CS 353

Project Final Report
Shipping Company Data Management System

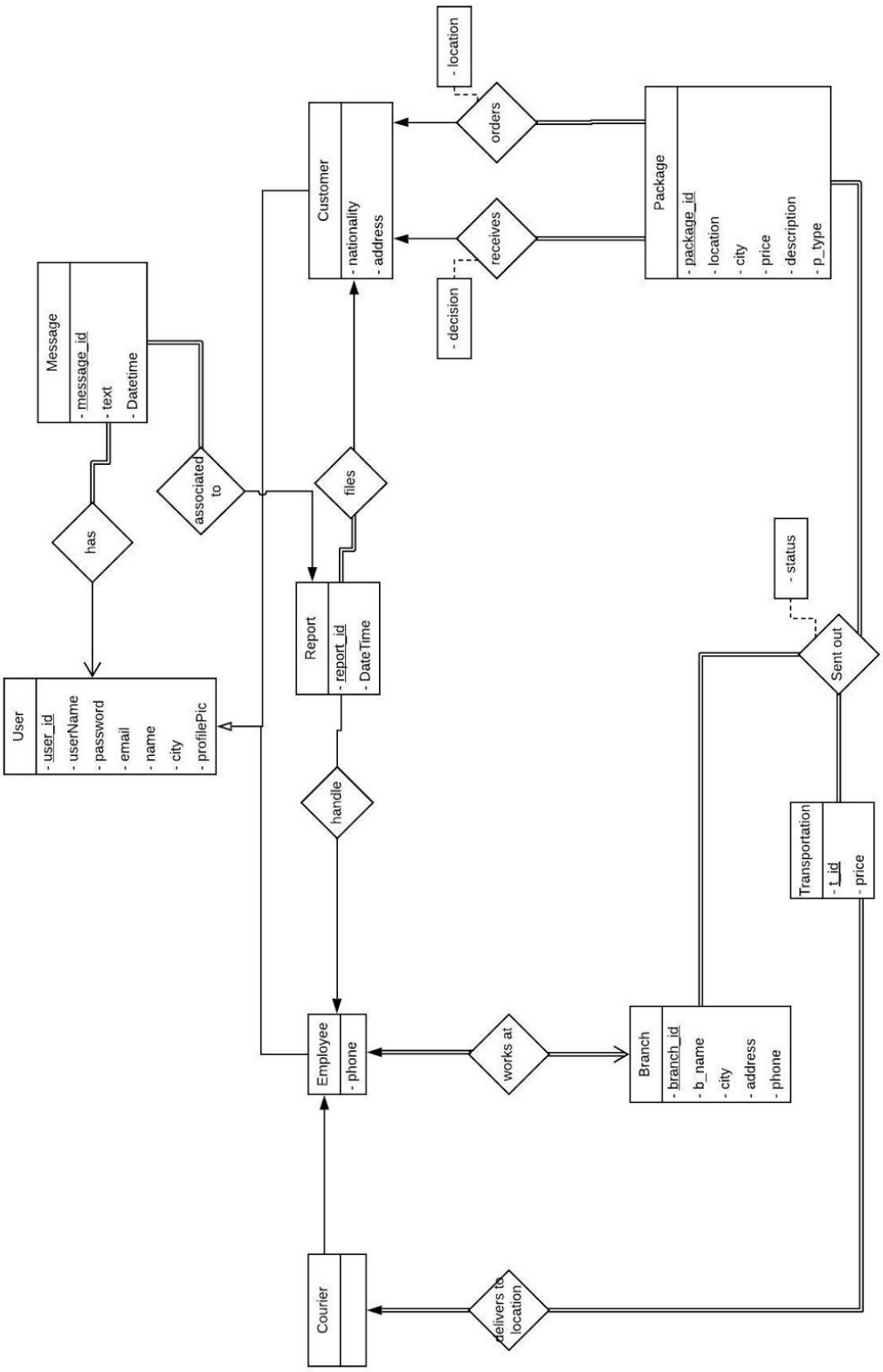
Group 14
Okan Şen 21202377

1. Introduction

Shipport is a shipment transportation company application which is for customers to order packages to other customers using location-based addresses, meaning that the location does not necessarily have to be the receiving customer’s saved address on the system. As soon as an order is given, the package is brought to the closest branch in the

city, and it is then, delivered to the given location by a courier, to the customer. The package and delivery prices are added and are demanded from the receiving customer. The receiving customer can check the package for any damage, and if found any, they can refuse the delivery, filing a report as well. They will be contacted by an employee and they will be compensated, by a new package, or by money.

2. Revised E/R Diagram


Customer
- nationality
- address
orders
- location
Package
- package_id
- location
- city
- price
- description
- p_type
receives
- decision
Message
- message_id
- text
- Datetime
has
associated to
files
User
- user_id
- userName
- password
- email
- name
- city
- profilePic
Report
- report_id
- DateTime
handle
- status
Sent out
Transportation
- t_id
- price
Employee
- phone
works at
Branch
- branch_id
- b_name
- city
- address
- phone
Courier
delivers to location

3. Final Relation Schemas

3.1 User

Relational Model:
User(user_id, userName, password, email, name, city, profilepic)

Primary Key:
user_id

Functional Dependencies:
user_id → userName, password, email, name, city, profilepic

Normal Form:
BCNF

3.2 Customer

Relational Model:
Customer(user_id, nationality, address)

Primary Key:
user_id

Functional Dependencies:
user_id → nationality, address

Normal Form:
BCNF

Foreign Key:
user_id references User(user_id)

3.3 Employee

Relational Model:
Employee(user_id, phone)

Primary Key:
user_id

Functional Dependencies:
user_id → phone

Normal Form:
BCNF

Foreign Key:
user_id references User(user_id)

3.4 Courier

Relational Model:
Courier(user_id)

Primary Key:
user_id

Normal Form:
BCNF

Foreign Key:
user_id references Employee(user_id)

3.5 Message

Relational Model:
Message(MessageID, text, DateTime, Sender, Recipient)

Primary Key:
MessageID

Functional Dependencies:
MessageID → text, DateTime, Sender, Recipient

Normal Form:
BCNF

Foreign Key:
Sender references Employee(user_id)
Recipient references Customer(user_id)

3.6 Package

Relational Model:
Package(package_id, city, price, description, p_type, delivery_type, payment_type)

Primary Key:
package_id

Functional Dependencies:
package_id → city, price, description, p_type, delivery_type, payment_type

Normal Form:
BCNF

3.7 Orders

Relational Model:
Orders(package_id, location, user_one_id, user_two_id)

Primary Key:
package_id

Functional Dependencies:
package_id→ location, user_one_id,user_two_id

Normal Form:
BCNF

Foreign Key:
package_id references Package(package_id)
user_one_id references Customer(user_id)
user_two_id references Customer(user_id)

3.8 Receives

Relational Model:
Receives(package_id, decision, user_one_id , user_two_id)

