Bilkent University
Department of Computer Engineering

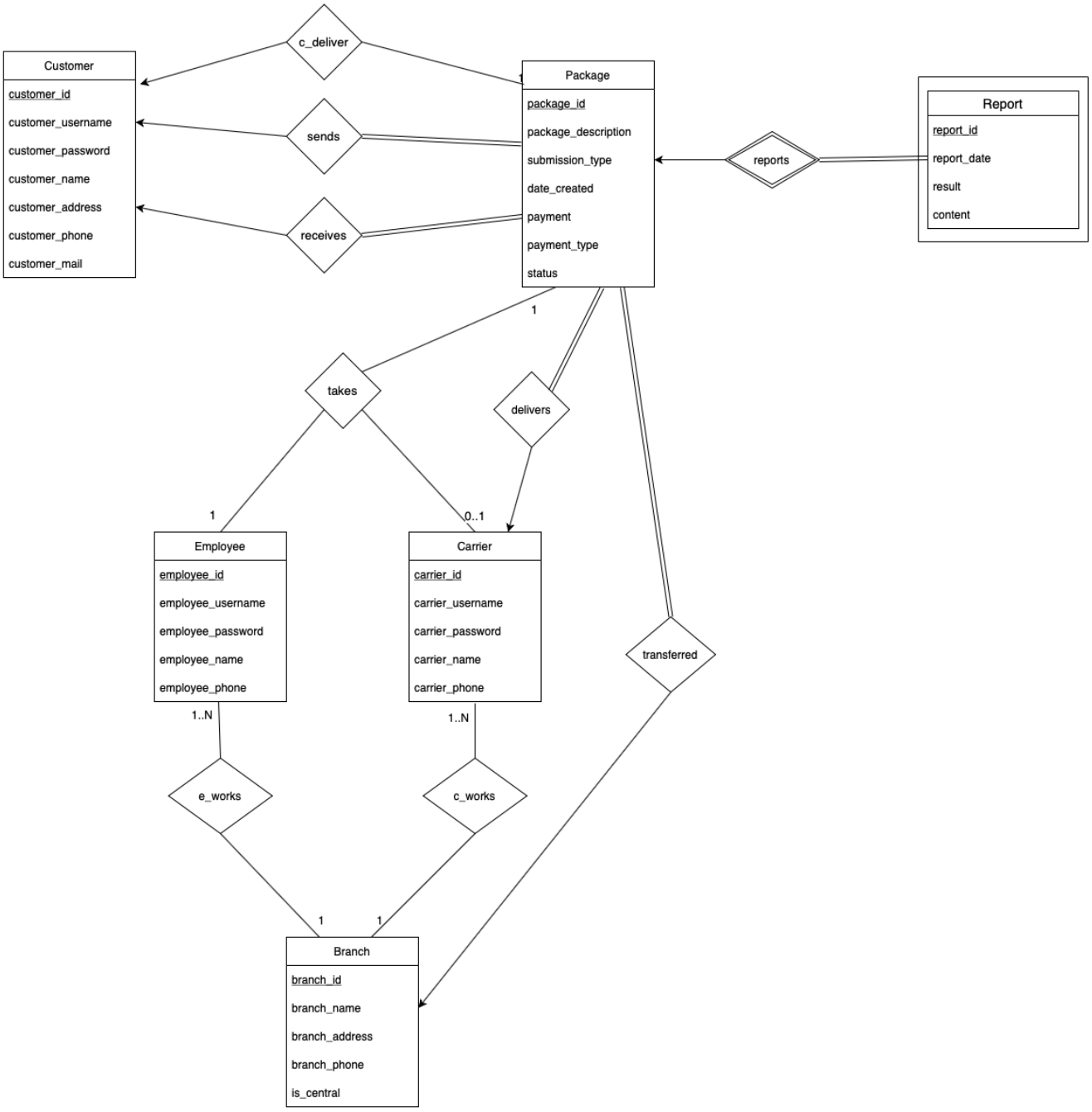
CS 353 Term Project Design Report
Shipping Company Data Management System
Group 3

Assigned TA
Mustafa Can Çavdar

Team Members
Saidcan Alemdaroğlu
Zeynep Korkunç
Samir Süleymanlı
Bartu Teber

Table of Contents

Revised ER Model



Changes in Revised ER Model

We added an extra functionality that allows customers to deliver packages and work as a carrier in order to gain money. Customers can see undelivered packages of all branches and can demand to deliver the package. If a package will be delivered by a customer instead of a carrier, the package will not participate in 'delivers' relationship and participates in c_deliver relationship instead. 'takes' relationship represents which employee the package is submitted to and which carrier the employee assigns to receive the package. So, 'carrier' entity set participates in the relationship just if the customer demands a carrier to receive the package. 'report' is a weak entity set since it cannot exist without a package to report. 'deliver' relationship represents which carrier will deliver the package to destination.

