

Electronics Online Store

ABSTRACT

After the COVID19 pandemic, e-commerce is quickly becoming a widely accepted and used business model. More and more businesses are developing web sites that allow them to conduct commercial transactions over the internet. It is fair to argue that online purchasing is growing more common. Moreover, due to the large segment that uses electronics nowadays and the emerging demand on electronics in the future, we decided to create an e-commerce website for Electronics Sale.



“E Shopper” Electronics Online Store

A ECEN424 “Intro. to Databases” Project Report

By

Ahmed Mohamed Fouad	18101467
Ahmed Ayman Ahmed	18101863
Emil Mourad Matta	18101962
Karim Mohamed	18101750
Mohamed Alsisi	18100695
Sara Hossam Anwar	18101273

Submitted in partial fulfillment of the requirements

for ECEN424 Project

Under the Supervision of

Dr. Tamer Arafa
Eng. Rameez Barakat

4/2/2022

TABLE OF CONTENTS

Chapter	Page
TABLE OF CONTENTS	iii
LIST OF FIGURES	iv
SECTION I: Introduction	1
SECTION II: Database Design	2
1.1 Database Design Diagram	2
1.2 Referential Integrity Constraint tables	4
1.3 Database column specifications	5
SECTION III: Business Rules & Assumptions	10
SECTION IV: List of Functions	11
4.1 Product Return	11
4.2 Invoice Issuance	14
4.3 Add Product	17
4.4 Update Product	18
4.3 Delete Product	19
4.4 Other Functions	19
SECTION V: User's Manual	20
5.1 User Friendly Home Page	20
5.2 Easy registration and easier login	21
5.3 Easy scroll between products	22
5.4 We care about our products	22
5.5 View your favorites list	23
5.6 View your cart and edit it	23
5.7 Easy checkout	24
5.8 Want to pay your balance? We got you	24
5.9 Contact us whenever you want	25
5.10 Don't like it anymore? Ok, we return it back	25
SECTION VI: Admin/ Manager's Manual	26
6.1 Manager's Dashboard	26
6.2 Generate customer reports	27
6.3 Manage products	27

LIST OF FIGURES

Figure	Page
Figure 1 Database Design Diagram	2
Figure 2 Referential Integrity I	4
Figure 3 Referential Integrity II	4
Figure 4 Product Column Specs	5
Figure 5 Invoice Column Specs	5
Figure 6 Prod_Inv Column Specs	6
Figure 7 Cards Column Specs	6
Figure 8 Customer Column Specs	7
Figure 9 Favorites Column Specs	7
Figure 10 Cart Column Specs	7
Figure 11 Manufacturer Column Specs	8
Figure 12 Categories Column Specs	8
Figure 13 Sub-Categories Column Specs	9
Figure 14 ProductReturn function header	11
Figure 15: ProductReturn function set of variables	11
Figure 16: ProductReturn refused cases	12
Figure 17: ProductReturn accepted	12
Figure 18:ProductReturn accepted	13
Figure 19: Return the result of the function	13
Figure 20 InvoiceIssuance function header	14
Figure 21: InvoiceIssuance function set of variables	14
Figure 22: InvoiceIssuance refused cases	15
Figure 23: InvoiceIssuance accepted	16
Figure 24 Add_Product function header	17
Figure 25: Add_Product function set of variables	17
Figure 26: Add_Product function body	17
Figure 27: UpdateProduct function header	18
Figure 28: UpdateProduct function set of variables	18
Figure 29: UpdateProduct function body	18
Figure 30: DeleteProduct function header	19
Figure 31: DeleteProduct function body	19
Figure 32 Home Page Frontend	20
Figure 33 Register Form	21
Figure 34 Login Form	21
Figure 35 Our Shop Page	22
Figure 36 Product Details Page	22
Figure 37 Favorites Page	23
Figure 38 Shopping Cart Page	23
Figure 39 Checkout Page	24
Figure 40 My Balance Page	24
Figure 41 Manager's Dashboard	26

Figure 42 Customer Reports I	27
Figure 43 Customer Reports II	27
Figure 44 Manage Products	27

SECTION I: Introduction

The objective of this project is to create an e-commerce website for Electronics Sale. It shows the user a catalogue of various products that may be purchased at the store. A shopping cart is supplied to the user to make online purchases easier. The goods in the shopping cart will be displayed as an order at the moment of checkout. More information will be required at that time to complete the transaction. The consumer will typically be prompted to fill out or select a billing address, a shipping address, a shipping option, and payment information, all of which will be saved in the user account. As soon as the order is placed, a customer service representative will contact the user to confirm the order.

The system is implemented using a 3-tier approach, with a backend database using MySQL, and php, a middle tier of PhpMyAdmin for communication between the database and the front end, and a web browser as the front-end client using BOOTSTRAP, HTML5, CSS3 and JS.

