**Foreign Key:** o\_id references Order(o\_id)

**SQL Definition:**

```

CREATE TABLE "Transaction"(
    t_id          SERIAL NOT NULL,
    o_id          INTEGER NOT NULL,
    tot_price     NUMERIC,
    PRIMARY KEY(t_id),
    FOREIGN KEY(o_id) REFERENCES "Order"(o_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

```

## 2.18. Package State

**Relational Model:** PackageState(ps\_id, state\_date, name)

**Primary Key:** ps\_id

**SQL Definition:**

```

CREATE TABLE PackageState(
    ps_id          SERIAL NOT NULL,
    "name"         VARCHAR(30) NOT NULL,
    state_date     DATE,
    PRIMARY KEY(ps_id)
);

```

## 2.19. Report

**Relational Model:** Report(r\_id, u\_id, explanation,result, date)

**Primary Key:** r\_id

**Foreign Key:** u\_id references Customer(u\_id)

**SQL Definition:**

```

CREATE TABLE Report(
    r_id          SERIAL NOT NULL,

```

```

u_id          VARCHAR(30) NOT NULL,
p_id          INT NOT NULL,
explanation    VARCHAR(300) NOT NULL,
"result"      VARCHAR(10),
"date"        DATE,
PRIMARY KEY(r_id),
FOREIGN KEY(u_id) REFERENCES "User"(u_id)
ON DELETE CASCADE
FOREIGN KEY(p_id) REFERENCES Package (p_id)
ON DELETE CASCADE
);

```

## 2.20. Vehicle

**Relational Model:** Vehicle(v\_id, capacity)

**Primary Key:** v\_id

**SQL Definition:**

```

CREATE TABLE Vehicle(
v_id          SERIAL NOT NULL,
capacity      NUMERIC,
PRIMARY KEY(v_id)
);

```

## 2.21. Courier Vehicle

**Relational Model:** CourierVehicle(v\_id, b\_id)

**Primary Key:** v\_id

**Foreign Key:** v\_id references Vehicle(v\_id)

**Foreign Key:** b\_id references Branch(b\_id)

**SQL Definition:**

```
CREATE TABLE CourierVehicle(  
    b_id          INTEGER,  
    FOREIGN KEY(b_id) REFERENCES Branch(b_id)  
    ON DELETE CASCADE  
    ) INHERITS(Vehicle);
```

**2.22. Shipper Vehicle**

**Relational Model:** ShipperVehicle(v\_id)

**Primary Key:** v\_id

**Foreign Key:** v\_id references Vehicle(v\_id)

**SQL Definition:**

```
CREATE TABLE ShipperVehicle(  
    ) INHERITS(Vehicle);
```

**2.23. Branch**

**Relational Model:** Branch(b\_id, address, name, employee\_num, cash\_register)

**Primary Key:** b\_id

**SQL Definition:**

```
CREATE TABLE Branch(  
    b_id          SERIAL NOT NULL,  
    address       VARCHAR(100) NT NULL,  
    "name"        VARCHAR(30) UNIQUE NOT NULL,  
    employee_num  INTEGER,  
    cash_register NUMERIC,  
    PRIMARY KEY(b_id)  
    );
```

## 2.24. Pac\_State

**Relational Model:** Pac\_State(ps\_id,p\_id,v\_id)

**Primary Key:** ps\_id, p\_id, v\_id

**Foreign Key:** v\_id references Vehicle(v\_id)

**Foreign Key:** ps\_id references PackageState(ps\_id)

**Foreign Key:** p\_id references Package(p\_id)

**SQL Definition:**

```
CREATE TABLE Pac_State(  
    ps_id          INTEGER NOT NULL,  
    p_id           INTEGER,  
    v_id           INTEGER,  
    PRIMARY KEY(ps_id, p_id, v_id),  
    FOREIGN KEY(ps_id) REFERENCES PackageState(ps_id)  
    ON DELETE CASCADE,  
    FOREIGN KEY(p_id) REFERENCES Package(p_id)  
    ON DELETE CASCADE,  
    FOREIGN KEY(v_id) REFERENCES Vehicle(v_id)  
);
```

## 2.25. 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 references Vehicle(v\_id)

**SQL Definition:**

```
CREATE TABLE Route(  
    destination_b_id, departure_b_id, v_id,  
    distance
```

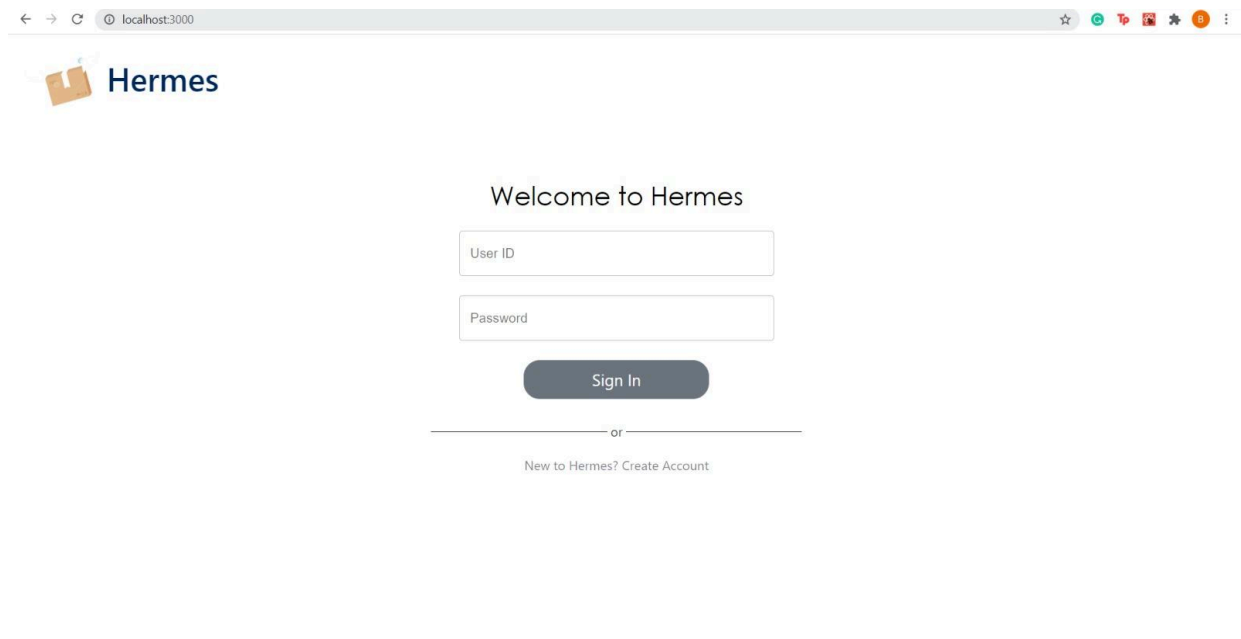
```

destination_b_id      INTEGER,
departure_b_id        INTEGER,
v_id                  INTEGER,
distance              NUMERIC,
PRIMARY KEY(destination_b_id, departure_b_id , v_id),
FOREIGN KEY(destination_b_id) REFERENCES Branch(b_id)
ON DELETE CASCADE,
FOREIGN KEY(departure_b_id) REFERENCES Branch(b_id)
ON DELETE CASCADE,
FOREIGN KEY(v_id) REFERENCES Vehicle(v_id)
);

```

### 3. User Interface Design and Needed SQL Statements

#### 3.1. Sign in



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page features the Hermes logo, which consists of a stylized orange and blue icon followed by the word 'Hermes' in a blue sans-serif font. Below the logo, the text 'Welcome to Hermes' is centered. Underneath this, there are two input fields: the first is labeled 'User ID' and the second is labeled 'Password'. A dark grey 'Sign In' button is positioned below the password field. A horizontal line with the word 'or' in the center separates the sign-in section from the registration section. At the bottom, the text 'New to Hermes? Create Account' is displayed as a link.

