CS313: Assignment-7 (Project-II)
Supply Chain Management System

B Siddharth Prabhu
200010003@iitdh.ac.in

Devdatt N
200010012@iitdh.ac.in

30 October 2022

1 Introduction

We have made a Supply Chain Management System using Database concepts, HTML, CSS, and Java technologies. It can be used by Suppliers, Distributors, Retailers and Customers. Further information on the entities can be obtained from the ER diagram attached below.

2 Entity Relationship Diagram

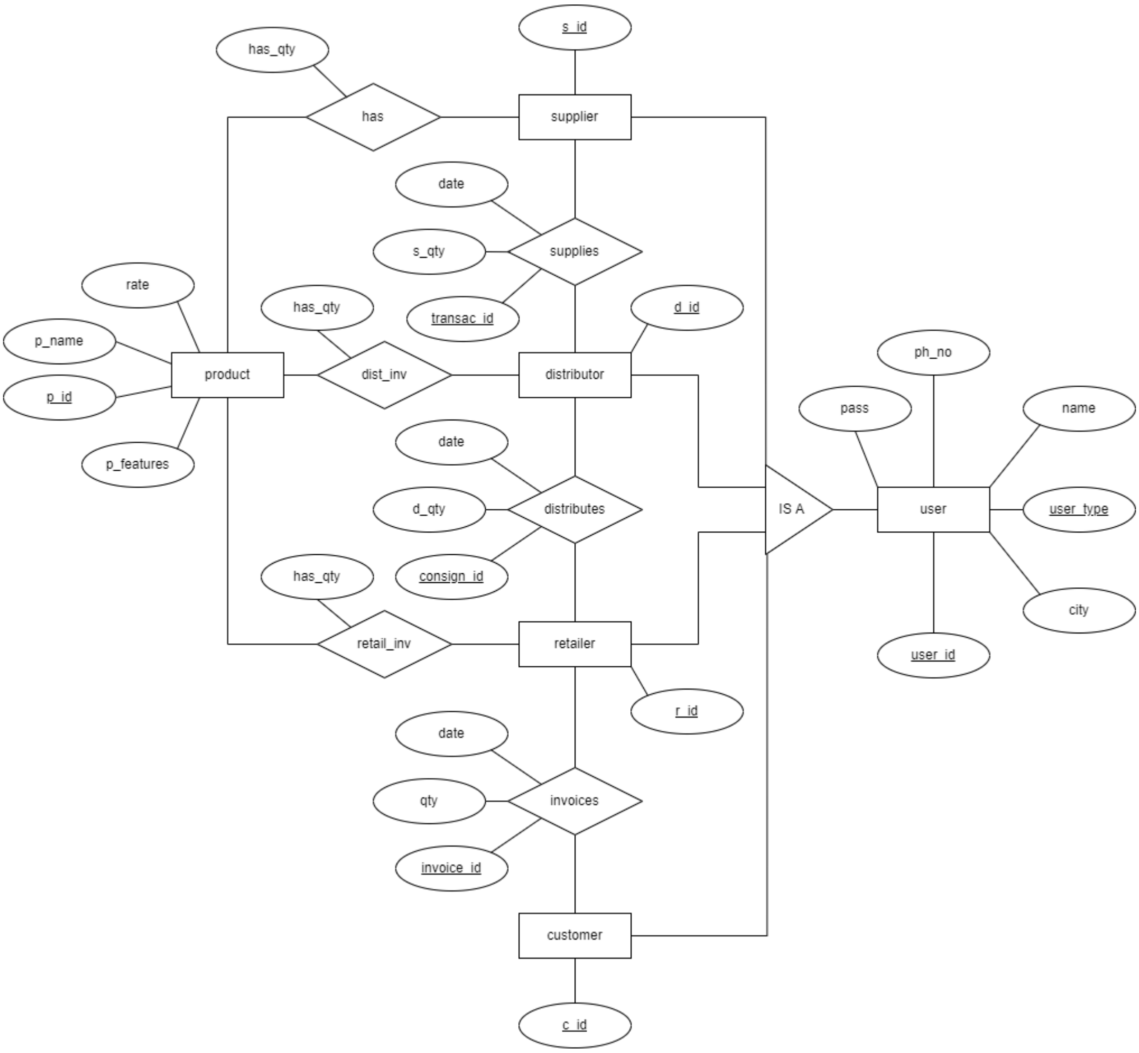


Figure 1: Entity-Relationship Diagram of the `supplychainDB` Database

The following points are to be noted:

- The entities ‘supplier’, ‘distributor’, ‘retailer’, and ‘customer’ are generalizations of the entity ‘user’. There are different ways to convert such generalization/specialization (IS A) relations into tables. We have made it so that the specialized entities inherit most of the attributes from user, except for `user_id` and `pass` .
- The inventories of the various entities are stored in their relations with ‘product’ table, in ‘has’, ‘dist_inv’, and ‘ret_inv’ respectively.
- The relations ‘supplies’, ‘distributes’, ‘invoices’ have their own primary keys ‘transac_id’, ‘consign_id’, and ‘invoice_id’ respectively.
- In each of the ‘inventories’, records are present only if the quantity of a given product is greater than zero.
- The password is hashed using SHA256 and stored for security reasons. It is not responsible to store raw password data in tables when creating an authenticating database.
- The goal is effectively for suppliers to add products and ship them to distributors. These distributors then distribute these packages among retailers. Finally, retailers send packages to customers.
- This system is to be used as the downstream part of an e-commerce system, where a request would be made from the customer’s side. Here, we only depict the products reaching the customer, assuming that they had been ordered on some other portal.

3 Database Schema and Users

First, we create a user for managing the supply chain database from the backend server :

```
CREATE USER 'susply'@'localhost' IDENTIFIED BY 'sus';
CREATE DATABASE supplychainDB;
GRANT ALL PRIVILEGES ON supplychainDB.* TO 'susply'@'localhost';
```