Primary Key:
package_id

Functional Dependencies:
package_id → decision, user_one_id , user_two_id

Normal Form:
BCNF

Foreign Key:
package_id references Package(package_id)
user_one_id references Customer(user_id)
user_two_id references Customer(user_id)

3.9 Report

Relational Model:
Report(report_id, DateTime)

Primary Key:
report_id

Functional Dependencies:
report_id → DateTime

Normal Form:
BCNF

3.10 Files

Relational Model:
Files(report_id, user_one_id, user_two_id)

Primary Key:
report_id

Functional Dependencies:
report_id → user_one_id, user_two_id

Normal Form:
BCNF

Foreign Key:
report_id references Report(report_id)
user_one_id references Customer(user_id)
user_two_id references Customer(user_id)

3.11 Handle

Relational Model:
Handle(report_id, user_id)

Primary Key:
report_id

Functional Dependencies:
report_id → user_id

Normal Form:

BCNF

Foreign Key:
report_id references Report(report_id)
User_id references Employee(user_id)

3.12 AssociatedTo

Relational Model:
AssociatedTo(message_id, report_id)

Primary Key:
message_id

Functional Dependencies:
message_id → report_id

Normal Form:
BCNF

Foreign Key:
message_id references Message(message_id)
report_id references Report(report_id)

3.13 Has

Relational Model:
Has(message_id, user_id)

Primary Key:
message_id

Functional Dependencies:
message_id → user_id

Normal Form:
BCNF

Foreign Key:
message_id references Message(message_id)
user_id references User(user_id)

3.14 Branch

Relational Model:
Branch(branch_id, b_name, city, address, phone)

Primary Key:
branch_id

Functional Dependencies:
branch_id → b_name, city, address, phone

Normal Form:
BCNF

3.15 Transportation

Relational Model:
Transportation(t_id, price)

Primary Key:
t_id

Functional Dependencies:
t_id → price

Normal Form:
BCNF

3.16 WorksAt

Relational Model:
WorksAt(user_id, branch_id)

Primary Key:
user_id

Functional Dependencies:
user_id → branch_id

Normal Form:
BCNF

Foreign Key:
user_id references Employee(user_id)
branch_id references Branch(branch_id)

3.17 DeliversToLocation

Relational Model:
DeliversToLocation(t_id)

Primary Key:
t_id

Normal Form:
BCNF

Foreign Key:
t_id references Transportation(t_id)

3.18 SentOut

Relational Model:
SentOut(package_id, t_id, branch_id, status)

Primary Key:
package_id, t_id, branch_id

Functional Dependencies:
package_id, t_id, branch_id → status

Normal Form:
BCNF

Foreign Key:
package_id references Package(package_id)
t_id references Transportation(t_id)
branch_id references Branch(branch_id)

4. Implementation Details

This project took the use of Xampp for its server creation and connecting MySQL and PHP for the website. It has built-in features for both MySQL and PHP. Xampp has its own shell command prompt, which is included in MySQL's standalone version as well, was used to create Shipport database using commands; "mysql -u root -p -h 127.0.0.1" to access MySQL, and then "create database shipport;" to create our database locally.

Later on, Eclipse platform was used to create the tables in our database in Java language, with mysqlconnectorjava as an additional JAR to be able to use MySQL codes. When the program was run, the tables were inserted inside Shipport database, and it could be viewed from "localhost/phpmyadmin".

PHP codes were written using Visual Studio because it recognizes the PHP labels. Accessing the database was problematic in the first stages of the project using PHP, because of its implementation limitations. PHP not being purely object-oriented

slows the code writing process while we could easily reach everything in object-based platforms.

In addition to PHP problems, using MySQL has its own set of problems as well. Using SQL queries can be a hassle sometimes, but especially when using nested queries it gets really messy and complicated. Instead of using MySQL, MongoDB could be a better choice.

5. Sample Output Reports

5.1 Total Sum of Price Received by Customer

```
SELECT SUM(t.price + p.price)
FROM Transportation t, SentOut s, Package p
WHERE t.t_id = s.t_id AND p.package_id = s.package_id
```

6. User's Manual

6.1 Login/Signup

Customers can log in or signup using usernames and passwords. If the user is an employee they can click on the "Click here to Get to Employee Page". If the user is an employee, they will be granted an account by the company, they do not need to sign up. They can only login with their given accounts. Customers can only sign up or login using the default login/signup page.