**Figure 1**

**Explanation:** Users will sign in to Hermes by using their user ids and passwords

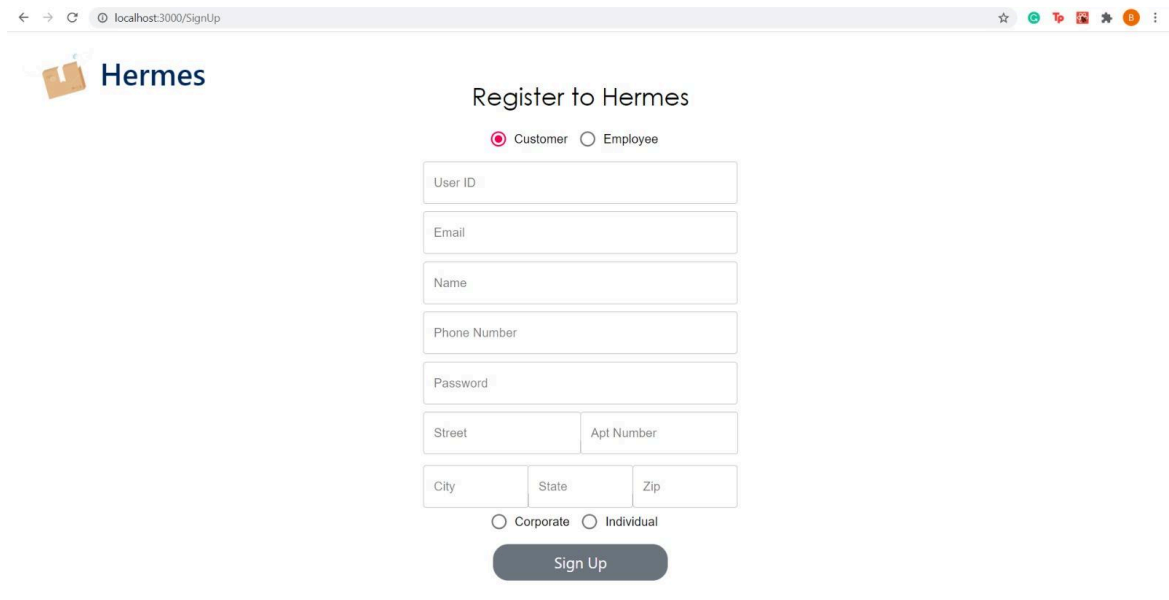
**Inputs:** @email, @password

**Needed SQL Statements:**


- `SELECT * FROM User WHERE u_id = @u_id AND password = @password`

## 3.2. Registration

### 3.2.1. Registration as Customer

A screenshot of a web browser showing the 'Register to Hermes' form. The browser's address bar shows 'localhost:3000/SignUp'. The form has the Hermes logo on the left. It includes radio buttons for 'Customer' (selected) and 'Employee'. The input fields are: 'User ID', 'Email', 'Name', 'Phone Number', 'Password', 'Street' (with 'Apt Number' as a sub-field), 'City', 'State', and 'Zip'. At the bottom, there are radio buttons for 'Corporate' and 'Individual', and a 'Sign Up' button.

← → ↻ localhost:3000/SignUp ☆ 🌐 📄 🏠 📱

 **Hermes**

Register to Hermes

☒ Customer ☐ Employee

User ID

Email

Name

Phone Number

Password

Street Apt Number

City State Zip

☐ Corporate ☐ Individual

Sign Up

**Figure 2**

**Explanation:** Customers will register to Hermes by entering their information. These are: user id that they will use when sign in to Hermes, email, name, password, phone number and their address information. Besides that they have to choose what kind of customer they are: corporate or individual.

**Inputs:** @userid, @email, @name,@phone, @password, @street, @aptnumber, @city, @state, @zipcode, @customertype

**Needed SQL Statements:**

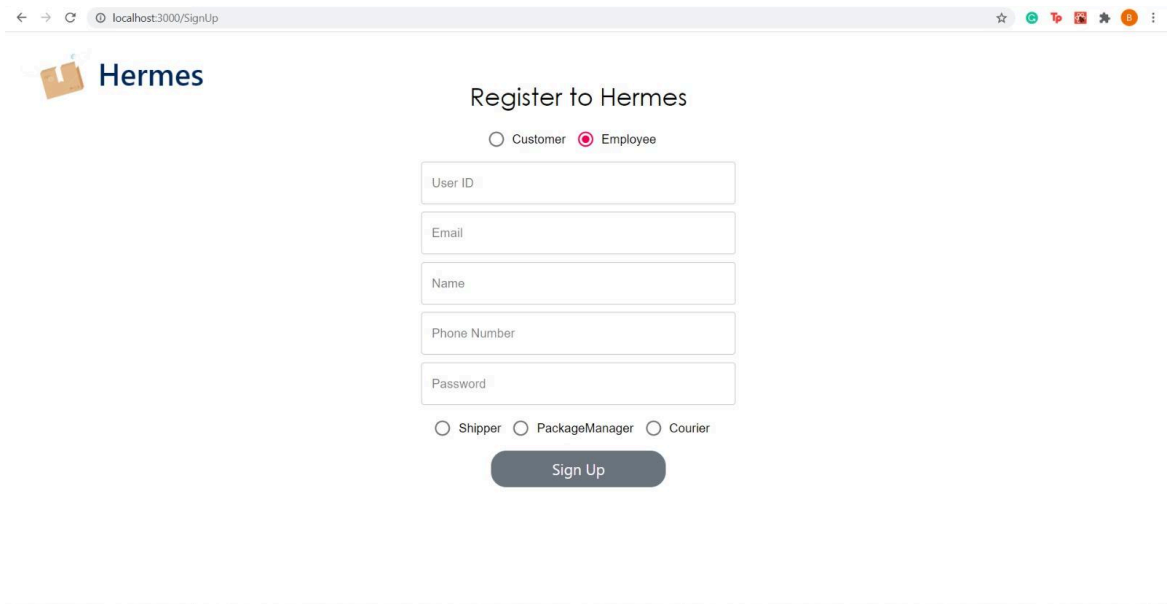
- **If @customertype = Corporate**

```
INSERT INTO Corporate VALUES (@user_id, @name,  
@password, @email, @phone, @city, @street, @apt_number,  
@state, @zip, 0);
```

- **If @customertype = Individual**

```
INSERT INTO Individual VALUES (@user_id, @name, @password,  
@email, @phone, @city, @street, @apt_number, @state, @zip, 0);
```

### 3.2.2. Registration as Employee



The screenshot shows a web browser window with the URL 'localhost:3000/SignUp'. The page features the Hermes logo (a brown box with a white 'H') and the text 'Hermes'. The main heading is 'Register to Hermes'. Below this, there are two radio buttons: 'Customer' (unselected) and 'Employee' (selected). The form contains five text input fields: 'User ID', 'Email', 'Name', 'Phone Number', and 'Password'. Below these fields, there are three radio buttons: 'Shipper' (unselected), 'PackageManager' (unselected), and 'Courier' (unselected). At the bottom of the form is a dark grey 'Sign Up' button.