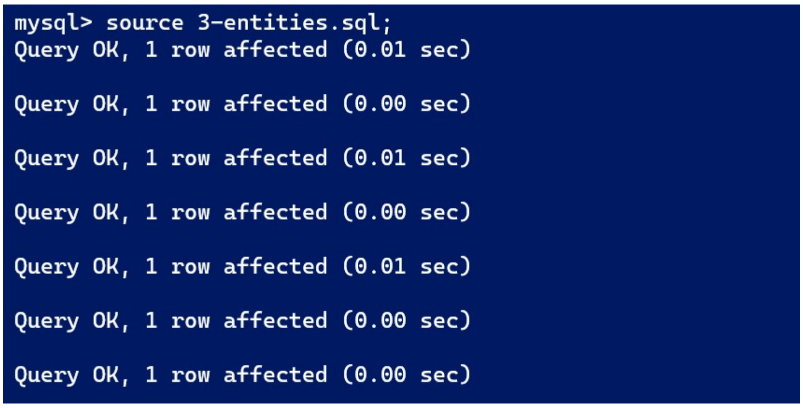
Next, we have the respective schema as shown below. We create the tables accordingly by sourcing the SQL file :

```
DROP DATABASE IF EXISTS supplychainDB;
CREATE DATABASE supplychainDB;

USE supplychainDB;

CREATE TABLE user(
    user_id VARCHAR(7) NOT NULL,
    user_type INT NOT NULL CHECK(user_type>=0 AND user_type<=3),
    pass VARCHAR(64), -- hash and store
    PRIMARY KEY(user_id, user_type)
);
CREATE TABLE supplier(
    s_id VARCHAR(7) PRIMARY KEY,
    s_name VARCHAR(35) NOT NULL,
    s_city VARCHAR(30),
    s_phno NUMERIC(10) NOT NULL,
    FOREIGN KEY(s_id) REFERENCES user(user_id)
);
CREATE TABLE distributor(
    d_id VARCHAR(7) PRIMARY KEY,
    d_name VARCHAR(35) NOT NULL,
    d_city VARCHAR(30),
    d_phno NUMERIC(10) NOT NULL,
    FOREIGN KEY(d_id) REFERENCES user(user_id)
);
```

```sql
CREATE TABLE retailer(
    r_id VARCHAR(7) PRIMARY KEY,
    r_name VARCHAR(35) NOT NULL,
    r_city VARCHAR(30),
    r_phno NUMERIC(10) NOT NULL,
    FOREIGN KEY(r_id) REFERENCES user(user_id)
);
CREATE TABLE customer(
    c_id VARCHAR(7) PRIMARY KEY,
    c_name VARCHAR(35) NOT NULL,
    c_city VARCHAR(30) NOT NULL,
    c_phno NUMERIC(10) NOT NULL,
    FOREIGN KEY(c_id) REFERENCES user(user_id)
);
CREATE TABLE product(
    p_id VARCHAR(7) PRIMARY KEY,
    p_name VARCHAR(35) NOT NULL,
    p_rate NUMERIC(9,2) NOT NULL,
    p_features VARCHAR(60)
);

CREATE TABLE has(
    s_id VARCHAR(7),
    p_id VARCHAR(7),
    has_qty INT CHECK(has_qty>0),
    PRIMARY KEY(s_id, p_id),
    FOREIGN KEY(p_id) REFERENCES product(p_id),
    FOREIGN KEY(s_id) REFERENCES supplier(s_id)
);
CREATE TABLE supplies(
    transac_id VARCHAR(7),
    p_id VARCHAR(7) NOT NULL,
    s_id VARCHAR(7) NOT NULL,
    d_id VARCHAR(7) NOT NULL,
    date DATE,
    s_qty INT CHECK(s_qty>0),
    PRIMARY KEY(transac_id),
    FOREIGN KEY(p_id) REFERENCES product(p_id),
    FOREIGN KEY(s_id) REFERENCES supplier(s_id),
    FOREIGN KEY(d_id) REFERENCES distributor(d_id)
);
CREATE TABLE distributes(
    consign_id VARCHAR(7),
    p_id VARCHAR(7) NOT NULL,
    d_id VARCHAR(7) NOT NULL,
    r_id VARCHAR(7) NOT NULL,
    date DATE,
    d_qty INT CHECK(d_qty>0),
    PRIMARY KEY(consign_id),
    FOREIGN KEY(p_id) REFERENCES product(p_id),
    FOREIGN KEY(r_id) REFERENCES retailer(r_id),
    FOREIGN KEY(d_id) REFERENCES distributor(d_id)
);
CREATE TABLE invoice_line(
    inv_line_id VARCHAR(7),
    p_id VARCHAR(7) NOT NULL,
    r_id VARCHAR(7) NOT NULL,
    c_id VARCHAR(7) NOT NULL,
    want_qty INT CHECK(want_qty>0),
    PRIMARY KEY(inv_line_id),
    FOREIGN KEY(p_id) REFERENCES product(p_id),
    FOREIGN KEY(r_id) REFERENCES retailer(r_id),
    FOREIGN KEY(c_id) REFERENCES customer(c_id)
);
```

```
CREATE TABLE invoice(
    inv_id VARCHAR(7),
    c_id VARCHAR(7) NOT NULL,
    inv_date DATE,
    PRIMARY KEY(inv_id),
    FOREIGN KEY(c_id) REFERENCES customer(c_id)
);



-- qty is passed on entirely until retailer.
--invoiceline will have qty less than retailer stock.
CREATE TABLE inv_has(
    inv_line_id VARCHAR(7),
    inv_id VARCHAR(7),
    PRIMARY KEY(inv_id, inv_line_id),
    FOREIGN KEY(inv_id) REFERENCES invoice(inv_id),
    FOREIGN KEY(inv_line_id) REFERENCES invoice_line(inv_line_id)
);
-- inventories
CREATE TABLE dist_inv(
    d_id VARCHAR(7),
    p_id VARCHAR(7),
    has_qty INT CHECK(has_qty>0),
    PRIMARY KEY(d_id, p_id),
    FOREIGN KEY(p_id) REFERENCES product(p_id),
    FOREIGN KEY(d_id) REFERENCES distributor(d_id)
);
CREATE TABLE retail_inv(
    r_id VARCHAR(7),
    p_id VARCHAR(7),
    has_qty INT CHECK(has_qty>0),
    PRIMARY KEY(r_id, p_id),
    FOREIGN KEY(p_id) REFERENCES product(p_id),
    FOREIGN KEY(r_id) REFERENCES retailer(r_id)
);
```

Next, we preload the database with some values to start with, which are available in the script files `scripts/3-entities.sql` and `src/4-insert.sql` . They're correctly sourced, as can be seen from the following screenshot :



```
mysql> source 3-entities.sql;
Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)
```

Figure 2: Preloading data

4 Languages and Technologies Used

We have used HTML and CSS assisted with Bootstrap and Pixel UI on the frontend. Java Server Pages (JSP) have been used to dynamically create web content so that web-elements like inventory tables have been rendered at time of execution.