Relational Model

Entity Sets

Customer(<u>customer_id</u>, customer_username, customer_password, customer_name, customer_adress, customer_phone)

Candidate keys -> customer_id
Primary key -> customer_id

FD -> customer_id -> customer_username, customer_password, customer_name, customer_adress, customer_phone

Normal Form -> BCNF

Create table **Customer**(customer_id int,
customer_username varchar(15) not NULL,
customer_password varchar(10) not NULL,
customer_name varchar(30) not NULL,
customer_adress varchar(50) not NULL,

```
    customer_phone varchar(12) not NULL
    primary key(customer_id)) ;
```

Employee(<u>employee_id</u>, employee_username, employee_password, employee_name, employee_phone, branch_id)

Candidate keys -> employee_id
Primary key -> employee_id

FD -> employee_id-> employee_username, employee_password, employee_name, employee_phone

Normal Form -> BCNF

branch_id is FK to Branch table

```
Create table Employee(    employee_id int,
                          employee_username varchar(15) not NULL,
                          employee_password varchar(10) not NULL,
                          employee_name varchar(30) not NULL,
                          employee_phone varchar(12) not NULL,
                          branch_id int,
                          primary key(employee_id),
                          foreign key( branch_id) references Branch ) ;
```

Carrier(<u>carrier_id</u>, carrier_username, carrier_password, carrier_name, carrier_phone, branch_id)

Candidate keys -> carrier_id
Primary key -> carrier_id

FD -> carrier_id-> carrier_username, carrier_password, carrier_name, carrier_phone

Normal Form -> BCNF

branch_id is FK to Branch table

```
Create table Carrier(     carrier_id int,
                          carrier_username varchar(15) not NULL,
                          carrier_password varchar(10) not NULL,
                          carrier_name varchar(30) not NULL,
                          carrier_phone varchar(12) not NULL,
                          branch_id int,
                          primary key(carrier_id),
                          foreign key( branch_id) references Branch ) ;
```

Branch(<u>branch_id</u>, branch_name, branch_address, branch_phone, is_central)

Candidate keys -> branch_id
Primary key -> branch_id

FD -> branch_id-> branch_name, branch_address, branch_phone, is_central

Normal Form -> BCNF

```
Create table Branch(    branch_id int,
                        branch_name varchar(30) not NULL,
                        branch_address varchar(30) not NULL,
                        branch_phone varchar(12) not NULL,
                        is_central char(1) not NULL,
                        primary key(branch_id)) ;
```

Package(<u>package_id</u>, package_description, submission_type, date, payment, status, sender_id, receiver_id, customer_deliver, courier_deliver, transferred_branch)

Candidate keys -> package_id
Primary key -> package_id

FD -> package_id-> package_description, submission_type, date, payment, status, sender_id, receiver_id

Normal Form -> BCNF

sender_id is FK to Customer table
receiver_id is FK to Customer table
customer_deliver is FK to Customer table
courier_deliver is FK to Courier table
transferred_branch is FK to Branch table

Create table **Package**(package_id int,
package_description varchar(30) not NULL,
submission_type varchar(30) not NULL,
date date_created not NULL,
payment varchar(15) not NULL,
status varchar(15) not NULL,
report_id int,
sender_id int,
receiver_id int,
Customer_carrier int,
transferred_branch int,
primary key(package_id),
foreign key(sender_id, receiver_id, customer_carrier) references Customer)
Foreign key(transferred_branch) references Branch) ;

Report(report_id, package_id, date, result, content)