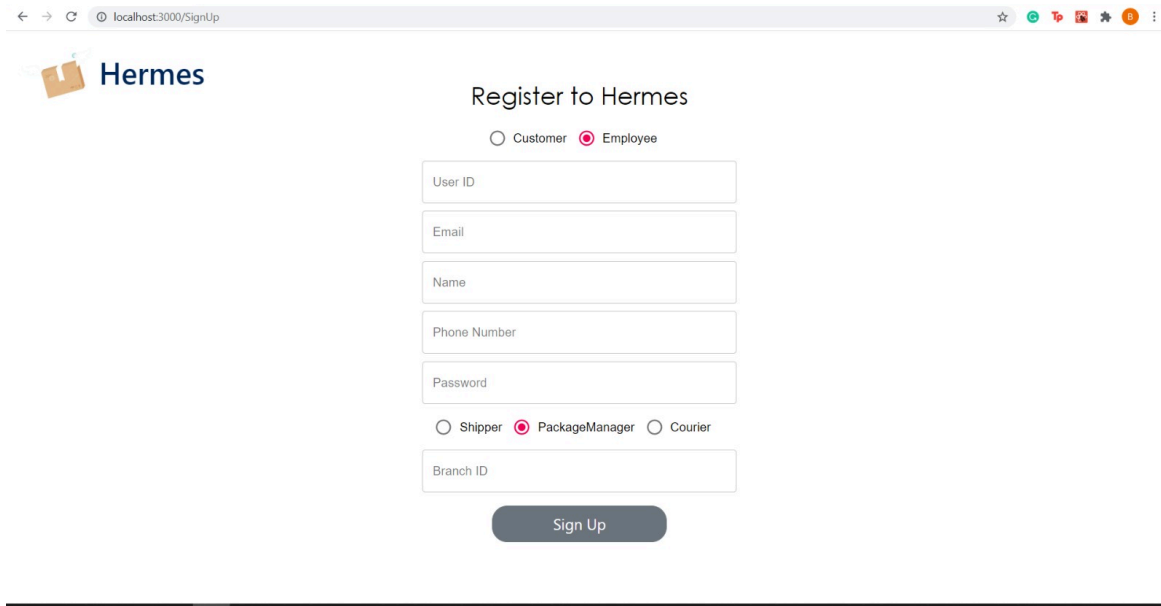
**Figure 3**

**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 .

**Inputs:** @userid, @email, @name,@phone, @password , @employeetype



### 3.2.2.1. Register as PackageManager



The screenshot shows a web browser window with the URL `localhost:3000/SignUp`. The page features the Hermes logo on the left. The main heading is "Register to Hermes". Below this, there are two radio buttons: "Customer" (unselected) and "Employee" (selected). The form contains several input fields: "User ID", "Email", "Name", "Phone Number", "Password", and "Branch ID". Below the "Password" field, there are three radio buttons: "Shipper" (unselected), "PackageManager" (selected), and "Courier" (unselected). A "Sign Up" button is located at the bottom of the form.

**Figure 4**

**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.

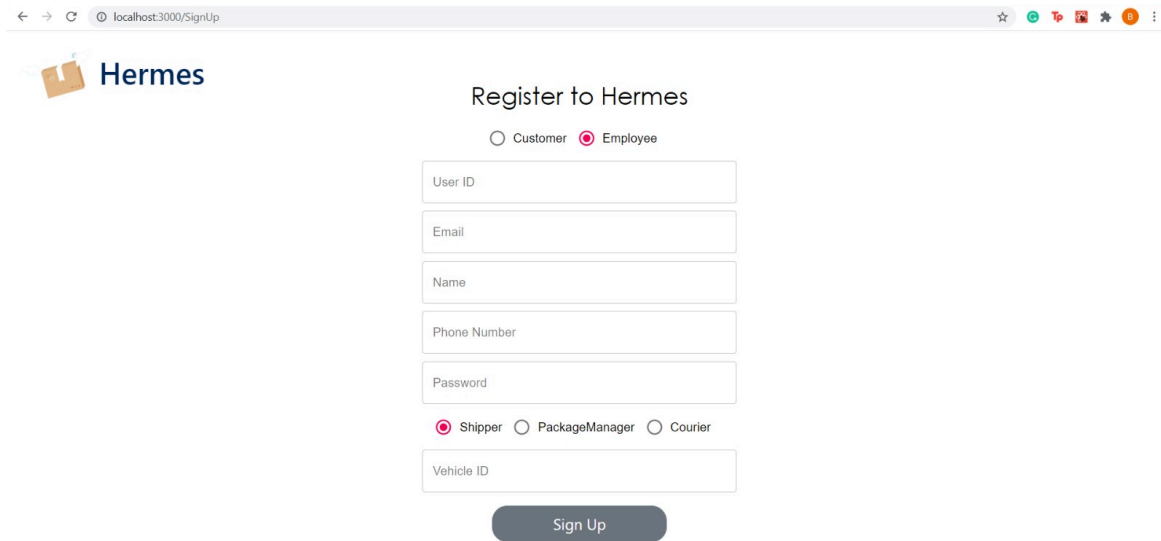
**Inputs:** @userid, @email, @name, @phone, @password, @employeetype, @branchid

**Needed SQL Statements:**

- If @employeetype = PackageManager

```
INSERT INTO PackageManager VALUES (@user_id, @name, @password, @email, @phone, 0, @b_id);
```

### 3.2.2.2. Register as Shipper or Courier



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/SignUp'. The page features the Hermes logo on the left. The main heading is 'Register to Hermes'. Below this, there are two radio buttons: 'Customer' (unselected) and 'Employee' (selected). The form contains several input fields: 'User ID', 'Email', 'Name', 'Phone Number', and 'Password'. Below these fields, there are three radio buttons for employee type: 'Shipper' (selected), 'PackageManager' (unselected), and 'Courier' (unselected). At the bottom of the form is a 'Vehicle ID' input field. A 'Sign Up' button is located at the bottom center of the form.

**Figure 5**

**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.

**Inputs:** @userid, @email, @name, @phone, @password, @employeetype, @vehicleid

#### **Needed SQL Statements:**

- **If @employeetype = Shipper**

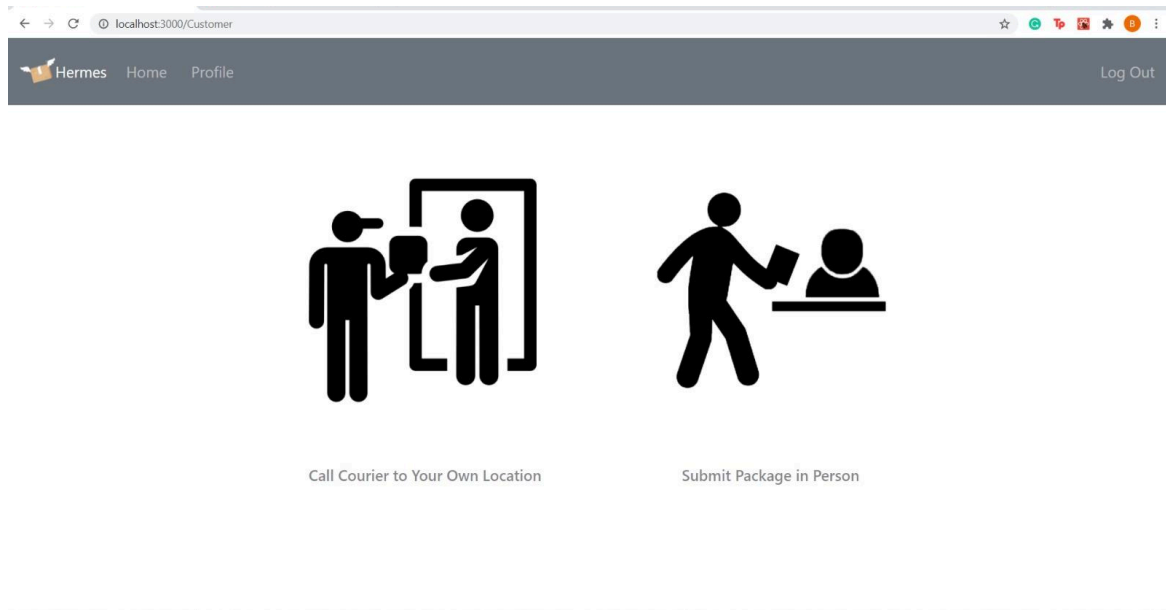
```
INSERT INTO Shipper VALUES (@user_id, @name, @password, @email, @phone, 0, @v_id);
```

- **If @employeetype = Courier**

```
INSERT INTO Courier VALUES (@user_id, @name, @password, @email, @phone, 0, @v_id);
```

### 3.3. Home Page

#### 3.3.1. Customer Home Page



**Figure 6**

**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 11**. If customers choose to submit the package in person option, they will be directed to the page shown in **Figure 12**.