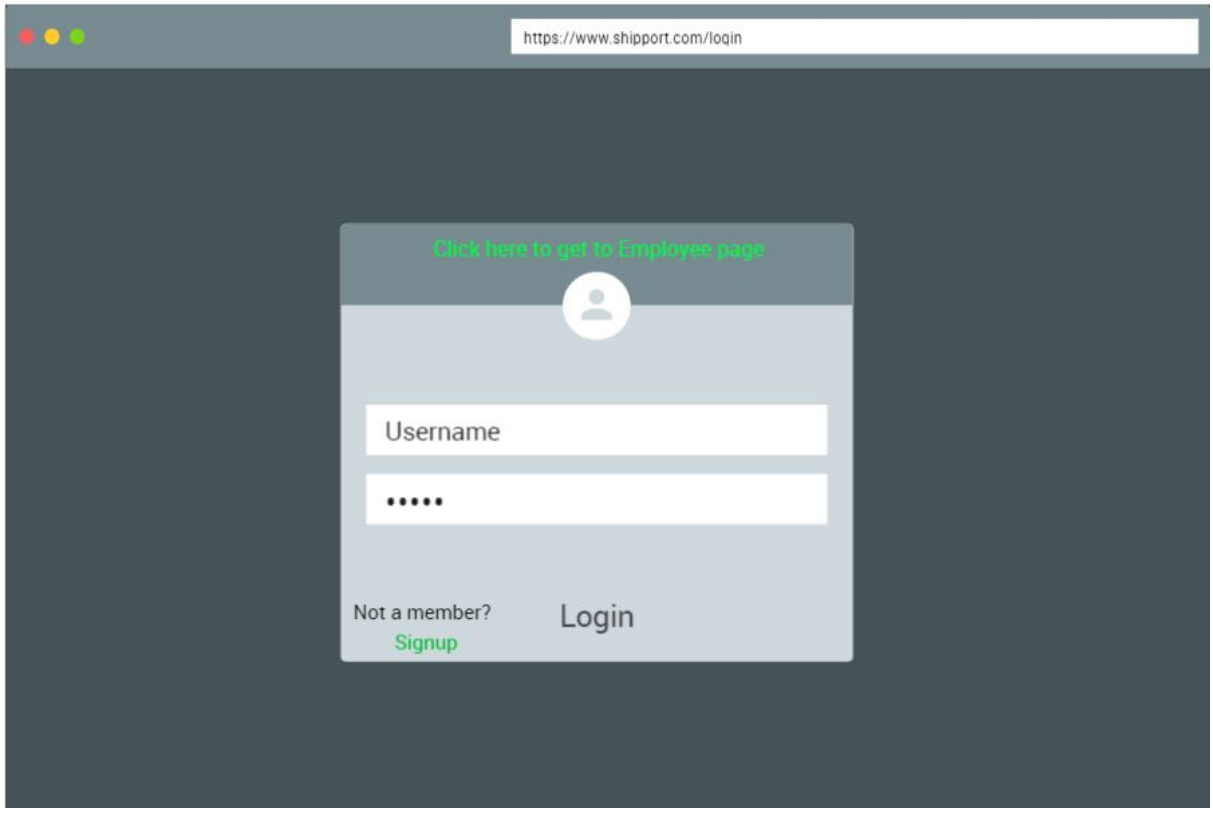


Figure 6.1.1: Login Default

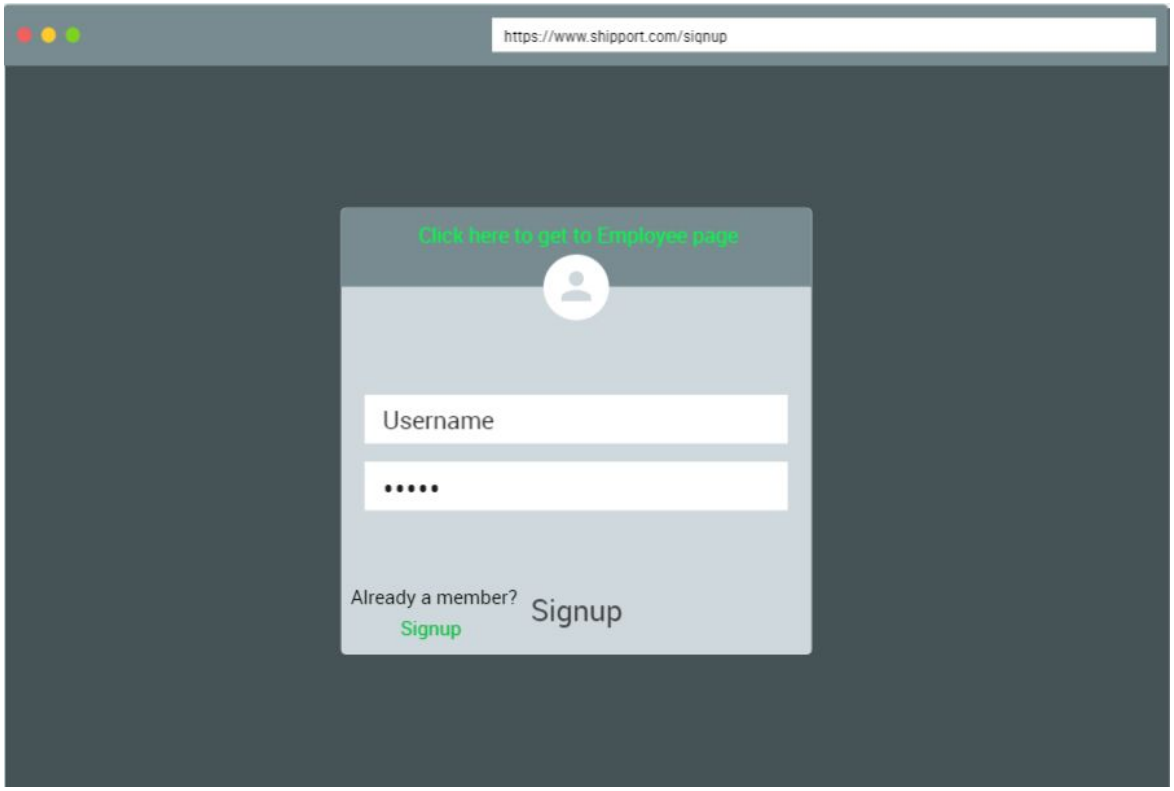


Figure 6.1.2: Signup Default

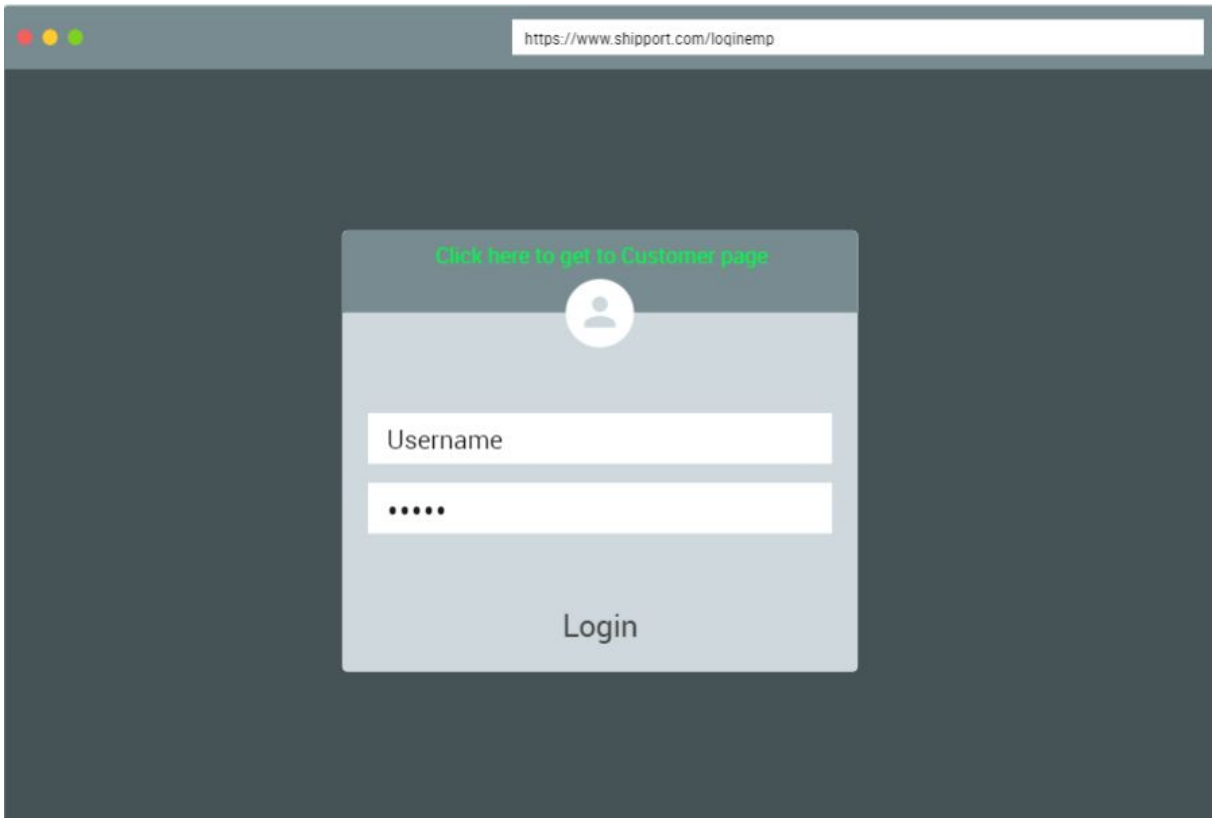


Figure 6.1.3: Login Employee

6.2 Customer Dashboard

Customers will be taken to this page when they signup or login. Here, they can send a package, check their current(ongoing) packages(either sent to them or sent by them), list all of their received and sent packages, and report a package.