Candidate keys -> report_id, package_id
Primary key ->report_id, package_id

FD -> report_id, package_id, date, result, content

Normal Form -> BCNF

package_id is FK to Package table

```
Create table Report(        report_id int,
                            package_id int,
                            report_date date not NULL,
                            result varchar(15) not NULL,
                            content varchar(30) not NULL,
                            primary key (report_id,package_id),
                            foreign key (package_id) references Package,
                            on delete cascade) ;
```

Relations

Takes(<u>package_id, employee_id, carrier_id</u>)

Candidate keys -> package_id, employee_id, carrier_id
Primary key -> package_id, employee_id, carrier_id

FD -> package_id, employee_id, carrier_id -> package_id, employee_id, carrier_id

Normal Form -> BCNF

package_id is FK to Package table
employee_id is FK to Employee table

carrier_id is FK to Carrier table

Create table **Takes**(package_id int,
employee_id int,
Courier_id int,
primary key(package_id, employee_id, Courier_id),
foreign key (package_id) references Package,
foreign key (employee_id) references Employee,
foreign key (Courier_id) references Courier)) ;

Sends(<u>customer_id, package_id</u>)

Candidate keys -> customer_id, package_id
Primary key -> customer_id, package_id

FD -> None

Normal Form -> BCNF

package_id is FK to Package table
customer_id is FK to Customer table

Receives(<u>customer_id, package_id</u>)

Candidate keys -> customer_id, package_id
Primary key -> customer_id, package_id

FD -> None

Normal Form -> BCNF

package_id is FK to Package table
customer_id is FK to Customer table

c_delivers(<u>customer_id, package_id</u>)

Candidate keys -> customer_id, package_id
Primary key -> customer_id, package_id

FD -> None

Normal Form -> BCNF

package_id is FK to Package table
customer_id is FK to Customer table

delivers(<u>carrier_id, package_id</u>)

Candidate keys -> carrier_id, package_id
Primary key -> carrier_id, package_id

FD -> None

Normal Form -> BCNF

package_id is FK to Package table

carrier_id is FK to Carrier table

transferred(package_id, branch_id)

Candidate keys -> package_id, branch_id
Primary key -> package_id, branch_id

FD -> None

Normal Form -> BCNF

package_id is FK to Package table
branch_id is FK to Branch table

reports(package_id, report_id)

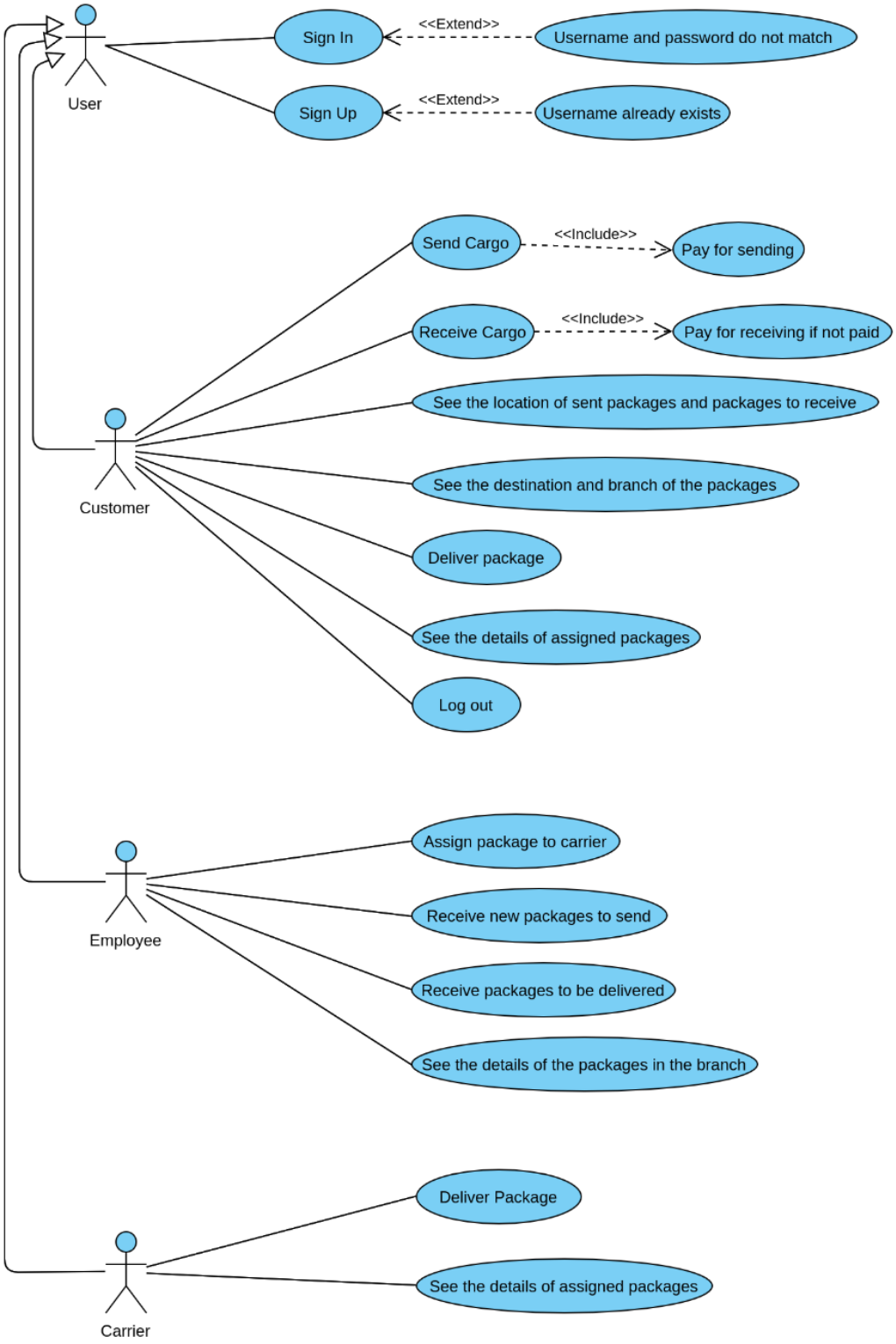
Candidate keys -> package_id, report_id
Primary key ->package_id, report_id

