Abstract

We developed a desktop application for the management of a supermarket system. This software will help owners in managing the various types of records delivered to their customers. This system is based on the sales transaction of items in a supermarket. The first activity is based on adding the items to the system along with the rate which are present in the supermarket and the name of the items which the supermarket will agree to sell. This management of the stocks is given to the stock managers. Any modifications to be done in the item name or the in the rate can be done by them. They also have the right to delete any item. As the customer buys the products and comes to the billing counter, the cashier is supposed to scan the barcode of the purchased product and the quantity he wanted to purchase. This is not a huge a task. The system will display shortcut buttons of all the items that doesn't have a barcode. He can select out of those displayed. Finally a bill will be generated for each customer. This will be saved in the database. Any periodic records can be viewed at any

time. If the stock is not available; the supermarket orders and buys from a prescribed supplier. The stock managers can view information about the stocks and check if any is below the minimum allowed. Admin provides a unique username and password for each employee through which he can login.

The product will help the employees to work in a highly effective and efficient environment. The salespersons have been recording the customer information in the past and even in the present. And indeed, it consumes their considerable time and energy that could be utilized in the better productive activities. Apart from that, with increasing customer Strength, the task of managing information of each individual customer is indeed a cumbersome task. There are a lot of reasons we implemented this project. In the manual System, there are number of inefficiencies that a salesperson faces. The information retrieval is one of the foremost problems. It is very difficult to gather the overall performance reports of the supermarket. Large records-books have to be maintained where relevant and irrelevant information has to be stored which is very untidy and clumsy process. On the other hand, there are many inherent problems that exist in any manual system. Usually, they lack efficiency. Less efficiency has a great impact on the productivity of any human being keeping the data up-to-date. The new system will cater to the need of the salespersons of any supermarket so that they can manage the system efficiently

Acknowledgements

We are thrilled to present to you the report about our completely new and revolutionary gym management system. We would like to express our gratitude & respect to our honorable teacher and project Supervisor Dr Abed El Safadi, Department of Computer Science, for his constant guidance, advice, encouragement & every possible help in the overall preparation of this report. We are also very much grateful to our families who always give us constant support and encouragement. We would like to thank our seniors who helped us greatly to complete this paper. In addition, we will mention our friends who also inspired and helped us to finish our work. We hope that this project paper has been prepared for the fulfillment of the course requirement. We would also like to thank our authority of Lebanese University.

Contents

Table Of Figures

Chapter 1: Introduction

This project deals with Super-Market automation. A Supermarket is a self-service store offering a wide variety of items related to food, household or daily use. Includes both purchase and sale of products. Designed to make the existing system more informative, reliable, fast and easy for all the stake-holders.

A. Objective

- ✓ To produce software which manage the activities done in a Super-Market.
- ✓ To maintain the records of the sales done in all times.
- ✓ To maintain the stock details.
- ✓ To reduce time in calculation of Sales activities.
- ✓ To store large amount of data in the database which will reduce clumsiness.

- ✓ To reduce paper work; so that users can spend more time on monitoring the Super-Market.

B. Advantages

<u>Advantages of the proposed system.</u>

- ✓ Reduced processing cost.
- ✓ Error reduction.
- ✓ Automatic updating of product details.
- ✓ Improved report generation and analysis.
- ✓ Better equipped to meet user requirements.
- ✓ Reduction in use of paper.
- ✓ Reduction in man power.
- ✓ Faster response time.

C. Overview

First we will start with a brief comparison between our application and some similar existing software. We will discuss why this application is really unique and surpasses all of the others. Then will we will proceed with the various techniques used in the design and implementation, justifying every choice. Finally we conclude our report with the results and different experiences lived throughout making this application.

Chapter 2 : Literature Review

There are some applications that provide similar services, such as: SAP Business One, Bright Pearl, Tylernet, and Lightspeed POS. The existing applications, presented above, can be synthesized in the following table:

	Accounting	Multi-Currency Support	Analytics / Reporting	Shortcut Product Buttons
SAP Business One	✓	✓		
Bright Pearl	✓	✓	✓	
Tylernet	✓		✓	
Lightspeed POS		✓	✓	

Figure 1Comparison of similar applications

As we can notice in the above table, no existing application meets all the requirement needed in a complete and efficient application, like the one we are developing.

Chapter 3: Design and Implementation

A. Application Architecture

The software is composed of three main parts: The database system, the graphical user interface, and the jdbc connector. In the database system we design the schema and create the tables holding the data. The graphical user interface provides a friendly and easy access to the database.

A.1 Database

The database contains all the data from the supermarket: customers, employees, products, stocks ,orders .....

The sql server used is Microsoft Sql Server 2014, with authentication mode.

A.2 Graphical User Interface

The graphical user interface is implemented in JavaFx, which is a new rich library for building modern applications.

A.3 Connection

The graphical user interface is connected to the database using jdbc : java database connectivity. It is an Application Programming Interface that allows the connection between java and different types of databases.

B. Design

B.1 Design Patterns

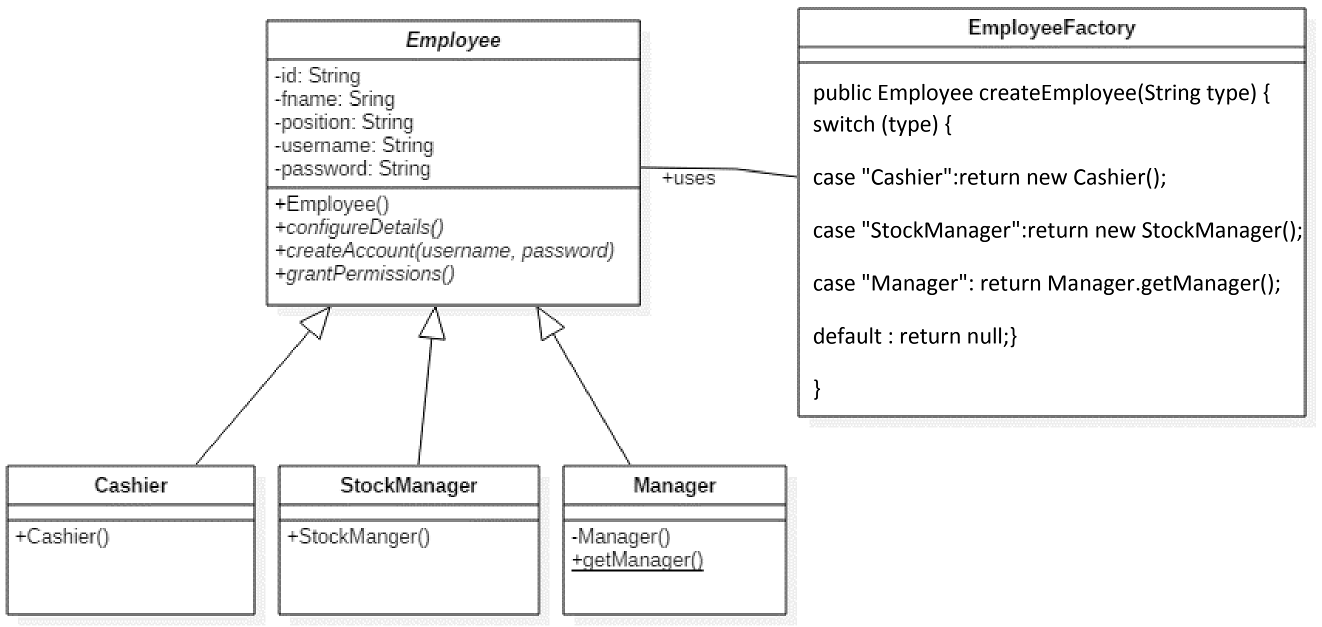
Singleton Pattern

In the application we have one and only one manager, so during runtime there must at most one instance of the object manager. The manager is also the system administrator of the database.

```
package entities;

public class Manager extends Employee {
	public static Manager manager;
	private Manager() {
		this.configureDetails();
	}
	public static Manager getManager() {
		if(manager==null) {
			manager=new Manager();
			System.out.println("Manager Created");
		}
		else {
			System.out.println("Manger already exists");
		}
		return manager;
	  }
  //code
}
```

Factory Pattern

In our application we have three types of employees: Cashier, StockManager, and Manager. In order to hide the creation process of each employee we implement the factory design pattern. All three types extend the abstract class Employee but each one implements the abstract methods differently.