They can also click on their profile picture on the top right side of the page to edit their profile settings. In Figure 6.2.2.

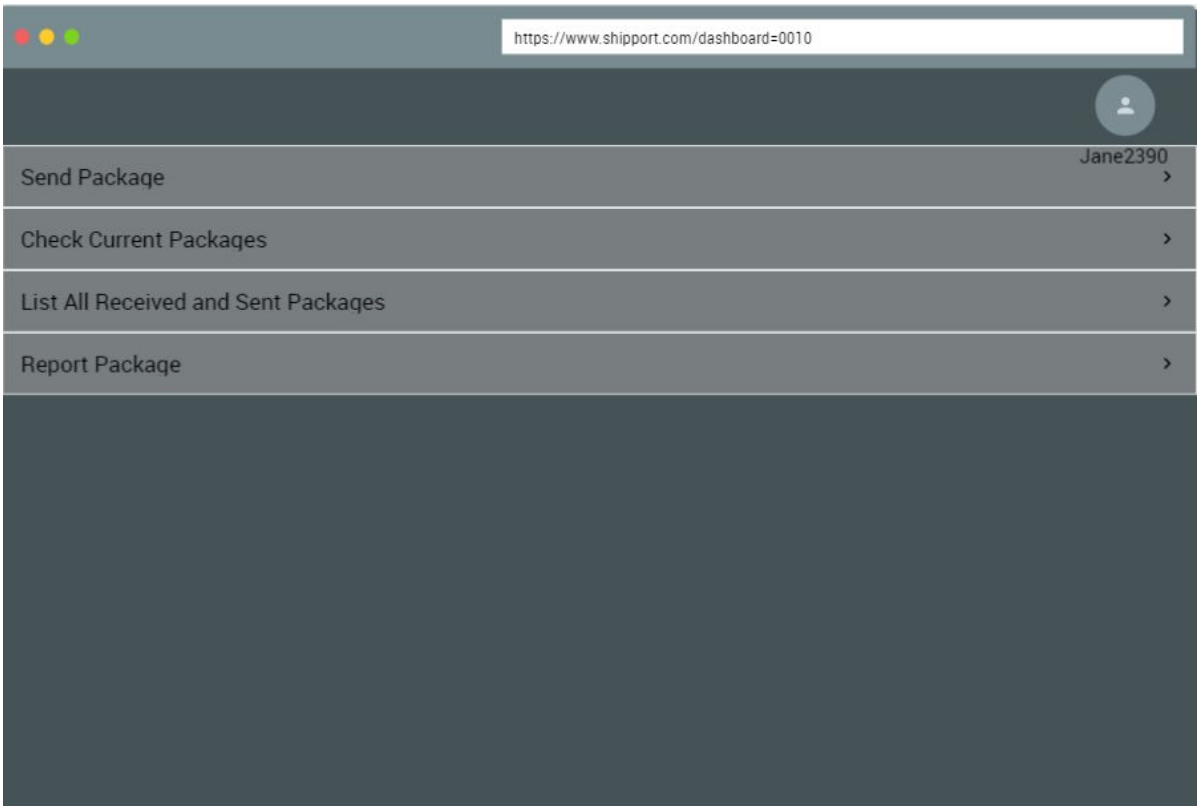


Figure 6.2.1: Customer Dashboard

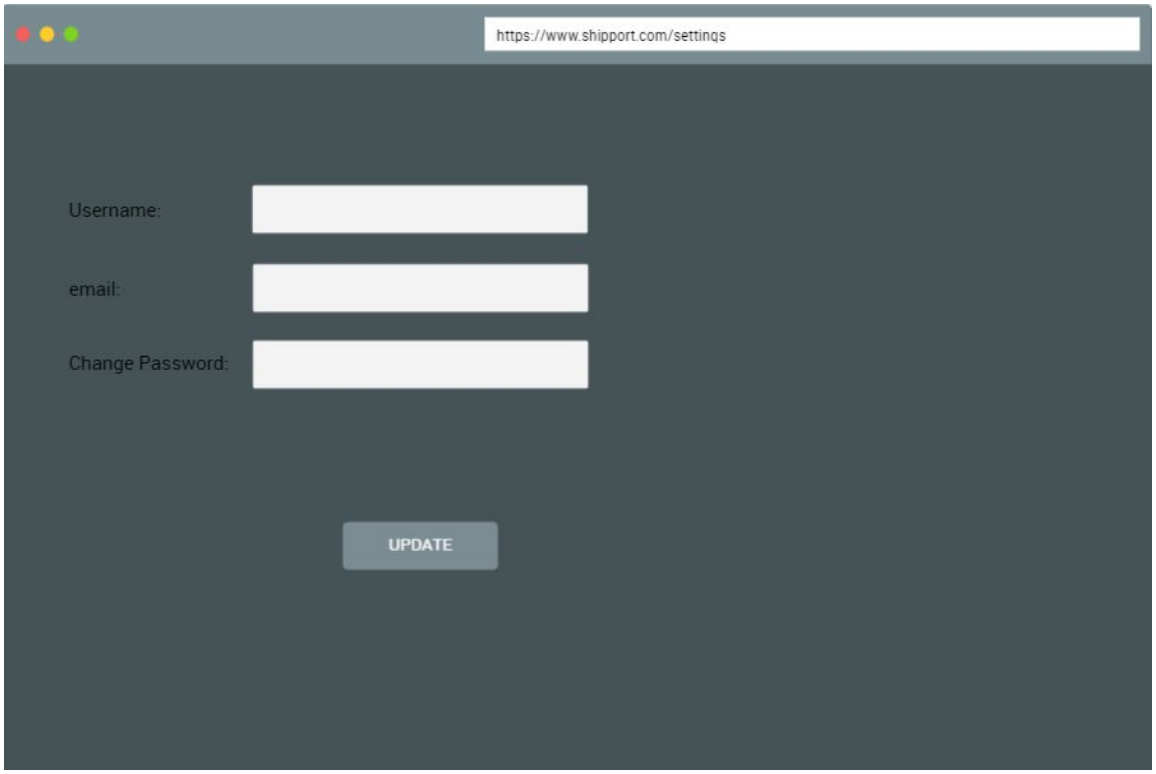


Figure 6.2.2: Customer Settings

6.3 Send Package

Customers can send a package when they open this page. They need to specify the receiving user's username, and the location they want it to be delivered. They can choose the package type from the dropdown menu; which lists all packaging types such as; cardboard, plastic bag, etc… These options will affect the price as well.

They can add a description for the order as well, for the receiving customer. They also should choose the city so that the delivery can be arranged for it by the system automatically.

