Shopping Cart System

Index:

Introduction

Objectives

Design

Database

Connecting the GUI

Graphical User Interface

Normalisation

Final E-R diagram

Conclusion

Group Members:

Kavan Patel - 211627045

Aryaman Sakthivel - 211627028

Aniket Mishra - 211627004

Satvik Rohira -211627032

Udit Sai

Introduction

An online shopping cart is software that makes it possible for customers to select products and buy them online.

In an archaic environment, a customer could browse, grab a product off the shelf, and head to the checkout counter to make a purchase. Or they could use a cart around the store to gather multiple products before checking out.

On e-commerce websites / apps, customers need a way to mimic this experience. An online shopping cart allows them to select products on their computer or mobile device and store them in a virtual cart. From here, they can make their purchase.

Our goal is to design a system which ensures streamlined use and optimum customer satisfaction. This project / application will aim to mimic some of the mainstream online shopping sites, which majority of the population uses while maintaining a simplistic model with functions relevant to our current project.

The application will contain major features like products browsing, cart management, automatic bill production,

price calculation and etc.



Objectives

Behind the scenes, e-commerce shopping carts are more functional for sellers. They present background information about your products–like inventory and shipping options–in a way that’s easy to grasp for your customer.

Thinking about the steps a customer goes through when using a website’s shopping cart can helps us understand how the cart system works.

Adding Products:

The fundamental function of an online shopping cart is the ability to add products. products should have an **Add to Cart** button so your potential customers can keep track of interesting items. The simpler it is for someone to add a product to their cart, the better.

When a customer adds a product to their cart, they may not buy it right away, or at all. Sometimes customers use their cart for considering products—which leads us to step 2

Reviewing the Cart:

Customers should have the opportunity to review their cart before making their purchase. The cart review feature in major

online shopping sites provides a unique opportunity for upselling. You can offer cross-product promotions, suggest products that pair well with items in their cart, or offer upgrades to their current selections.

This lead us to our potentially final step of the shopping cart process.

#### **Completing the purchase:**

After all the products have been added to the cart, and have been reviewed for one last time, its time for finalising purchase. Following the purchase, a final bill is generated with a bill id, total purchase amount and a purchase date

Design

**Creating the Database and setting-up Tables**

For this part, we use the MySQL database.

We can now proceed to create the **Items** table. We'll use this table to store the product IDs, along with the product name, description, and price.

* *create table Items(*
* *I\_ID numeric(4) Primary Key Not Null,*
* *Name varchar(25),*
* *Type varchar(25),*
* *Price numeric(8,2));*

Next table is the **Manufacture** table containing M-id as Primary Key, I\_ID as foreign key, Manufacture price, factory name and address.

* *create table Manufacture(*
* *M\_ID numeric(5) Primary key Not Null,*
* *I\_ID numeric(4), foreign key (I\_ID) references Items(I\_ID),*
* *M\_Price numeric(6,2),*
* *Factory varchar(25),*
* *Address varchar(30));*

**Inventory** table contains I\_ID and M\_ID as foreign keys and inventory quantity

* *create table Inventory(*
* *I\_ID numeric(4),*
* *M\_ID numeric(5),*
* *Quantity int,*
* *foreign key (I\_ID) references Items(I\_ID) on delete cascade,*
* *foreign key (M\_ID) references Manufacture(M\_ID) on delete cascade*);

**Customer** table contains customer id as primary key, customer name, phone and address.

* *create table Customers(*
* *C\_ID numeric(3) primary key,*
* *C\_Name varchar(30),*
* *C\_Phone numeric(10),*
* *C\_Address varchar(35));*

**Order** table only contains order id and c\_id which references Customer table’s c\_id

* *create table Orders(*
* *O\_ID varchar(4) Primary key,*
* *C\_ID numeric(3), foreign key (C\_ID) references Customers(C\_ID));*

**Cart** table is the second last table with the order no as foreign key O\_ID of Orders and I\_ID referencing I\_ID of Items, Cart quantity and item price.

* *create table Cart(*
* *Order\_No varchar(4), foreign key (Order\_No) references Orders(O\_ID),*
* *I\_ID numeric(4), foreign key (I\_ID) references Items(I\_ID),*
* *Quantity int,*
* *Item\_Price int);*

Last table **Bill** has B\_ID being the Primary Key, C\_ID as foreign key referencing Customers’ C\_ID, Total price and Order Date.