FD -> None

Normal Form -> BCNF

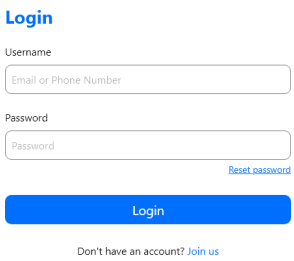
package_id is FK to Package table
report_id is FK to Report table

Use Case Diagram



User Interface Design & Corresponding SQL Statements

Login

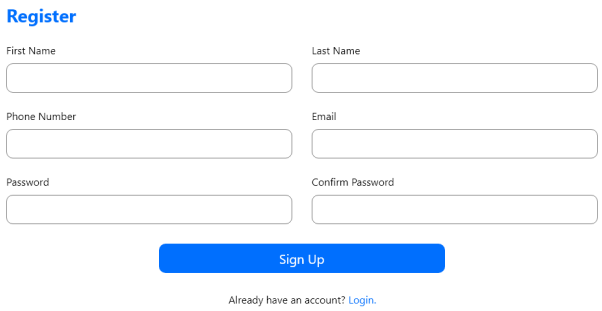


SELECT * FROM Customer WHERE (@username = customer_username AND sha(@password) = customer_password)

SELECT * FROM Employee WHERE (@username = employee_username AND sha(@password) = employee_password)