SECTION II: Database Design

1.1 Database Design Diagram

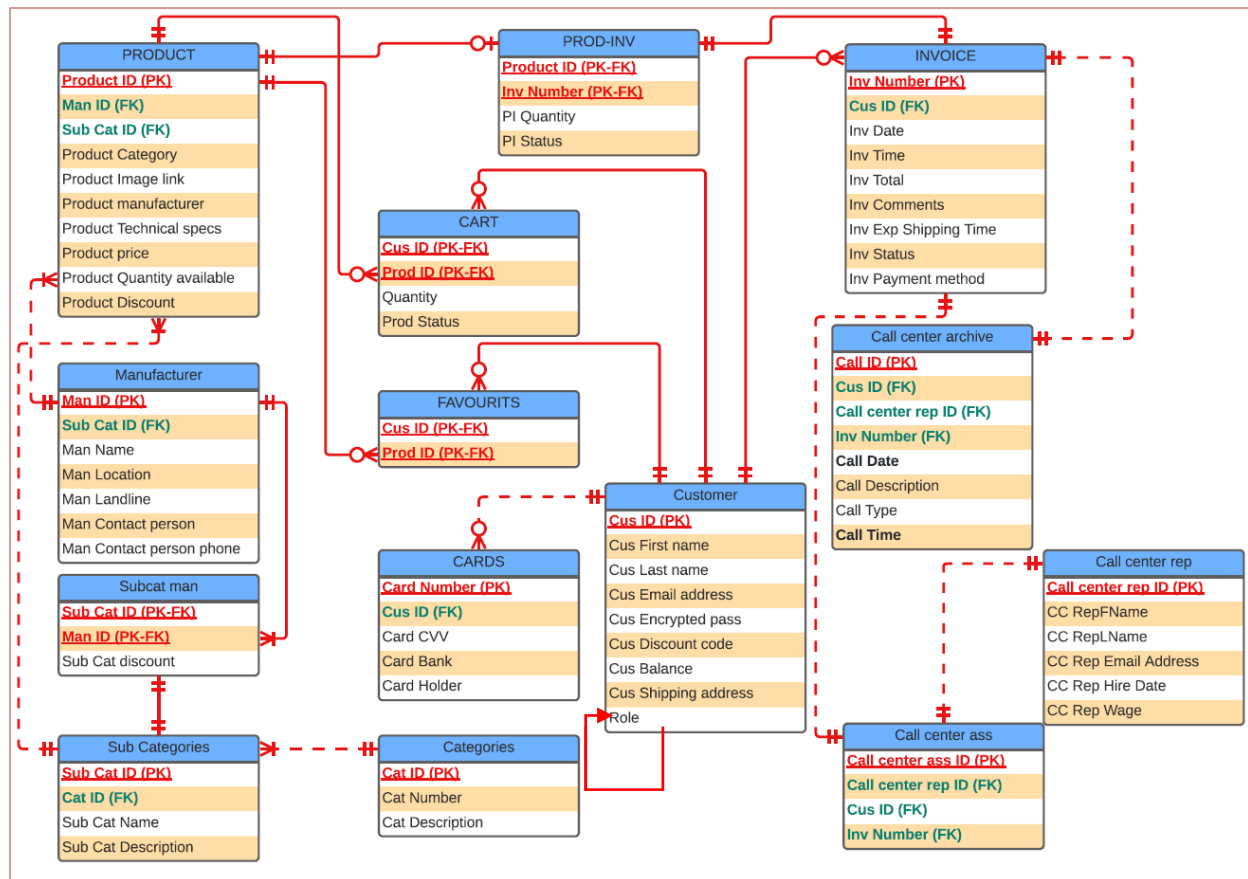


Figure 1 Database Design Diagram

The above shown diagram shows the proposed database design diagram, which have 4 major tables (PRODUCT, INVOICE, CUSTOMER, & MANUFACTURER) and the rest of the table are connecting these major ones together as follows:

- Each INVOICE can have more than one row referencing it in the PROD_INV table indicating that the user has purchased more than one product in the same invoice (**1:1 Weak ID-Dependent relationship**)
- CUSTOMER table has attributes representing information about the user, one of which is the CusRole which can have one 4 values:
 - 0: for normal User (can browse the website without having access for the database)
 - 1: for Manager (have the same roles as the admin with one extra permission which is changing the role of another user) (**1:1 Recursive relationship**)
 - 2: for Admin (can locate product, update inventory, check user payments, balance and fees)
 - 3: for Call Center Representatives (can only view his call assignments)
- PRODUCT table has attributes representing information about the product, including:

- SubCatID: a foreign key representing the subcategory of each product (to display each subcategory items in a separate page of the website) (**1:M strong relationship**)
- ManID: a foreign key representing the manufacturer supplying this product (**1:M strong relationship**)
- ProdQuantityAvailableRef/ ProdQuantityAvailableRef: representing amount of product in inventory refurbished and amount new
- A CUSTOMER may have more than one card as shown in CARDS (1:M strong relationship)
- CART, SUBCAT_MAN, & PROD_INV are all **Associative entities**
- On the issuance of each INVOICE a call center assignment is issued for a call center representative to make a follow up call within 24 hours confirming the order done by the customer
- Another type of calls is the Inquiry Call in which the customer contacts us for an inquiry
- These 2 call types (Follow up/ Inquiry calls) are stored in CallCenterArchive table

1.2 Referential Integrity Constraint tables

Relationship		Referential Integrity Constraint	Cascading Behavior	
Parent	Child		On Update	On Delete
PRODUCT	PROD_INV	ProdID in PROD_INV must exist in ProdID in PRODUCT	Yes	No
INVOICE	PROD_INV	InvNumber in PROD_INV must exist in InvNumber in INVOICE	Yes	Yes
CUSTOMER	CARDS	CusID in CARDS must exist in CusID in CUSTOMER	Yes	Yes
CUSTOMER	FAVOURITS	CusID in FAVOURITS must exist in CusID in CUSTOMER	Yes	Yes
CUSTOMER	CART	CusID in CART must exist in CusID in CUSTOMER	Yes	Yes
PRODUCT	FAVOURITS	ProdID in FAVOURITS must exist in ProdID in PRODUCT	Yes	Yes
PRODUCT	CART	ProdID in CART must exist in ProdID in PRODUCT	Yes	Yes