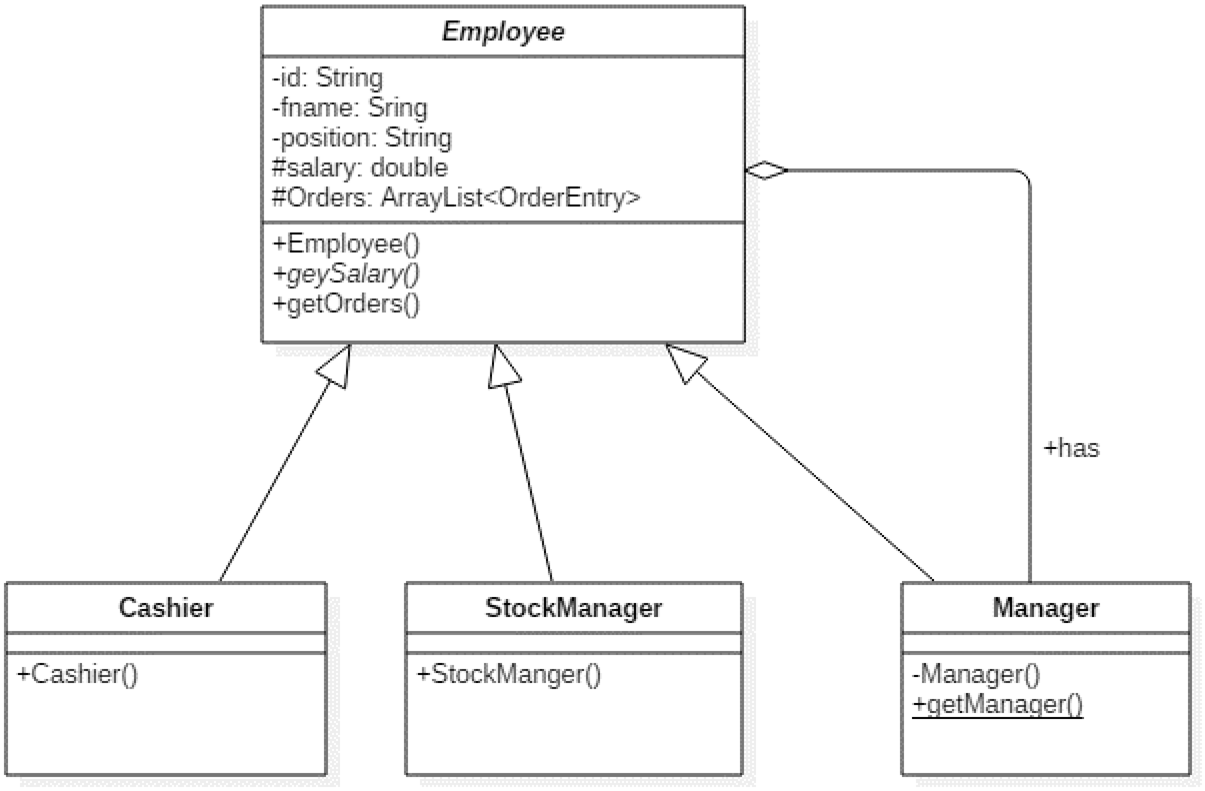


```
public Employee createEmployee(String type) {
switch (type) {

case "Cashier":return new Cashier();

case "StockManager":return new StockManager();

case "Manager": return Manager.getManager();

default : return null;}

}
```

The three methods : configureDetails(), createAccount(username,password), and grantPermissions() are abstract and have different implementation for each class. The grantPermissions() grants the cashier certain privileges like adding orders and searching for products, and the StockManager different permissions like adding stocks and changing quantities inside them, while the manager gets the all privileges permission.

Composite Pattern

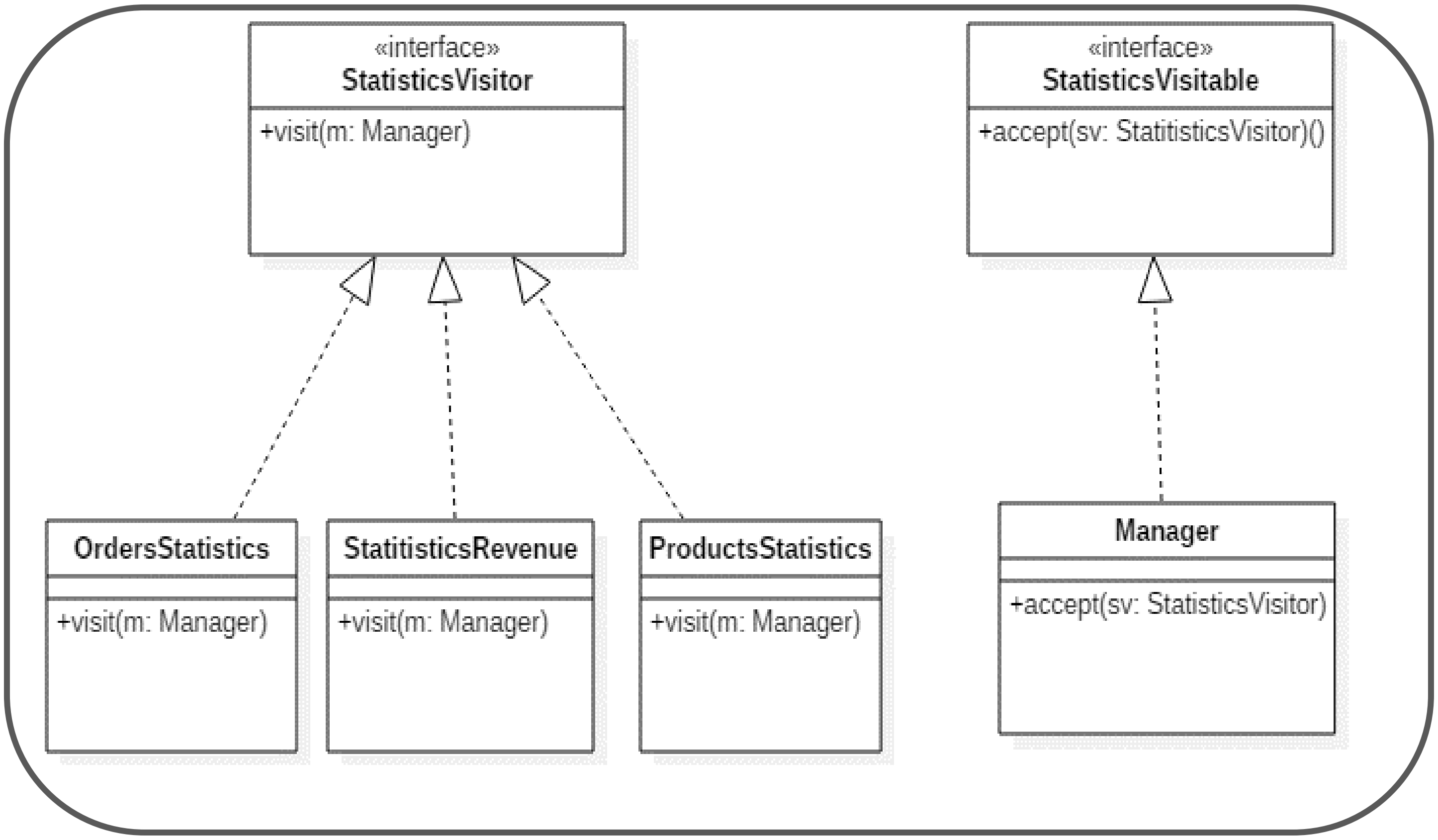
The manager has a group of employees, so he is represented as a composite class. Every employee has a salary except for the manager (the owner of the supermarket) so the getSalary() in the Manager class calculates and returns the sum of the salries of all the employees of the Arraylist<Employee> in Manager.