### 3.3.2. Employee Home Page

#### 3.3.2.1. Package Manager Home Page

The screenshot displays the 'Package Manager Home Page' in a web browser. The page has a dark header with the 'Hermes' logo, 'Home', 'Profile', and 'Log Out' links. The main content area contains four assignment forms and a reports section. The 'Assign Shipper' forms (top-left and bottom-left) are for Package ID 276, User ID ahmet.yildiz, Address Bilkent/ANKARA, and Recipient ID cem.alkan. They include dropdowns for 'Select Shipper' and 'Select Branch', and 'Accept' and 'Deny' buttons. The 'Assign Courier' forms (top-right and bottom-right) are for Package ID 289, User ID mehmet.efkan, Address Kızılay/ANKARA, and Recipient ID rasim.yasar. They include a dropdown for 'Select Courier' and an 'Accept' button. The 'Reports' section (right) lists Package IDs 725 and 532 with 'See Details' buttons.

**Figure 7**

**Explanation:** Package managers can assign shippers to the packages that come to the branch by a courier. They should first choose the shipper who is currently in this branch and press accept. Then, the shipper will be assigned. If they deny one of the assign shipper sections, the denied package will be shown in the assign courier section of the package manager's home page in order to send the package back to the customer via courier.

They also can assign couriers to the packages that come to the branch by a shipper. They should first choose the courier and press accept. Besides these, package managers can see the list of reports filled by customers in their homepages and finalize the report as positive or negative as in **Figure 8**.

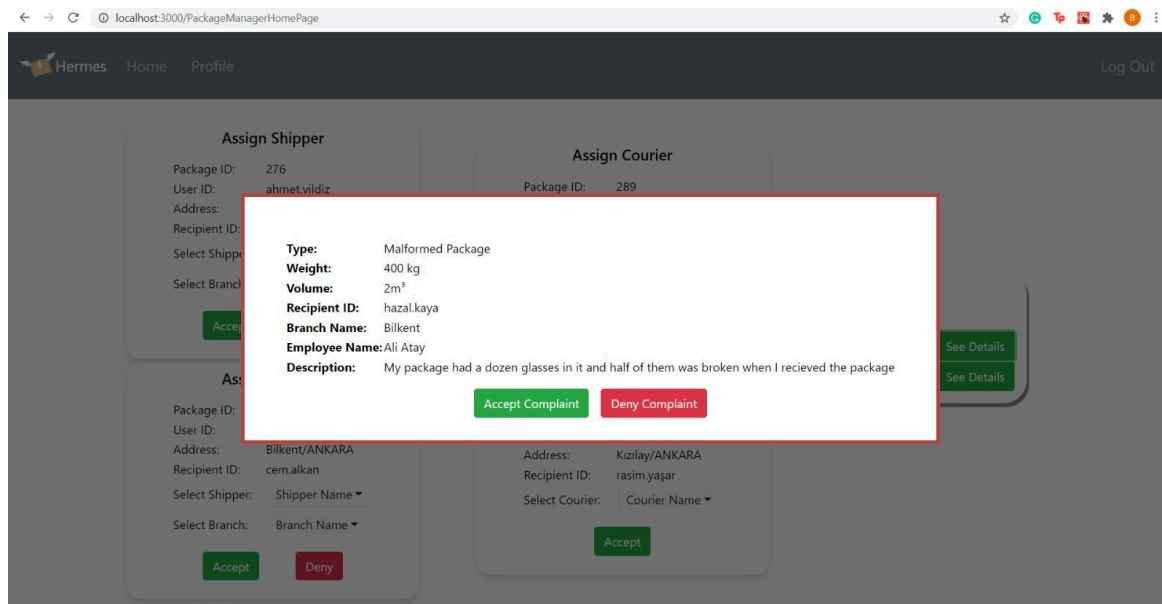
**Inputs: @p\_id**

**Needed SQL Statements:**

- ```
SELECT p.p_id, o.send_customer_id, o.take_indv_id, b.address  
FROM (Package p NATURAL JOIN "Order" o) INNER  
JOIN "User" u ON o.send_customer_id= u.u_id NATURAL
```

JOIN PackageManager pm NATURAL JOIN Branch b  
WHERE p.p\_id = @p\_id;

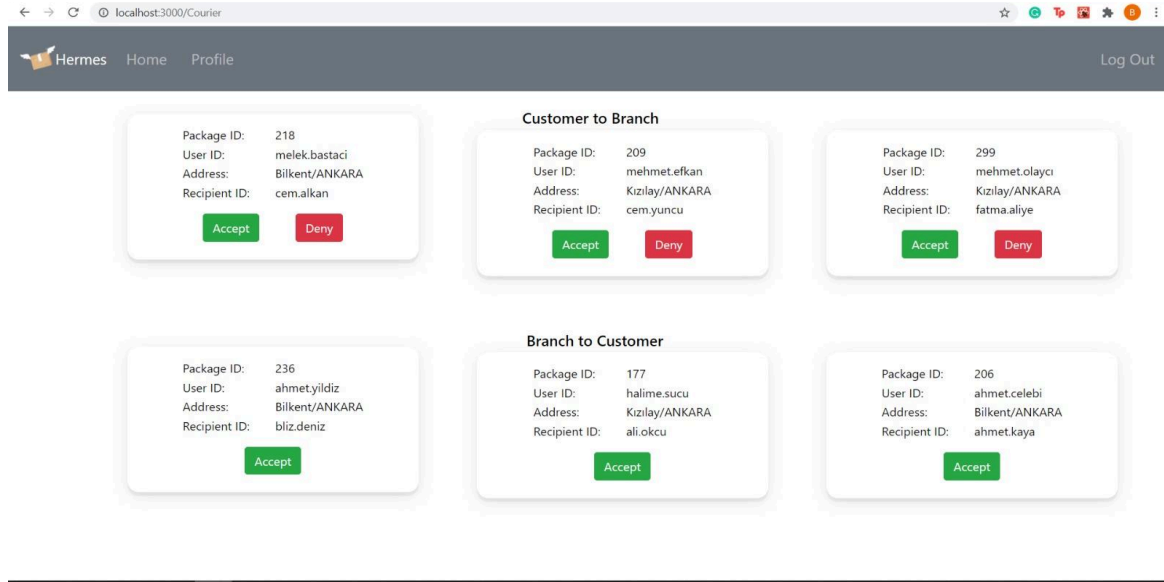
- SELECT b."name" FROM branch b;
- SELECT s."name", s.u\_id, s.v\_id FROM (Shipper s NATURAL JOIN Route r) INNER JOIN Branch b ON r.departure\_b\_id = b.b\_id WHERE b."name" = @name
- INSERT INTO PackageState VALUES('Shipper', @date);
- INSERT INTO Pac\_State VALUES(@ps\_id, @p\_id, @v\_id);
- SELECT c."name", c.v\_id FROM Courier c;
- INSERT INTO PackageState VALUES('Courier', @date);
- INSERT INTO Pac\_State VALUES(@ps\_id, @p\_id, @v\_id);
- SELECT r.p\_id FROM Report;



**Figure 8**

- SELECT r.p\_id FROM Report r;
  - SELECT p.item\_dscrptn, p.weight, p.volume, o.take\_indv\_id, b."name", pm."name"
- FROM (Package p NATURAL JOIN "Order" o) INNER JOIN "User" u  
ON o.send\_customer\_id= u.u\_id NATURAL JOIN PackageManager pm  
NATURAL JOIN Branch b WHERE p.p\_id = @p\_id;
- SELECT \* FROM MalformedPackages mp where mp.p\_id = @p\_id;
  - SELECT \* FROM LostPackages lp where lp.p\_id = @p\_id;

### 3.3.2.2. Courier Home Page



**Figure 9**

**Explanation:** Couriers can see the assigned packages that will be delivered from customer address to branch at the top of their home page. Couriers also can see the assigned packages that will be delivered from branch to customer address. They can accept or deny the packages. If they deny the packages in the customer to branch section, notification will be sent to the customer. 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 7**. They will accept the package to deliver it from the branch to the customer.