Figure 2 Referential Integrity I

Relationship		Referential Integrity Constraint	Cascading Behavior	
Parent	Child		On Update	On Delete
MANUFACTURER	SubCat_Man	ManID in SubCat_Man must exist in ManID in MANUFACTURER	Yes	Yes
MANUFACTURER	Product	ManID in Product must exist in ManID in MANUFACTURER	Yes	Yes
Sub_Categories	product	SubCatID in Sub_Categories must exist in SubCatID in Product	Yes	Yes
Sub_Categories	SubCat_Man	SubCatID in Sub_Categories must exist in SubCatID in SubCat_Man	Yes	Yes
Categories	Sub_Categories	CatID in Categories must exist in CatID in Sub_Categories	Yes	No
Customer	Invoice	CusID in Customer must exist in CusID in Invoice	Yes	No

Figure 3 Referential Integrity II

1.3 Database column specifications

PRODUCT					
Column Name	Data Type	Key	Required	Default Values	Remarks
ProdID	Integer	Primary Key	Yes	None	
ProdImageLink	VARCHAR(50)	NO	Yes	None	
ProdManufacturer	VARCHAR(50)	NO	Yes	None	
<u>ProdTechnicalSpecs</u>	VARCHAR(200)	NO	Yes	None	
<u>ProdPrice</u>	Double	NO	Yes	None	
<u>ProdQuantityAvailableNew</u>	Integer	NO	Yes	None	
<u>ProdQuantityAvailableRef</u>	Integer	NO	Yes	None	
<u>ProdDiscount</u>	Double	NO	NO	None	

Figure 4 Product Column Specs

INVOICE					
Column Name	Data Type	Key	Required	Default Values	Remarks
<u>InvNumber</u>	Integer	Primary Key	Yes	None	
<u>InvDate</u>	Date	NO	Yes	None	Format: <u>yyyy-mm-dd</u>
<u>InvTime</u>	time	NO	Yes	None	Format: 00:00:00
<u>InvComments</u>	VARCHAR(200)	NO	NO	None	
<u>InvExpShipingTime</u>	VARCHAR(50)	NO	Yes	None	
<u>InvStatus</u>	VARCHAR(50)	NO	Yes	Processing	Must be a value of ('Processing', 'Shipped', 'Canceled')
<u>InvPaymentMethod</u>	VARCHAR(50)	NO	Yes	None	

Figure 5 Invoice Column Specs

PROD_INV

Column Name	Data Type	Key	Required	Default Values	Remarks
ProdID	Integer	Primary Key, Foreign Key	Yes	None	
<u>InvNumber</u>	Integer	Primary Key, Foreign Key	Yes	None	
<u>PIQuantity</u>	Integer	NO	Yes	None	
<u>PIStatus</u>	VARCHAR(50)	NO	Yes	New	Must be a value of ('New', 'Refurbished')

Figure 6 Prod_Inv Column Specs

CARDS

Column Name	Data Type	Key	Required	Default Values	Remarks
<u>CardNumber</u>	VARCHAR(12)	Primary Key	Yes	None	
<u>CardHolder</u>	VARCHAR(20)	NO	Yes	None	
<u>CardCVV</u>	VARCHAR (3)	NO	Yes	None	
<u>CardBank</u>	VARCHAR(50)	NO	Yes	None	
<u>CusID</u>	INT	Foreign Key	Yes	None	

Figure 7 Cards Column Specs

CUSTOMER

Column Name	Data Type	Key	Required	Default Values	Remarks
<u>CusID</u>	INT	Primary Key	Yes	None	
<u>CusFirstName</u>	VARCHAR(20)	NO	Yes	None	
<u>CusEmailAddress</u>	VARCHAR (50)	NO	Yes	None	
<u>CusEncryptedPass</u>	VARCHAR(50)	NO	Yes	None	
<u>CusBalance</u>	DOUBLE	NO	No	None	

Figure 8 Customer Column Specs

FAVOURITS

Column Name	Data Type	Key	Required	Default Values	Remarks
<u>CusID</u>	INT	Primary Key, Foreign key	Yes	None	
ProdID	INT	Primary Key, Foreign key	Yes	None	

Figure 9 Favorites Column Specs

CART

Column Name	Data Type	Key	Required	Default Values	Remarks
<u>CusID</u>	INT	Primary Key, Foreign key	Yes	None	
ProdID	INT	Primary Key, Foreign key	Yes	None	
Quantity	INT	NO	Yes	None	

Figure 10 Cart Column Specs

MANUFACTURER

Column Name	Data Type	Key	Required	Default Values	Remarks
<u>ManID</u>	INT	Primary Key	Yes	None	
<u>ManName</u>	VARCHAR (50)	NO	Yes	None	
<u>ManDiscount</u>	DOUBLE	NO	Yes	None	
<u>SubCatID</u>	INT	Foreign Key	Yes	None	

Figure 11 Manufacturer Column Specs

Categories

Column Name	Data Type	Key	Required	Default Values	Remarks
<u>CatID</u>	INT	Primary Key	Yes	None	
<u>CatName</u>	VARCHAR (50)	NO	Yes	None	
<u>CatDescription</u>	VARCHAR(200)	NO	Yes	None	

Figure 12 Categories Column Specs

Sub-Categories

Column Name	Data Type	Key	Required	Default Values	Remarks
<u>SubCatID</u>	INT	Primary Key	Yes	None	
<u>SubCatName</u>	VARCHAR (50)	NO	Yes	None	
<u>SubCatDescription</u>	VARCHAR(200)	NO	Yes	None	
<u>CatID</u>	INT	Foreign Key	Yes	None	