* *create table Bill(*
* *B\_ID varchar(4) primary key,*
* *C\_ID numeric(3), foreign key (C\_ID) references Customers(C\_ID),*
* *Total\_Price int,*
* *Order\_Date date);*

For practical and presentation purposes we will also add a plethora of sample values to test the shopping cart system and its many features.

Connecting The GUI

The GUI for this project is to be made using Python via Visual Studio and is going to be connected to the various tables in MySQL via this code below:

import mysql.connector

class DB:

def connectToDatabase(self):

try:

self.db = mysql.connector.connect

(host='localhost',user='\*\*\*\*\*',password='\*\*\*\*\*’, database='miniproject')



self.dbcursor = self.db.cursor()

self.db.autocommit = True

print("Connected to Database Successfully")

return self.dbcursor

except Exception as e:

print("Error connecting to database")

print(e)

quit(-1)

def \_init\_(self):

self.connectToDatabase()

Graphical User Interface (GUI)

The final interface for the application is made using the code written below:

self.ui.page

self.ui.b1.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.p2))

self.ui.show.clicked.connect(self.loadDataFromatt)

self.ui.send.clicked.connect(self.Add\_to\_cart)

self.ui.home.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.page))

 self.ui.order\_b.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.cart\_page))

self.ui.go\_to\_cart.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.cart\_page))

self.ui.bill\_b.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.bill\_page))

self.ui.go\_home1.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.page))

self.ui.go\_home2.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.page))

self.ui.cart\_show.clicked.connect(self.loadOrders)

self.ui.bill\_show.clicked.connect(self.loadBills)

def loadDataFromatt(self):

query = "SELECT \* FROM Items"

self.cur.execute(query)

result=self.cur.fetchall()

self.ui.tables.setRowCount(len(result))

for row\_number, row\_data in enumerate(result):

self.ui.tables.insertRow(row\_number)

for column\_number, data in enumerate(row\_data):

self.ui.tables.setItem(row\_number, column\_number, QtWidgets.QTableWidgetItem(str(data)))

def loadOrders(self):

query = "SELECT \* FROM Cart"

self.cur.execute(query)

result=self.cur.fetchall()

self.ui.cart\_table.setRowCount(len(result))

for row\_number, row\_data in enumerate(result):

 self.ui.cart\_table.insertRow(row\_number)

for column\_number, data in enumerate(row\_data):

self.ui.cart\_table.setItem(row\_number, column\_number, QtWidgets.QTableWidgetItem(str(data)))

def loadBills(self):

query = "SELECT \* FROM Bill"

self.cur.execute(query)

result=self.cur.fetchall()

self.ui.bill\_table.setRowCount(len(result))

for row\_number, row\_data in enumerate(result):

self.ui.bill\_table.insertRow(row\_number)

for column\_number, data in enumerate(row\_data):

self.ui.bill\_table.setItem(row\_number, column\_number, QtWidgets.QTableWidgetItem(str(data)))

def InsertData(self):

self.cur.execute("INSERT INTO att(student\_id, student\_name, AttendanceStatus, attendance\_date) VALUES(%s, %s, %s, %s)" ,

(self.ui.q1.text(),self.ui.student\_name.text(), self.ui.attendance\_status.text(), d))

print("Data Sent to DataBase Successfully!")

def Add\_to\_cart(self):

s = self.ui.qute.text()

l = self.ui.order.text()

print(s)

self.cur.execute("call AddtoCart(%s, %s, %s)" , (l, s, self.ui.quantity.text()))

 def generatebill(self):

b = self.ui.bill.text()

o = self.ui.ord.text()

self.cur.execute("call generate\_bill(%s, %s)" , (b, o))

def showbill(self):

query2 = "SELECT \* FROM Bill"

self.cur.execute(query2)

result=self.cur.fetchall()

self.ui.billtable.setRowCount(len(result))