#### Needed SQL Statements:

##### --For Customer to Branch

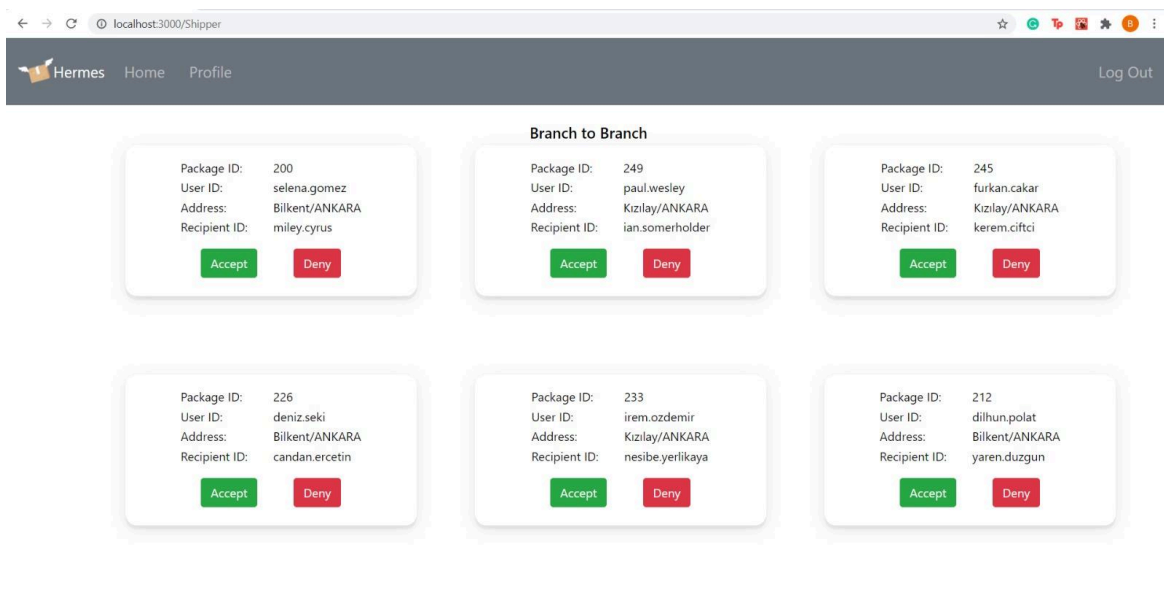
- ```
SELECT p.p_id, o.send_customer_id, o.take_indv_id,
b.address
FROM (Package p NATURAL JOIN "Order" o)
INNER JOIN "User" u ON o.send_customer_id= u.u_id
NATURAL JOIN PackageManager pm NATURAL JOIN
Branch b WHERE p.p_id = @p_id;
```
- ```
SELECT c.v_id FROM Courier c WHERE c.u_id = @u_id;
```

- INSERT INTO PackageState VALUES('Courier', @date);
- INSERT INTO Pac\_State VALUES(@ps\_id, @p\_id, @v\_id);

#### --For Branch to Customer

- SELECT p.p\_id, o.send\_customer\_id, o.take\_indv\_id, c.state, c.street, c.apt\_number  
FROM (Package p NATURAL JOIN "Order" o)  
INNER JOIN Customer c ON o.take\_indv\_id= c.u\_id  
WHERE p.p\_id = @p\_id;
- SELECT c.v\_id FROM Courier c WHERE c.u\_id = @u\_id;
- INSERT INTO PackageState VALUES('Courier', @date);
- INSERT INTO Pac\_State VALUES(@ps\_id, @p\_id, @v\_id);

### 3.3.2.3. Shipper Home Page



**Figure 10**

**Explanation:** Shippers can see the assigned packages that will be delivered from the current branch to the destination branch. They can accept or deny the assigned packages. If they deny the assigned packages, the package will be shown in the package managers' home page to assign the package again in **Figure 7**.

### Needed SQL Statements:

- `SELECT p.p_id, o.send_customer_id, o.take_indv_id, b.address FROM (Package p NATURAL JOIN "Order" o) INNER JOIN "User" u ON o.send_customer_id= u.u_id NATURAL JOIN PackageManager pm NATURAL JOIN Branch b WHERE p.p_id = @p_id;`
- `SELECT s.v_id FROM Shipper s WHERE s.u_id = @u_id;`
- `INSERT INTO PackageState VALUES('Shipper', @date);`
- `INSERT INTO Pac_State VALUES(@ps_id, @p_id, @v_id );`

## 3.4. Submitting Packages For Customers

### 3.4.1. Submit Package to Courier



Please fill out the form below to submit your package to the courier.

Describe the item:

Enter the weight of the item:

Enter the volume of the item:

m<sup>3</sup>

Select recipient of the item:

Recipient ID ▾

Select branch:

Branch Name ▾

Current Total Price: 47.93₺

Submit

or

Go Back to Selection

**Figure 11**

**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. According to this information, they will be able to see the calculated price for the package. They also need to select the closest branch to them to call



the courier. Then they will submit it to the courier. An available courier from the selected branch will be automatically assigned by the system to take the package from the customer.

**Inputs:** @description, @weight, @volume, @recipientid, @branch

**Needed SQL Statements:**

- INSERT INTO "Order" VALUES(@take\_indv\_id , @send\_customer\_id, @price, @rate);
- INSERT INTO package VALUES(@o\_id , @weight, @volume);
- SELECT b."name" FROM branch b;
- SELECT c."name" FROM customer c;

### 3.4.2. Submit Package In Person



Please fill out the form below to submit your package to the branch in person.

Describe the item:

Enter the weight of the item:

kg

Enter the volume of the item:

m³

Select recipient of the item:

Recipient ID ▾

Select branch:

Branch Name ▾

Select employee:

Employee Name ▾

Current Total Price: 40.52₺

Submit

or

Go Back to Selection

**Figure 12**

**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 one of the employees in that branch. Then they are able to calculate the price of the package and submit it to the package manager.