Figure 13 Sub-Categories Column Specs

SECTION III: Business Rules & Assumptions

In this section, we demonstrate the most important business rules and assumptions:

1. The product category is standard; not all categories are necessarily in stock.
2. The product list is updated as necessary, yet a product on that list may not be ordered if the product shop owner decides that a product is not desirable for some reason.
3. Product will be classified as new or refurbished (if returned between 15 and 30 days from purchase)
4. Installment option is monthly payments (for one year) and would cost 20% extra for interest rate.
5. Refurbished products are sold with 10% discount from original price.
6. When a customer buys a product, the purchase date is stored, and no returns are allowed after 30 days.
7. Returns are allowed only for cash products.
8. Returns less than 15 days come with no extra fees.
9. Returns between 15 and 30 days have 15% restocking fees.
10. A customer cannot check out products unless they have zero balance.
11. Managerial Hierarchy:
 - Normal User (can browse the website without having access for the database)
 - Manager (have the same roles as the admin with one extra permission which is changing the role of another user)
 - Admin (can locate product, update inventory, check user payments, balance and fees)
 - Call Center Representatives (can only view his call assignments)

SECTION IV: List of Functions

4.1 Product Return

In this part, we describe how we implemented the following points using **sql** function.

9. When a customer buys a product, the purchase date is stored, and no returns are allowed after 30 days.
10. Returns are allowed only for cash products.
11. Returns less than 15 days come with no extra fees.
12. Returns between 15 and 30 days have 15% restocking fees.

To implement these points, we created a **sql** function called **ProductReturn** that takes CustomerID, InvoiceNumber, ProductName and Quantity as inputs and return varchar output.

```
CREATE FUNCTION ProductReturn(CustomerID INT, InvoiceNumber INT, ProductName VARCHAR(50), Quantity INT)
RETURNS VARCHAR(100)
NOT DETERMINISTIC
```

Figure 14 ProductReturn function header

So first we declare a set of variables that can help us in implementing the function.

```
Declare Result VARCHAR(100);
SET @CurrentDate = NOW();
SET @ExistInvoice = (SELECT Count(*) FROM INVOICE WHERE INVOICE.InvNumber=InvoiceNumber );
SET @ExistID = (SELECT Count(*) FROM INVOICE WHERE INVOICE.InvNumber=InvoiceNumber AND INVOICE.CusID=CustomerID );
SET @ProductID = (SELECT ProdID FROM PRODUCT WHERE PRODUCT.ProdName=ProductName);
SET @InvoiceDate= (SELECT InvDate FROM INVOICE WHERE INVOICE.InvNumber=InvoiceNumber);
SET @ValidQuantity = (SELECT PIQuantity FROM PROD_INV WHERE PROD_INV.ProdID=@ProductID AND PROD_INV.InvNumber=InvoiceNumber);
Set @DiffDate = DATEDIFF( @CurrentDate, @InvoiceDate );
Set @ProductPrice = (SELECT ProdPrice FROM PRODUCT WHERE PRODUCT.ProdID=@ProductID);
SET @Method = (SELECT InvPaymentMethod FROM INVOICE WHERE INVOICE.InvNumber=InvoiceNumber);
SET @Equal = @Quantinty-@ValidQuantity;
```

Figure 15: ProductReturn function set of variables

CurrentDate: is the current date of the product return request.

ExistInvoice: equals one if there an invoice satisfying the InvoiceNumber that was entered by the user, otherwise its equal to zero.

ExistID: equals one if the invoice was assigned to the user who tries to return his product, otherwise its equal to zero.

ProductID: is the product ID that can be obtained from the product table.

InvoiceDate: is the date of the invoice issuance.

ValidQuantity: is the maximum quantity that user can return of a single product that he purchased before.

DiffDate: is difference between the current date and the invoice date in days.

Second, we check the cases that the user can't return his products which are the following.

```
IF @ExistInvoice=0 THEN
    SET Result ="The Invoice number is not found";
ELSEIF @ExistID=0 THEN
    SET Result="The invoice number doesn't match with the customer invoices";
ELSEIF @Method!="Cash" THEN
    SET Result="Products only could be returned for Cash payment method";

ELSEIF Quantity> @ValidQuantity THEN
    SET Result="the Input quantity is not valid";
```

Figure 16: ProductReturn refused cases

The first case is when the invoice number entered by the user isn't valid.

The second case is when the invoice number entered by the user is assigned to another user.

The third case is when the user selects to buy the products by installment so he can't return these products.

The fourth case is when the quantity that the user wants to return is more than the quantity he asked when the purchase was made.

```
ELSEIF @DiffDate<15 THEN
    SET Result="Your request is accepted";

    UPDATE PRODUCT
    SET
        ProdQuantityAvailableNew=ProdQuantityAvailableNew+Quantity
    WHERE
        PRODUCT.ProdID=@ProductID;

    UPDATE INVOICE
    SET
        InvTotal=InvTotal- (@ProductPrice * Quantity)
    WHERE
        INVOICE.InvNumber=InvoiceNumber;

    UPDATE PROD_INV
    SET
        PIQuantity=PIQuantity-Quantity
    WHERE
        PROD_INV.ProdID=@ProductID AND PROD_INV.InvNumber=InvoiceNumber;

    SET @Total= (SELECT InvTotal FROM INVOICE WHERE INVOICE.InvNumber=InvoiceNumber);

    IF @Equal=0 THEN
        DELETE FROM PROD_INV
        WHERE PROD_INV.InvNumber=InvoiceNumber AND PROD_INV.ProdID=@ProductID;
    END IF;
```