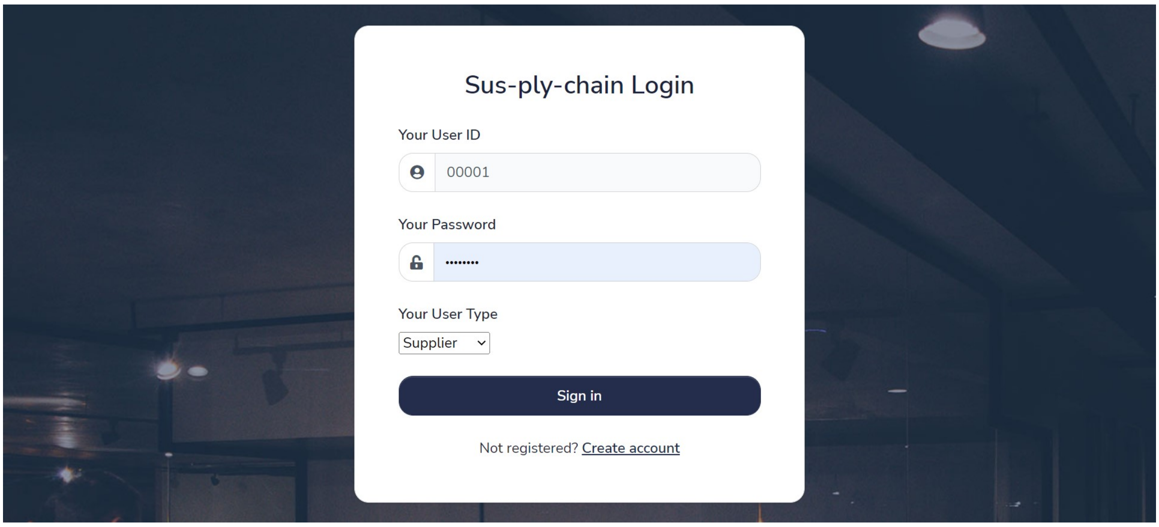
On the backend, we have used Java running on Apache's Tomcat server as a runtime environment with a JDBC connector for connecting to our MySQL database. Our editors of choice have been VS Code and Eclipse, with Eclipse being the main development environment for executing the application. Furthermore, recommended HTML and CSS extensions in VS Code have been used.

5 UI Design and Pages

We have tried to follow a Material UI theme with rounded corners for containers and this has been assisted greatly by Bootstrap and Pixel.

There are four main entities which interact with each other in the Supply Chain - Supplier, Distributor, Retailer and Customer. We shall go through the login page, then each entity's dashboard.