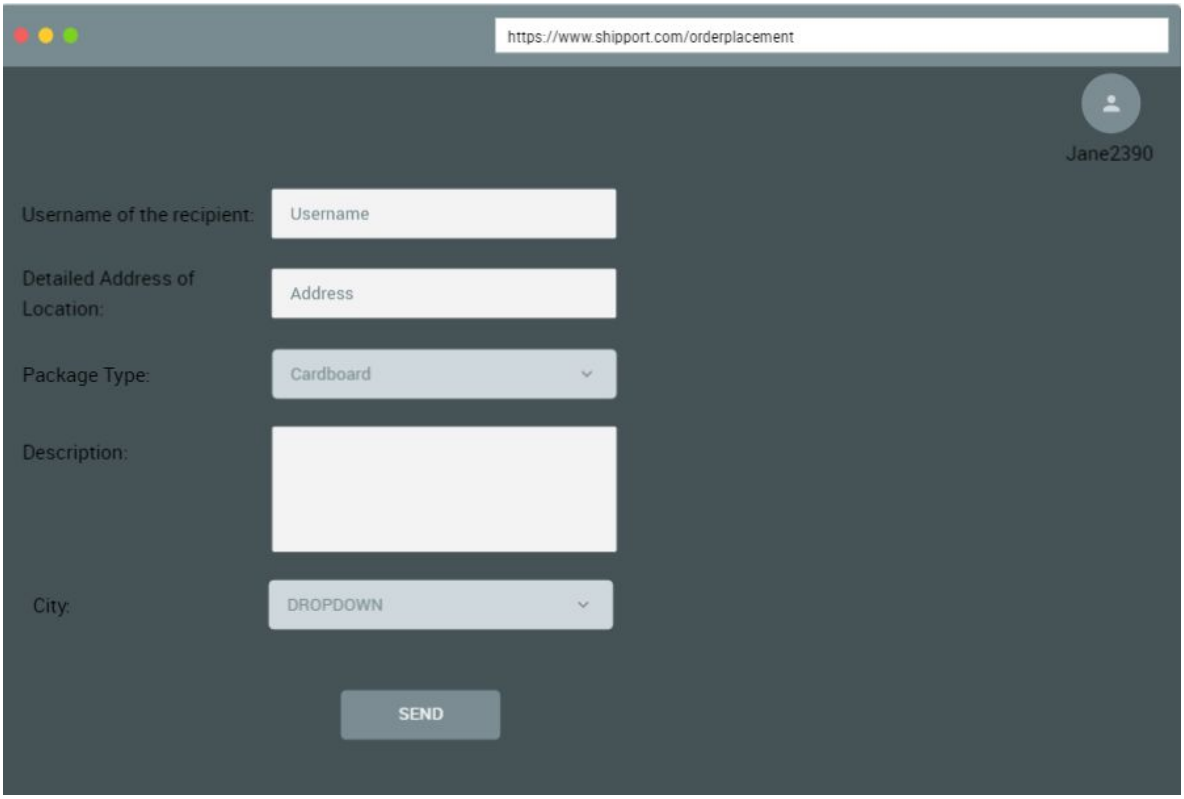


Figure 6.3.1: Send Package

6.4 Check Current Packages

Upon clicking on "list current packages" button, the customer will be directed here. They can see the status of the package, who it is from and who it is for. They can click on one of them to see the full details about them; provided by the customer sending the package, such as; city, full address, package type in addition to these three; sender, receiver name and status.

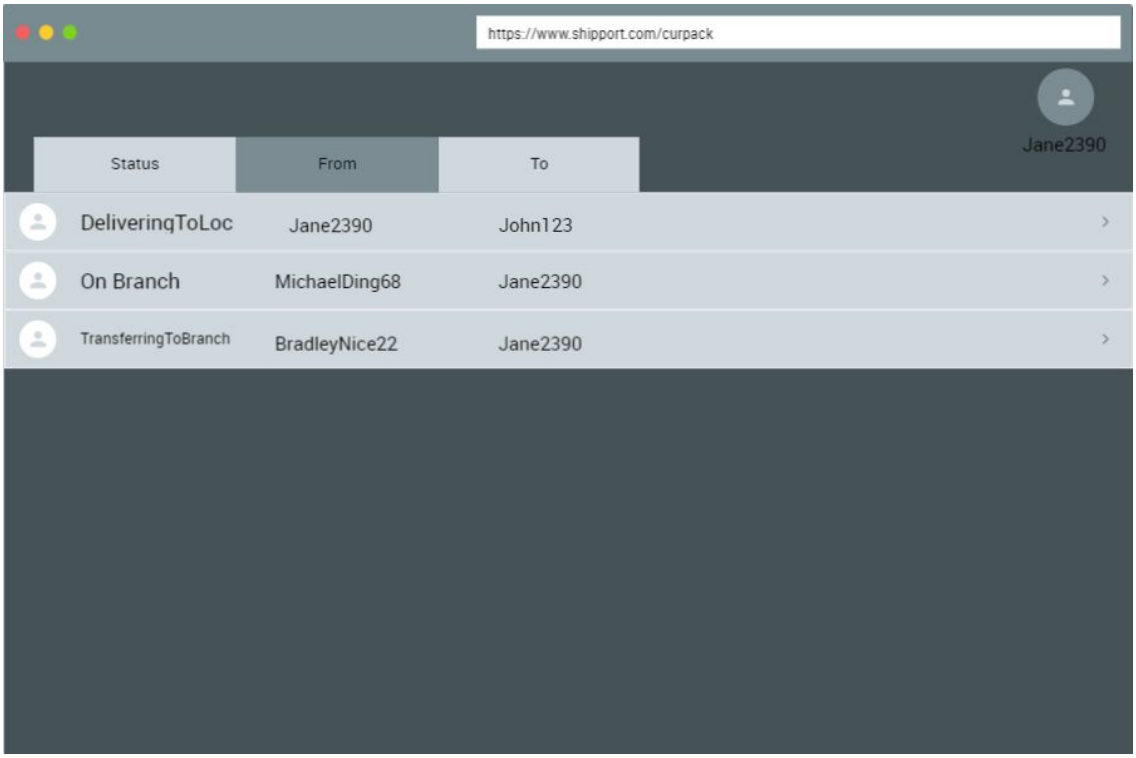


Figure 6.4.1: List Current Packages

Figure 6.4.2: Package Details

6.5 List All Received and Sent Packages

Customers can view their packages history from here. It is basically the same as list current packages but they have been delivered, so they don't have a status attribute. These packages can be clicked to view further info as well, but it is the same as the previous one so the UI isn't included.