Figure 17: ProductReturn accepted

After that we check if the difference date is less than 15 days, so we accept the user return request and return this quantity as new and return to the customer his money.

```
ELSEIF @DiffDate>15 AND @DiffDate<30 THEN
    SET Result="Your request is accepted";

    UPDATE PRODUCT
    SET
        ProdQuantityAvailableRef=ProdQuantityAvailableRef+Quantity
    WHERE
        PRODUCT.ProdID=@ProductID;

    UPDATE INVOICE
    SET
        InvTotal=InvTotal- (@ProductPrice * Quantity * 0.85)
    WHERE
        INVOICE.InvNumber=@InvoiceNumber;

    UPDATE PROD_INV
    SET
        PIQuantity=PIQuantity-Quantity
    WHERE
        PROD_INV.ProdID=@ProductID AND PROD_INV.InvNumber=InvoiceNumber;

    SET @Total= (SELECT InvTotal FROM INVOICE WHERE INVOICE.InvNumber=InvoiceNumber);

    IF @Equal=0 THEN
        DELETE FROM PROD_INV
        WHERE PROD_INV.InvNumber=InvoiceNumber AND PROD_INV.ProdID=@ProductID;
    END IF;
```

Figure 18:ProductReturn accepted

If the difference date is more than 15 days but less than 30 days, we accept the user return request but in this case we only return to him 0.85 of the money he paid due to 15% stocking fees.

```
ELSE
    SET Result="Your request has passed the allowed duration (30 days)";

END IF;

RETURN Result;
```

Figure 19: Return the result of the function

Now if the difference date is more than 30 days, we refuse the customer return request and finally we return the result.

4.2 Invoice Issuance

In this part, we describe how we implemented the following points using **sql** function.

5. When a customer selects one or more products, an invoice is issued. Each invoice may contain charges for one or more products.
6. Users can buy in cash, or using installments, thus having multiple outstanding invoices for a purchase.
7. Installment option is monthly payments (for one year) and would cost 20% extra for interest rate.
8. Refurbished products are sold with 10% discount from original price.

To implement these points, we created a **sql** function called **InvoiceIssuance** that takes CustomerID, ProductName, InvoiceComments, Quantity, Method and ProductState as inputs and return varchar output.

ProductState is equal to one if the user wants to buy new products otherwise it's equal to zero.

```
CREATE FUNCTION InvoiceIssuance( CustomerID INT,ProductName VARCHAR(50),InvoiceComments VARCHAR(100), Quantity INT,Method VARCHAR(100),InvoiceState BOOL,ProductState BOOL)
RETURNS VARCHAR(100)
NOT DETERMINISTIC
```

Figure 20 InvoiceIssuance function header

So first we declare a set of variables that can help us in implementing the function.

```
SET @CurrentDate=NOW();
SET @ProductID= (SELECT ProdID FROM PRODUCT WHERE PRODUCT.ProdName=ProductName);

SET @ValidQuantityNew =(SELECT ProdQuantityAvalilableNew FROM PRODUCT WHERE PRODUCT.ProdName=ProductName);
SET @ValidQuantityRef =(SELECT ProdQuantityAvalilableRef FROM PRODUCT WHERE PRODUCT.ProdName=ProductName);
SET @ProductPrice =(SELECT ProdPrice FROM PRODUCT WHERE PRODUCT.ProdID=@ProductID);
SET @CustomerBalance =( SELECT CusBalance FROM CUSTOMER WHERE CUSTOMER.CusID=CustomerID);
```

Figure 21: InvoiceIssuance function set of variables

CurrentDate: is the current date of the product return request.

ProductID: is the product ID that can be obtained from the product table.

ValidQuantityNew: is the maximum quantity (New) that user can buy of a single product.

ValidQuantityRef: is the maximum quantity (Refurbished) that user can buy of a single product.

Product Price: is the product price (New) that can be obtained from the product table.

CustomerBalance: is the current Balance of the customer.

```
IF @CustomerBalance>0 THEN
    SET Result="You can't place any new order untill your balance become zero";

ELSEIF Quantity>@ValidQuantityNew AND ProductState=1 THEN
    SET Result="there isn't enough quantity";

ELSEIF Quantity>@ValidQuantityRef AND ProductState=0 THEN
    SET Result="there isn't enough quantity";
```

Figure 22: InvoiceIssuance refused cases

The first case is when customer balance is more than zero.

The second case is when user wants to buy new products more than the available quantity.

The third case is when user wants to buy refurbished products more than the available quantity.

```

ELSEIF InvoiceState=1 THEN
    SET Result="Your invoice has been created ";
    SET @InvoiceNumber=(SELECT InvNumber FROM INVOICE ORDER BY InvNumber DESC LIMIT 1);
    IF Method="Cash" THEN
        UPDATE INVOICE
        SET
            InvTotal=InvTotal+(@ProductPrice * Quantity)
        WHERE
            INVOICE.InvNumber=@InvoiceNumber;
    ELSE
        UPDATE INVOICE
        SET
            InvTotal=InvTotal+(1.2*@ProductPrice * Quantity)
        WHERE
            INVOICE.InvNumber=@InvoiceNumber;
        UPDATE CUSTOMER
        SET
            CusBalance=CusBalance+(1.2*@ProductPrice * Quantity)
        WHERE
            CUSTOMER.CusID=CustomerID;
    END IF;

INSERT INTO PROD_INV(ProdID, InvNumber, PIQuantity,PIStatus) VALUES (@ProductID,@InvoiceNumber,Quantity,@ProductType);
IF @ProductType='NEW' THEN
    UPDATE PRODUCT
    SET
        ProdQuantityAvailableNew=ProdQuantityAvailableNew - Quantity
    WHERE
        PRODUCT.ProdID=@ProductID;
ELSE
    UPDATE PRODUCT
    SET
        ProdQuantityAvailableRef=ProdQuantityAvailableRef - Quantity
    WHERE
        PRODUCT.ProdID=@ProductID;
END IF;