5.a Login Page



Sus-ply-chain Login

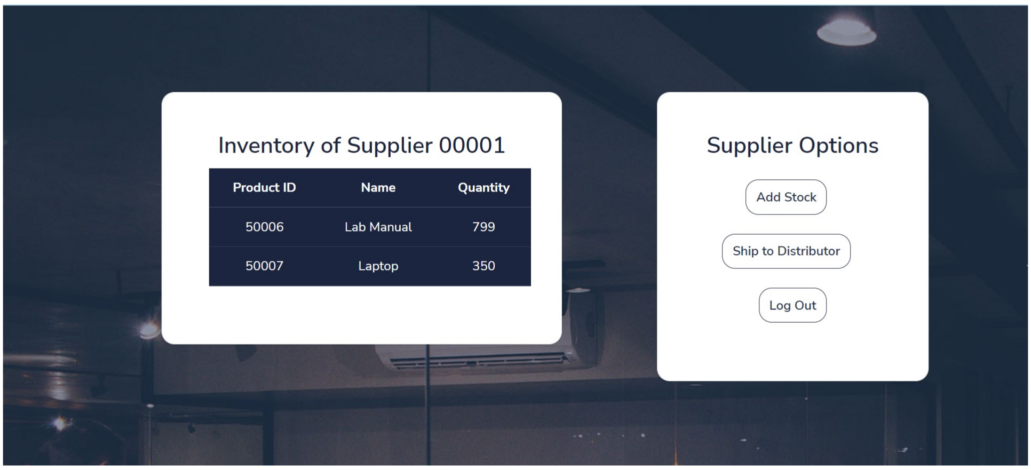
Your User ID: 00001

Your Password: ••••••••

Your User Type: Supplier

Figure 3: Login Page

5.b Supplier Dashboard

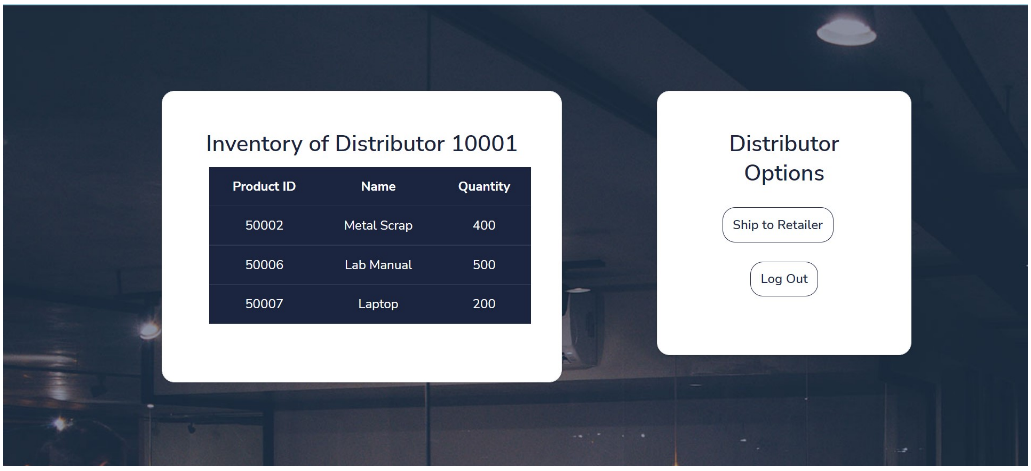


Inventory of Supplier 00001

Product ID	Name	Quantity
50006	Lab Manual	799
50007	Laptop	350

Figure 4: Supplier Dashboard

5.c Distributor Dashboard

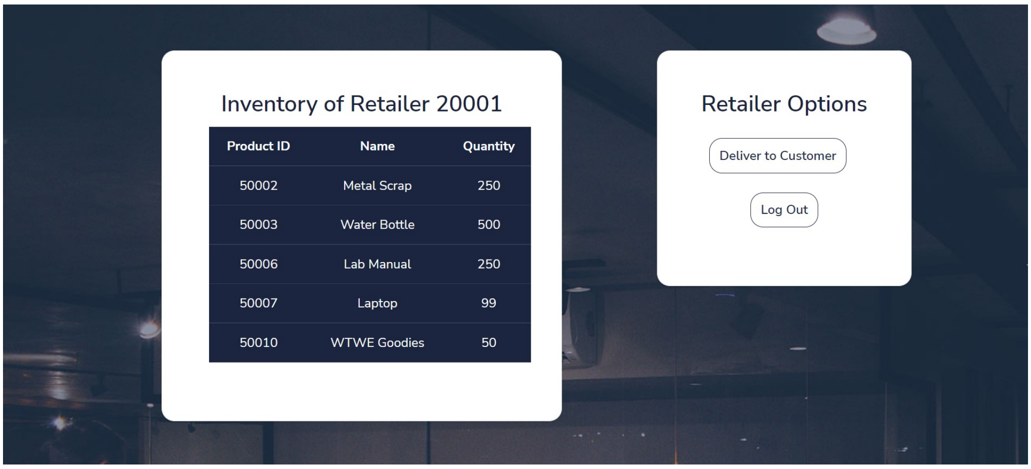


Inventory of Distributor 10001

Product ID	Name	Quantity
50002	Metal Scrap	400
50006	Lab Manual	500
50007	Laptop	200

Figure 5: Distributor Dashboard

5.d Retailer Dashboard

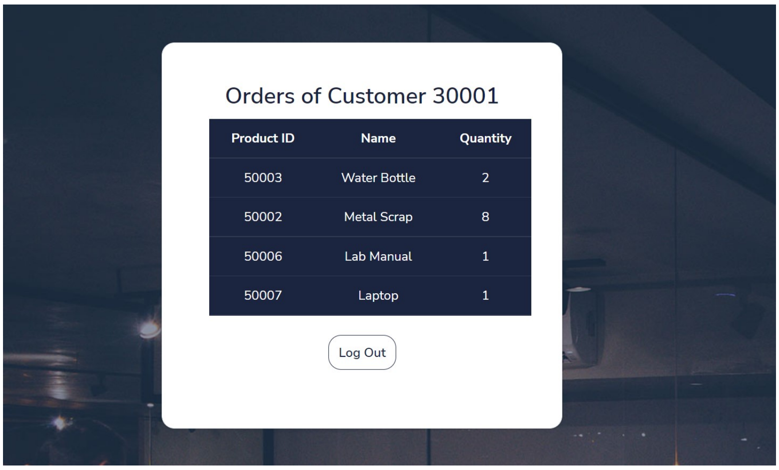


Inventory of Retailer 20001

Product ID	Name	Quantity
50002	Metal Scrap	250
50003	Water Bottle	500
50006	Lab Manual	250
50007	Laptop	99
50010	WTWE Goodies	50