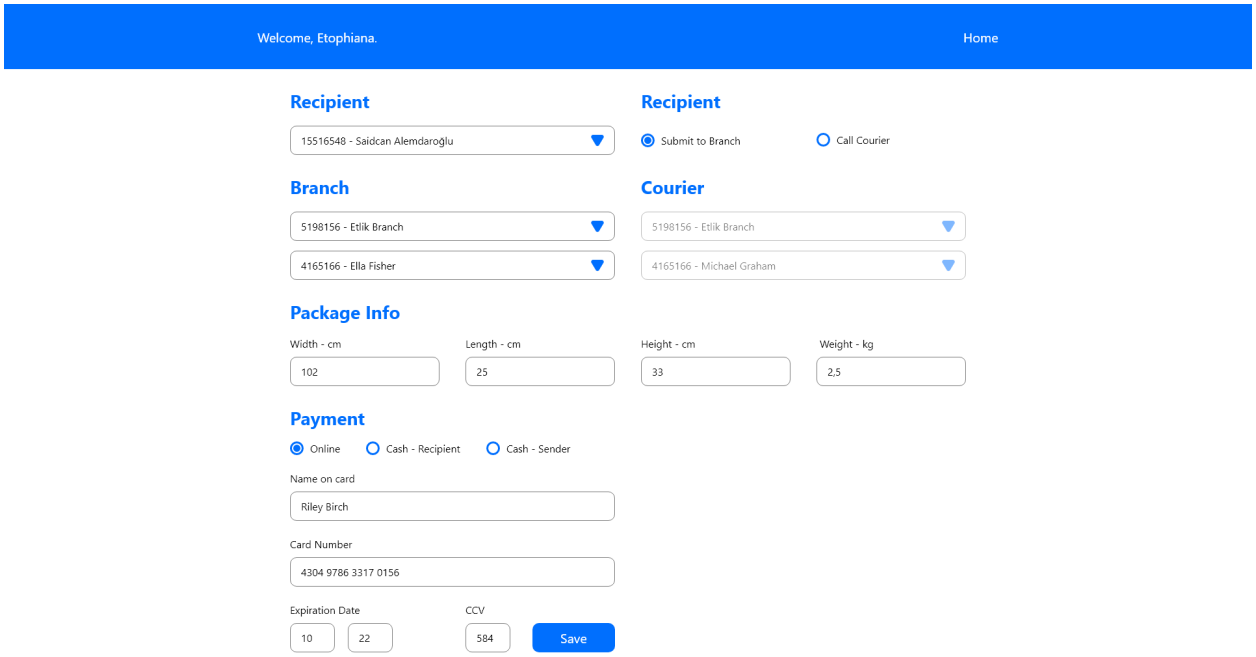
SELECT * FROM Courier WHERE (@username = courier_username AND sha(@password) = courier_password)

Sign Up



INSERT INTO customers(username, password, name, mail) VALUES (@username, @sha(password), @name, @mail)

Customer - Send a package



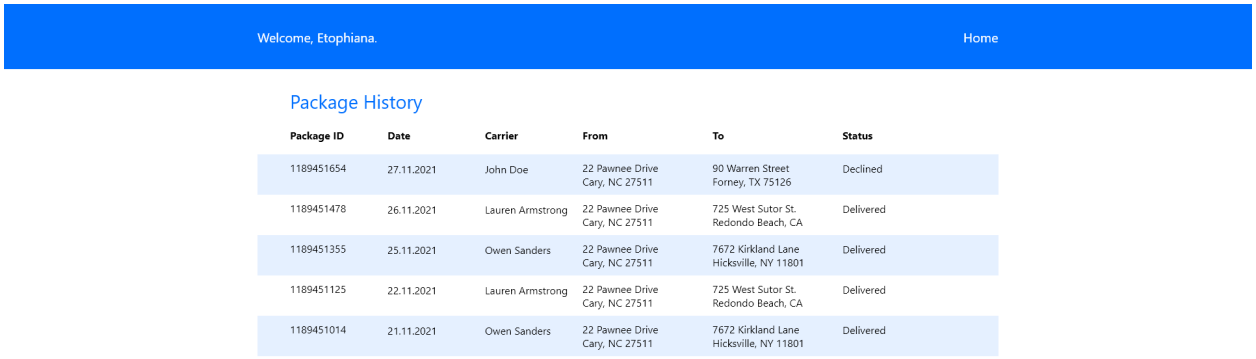
SELECT customer_id, customer_name FROM Customer