```

Figure 23: InvoiceIssuance accepted

After that we check if the InvoiceState is equal to one which means that the customer wants to buy more than one type of the product in the same Invoice, we add these products to the same invoice of the customer and check the payment method is cash then he will pay the real price of products, otherwise he will pay extra 20% as an interest rate.

If the InvoiceState is equal to zero , then we will do the same thing expect that we will create a new invoice instead of updating another one.

4.3 Add Product

In this part, we describe how we implemented the add product to the DB using **sql** function.

To implement these points, we created a **sql** function called **Add_Product** that takes (ProductName, CategoryName, ProductImageLink, ManufacturerName, TechnicalSpecs, Price, QuantityAvailableNew, QuantityAvailableRef and Discount) as inputs and return “1” output if the product has been added successfully.

```
1  DELIMITER //
2  CREATE FUNCTION Add_Product (ProductName VARCHAR(100), CategoryName VARCHAR (150) , ProductImageLink VARCHAR(150)
3  , ManufacturerName VARCHAR(150), TechnicalSpecs VARCHAR(200), Price Double , QuantityAvailableNew INT,
4  QuantityAvailableRef INT, Discount Double)
```

Figure 24 Add_Product function header

So first we declare a set of variables that can help us in implementing the function.

```
8  BEGIN
9  Declare Result VARCHAR(100);
10 SET @ProductID = (SELECT ProID FROM PRODUCT WHERE PRODUCT.ProdName=ProductName);
11 Set @MANUFACTURERID = (Select ManID from MANUFACTURER where MANUFACTURER.ManName = ManufacturerName);
12 Set @CatID = (Select SubCatID from SUB_CATEGORIES where SUB_CATEGORIES.SubCatName = CategoryName);
```

Figure 25: Add_Product function set of variables

ProductID: is the product ID that can be obtained from the product table.

MANUFACTURERID: is the Manufacturer ID that can be obtained from the Manufacturer table.

CatID: is the SubCatID that can be obtained from the SUB_CATEGORIES table.

```
13 INSERT INTO product (ProdName, SubCatID, ProdImageLink, ManID, ProdTechnicalSpecs, ProdPrice,
14 ProdQuantityAvailableNew, ProdQuantityAvailableRef, ProdDiscount)
15 VALUES(ProductName, CatID, ProductImageLink , MANUFACTURERID, TechnicalSpecs, Price,
16 QuantityAvailableNew, QuantityAvailableRef,Discount);
17
18 RETURN '1';
19 END //
20 DELIMITER ;
21
```

Figure 26: Add_Product function body

4.4 Update Product

In this part, we describe how we implemented the add product to the DB using **sql** function.

To implement these points, we created a **sql** function called **UpdateProduct** that takes (ProductName, CategoryName, ProductImageLink, ManufacturerName, TechnicalSpecs, Price, QuantityAvailableNew, QuantityAvailableRef and Discount) as inputs and return “1” output if the product has been updated successfully.

```
1
2 DELIMITER //
3 CREATE FUNCTION UpdateProduct (ProductName VARCHAR(100), CategoryName VARCHAR (150) , ProductImageLink VARCHAR(150) ,
4 ManufacturerName VARCHAR(150), TechnicalSpecs VARCHAR(200),
5 Price Double , QuantityAvailableNew INT, QuantityAvailableRef INT, Discount Double)
6 RETURNS varchar(100)
```

Figure 27: UpdateProduct function header

So first we declare a set of variables that can help us in implementing the function.

```
8 BEGIN
9     Declare Result VARCHAR(100);
10    SET @ProductID = (SELECT ProID FROM PRODUCT WHERE PRODUCT.ProdName=ProductName);
11    Set @MANUFCAURERID = (Select ManID from MANUFCAURER where MANUFCAURER.ManName = ManufacturerName);
12    Set @CatID = (Select SubCatID from SUB_CATEGORIES where SUB_CATEGORIES.SubCatName = CategoryName);
```

Figure 28: UpdateProduct function set of variables

ProductID: is the product ID that can be obtained from the product table.

MANUFCAURERID: is the Manufacturer ID that can be obtained from the Manufacturer table.

CatID: is the SubCatID that can be obtained from the SUB_CATEGORIES table.

```
13 UPDATE product
14 SET ProdName = ProductName, SubCatID = CatID , ProdImageLink =ProductImageLink, ManID = MANUFCAURERID,
15 ProdTechnicalSpecs = TechnicalSpecs,
16 ProdPrice = Price , ProdQuantityAvailableNew = QuantityAvailableNew , ProdQuantityAvailableRef = QuantityAvailableRef,
17 ProdDiscount = Discount
18 WHERE ProdID = ProductID ;
19 return '1';
20 END //
21 DELIMITER ;
```

Figure 29: UpdateProduct function body

4.3 Delete Product

In this part, we describe how we implemented the add product to the DB using **sql** function.