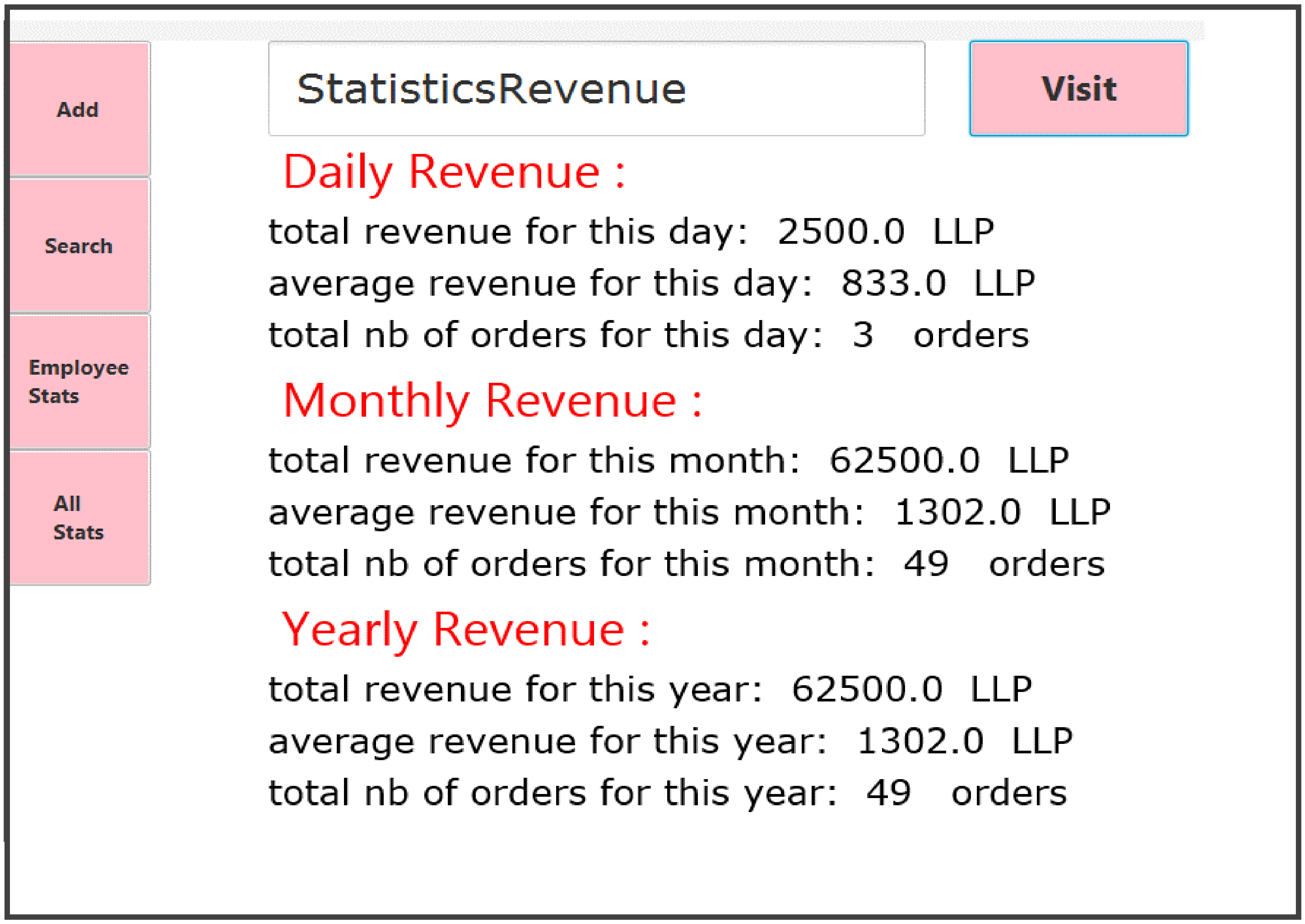


```
public double getSalary() {
            double s=0;
            for(int i=0;i<employees.size();i++)
                  s+=employees.get(i).getSalary();
            return s;}
public ArrayList<OrderEntry> getOrders() {
            ArrayList<OrderEntry> Orders=new
ArrayList<OrderEntry>();
            for(int i=0;i<employees.size();i++)
                  Orders.addAll(employees.get(i).getOrders());
            return Orders;}
```

Visitor Pattern

The application is designed in a way that it accepts new services without making significant changes to the code. The manger can generate as many new statistics as much as the developers keep writing new classes that implement the StatisticsVisitor interface.





Daily Revenue :

total revenue for this day: 2500.0 LLP
average revenue for this day: 833.0 LLP
total nb of orders for this day: 3 orders

Monthly Revenue :

total revenue for this month: 62500.0 LLP
average revenue for this month: 1302.0 LLP
total nb of orders for this month: 49 orders

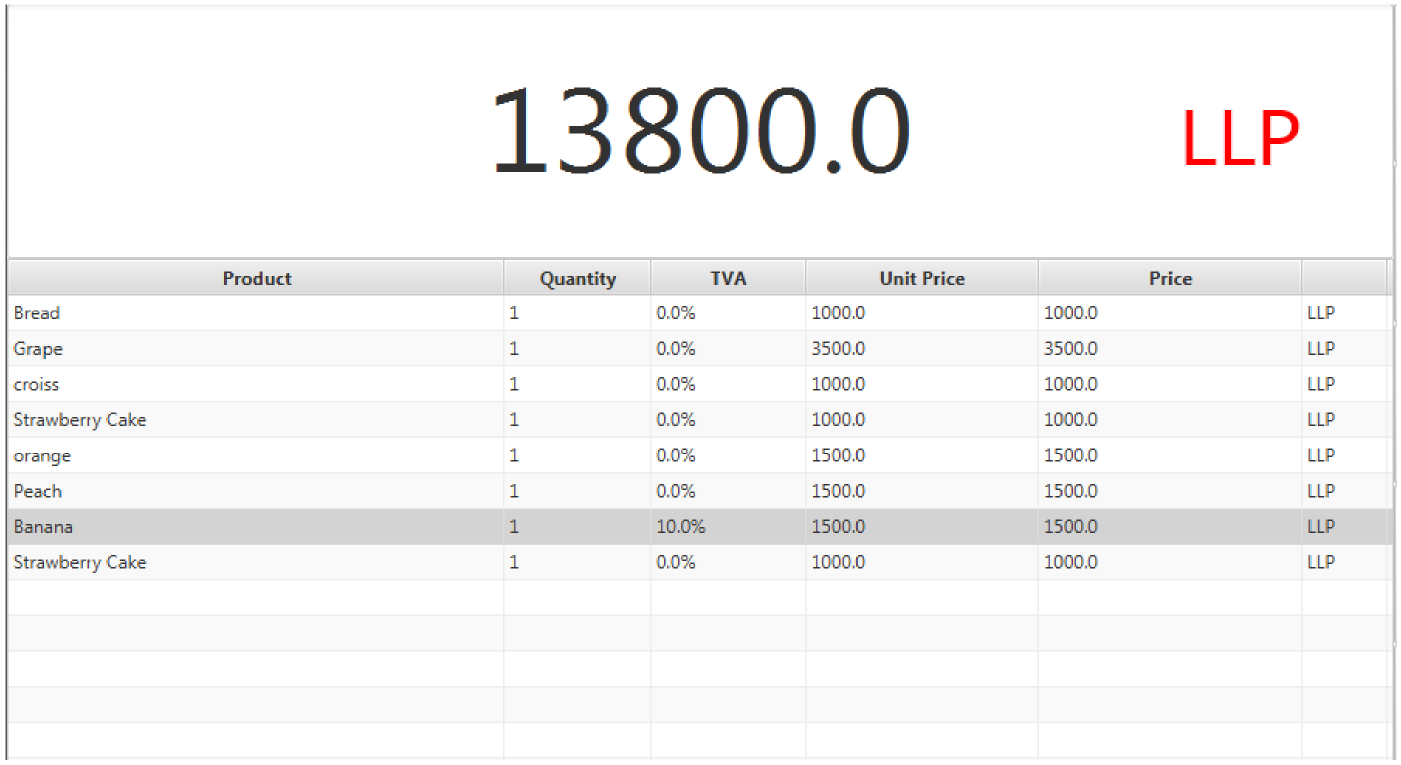
Yearly Revenue :

total revenue for this year: 62500.0 LLP
average revenue for this year: 1302.0 LLP
total nb of orders for this year: 49 orders

Figure 2 Visitor Pattern

Observer Pattern

The application is filled with tables, each table is an observer and the observable is a list data for each table. When we add an OrderEntry to the list data the table is automatically updated ,and the same happens when we remove an OrderEntry from the data. The total textfield is also updated at the same time.



13800.0 LLP

Product	Quantity	TVA	Unit Price	Price	
Bread	1	0.0%	1000.0	1000.0	LLP
Grape	1	0.0%	3500.0	3500.0	LLP
croiss	1	0.0%	1000.0	1000.0	LLP
Strawberry Cake	1	0.0%	1000.0	1000.0	LLP
orange	1	0.0%	1500.0	1500.0	LLP
Peach	1	0.0%	1500.0	1500.0	LLP
Banana	1	10.0%	1500.0	1500.0	LLP
Strawberry Cake	1	0.0%	1000.0	1000.0	LLP

```
data.add( new OrderEntry(rs.getString("PID"),
                rs.getString("PNAME").trim(),Quantity,
            (int)(rs.getDouble("TVA") * 100) + "%",
                rs.getDouble("PRICE"),
                rs.getString("PRICE_Currancy").trim()));
```