Figure 6: Retailer Dashboard

5.e Customer Dashboard

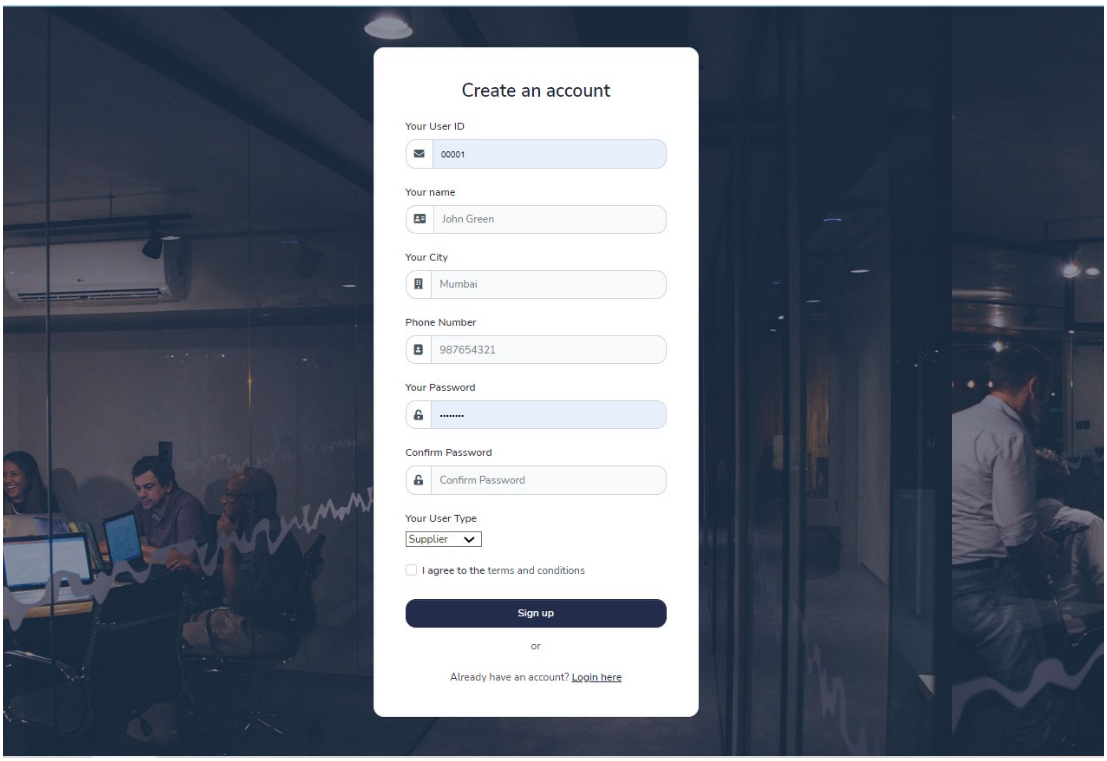


Orders of Customer 30001

Product ID	Name	Quantity
50003	Water Bottle	2
50002	Metal Scrap	8
50006	Lab Manual	1
50007	Laptop	1