SELECT branch_id, branch_name FROM Branch

SELECT employee_id, employee_name FROM Employee WHERE branch_id = @branch_id

SELECT Carrier_id, carrier_name FROM Carrier WHERE branch_id = @branch_id

Customer - Package History

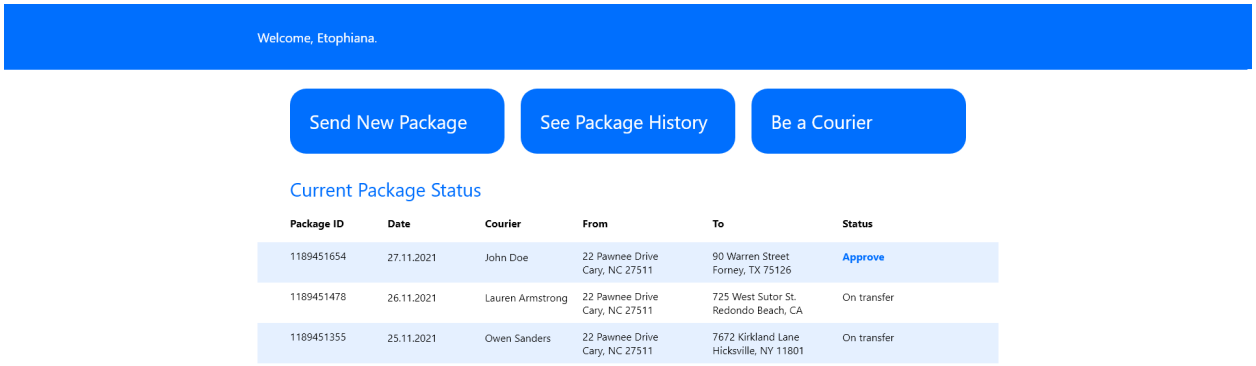


SELECT p.package_id, p.date_created, carrier.carrier_name, deliverer.customer_name, sender.address, receiver.address, p.status

FROM Package p, Carrier carrier, Customer deliverer, Customer sender, Customer receiver

WHERE p.sender_id = sender.customer_id AND p.receiver_id = receiver.customer_id AND (p.carrier_deliver = carrier.carrier_id OR p.customer_deliver = deliverer.customer_id) and (p.sender_id = @my_id OR p.receiver_id = @my_id)

Customer - Current packages (Home)

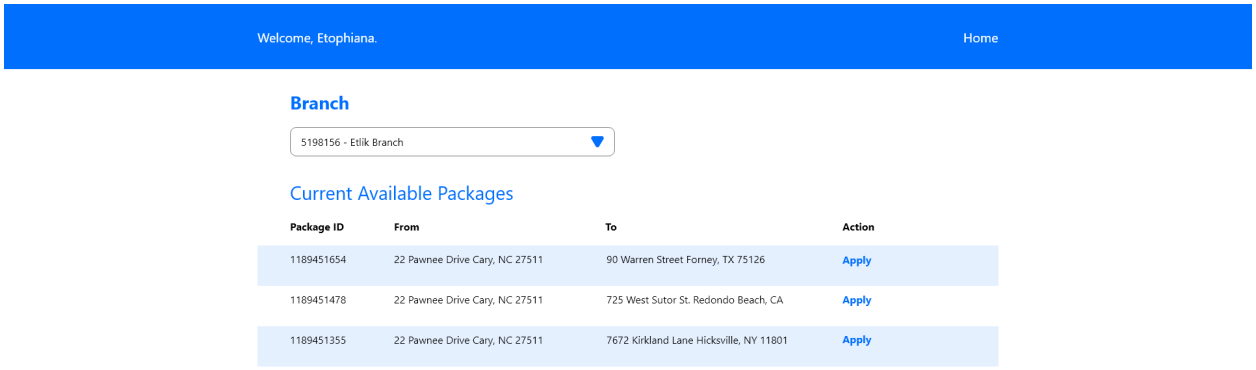


SELECT p.package_id, p.date_created, carrier.carrier_name, deliverer.customer_name, sender.address, receiver.address, p.status

FROM Package p, Carrier carrier, Customer deliverer, Customer sender, Customer receiver

WHERE p.sender_id = sender.customer_id AND p.receiver_id = receiver.customer_id AND (p.carrier_deliver = carrier.carrier_id OR p.customer_deliver = deliverer.customer_id) AND (p.sender_id = @my_id OR p.receiver_id = @my_id) AND p.status <> “Delivered”

Customer - Be a Courier



Get all the branches

SELECT branch_id, branch_name
FROM Branch

Get all the packages that are waiting for delivery

SELECT p.package_id, sender.address, receiver.address, p.status
FROM Package p, Customer sender, Customer receiver
WHERE p.sender_id = sender.customer_id AND p.receiver_id = receiver.customer_id AND p.transferred_branch = @branch_id AND p.status = “Waiting for Delivery”