Model-View-Controller Pattern

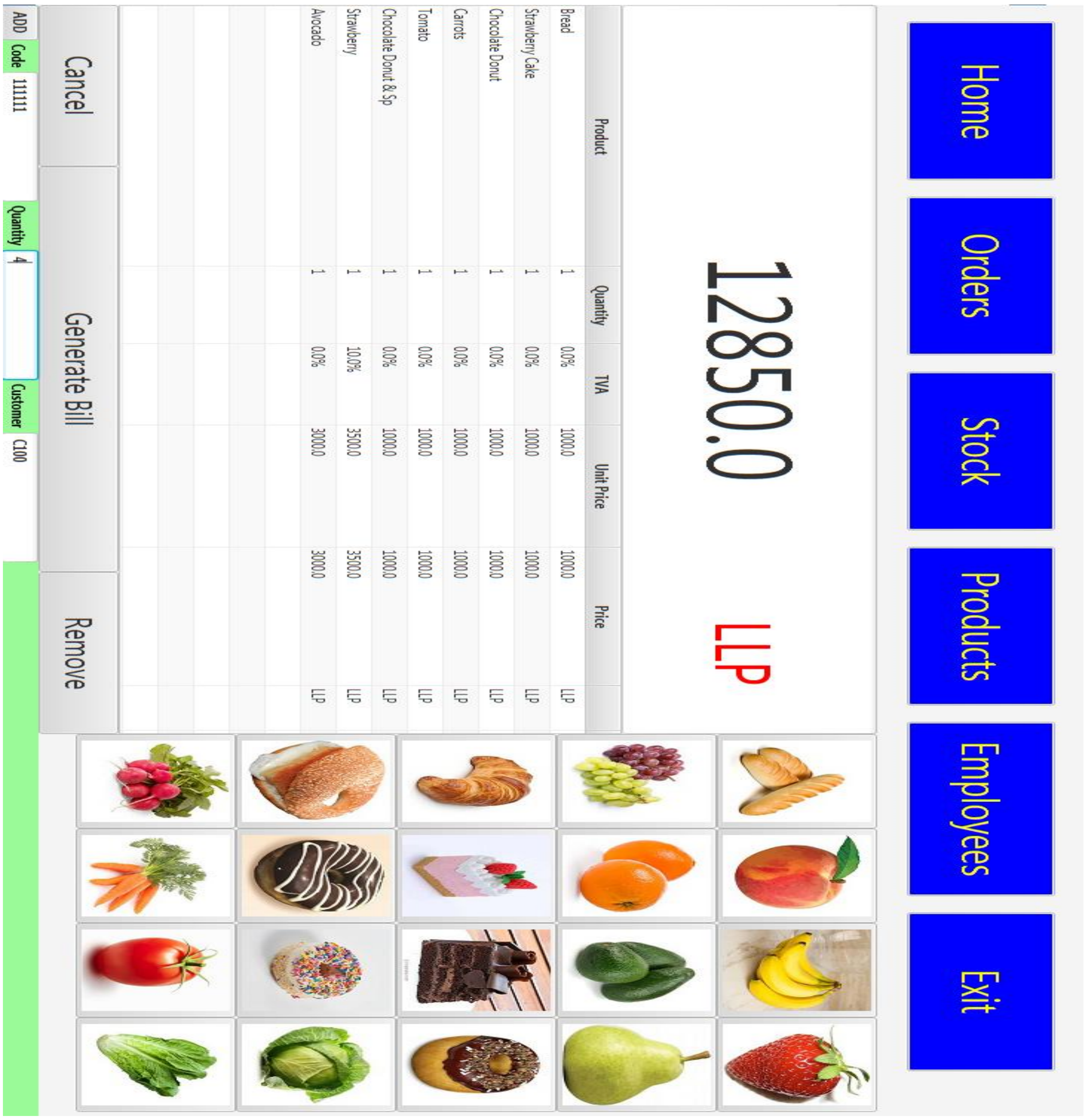
The application is composed of one main MVC pattern and several secondary MVC Patterns. In the main MVC we have Model, Home (View), and Controller. The Home Page contains 6 buttons : Home, Orders, Stocks, Products, Employees, Exit. When the Order button is clicked we create a new OrderModel, ViewModel and connect them using new OrderController. The same scenario happens for the StockModel-StockView-StockContoller, ProductModel-ProductView-ProductContoller, EmployeeModel-EmployeeView-EmployeeContoller.

Sample of the Controller :

The top buttons are stored in the array Button[] top

```
V.top[1].setOnAction(e -> {  //top[1] is the second button =>Orders

    //Code      });

       OrderModel OM = new OrderModel(M.Connection);
     OrdersView OV = new OrdersView()
     new OrderController(OM, OV, M.Connection,V);
     V.root.setCenter(OV.OrderPane);
     OV.OrderPane.setCenter(OV.InnerPages[0]);
       OV.Orders.setItems(OM.data);
     OM.getOrders();
     OV.InnerPages[0].getChildren().remove(OV.bottom1);
     OV.InnerPages[0].getChildren().remove(OV.bottom2);

     //Code    });
```

B.3 Samples From the Application

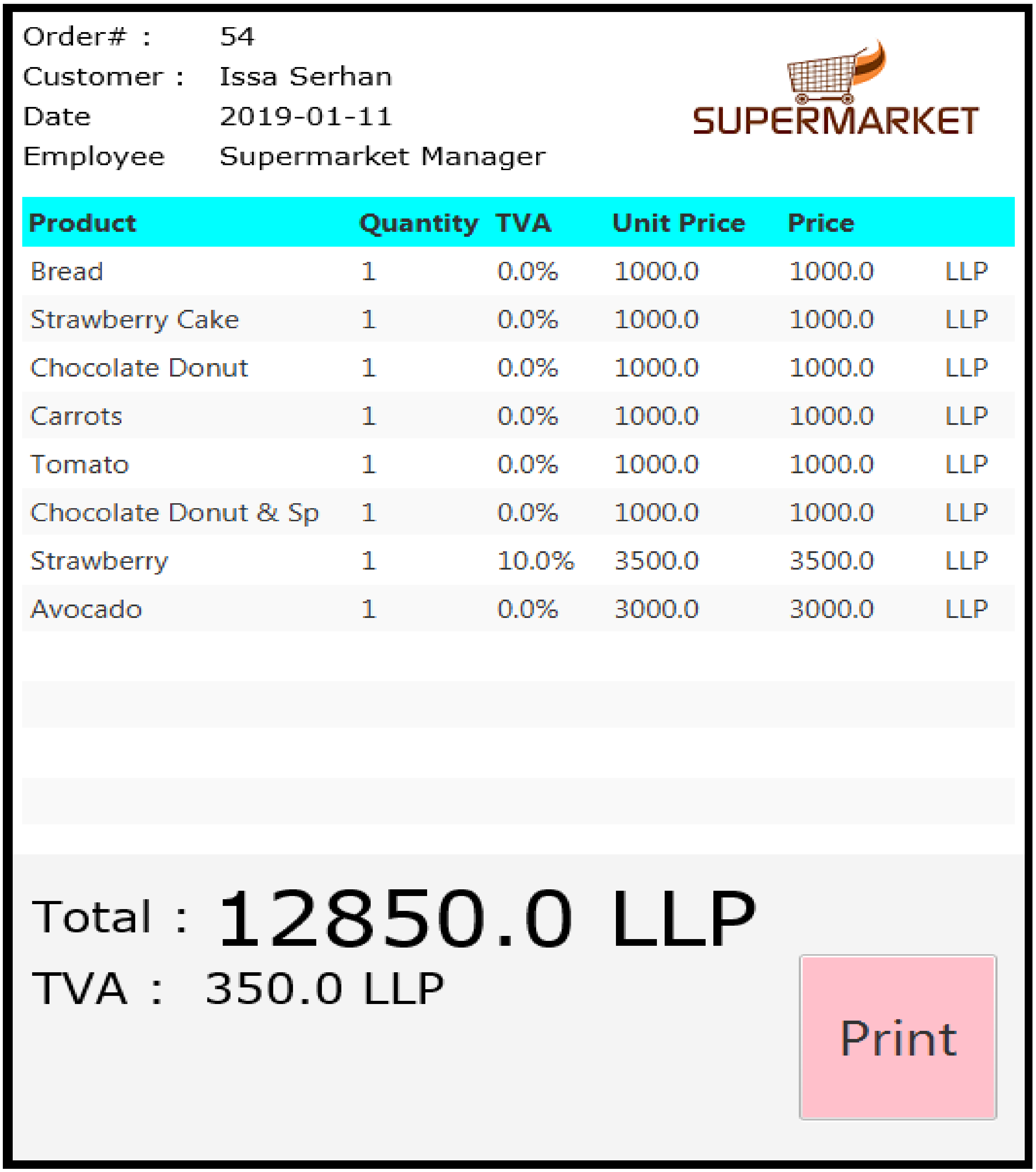


12850.0 LLP

Product	Quantity	TVA	Unit Price	Price	
Bread	1	0.0%	1000.0	1000.0	LLP
Strawberry Cake	1	0.0%	1000.0	1000.0	LLP
Chocolate Donut	1	0.0%	1000.0	1000.0	LLP
Carrots	1	0.0%	1000.0	1000.0	LLP
Tomato	1	0.0%	1000.0	1000.0	LLP
Chocolate Donut & Sp	1	0.0%	1000.0	1000.0	LLP
Strawberry	1	10.0%	3500.0	3500.0	LLP
Avocado	1	0.0%	3000.0	3000.0	LLP