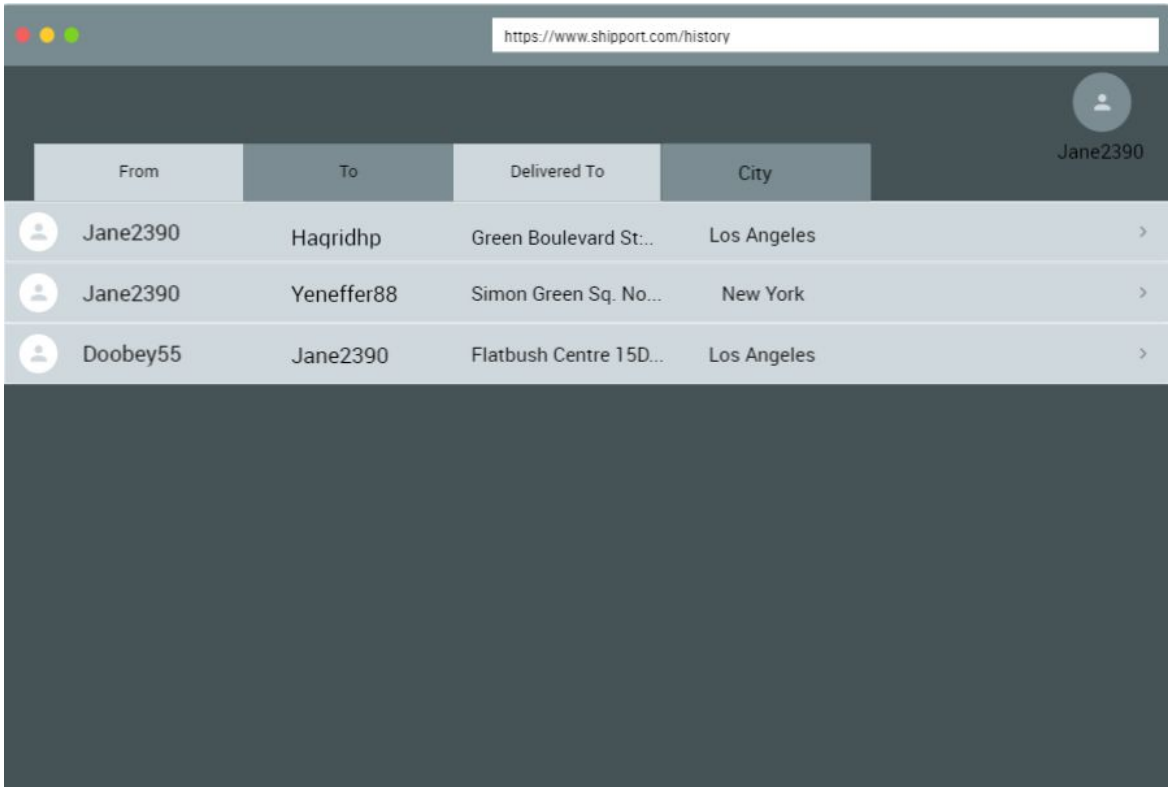


Figure 6.5.1: History of Packages

6.6 Report Package

If the receiving customer finds damage on the package they can decline the delivery and report the package or if the package is missing they can report as well. They will have to specify the reason for the report from the dropdown menu at the bottom. The menu includes options such as; damage to the package, damage to the product, missing package. When they report it, the report will be added to the reports depository which will be handled by employees later on.

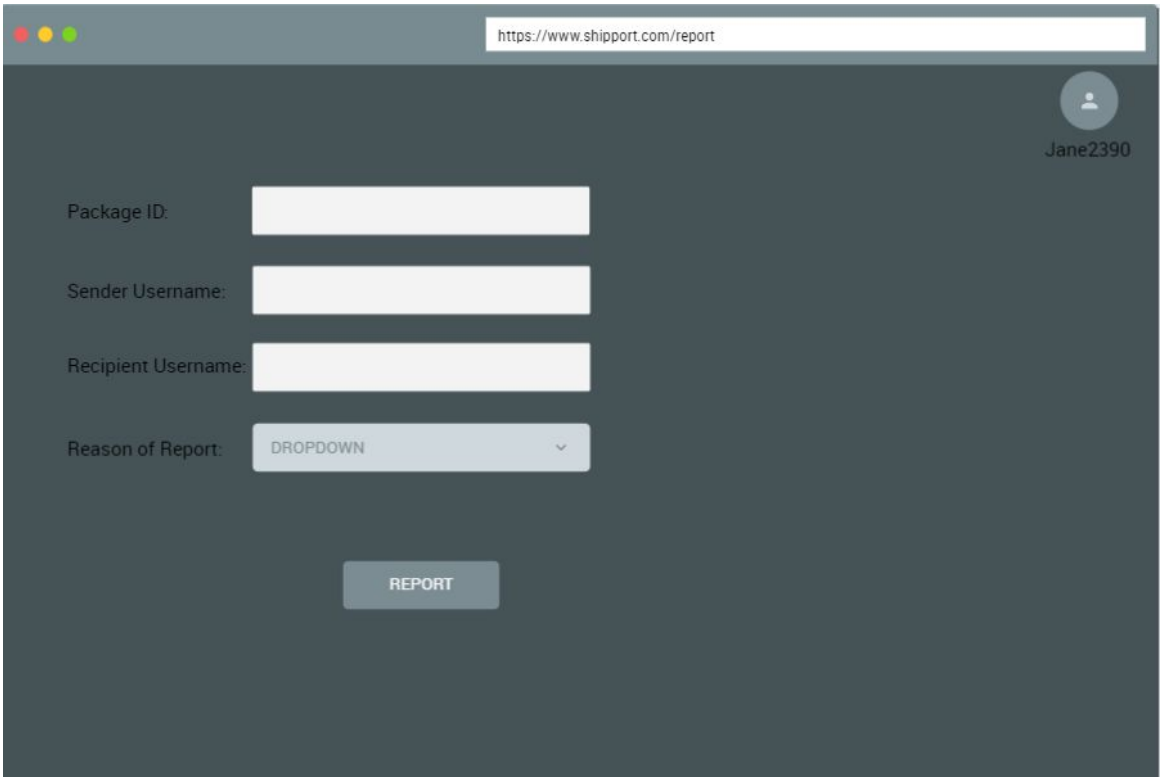


Figure 6.6.1: Report Package

6.7 Employee Dashboard

Employees can handle reports, filed by customers. When they list the reports they will be directed to all reports page. All the reports have information about them, including date and time, report id, sender id, receiver id, package id, and the status of the report which is separated in three ways; open(can be claimed by any employee), onOthers(one employee has claimed the report and is already working on it), and Handled (this report has been handled and it does not need further action).