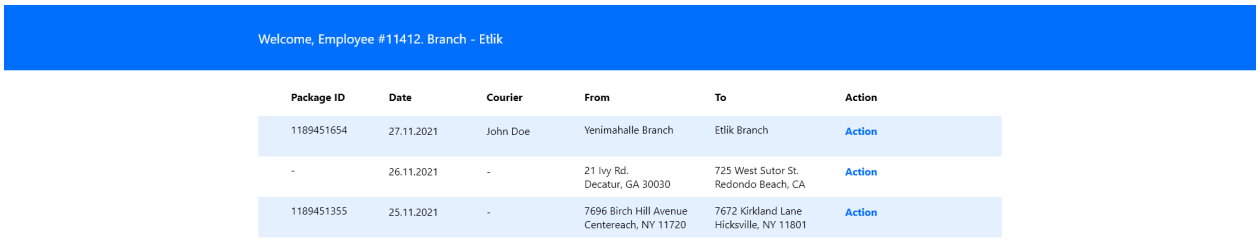
Update status and customer_delivery fields of the package

UPDATE Package SET status = “On Delivery” and customer_deliver = @my_id WHERE package_id = @package_id

Customer - Report


```
INSERT INTO     Report (content, package_id, curdate())
VALUES          (@content, @package_id)
```

Courier & Employee - Current packages (Home)



Get packages for Employee

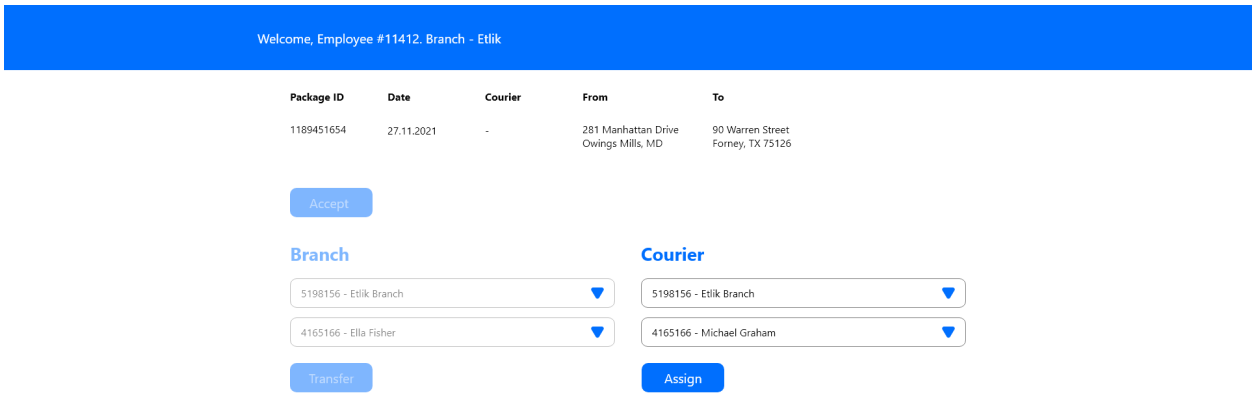
SELECT p.package_id, p.date_created, carrier.carrier_name, deliverer.customer_name, sender.address, receiver.address
FROM Package p, Carrier carrier, Customer deliverer, Customer sender, Customer receiver, Employee employee
WHERE p.sender_id = sender.customer_id AND p.receiver_id = receiver.customer_id AND (p.carrier_deliver = carrier.carrier_id OR p.customer_deliver = deliverer.customer_id) AND p.employee_id = @my_id AND p.status <> “Delivered” AND p.status <> “Declined”

Get packages for Carrier

SELECT p.package_id, p.date_created, carrier.carrier_name, deliverer.customer_name, sender.address, receiver.address
FROM Package p, Carrier carrier, Customer deliverer, Customer sender, Customer receiver, Employee employee

WHERE p.sender_id = sender.customer_id AND p.receiver_id = receiver.customer_id AND (p.carrier_deliver = carrier.carrier_id OR p.customer_deliver = deliverer.customer_id) AND p.carrier_id = @my_id AND p.status <> “Delivered” AND p.status <> “Declined”

Courier & Employee - Package Actions



Transfer package to another branch

UPDATE Package SET transferred_branch = @branch_id WHERE package_id = @package_id