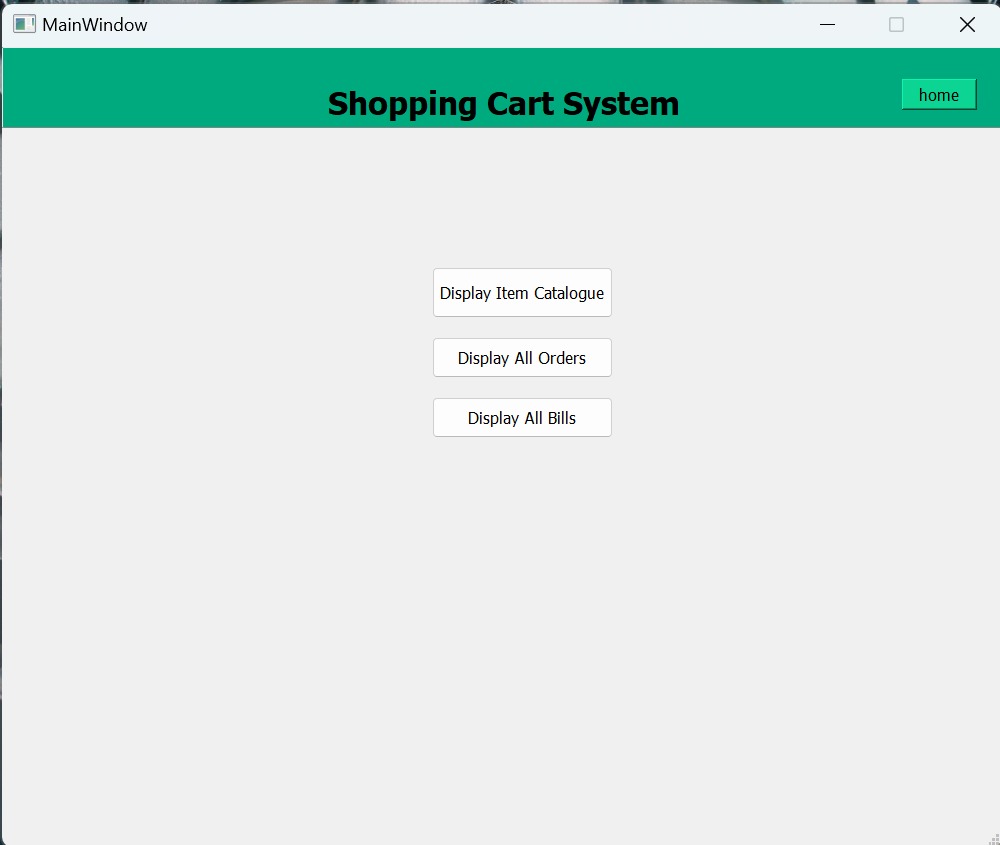
for row\_number, row\_data in enumerate(result):

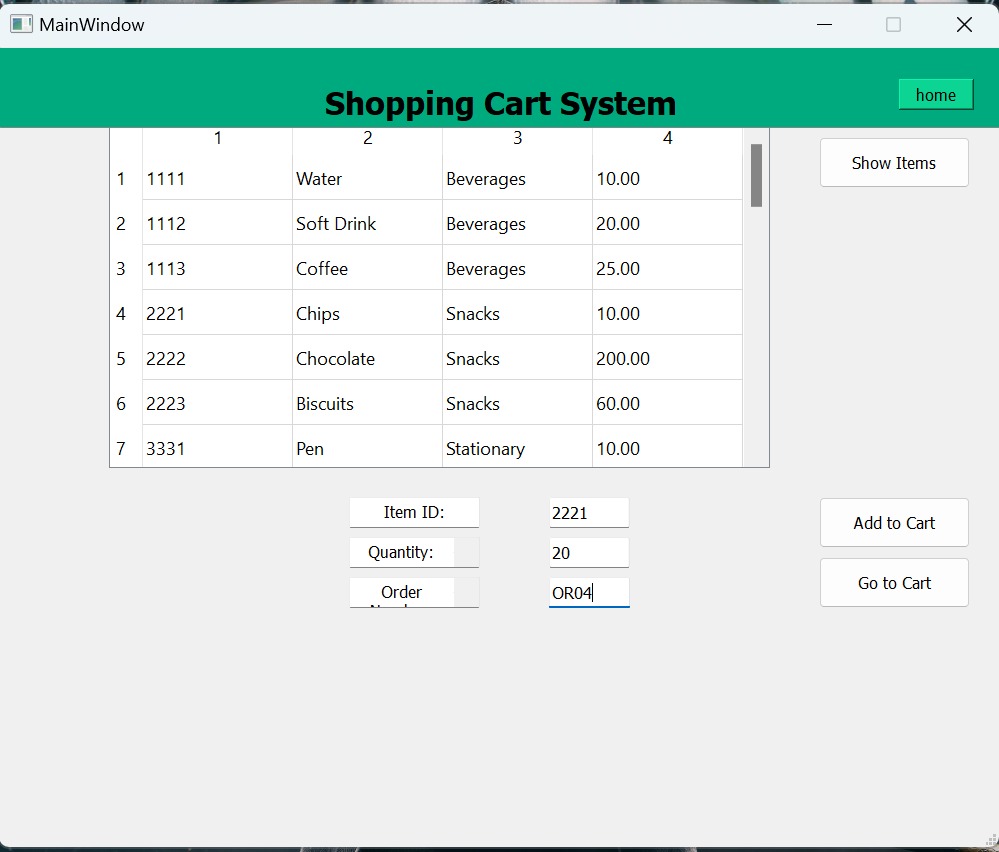
self.ui.billtable.insertRow(row\_number)