**Inputs:** @description, @weight, @volume, @recipientid, @branch

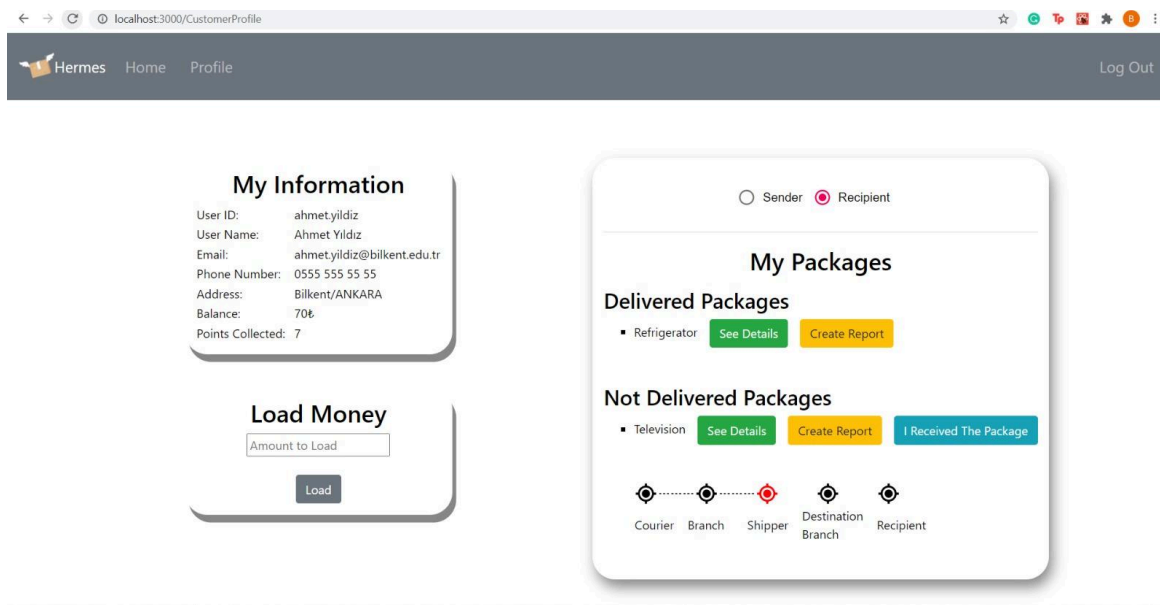
### Needed SQL Statements:

- INSERT INTO "Order" VALUES(@take\_indv\_id , @send\_customer\_id, @price, @rate);
- INSERT INTO package VALUES(@o\_id , @weight, @volume);
- SELECT b."name" FROM branch b;
- SELECT pm."name" , pm.u\_id FROM packagemanager pm WHERE b\_id = @b\_id;
- SELECT c."name" FROM customer c;

## 3.5. Profile Page

### 3.5.1. Customer Profile Page

#### 3.5.1.1. Individual Profile Page



**Figure 13**

**Explanation:** Individual customers can see their personal information in their profile pages. They can see the list of delivered and not delivered packages. Besides that, they can see the status of not delivered packages such as, on branch, on courier etc. They are able to load money to their card by using the load button. They can see details of the packages by clicking the “See Details” button as in **Figure 14**. They also can create malformed reports for delivered packages as in **Figure 20**. Besides, they

can create lost reports for undelivered packages as in *Figure 21*. At the final step, they will use the button “I Received The Package” as shown in *Figure 15*, if they successfully take the package. This button will be disabled until the courier takes the package in order to send it to the customer.

**Inputs:** @addedmoney

**Needed SQL Statements:**

**--For Customer Info**

- SELECT i.u\_id , i."name", i.email , i.phone , i.street , i.state , i.zip ,i.city, c".money"  
FROM Individual i NATURAL JOIN customercard c  
WHERE i.u\_id = @u\_id);

**--For Delivered Packages for Sender**

- (SELECT p1.item\_dscrptn  
FROM "Order" o NATURAL JOIN SecurelyDeliveredPackages p1  
WHERE o.send\_customer\_id = @send\_customer\_id  
UNION  
SELECT p2.item\_dscrptn  
FROM "Order" o NATURAL JOIN MalformedDeliveredPackage  
p2  
WHERE o.send\_customer\_id = @send\_customer\_id);

**--For Non-delivered Packages for Sender**

- (SELECT p1.item\_dscrptn  
FROM "Order" o NATURAL JOIN LostPackages p1  
WHERE o.send\_customer\_id = @send\_customer\_id  
UNION  
SELECT p2.item\_dscrptn  
FROM "Order" o NATURAL JOIN HoldoutPackages p2

WHERE o.send\_customer\_id = @send\_customer\_id);

**--For Delivered Packages for Recipient**

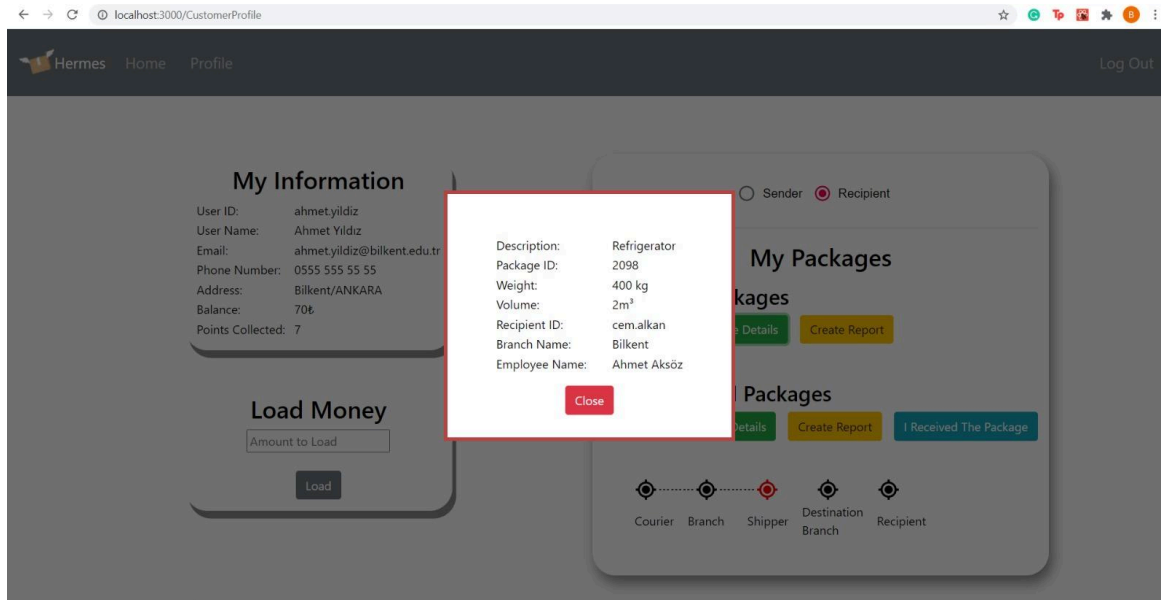
```
(SELECT p1.item_dscrptn
FROM "Order" o NATURAL JOIN SecurelyDeliveredPackages p1
WHERE o.take_indv_id = @take_indv_id
UNION
SELECT p2.item_dscrptn
FROM "Order" o natural join MalformedDeliveredPackage p2
WHERE o.take_indv_id = @take_indv_id);
```

**--For Non-delivered Packages for Recipient**

- (SELECT p1.item\_dscrptn  
FROM "Order" o NATURAL JOIN LostPackages p1  
WHERE o.take\_indv\_id = @take\_indv\_id  
UNION  
SELECT p2.item\_dscrptn  
FROM "Order" o NATURAL JOIN HoldoutPackages p2  
WHERE o.take\_indv\_id = @take\_indv\_id);