Figure 7: Customer Dashboard

5.f Sign Up Page



Create an account

Your User ID: 00001

Your name: John Green

Your City: Mumbai

Phone Number: 987654321

Your Password: ········

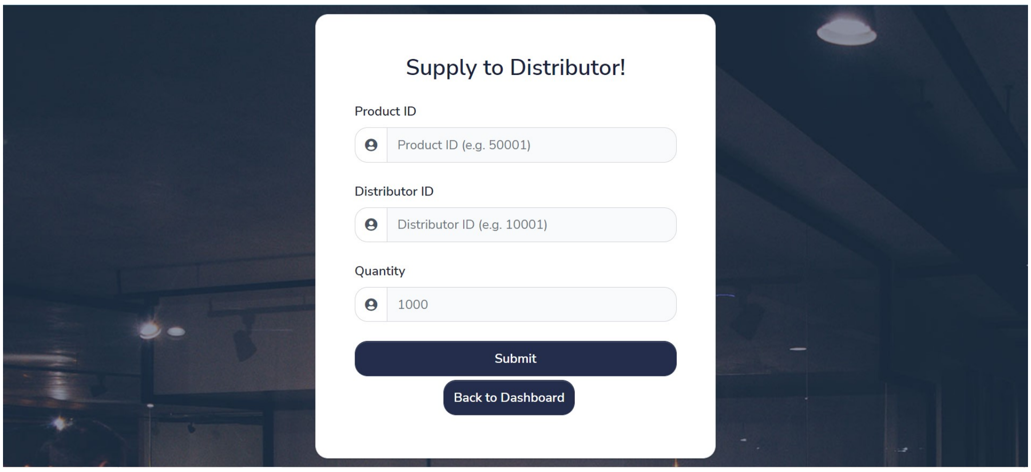
Confirm Password: Confirm Password

Your User Type: Supplier

□ I agree to the terms and conditions

Figure 8: Sign Up Page

5.g Supplier Shipping to Distributor


Supply to Distributor!

Product ID

Product ID (e.g. 50001)

Distributor ID

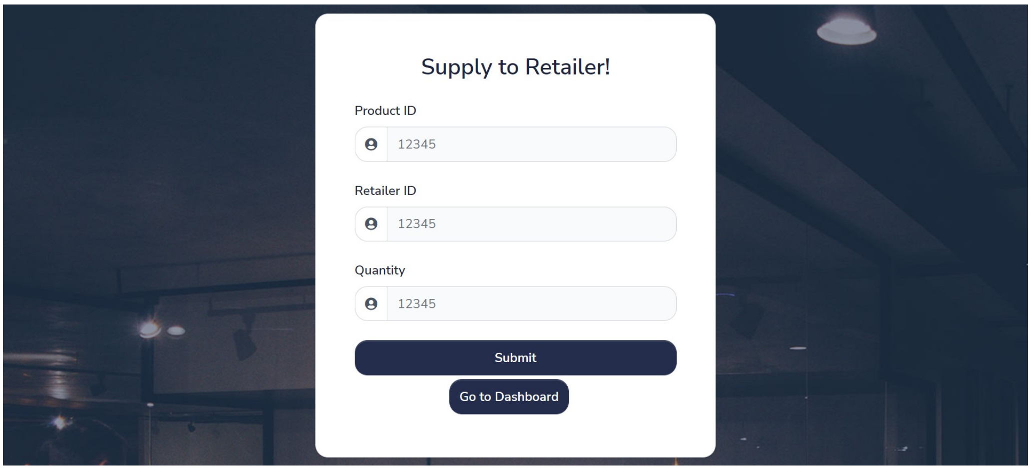
Distributor ID (e.g. 10001)

Quantity

1000

Figure 9: Supplier Shipping to Distributor

5.h Distributor Distributing Among Retailers


Supply to Retailer!

Product ID

12345

Retailer ID

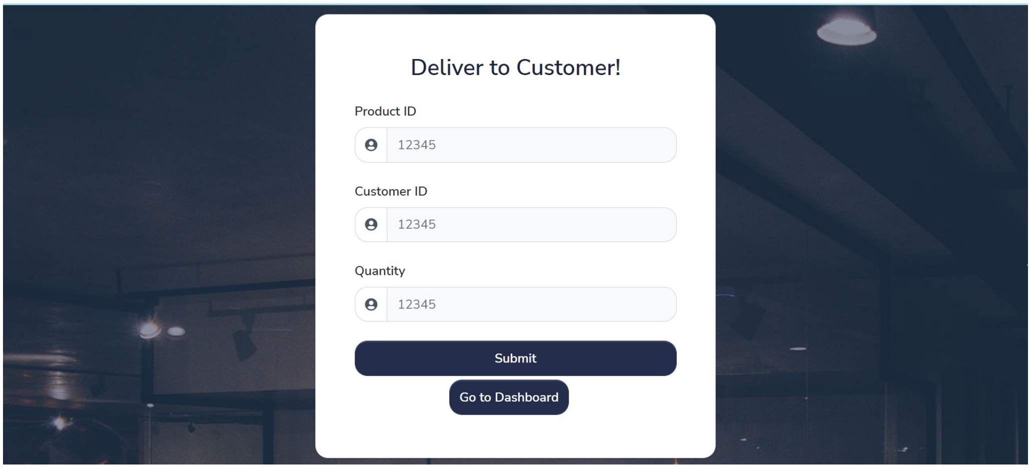
12345

Quantity

12345

Figure 10: Distributor Distributing Among Retailers

5.i Retailer Delivering to Customers


Deliver to Customer!