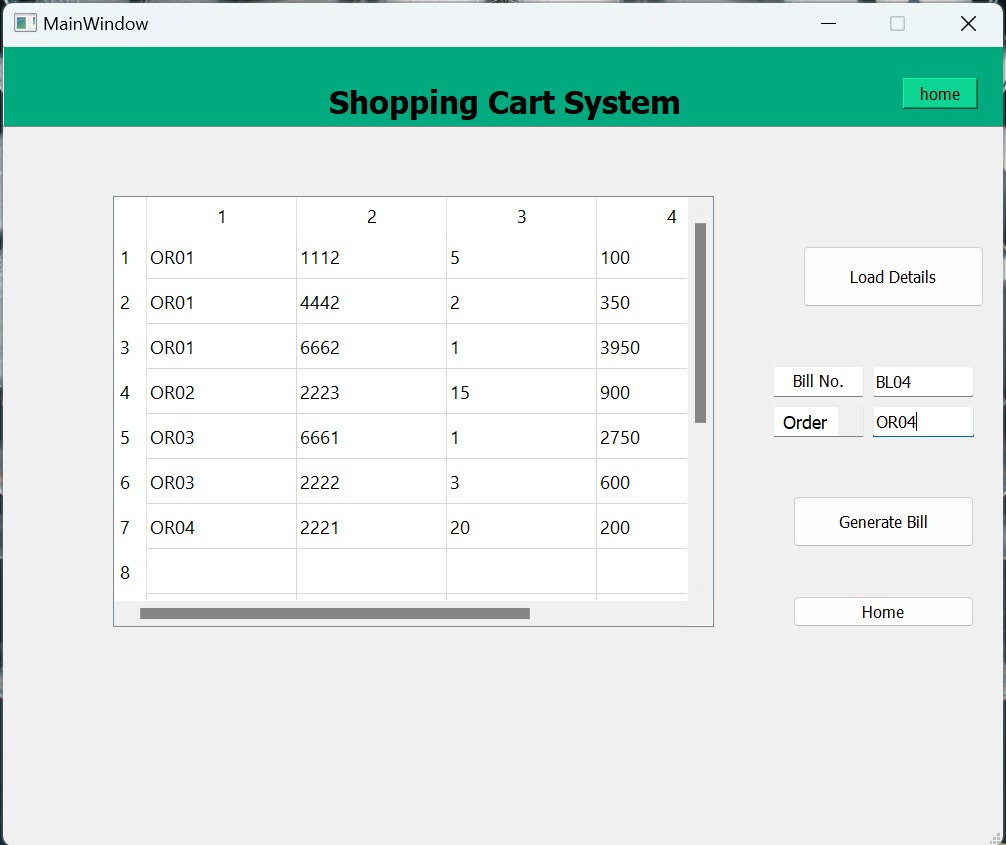
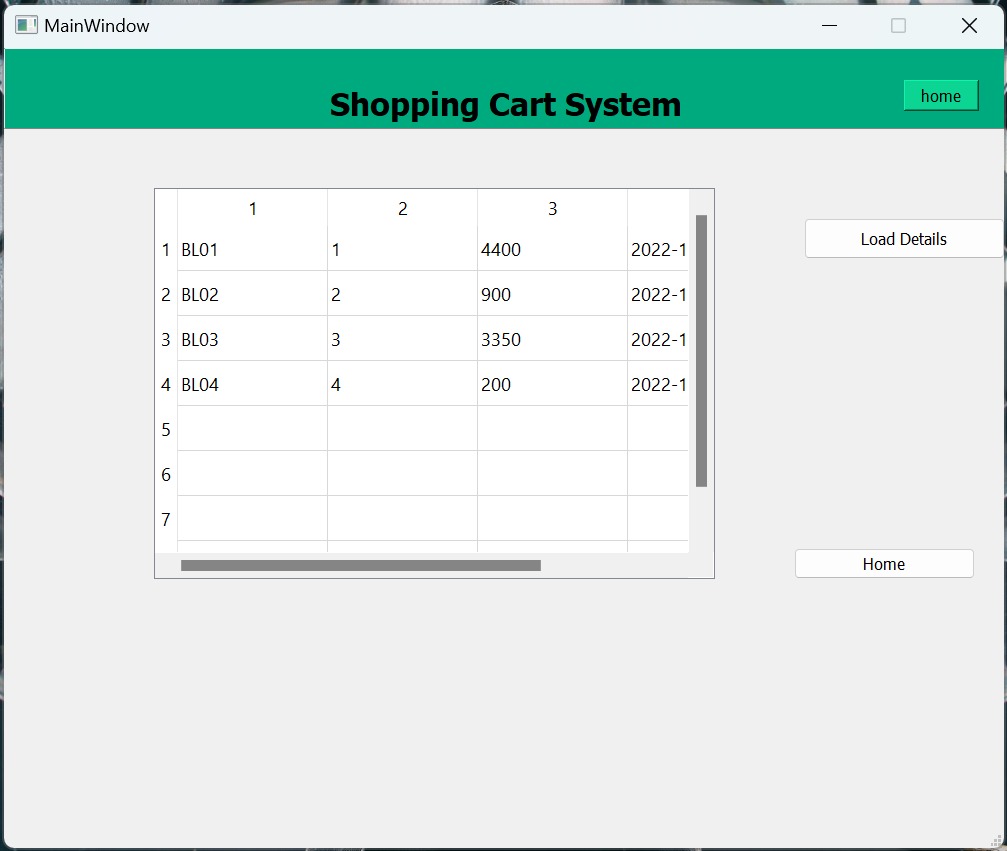
for column\_number, data in enumerate(row\_data):