Code: 111111
Quantity: 4
Customer: C100

Figure3: Home Page



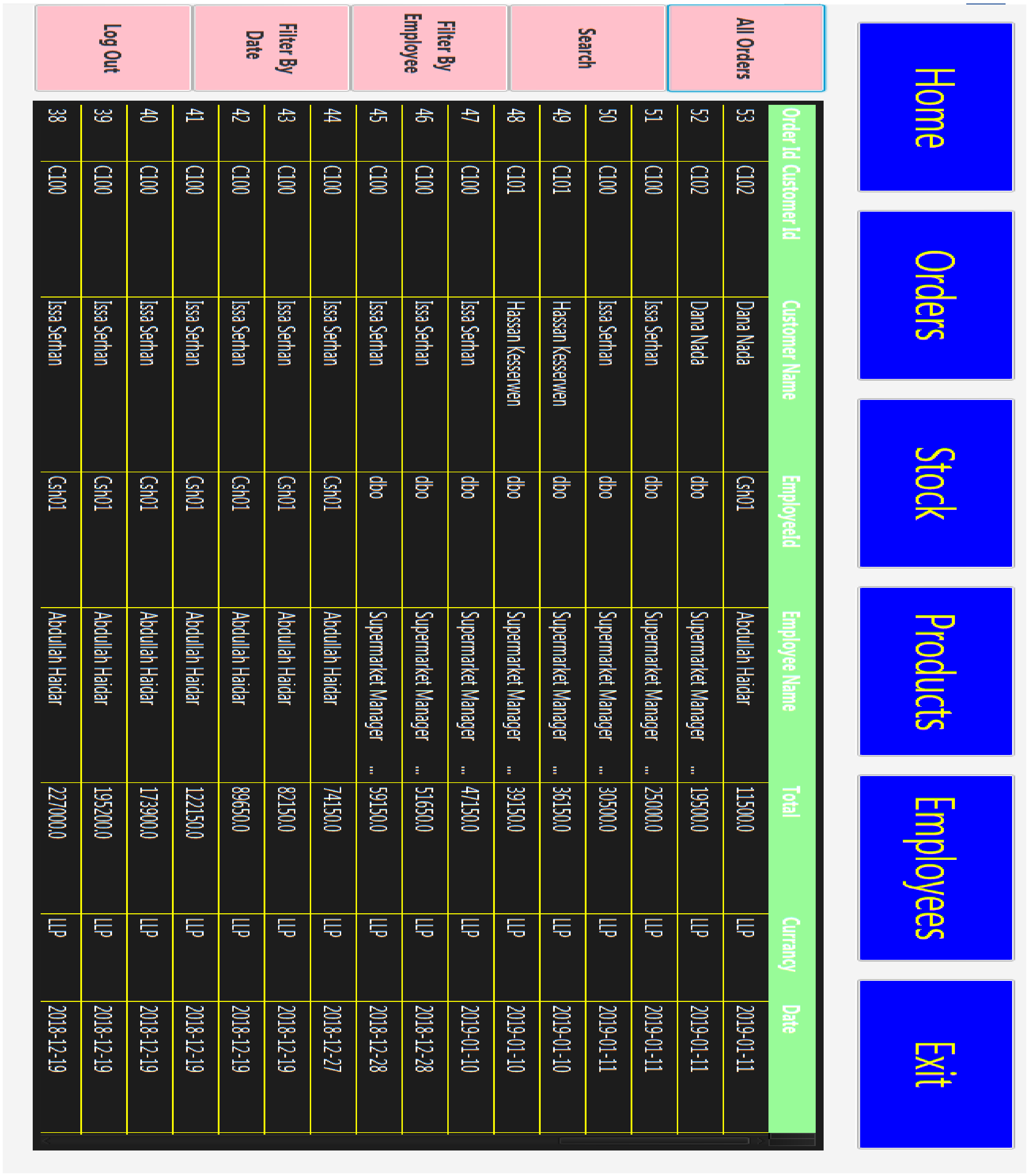
Order# :	54
Customer :	Issa Serhan
Date	2019-01-11
Employee	Supermarket Manager

Product	Quantity	TVA	Unit Price	Price	
Bread	1	0.0%	1000.0	1000.0	LLP
Strawberry Cake	1	0.0%	1000.0	1000.0	LLP
Chocolate Donut	1	0.0%	1000.0	1000.0	LLP
Carrots	1	0.0%	1000.0	1000.0	LLP
Tomato	1	0.0%	1000.0	1000.0	LLP
Chocolate Donut & Sp	1	0.0%	1000.0	1000.0	LLP
Strawberry	1	10.0%	3500.0	3500.0	LLP
Avocado	1	0.0%	3000.0	3000.0	LLP

Total : 12850.0 LLP

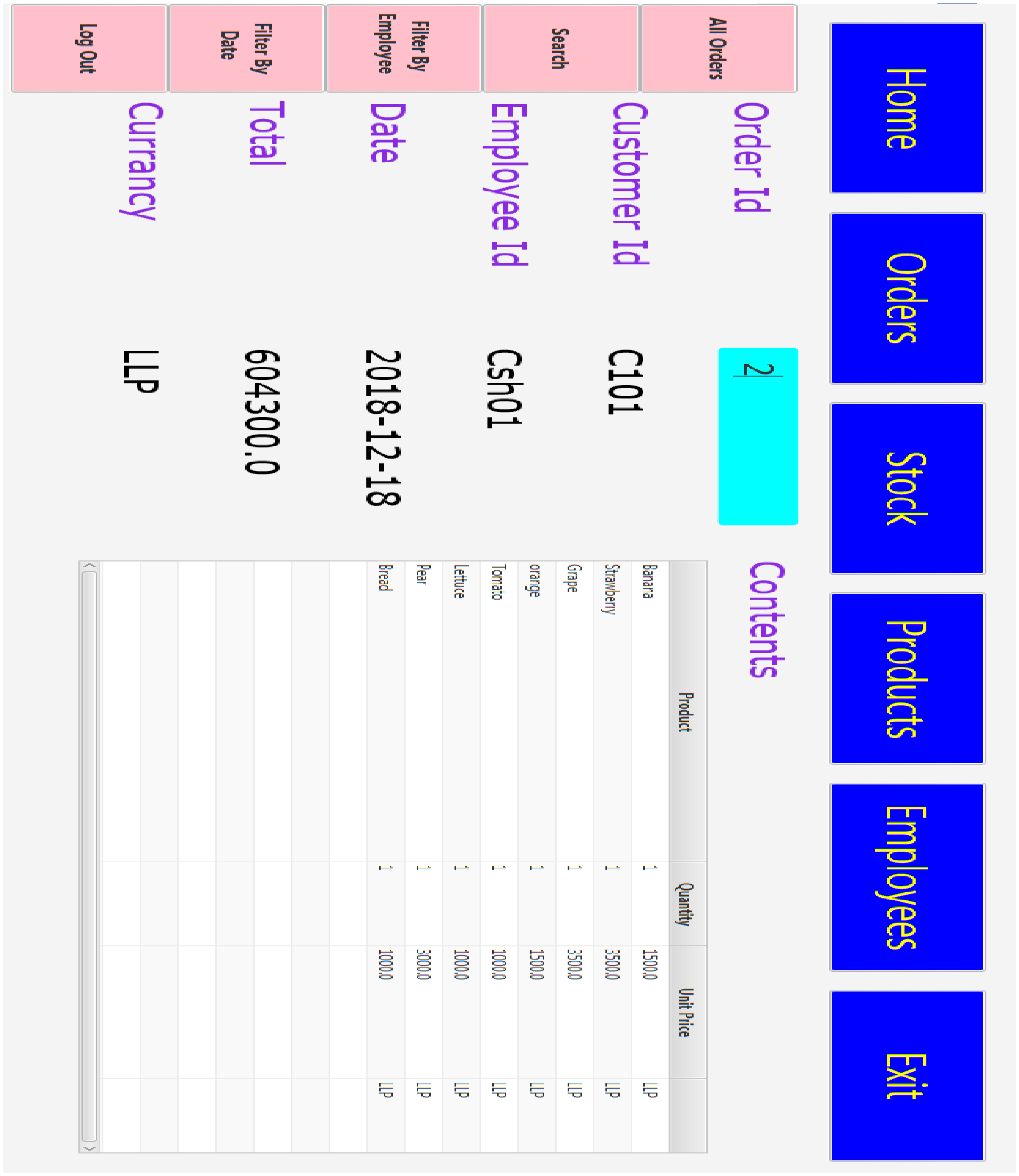
TVA : 350.0 LLP

Figure4: Bill sample



Order Id	Customer Id	Customer Name	EmployeeId	Employee Name	Total	Currancy	Date
53	C102	Dana Nada	Csh01	Abdullah Haidar	11500.0	LLP	2019-01-11
52	C102	Dana Nada	dbo	Supermarket Manager ...	19500.0	LLP	2019-01-11
51	C100	Issa Serhan	dbo	Supermarket Manager ...	25000.0	LLP	2019-01-11
50	C100	Issa Serhan	dbo	Supermarket Manager ...	30500.0	LLP	2019-01-11
49	C101	Hassan Kesserwen	dbo	Supermarket Manager ...	36150.0	LLP	2019-01-10
48	C101	Hassan Kesserwen	dbo	Supermarket Manager ...	39150.0	LLP	2019-01-10
47	C100	Issa Serhan	dbo	Supermarket Manager ...	47150.0	LLP	2019-01-10
46	C100	Issa Serhan	dbo	Supermarket Manager ...	51650.0	LLP	2018-12-28
45	C100	Issa Serhan	dbo	Supermarket Manager ...	59150.0	LLP	2018-12-28
44	C100	Issa Serhan	Csh01	Abdullah Haidar	74150.0	LLP	2018-12-27
43	C100	Issa Serhan	Csh01	Abdullah Haidar	82150.0	LLP	2018-12-19
42	C100	Issa Serhan	Csh01	Abdullah Haidar	89650.0	LLP	2018-12-19
41	C100	Issa Serhan	Csh01	Abdullah Haidar	122150.0	LLP	2018-12-19
40	C100	Issa Serhan	Csh01	Abdullah Haidar	173900.0	LLP	2018-12-19
39	C100	Issa Serhan	Csh01	Abdullah Haidar	195200.0	LLP	2018-12-19
38	C100	Issa Serhan	Csh01	Abdullah Haidar	227000.0	LLP	2018-12-19