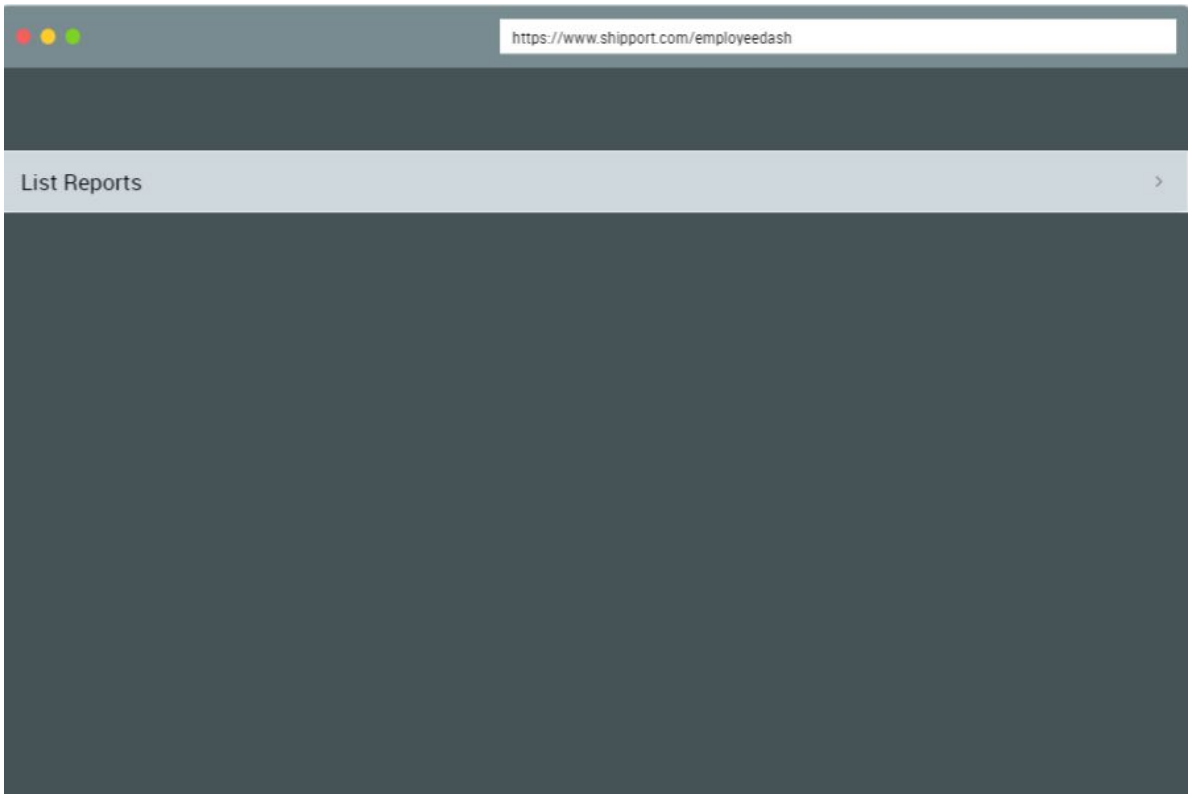
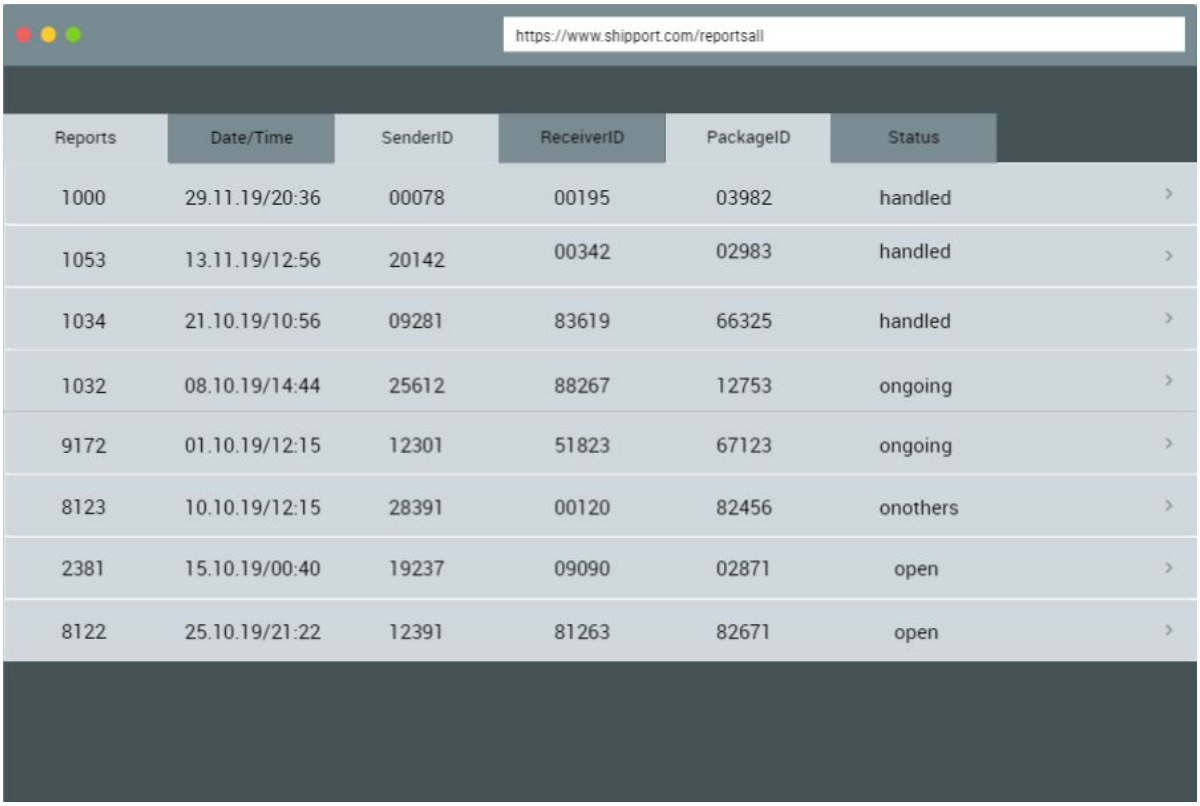


Figure 6.7.1: Employee Dashboard



Reports	Date/Time	SenderID	ReceiverID	PackageID	Status
1000	29.11.19/20:36	00078	00195	03982	handled
1053	13.11.19/12:56	20142	00342	02983	handled
1034	21.10.19/10:56	09281	83619	66325	handled
1032	08.10.19/14:44	25612	88267	12753	ongoing
9172	01.10.19/12:15	12301	51823	67123	ongoing
8123	10.10.19/12:15	28391	00120	82456	onothers
2381	15.10.19/00:40	19237	09090	02871	open
8122	25.10.19/21:22	12391	81263	82671	open

Figure 6.7.2: All Reports

6.8 Handling Report

When the employee opens an open report, they see this view. They can see all of the necessary information and they can even message both of the customers, for further info or to inform them about the process. The messaging screen can be viewed in 6.8.2.