Product ID

12345

Customer ID

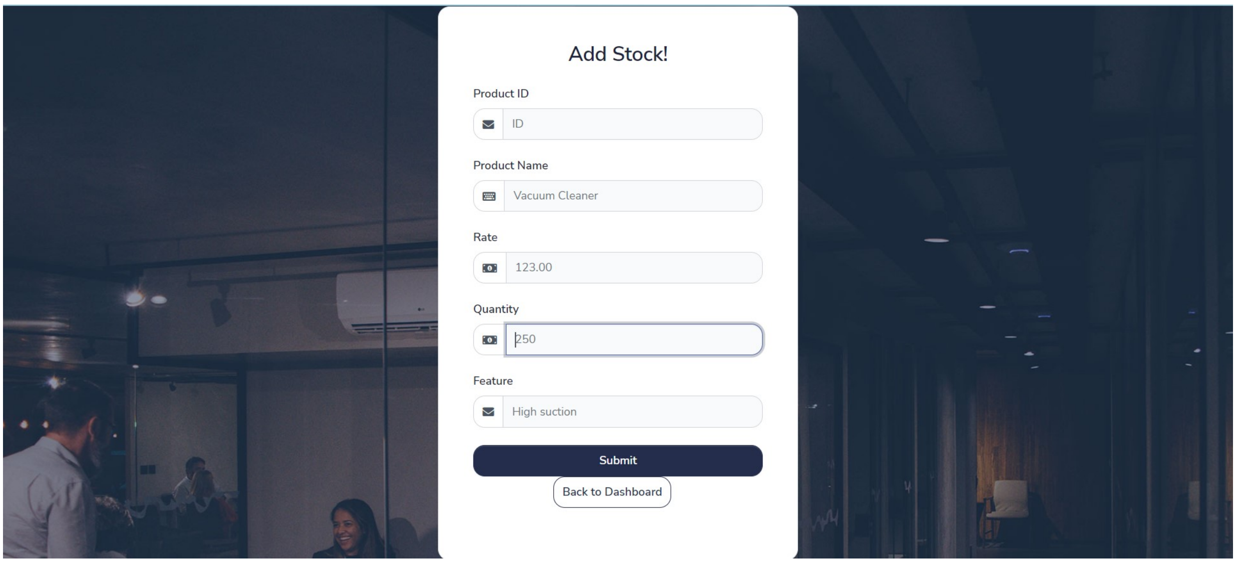
12345

Quantity

12345

Figure 11: Retailer Delivering to Customers

5.j Supplier Adding Stocks



Add Stock!

Product ID: ID

Product Name: Vacuum Cleaner

Rate: 123.00

Quantity: 250

Feature: High suction

Figure 12: Supplier Adding Stock

6 Workflow and Version Control

For Version Control, Git and GitHub were utilized. The repository corresponding to this project can be found at `https://github.com/devblixt/Sus-ply-chain` . We initially came up with a rough diagram to draft a rough structure of our database. Then, we formalized it into Entities and Relationships. These were then converted into tables, which could be created in SQL, and related using Foreign Key constraints. Front-End technologies are responsible for updating and inserting values into these tables, via forms that are present in JSP files. Webpages were linked by HTTP Requests, Responses, and even Redirects. Hence, we created a website that simulated the supply-chain, all the way from the suppliers to the customers.

7 Conclusions and Beyond

We made a Supply Chain Management System using various front-end and back-end tools. MySQL functions well as a back-end, but to a client-side user, the direct use of MySQL would be infeasible. Hence, using Front-End tools and technologies greatly help in taking advantage of the data storage and organization features that MySQL possesses. Version Control tools like Git greatly help workflow and were extensively used in this assignment by us.

Further developments that could be done in this project include making the ‘invoices’ table to actually form lines of a customer invoice, then allow customers to generate invoices of their purchases. A full e-commerce website could be made at the retailer-customer end of our schema, by building on the existing tables and relations.