Figure5: Orders Page



Order Id: 2

Customer Id: C101

Employee Id: Csh01

Date: 2018-12-18

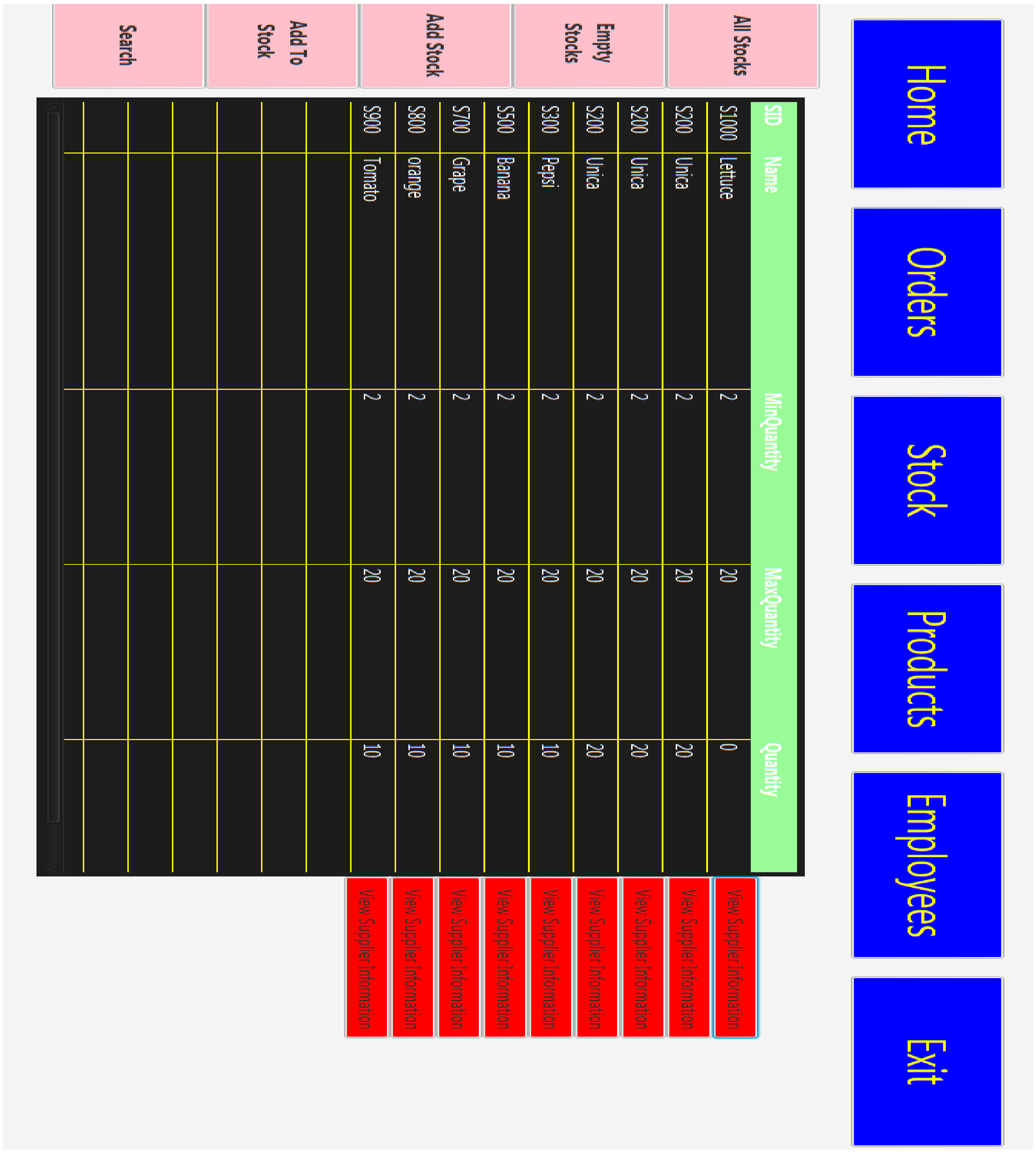
Total: 604300.0

Currancy: LLP

Contents

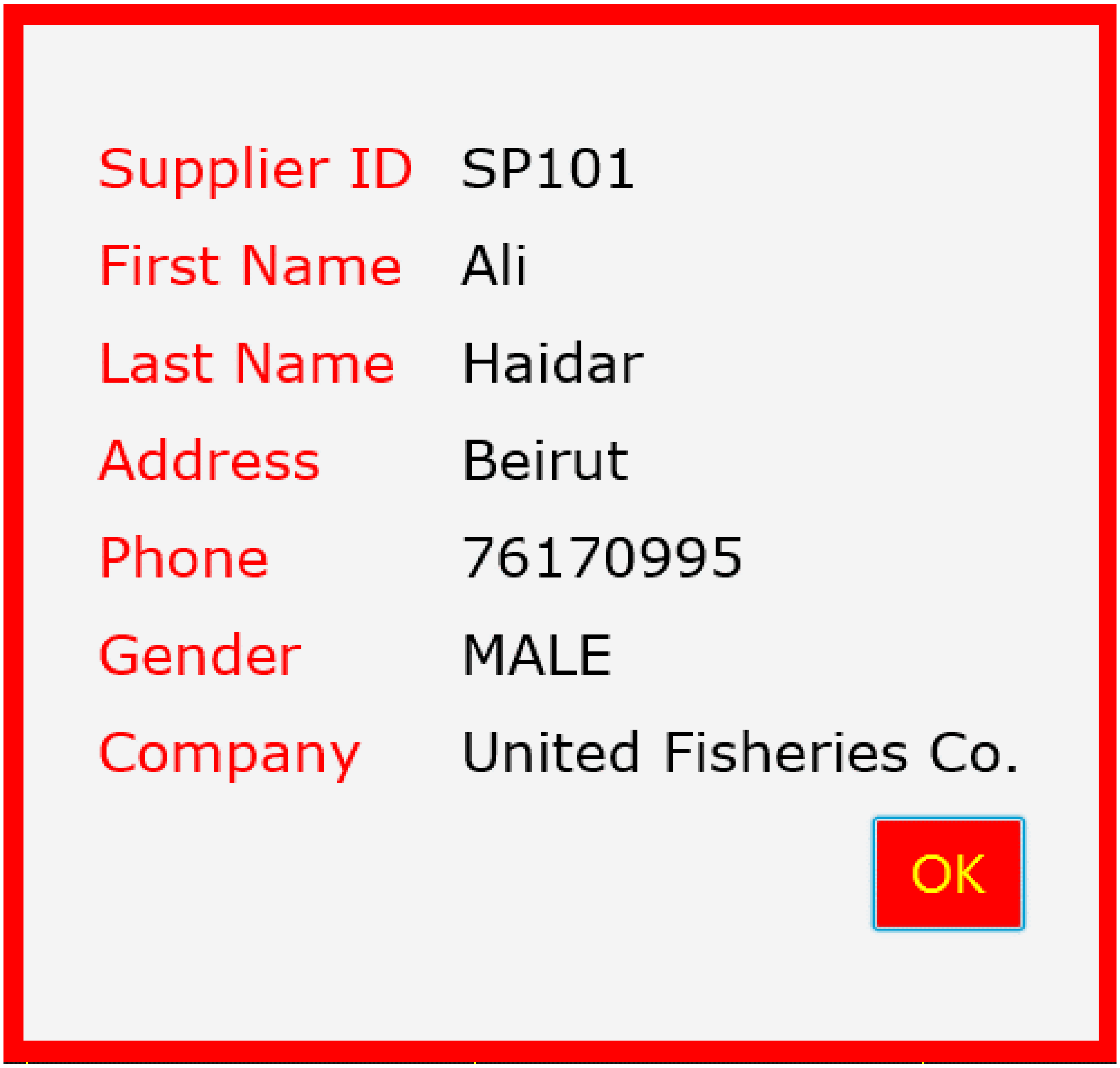
Product	Quantity	Unit Price	
Banana	1	1500.0	LLP
Strawberry	1	3500.0	LLP
Grape	1	3500.0	LLP
orange	1	1500.0	LLP
Tomato	1	1000.0	LLP
Lettuce	1	1000.0	LLP
Pear	1	3000.0	LLP
Bread	1	1000.0	LLP

Figure6: Orders search section



SID	Name	MinQuantity	MaxQuantity	Quantity
S1000	Lettuce	2	20	0
S200	Unica	2	20	20
S200	Unica	2	20	20
S200	Unica	2	20	20
S300	Pepsi	2	20	10
S500	Banana	2	20	10
S700	Grape	2	20	10
S800	orange	2	20	10
S900	Tomato	2	20	10

Figure8: Stocks Page



Supplier ID	SP101
First Name	Ali
Last Name	Haidar
Address	Beirut
Phone	76170995
Gender	MALE
Company	United Fisheries Co.