self.ui.billtable.setItem(row\_number, column\_number, QtWidgets.QTableWidgetItem(str(data)))

The final GUI looks like this:

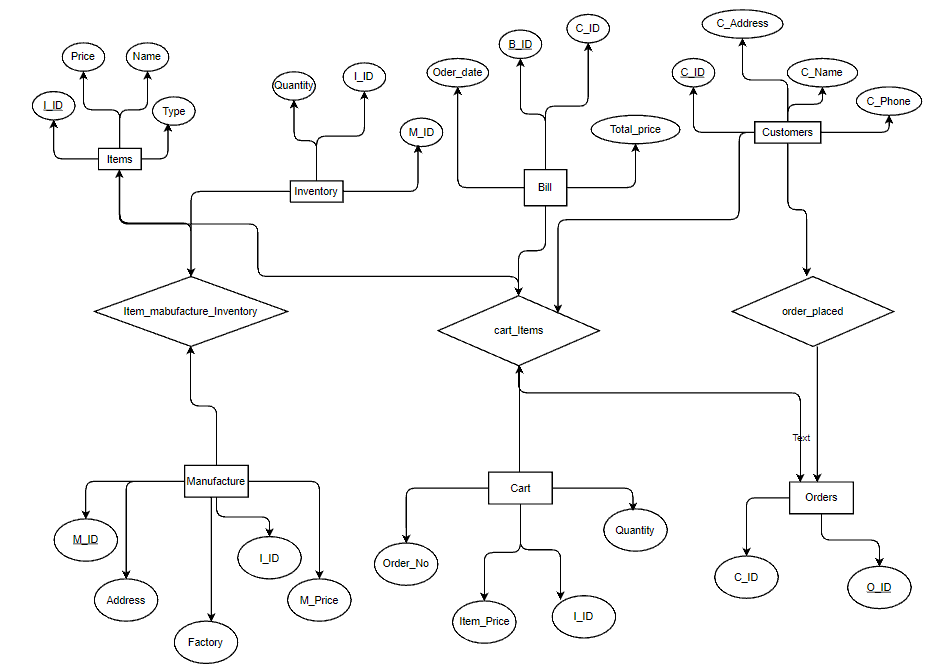




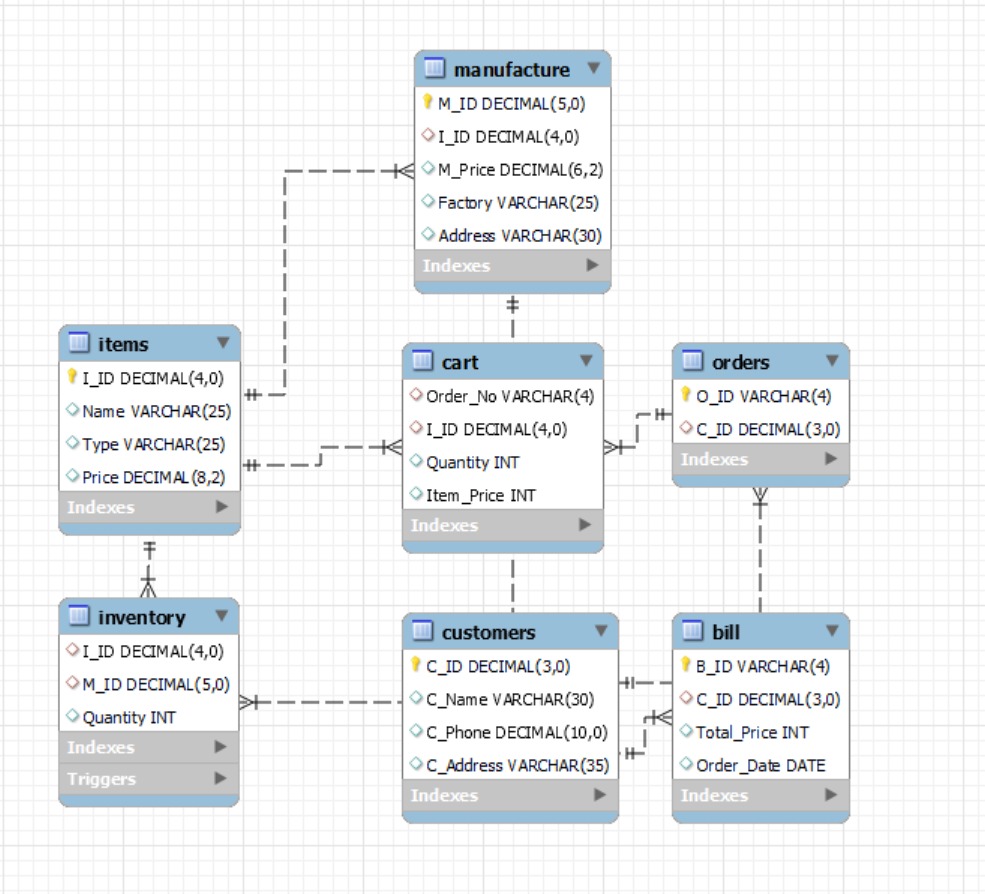


ER Diagrams

Entity Relation



Normalised Diagram



Conclusion

The purpose of this project thesis was to create a new shopping process concept for the groceries and other items, with the aim to simplify the whole shopping process for customers by using a computerized shopping cart.

Smart shopping cart system was developed from the perspective of customer whose needs were not fully fulfilled once he takes part of the shopping system process

We can conclude without doubts after plethora of surveys and customer feedbacks that the quality of customer service in grocery industry in general is totally poor and insufficient. This absence comes automatically due to the grocery industry not being driven by product innovations rather than profitability.

Research shows that service innovation can be a key differentiator and driver for competitiveness in the future. Therefore, by using this new innovative system all part of shopping process will be winners: Customers will reduce wait time, simplify the shopping process, knowing what and where to buy it, making their shopping process more pleasant and manageable.

For stores: They can increase sales (both in terms of immediate impulse buys and future customer loyalty) by producing happier customers. Additionally, it will increase their staff productivity (thus decreasing their costs) by efficiently matching staff members to customers.

Furthermore, it gives them management data about their customers and their processes, which can lead to ongoing competitive advantages.