**--For load money**

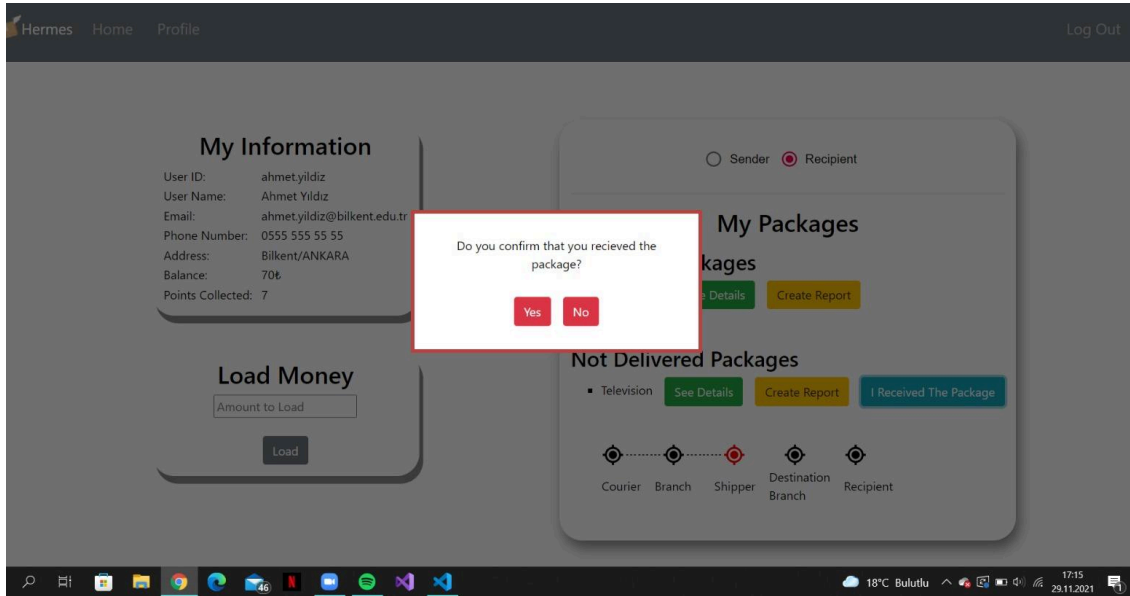
- UPDATE CustomerCard SET "money" = "money"+@addedmoney  
WHERE u\_id= @u\_id;



*Figure 14*

#### Needed SQL Statements:

- ```
SELECT p.p_id, p.item_descrptn, p.weight, p.volume, o.take_indv_id,
b."name"
FROM (Package p NATURAL JOIN "Order" o)
INNER JOIN "User" u ON o.send_customer_id= u.u_id NATURAL JOIN
PackageManager pm NATURAL JOIN Branch b WHERE p.p_id = @p_id;
```



*Figure 15*

**Needed SQL Statements:**

```
INSERT INTO PackageState VALUES('Recipient', @date);
```

```
INSERT INTO Pac_State VALUES(@ps_id, @p_id, @v_id );
```

```
SELECT * FROM Package p WHERE p.p_id = @p_id;
```

```
INSERT INTO SecurelyDelivered VALUES(@p_id, @o_id, @weight,  
@item_dscrptn, @volume);
```

### 3.5.1.2. Corporate Profile Page

← → localhost:3000/CompanyProfile ☆ 📄 📄 📄 📄 📄

Hermes Home Profile Log Out

#### My Information

User ID: trendyol  
User Name: Trendyol  
Email: deliveries@trendyol  
Phone Number: 0555 555 55 55  
Address: Trendyol A.Ş./ANKARA  
Budget: 75000

#### Load Money

Amount to Load

Load

**Figure 16**

**Explanation:** Corporate customers can see their institutional information in their profile pages. They are able to load money to budget by using the load button.

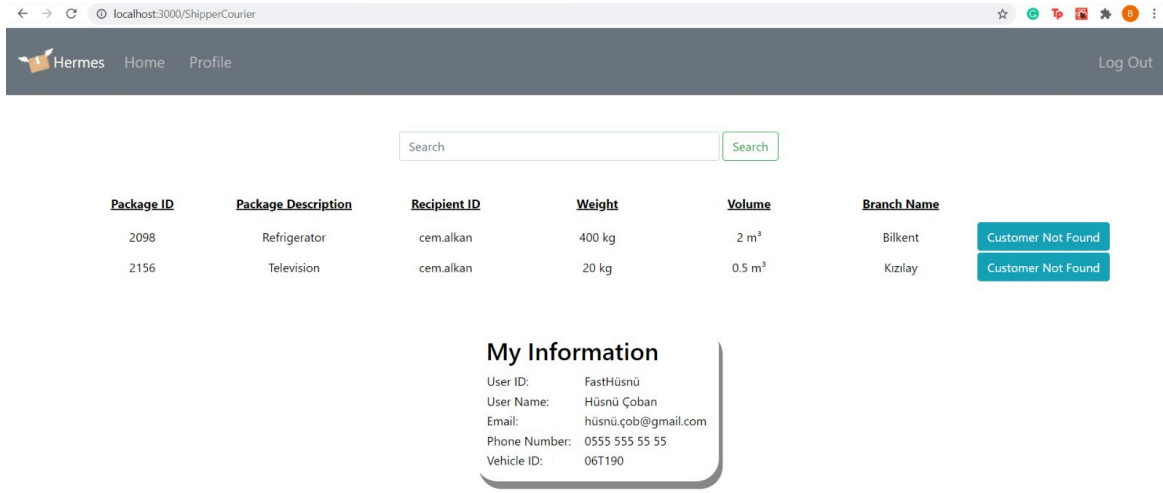
**Inputs:** @addedmoney

**Needed SQL Statements:**

- SELECT \* FROM corporate c WHERE c.u\_id = @u\_id;
- UPDATE corporate SET budget = budget + @addedmoney WHERE u\_id= @u\_id;

### 3.5.2. Employee Profile Page

#### 3.5.2.1. Courier Profile Page



**Figure 17**

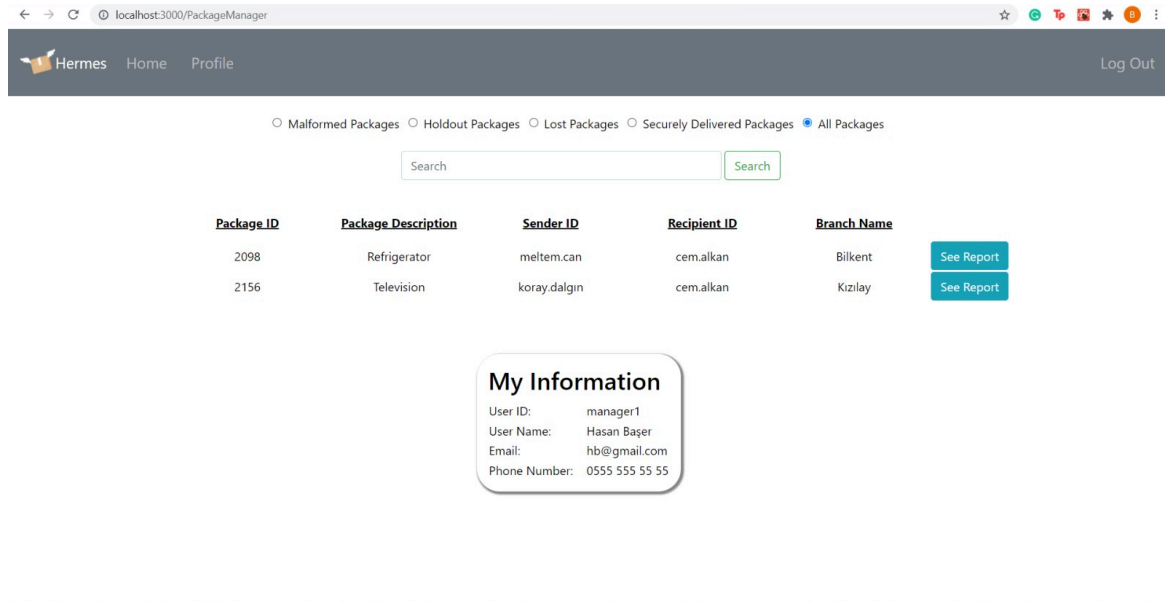
**Explanation:** Couriers can see their information in their profile pages. Besides, couriers can see the packages they accepted to deliver here. When they try to deliver the package to a customer and cannot reach them, the courier presses the customer not found the button. Then, the package state changes to the holdout and the package takes place in the holdout packages table.