Figure9: Supplier Information

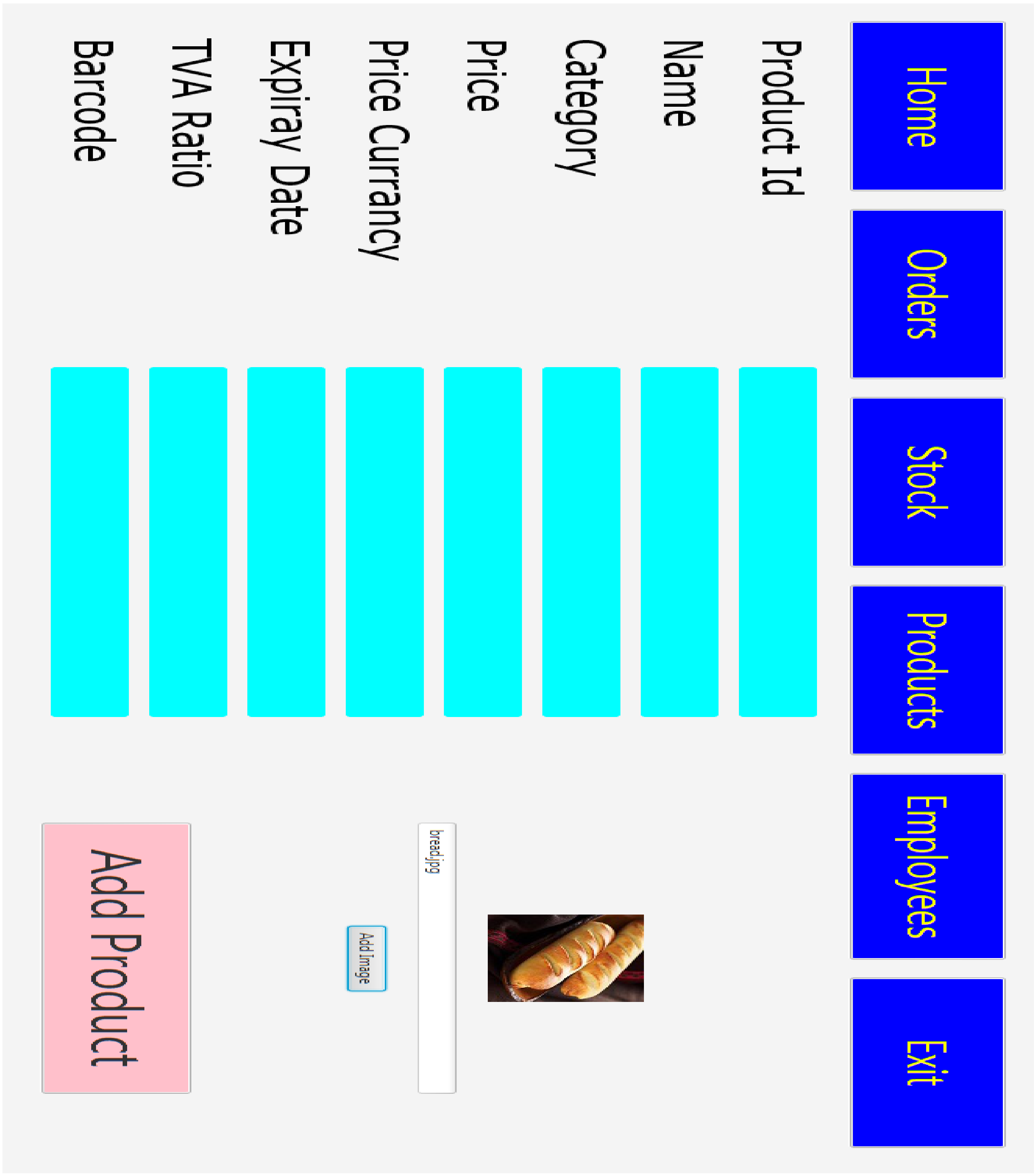
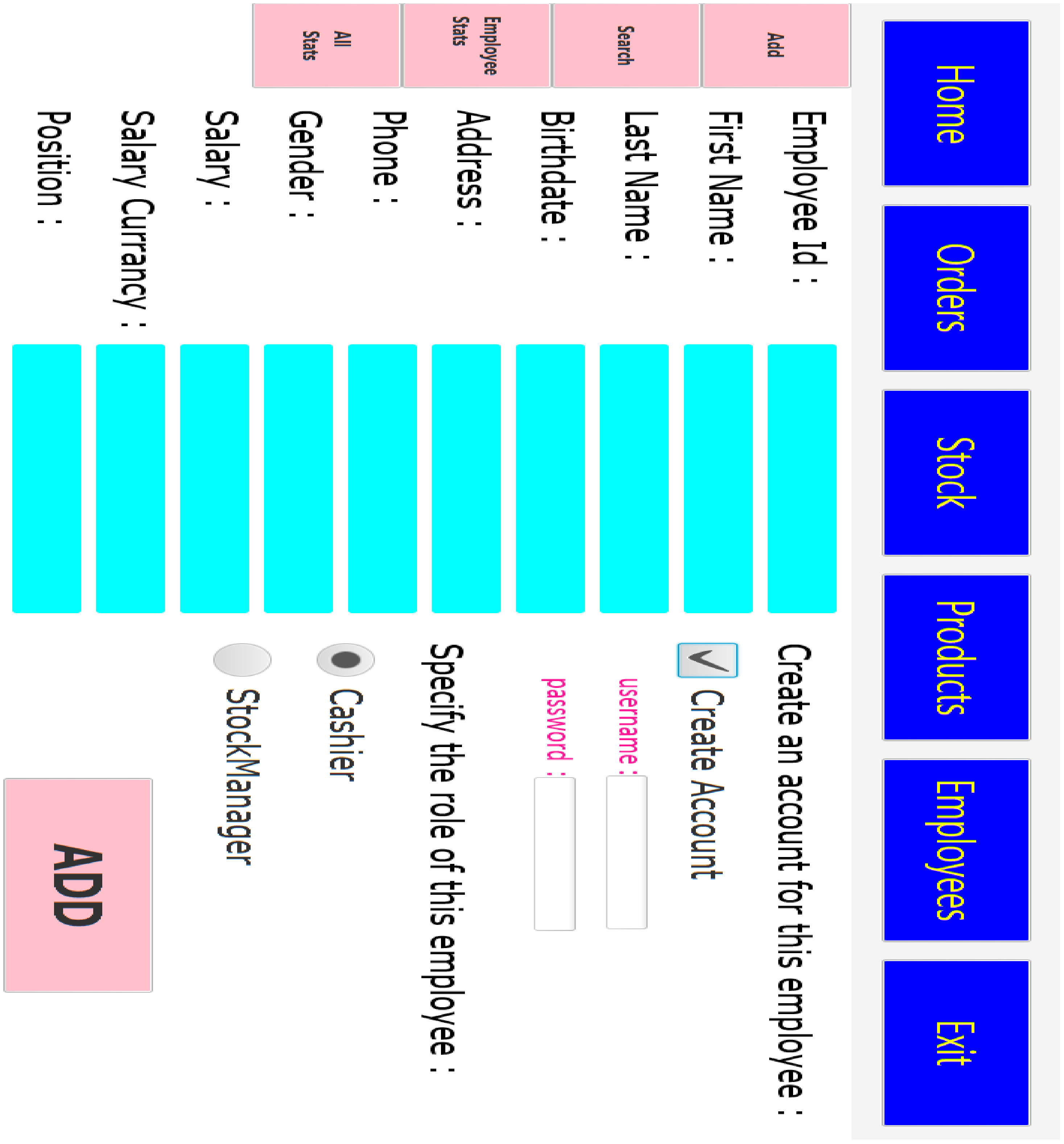


Figure10: Product Page



Employee Id :
First Name :
Last Name :
Birthdate :
Address :
Phone :
Gender :
Salary :
Salary Currancy :
Position :