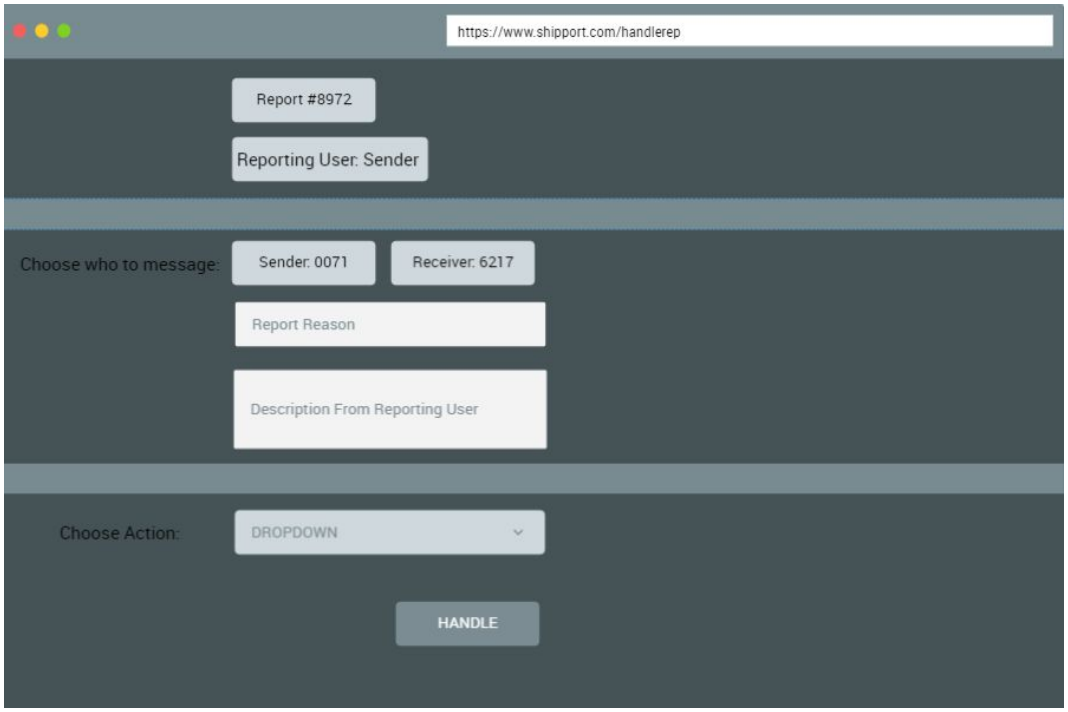


Figure 6.8.1: Handling Report

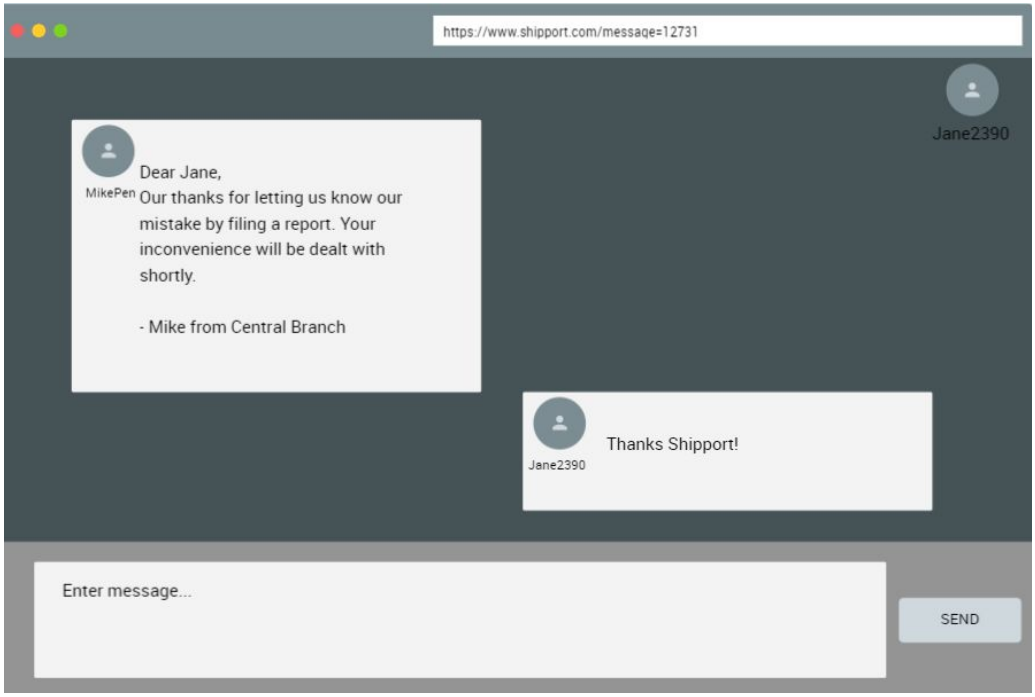


Figure 6.8.2: Message View

6.9 Courier View

Couriers are only able to see what deliveries are given to them and update the status of these deliveries. The status attribute is also useful for couriers since they have to pick a package from their branch once it arrives.

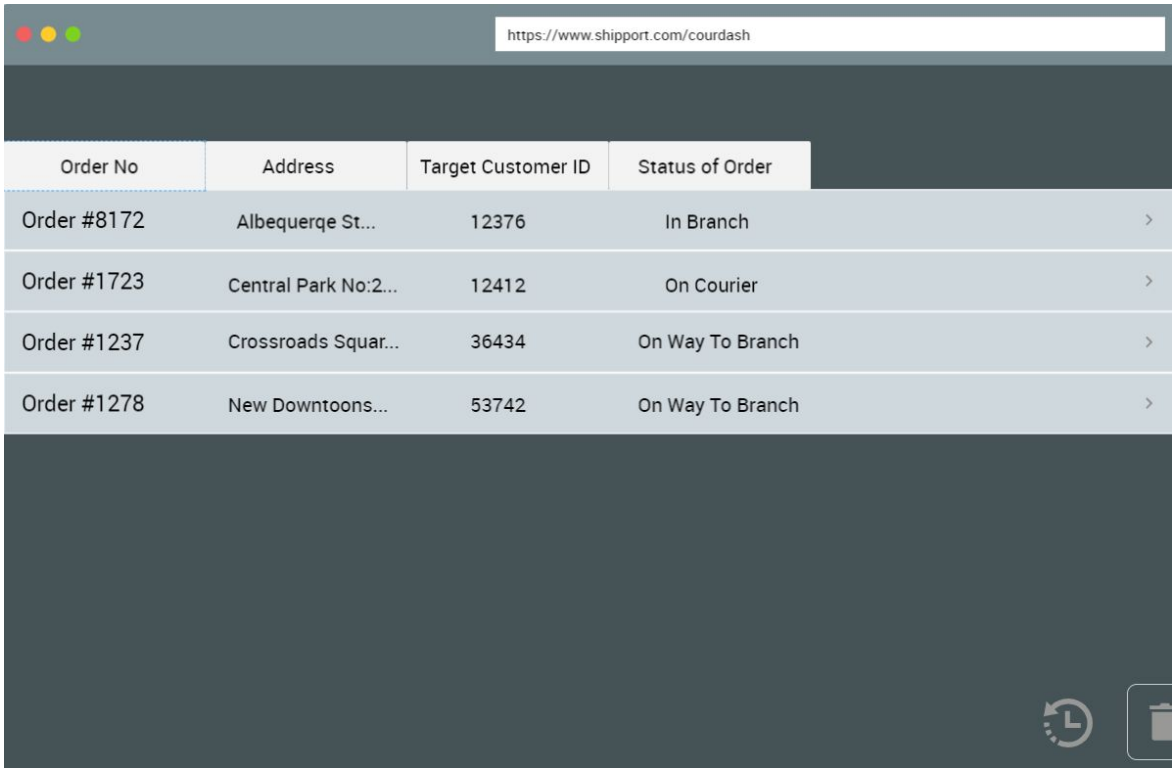


Figure 6.9.1: Courier View