#### Needed SQL Statements:

- `SELECT * FROM courier c WHERE c.u_id = @u_id;`
- `SELECT pk.p_id FROM packagestate pk WHERE pk.name = 'Courier';`
- `SELECT p.p_id, p.item_dscrptn, p.volume, p.weight, o.take_indv_id, o.o_id, b.name FROM (Package p NATURAL JOIN "Order" o) INNER JOIN "User" u ON o.send_customer_id = u.u_id NATURAL JOIN PackageManager pm NATURAL JOIN Branch b WHERE p.p_id = @p_id;`
- `INSERT INTO HoldoutPackages VALUES (@p_id, @o_id, @weight, @item_dscrptn, @volume);`
- `UPDATE packagestate ps SET ps. "Name" = 'Holdout' WHERE ps.p_id = @p_id`



### 3.5.2.2. Package Manager Profile Page



**Figure 18**

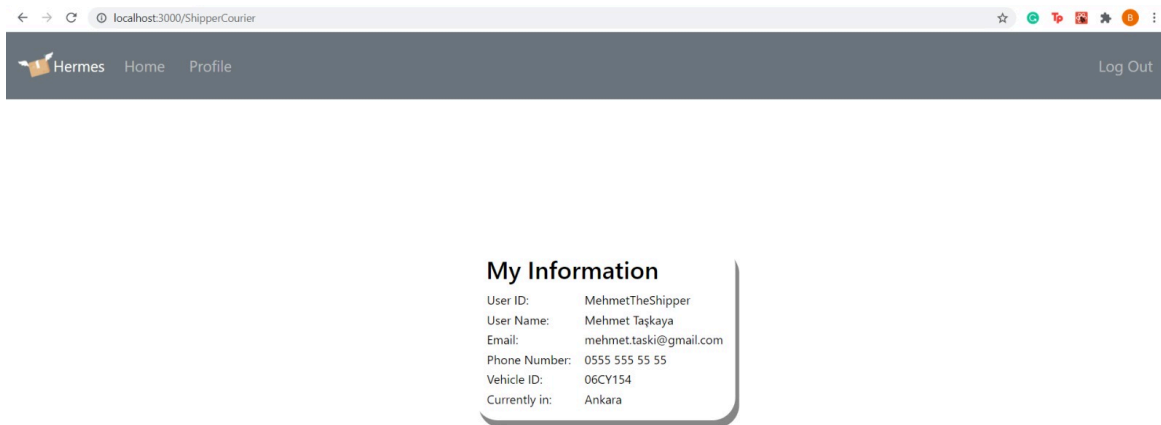
**Explanation:** Package Managers can see their information in their profile pages. Besides these, they can see malformed packages, holdout packages, lost packages, securely delivered packages and all packages by using radio buttons at the top of the page. They also can search packages by using the search bar. They can see reports by clicking the “See Report” button.

#### Needed SQL Statements:

- `SELECT * FROM PackageManager pm where pm.u_id = @u_id;`
- `SELECT p.p_id, p.item_dscrptn, o.take_indv_id, b."name"  
FROM (Package p NATURAL JOIN "Order" o) INNER JOIN "User" u  
ON o.send_customer_id= u.u_id NATURAL JOIN PackageManager pm  
NATURAL JOIN Branch b;`
- `SELECT sp.p_id, sp.item_dscrptn, o.take_indv_id, b."name"  
FROM (SecurelyDeliveredPackage sp NATURAL JOIN "Order" o)  
INNER JOIN "User" u ON o.send_customer_id= u.u_id NATURAL JOIN  
PackageManager pm NATURAL JOIN Branch b;`

- ```
SELECT lp.p_id, lp.item_dscrptn, o.take_indv_id, b."name"
FROM (LostPackages lp NATURAL JOIN "Order" o) INNER JOIN
"User" u ON o.send_customer_id= u.u_id NATURAL JOIN
PackageManager pm NATURAL JOIN Branch b;
```
- ```
SELECT hp.p_id, hp.item_dscrptn, o.take_indv_id, b."name"
FROM (HouldoutPackages hp NATURAL JOIN "Order" o) INNER JOIN
"User" u ON o.send_customer_id= u.u_id NATURAL JOIN
PackageManager pm NATURAL JOIN Branch b;
```
- ```
SELECT hp.p_id, hp.item_dscrptn, o.take_indv_id, b."name"
FROM (MalformedPackages p NATURAL JOIN "Order" o) INNER JOIN
"User" u ON o.send_customer_id= u.u_id NATURAL JOIN
PackageManager pm NATURAL JOIN Branch b;
```

### 3.5.2.3. Shipper Profile Page



**Figure 19**

**Explanation:** Shippers can see their information in their profile pages.

**Needed SQL Statements:**

- ```
SELECT * FROM shipper s WHERE s.u_id = @u_id;
```

## 3.6. Reports

### 3.6.1. File a Report Customer

#### 3.6.1.1. Malformed Report

The screenshot shows a web browser window at localhost:3000/CustomerProfile. The page has a dark header with 'Hermes', 'Home', 'Profile', and 'Log Out' links. The main content area is titled 'My Info' and displays user details: User ID, User Name, Email, Phone Number, Address, Balance, and Points Collected. A modal form titled 'Malformed Package' is overlaid on the page. The form contains the following text: 'Creating report for a delivered package means that your package is malformed. If it is malformed, fill out the form below.' It also displays 'Description: Refrigerator' and 'Package ID: 2098'. Below this is a text area labeled 'Give information about the package's situation:' and two buttons: 'Submit' (green) and 'Close' (red). In the background, there is a 'Load' button and a 'I Received The Package' button. At the bottom, a flow diagram shows the package journey: Courier -> Branch -> Shipper -> Destination Branch -> Recipient.

*Figure 20*

**Explanation:** Individual customers can report if their delivered package is malformed. They will fill the input text to describe the malformed package's situation.

**Inputs:** @text

**Needed SQL Statements:**

**--For Package Details**

- `SELECT p.p_id, p.item_dscrptn FROM MalformedDeliveredPackage p WHERE p.p_id = @p_id ;`

**--For Report Creation**

- `INSERT INTO Report VALUES(@u_id, @p_id, @explanation, null, @date);`

### 3.6.1.2. Lost Report

localhost:3000/CustomerProfile

Hermes Home Profile Log Out

**My Info**

User ID: 4  
User Name: A  
Email: a  
Phone Number: 0  
Address: 8  
Balance: 7  
Points Collected: 7

**Lost Package**

Creating report for a not delivered package means that your package is lost.  
If it is lost, fill out the form below.

Description: Television  
Package ID: 2098

Give information about the package's situation:

Submit Close

I Received The Package

Courier Branch Shipper Destination Branch Recipient

*Figure 21*

**Explanation:** Individual customers can report if their undelivered package is lost. They will fill the input text to describe the package's situation.

**Inputs:** @text

**Needed SQL Statements:**

**--For Package Details**

- `SELECT p.p_id, p.item_dscrptn FROM LostPackages p WHERE p.p_id = @p_id;`

**--For Report Creation**

- `INSERT INTO Report VALUES(@u_id, @p_id, @explanation, null, @date);`

## **4. Implementation Plan**

Our projects will be implemented by several programs. The front-end of the website will be implemented using React.js. For the back-end and API handling Node.js will be used. As the query language PostgreSQL will be used. We have implemented user interfaces using React.js and the photos in figures above are screen captures of the program implementation. We have used VisualStudio Code for implementation. For the database management system, pgAdmin will be used in order to keep track of the records.