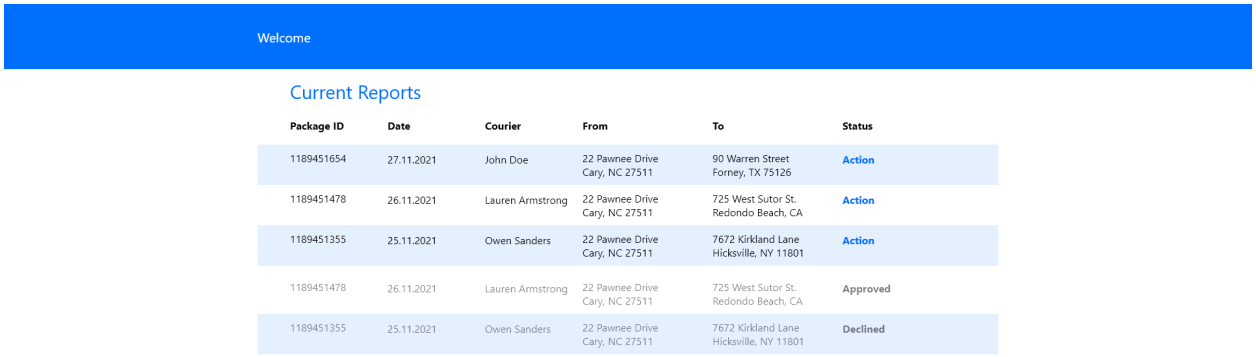
Assign carrier to a package

UPDATE Package SET carrier_deliver = @carrier_id WHERE package_id = @package_id

Update package status to “On Transfer”

UPDATE Package SET status = “On Transfer” WHERE package_id = @package_id

Reports

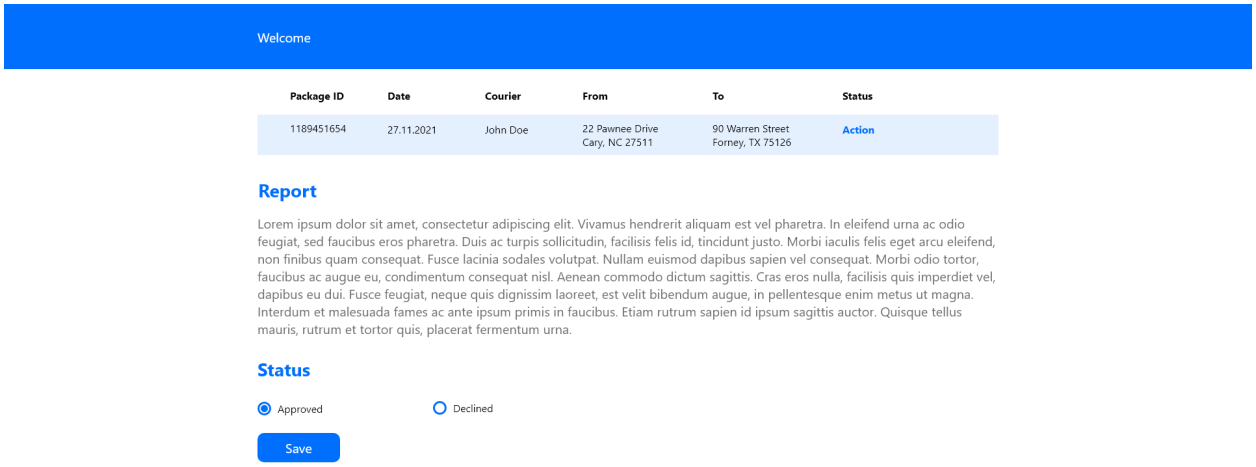


SELECT p.package_id, p.date_created, carrier.carrier_name, deliverer.customer_name, sender.address, receiver.address, p.status

FROM Package p, Carrier carrier, Customer deliverer, Customer sender, Customer receiver, Report report

WHERE p.sender_id = sender.customer_id AND p.receiver_id = receiver.customer_id AND (p.carrier_deliver = carrier.carrier_id OR p.customer_deliver = deliverer.customer_id) and report.package_id = p.package_id

Reports - Actions



UPDATE Report SET status = @status WHERE report_id = @report_id

Implementation Plan

We will implement the backend server using Django and will use MySQL as our database. The frontend will be coded with HTML, CSS, and JavaScript.

Website

https://github.com/SMRSLYMNL/CS_353