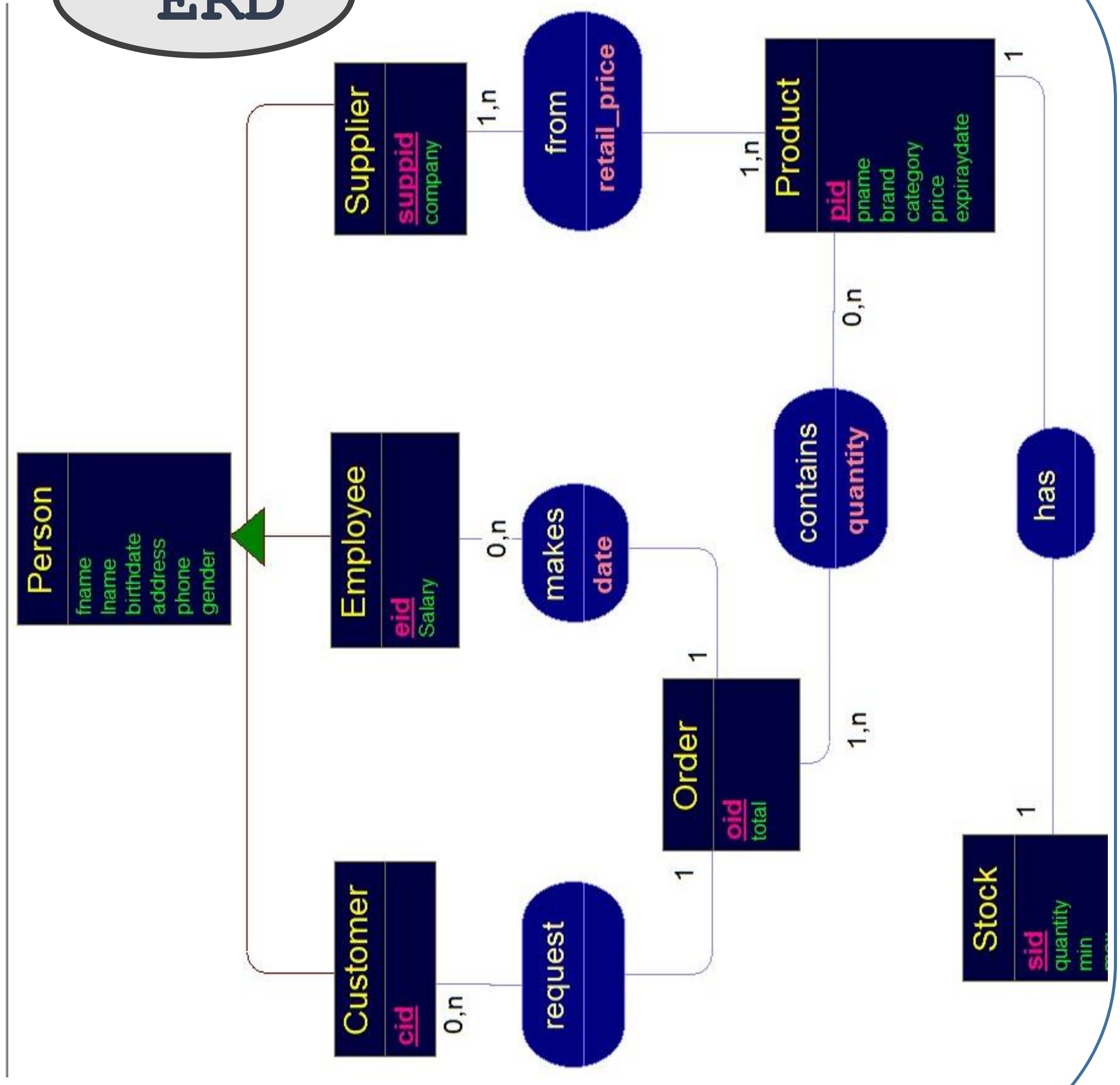
Create an account for this employee :
✓ Create Account
username :
password :

Specify the role of this employee :
● Cashier
○ StockManager

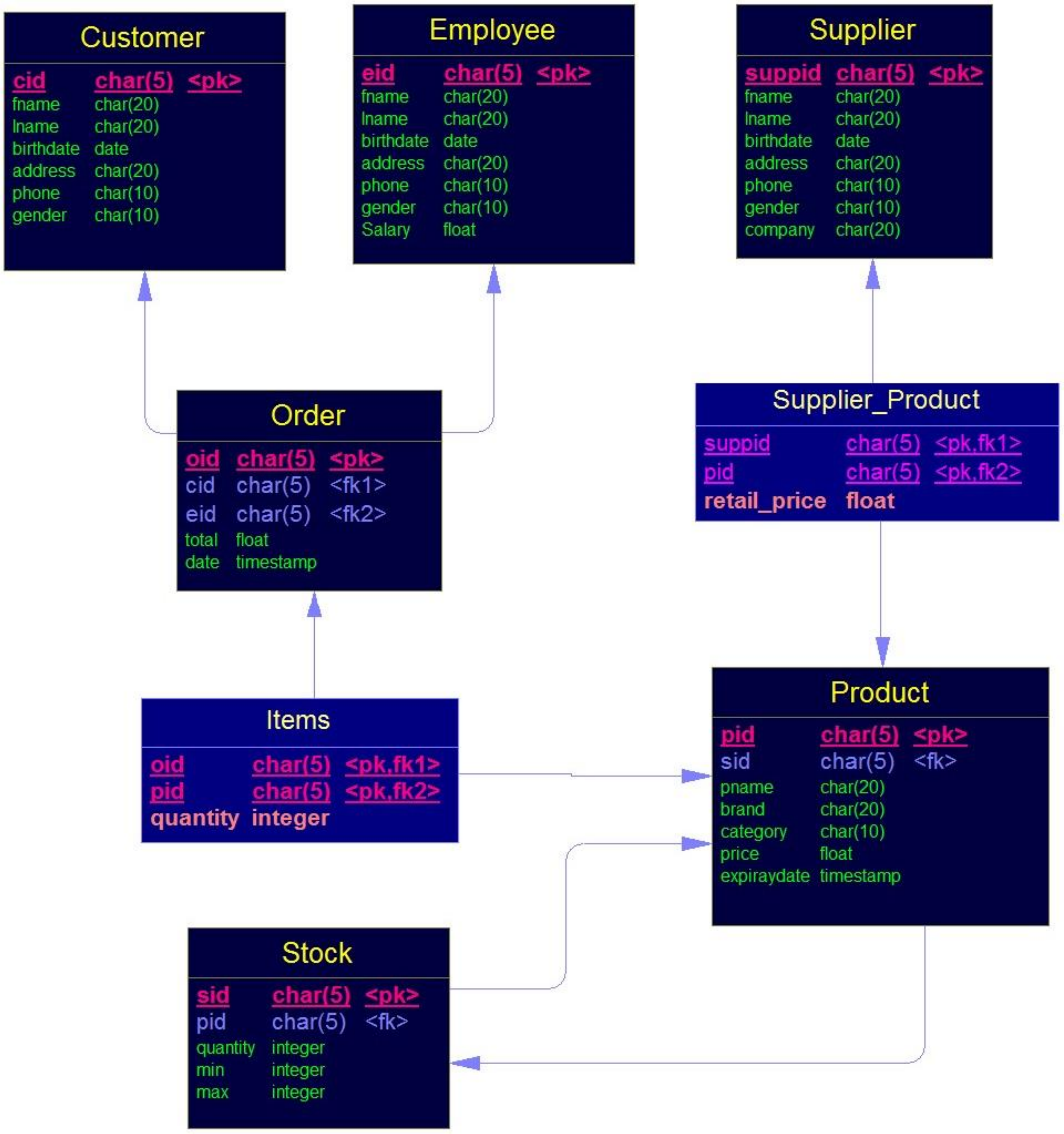
Figure11: Add Employee

C. Database Structure

ERD



RDM


Customer
cid char(5) <pk>
fname char(20)
lname char(20)
birthdate date
address char(20)
phone char(10)
gender char(10)
Employee
eid char(5) <pk>
fname char(20)
lname char(20)
birthdate date
address char(20)
phone char(10)
gender char(10)
Salary float
Supplier
suppid char(5) <pk>
fname char(20)
lname char(20)
birthdate date
address char(20)
phone char(10)
gender char(10)
company char(20)
Order
oid char(5) <pk>
cid char(5) <fk1>
eid char(5) <fk2>
total float
date timestamp
Supplier_Product
suppid char(5) <pk,fk1>
pid char(5) <pk,fk2>
retail_price float
Items
oid char(5) <pk,fk1>
pid char(5) <pk,fk2>
quantity integer
Product
pid char(5) <pk>
sid char(5) <fk>
pname char(20)
brand char(20)
category char(10)
price float
expiraydate timestamp
Stock
sid char(5) <pk>
pid char(5) <fk>
quantity integer
min integer
max integer

Conclusion

Working on this project has been a great deal of entertainment and benefit for us. We sharpened our programming and developing skills in all different aspects.

We definitely look forward to add some new features for this product, and we will try to improve it as much as possible.