To implement these points, we created a **sql** function called **DeleteProduct** that takes (ProductName) inputs and return “1” output if the product has been deleted successfully.

```
1  # delete product
2
3  DELIMITER //
4  CREATE FUNCTION DeleteProduct(ProductName varchar(30))
5  RETURNS VARCHAR(100)
```

Figure 30: DeleteProduct function header

```
6  NOT DETERMINISTIC
7  BEGIN
8      Declare Result VARCHAR(100);
9      DELETE FROM product WHERE product.ProdName = ProductName;
10     return '1';
11
12 END //
13 DELIMITER ;
```

Figure 31: DeleteProduct function body

4.4 Other Functions

- **Add to Cart:** Insert to cart the products ID that the customer wants to add to cart and the quantity of each product
- **Display Cart:** Display to the user the name of products in cart, the price of each product, the quantity available for each product(new/refurbished), the discount on each product and the quantity of each product in the cart
- **Add to Favorites:** Insert to Favorites section of the customer the products IDs
- **Display Favorites:** Display to the user the name of products in favorites, the price of each product, the quantity available for each product(new/refurbished), the discount on each product, the manufacturer ID for each product and the quantity of each product.
- **Search by Product Name:** Function used to search for the closest results to what the user has written (to concatenate two or more than two-character expressions into a single string).
- **Check Customer Balance:** Check the balance of the customer if has installments he needs to pay.
- **Admin View User Report:** Display to the admin the customer invoice, Customer ID, customer first and last name, customer balance, invoice date, products IDs and name listed in the invoice and the quantity of each product(can be displayed only to the admin).

SECTION V: User's Manual

5.1 User Friendly Home Page

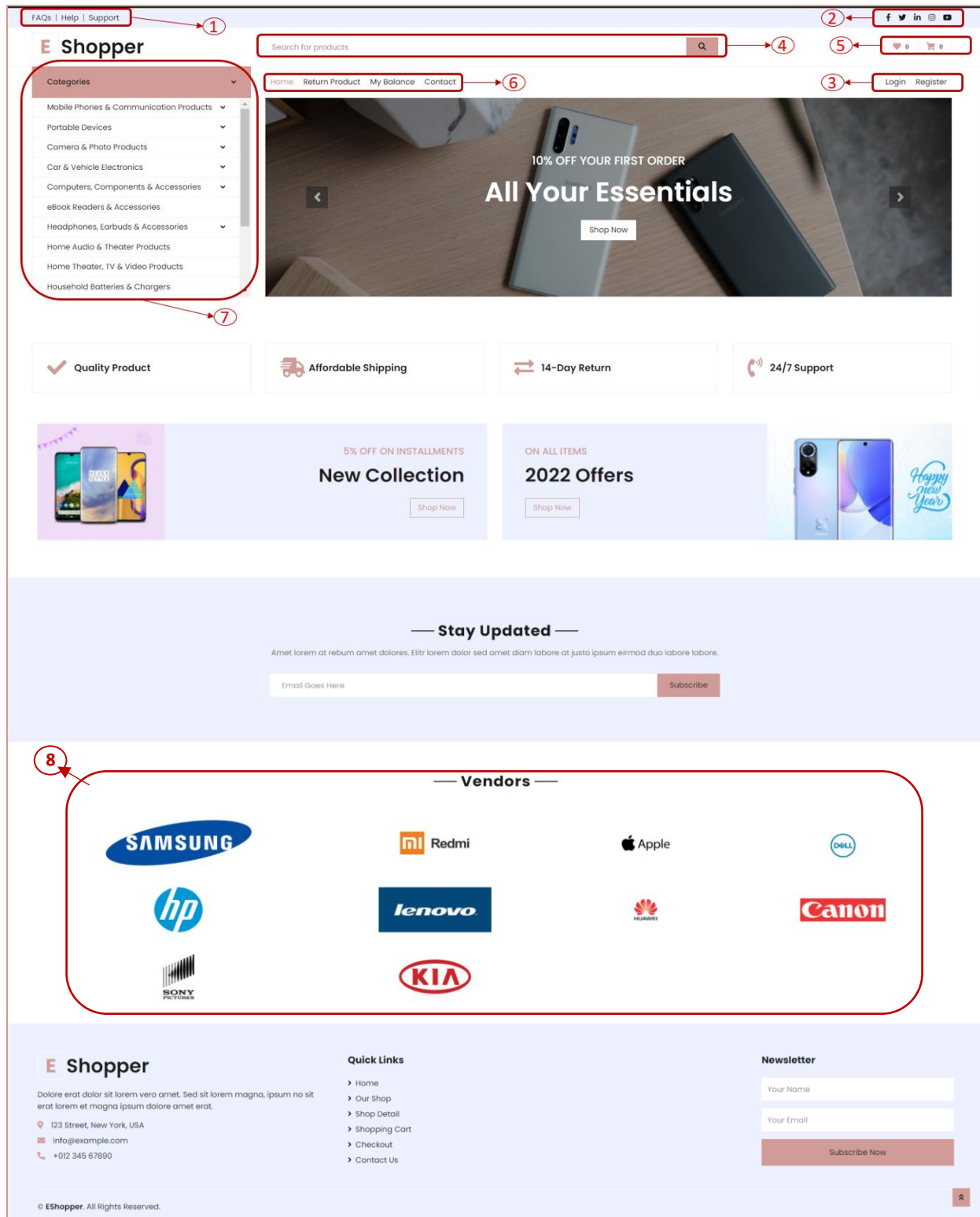


Figure 32 Home Page Frontend

The labels shown above are:

1. Get help & talk to support
2. Follow us on social media
3. Login or register new account
4. Search for a specific product (by full or part of the product name)
5. View your cart & favorite list (count changes when items are added to favorites or cart)
6. Quick Links for the website pages
 - Home: View latest products (Returns to Home Page)
 - Return Product: Return already purchased products, under the following conditions:
 - i. No returns are allowed after 30 days.
 - ii. Returns are allowed only for cash products.
 - iii. Returns less than 15 days come with no extra fees.
 - iv. Returns between 15 and 30 days have 15% restocking fees.
 - View your balance (If the user has any installments or restocking fees due)
 - Contact us for inquiries
7. View products by category (if a category has a subcategory, a small arrow is present at the end of the tile displaying the subcategories for the selected category)
8. Vendors displaying their products in our website

5.2 Easy registration and easier login

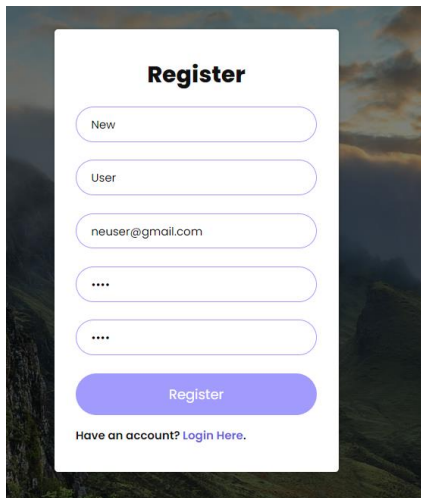
A screenshot of a 'Register' form. The form has a title 'Register' at the top. Below it are five input fields: 'New' (with a dropdown arrow), 'User', 'neuser@gmail.com', two fields with four dots (passwords), and a 'Register' button. At the bottom, there is a link: 'Have an account? Login Here.'

Figure 33 Register Form

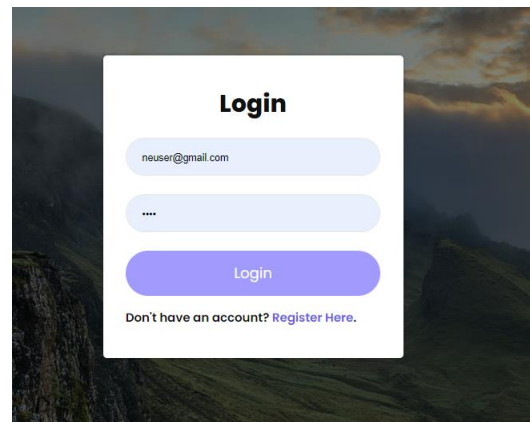
A screenshot of a 'Login' form. The form has a title 'Login' at the top. Below it are two input fields: 'neuser@gmail.com' and a field with four dots (password). Below these is a 'Login' button. At the bottom, there is a link: 'Don't have an account? Register Here.'

Figure 34 Login Form

5.3 Easy scroll between products

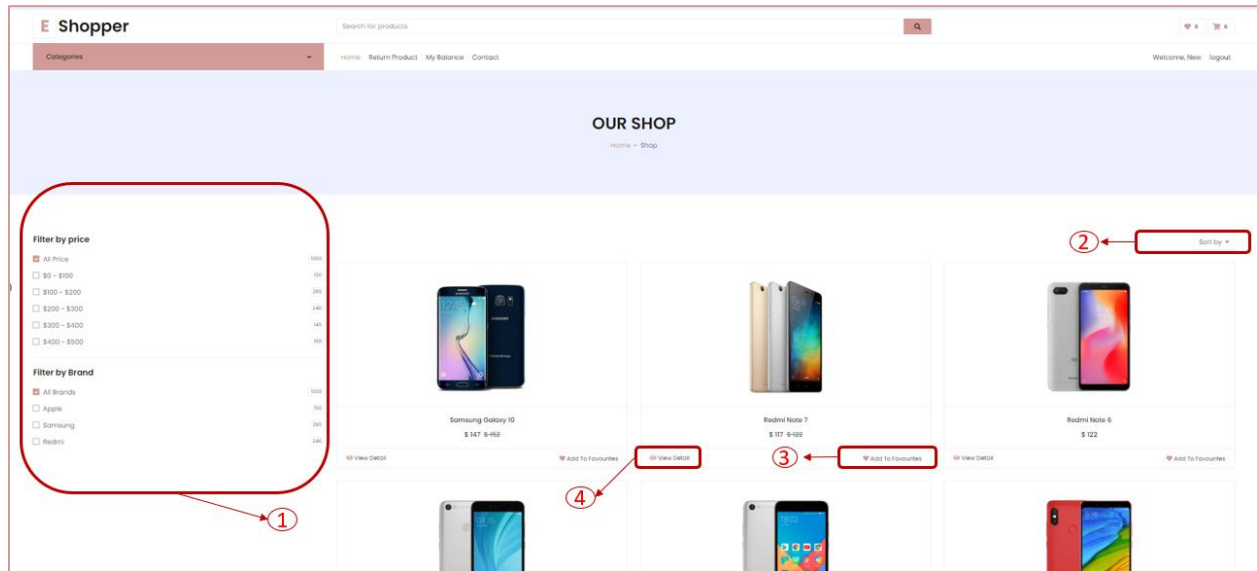


Figure 35 Our Shop Page

1. Filter products as you wish
2. Sort products
3. Add product to your favorites list
4. View more details about the product

5.4 We care about our products

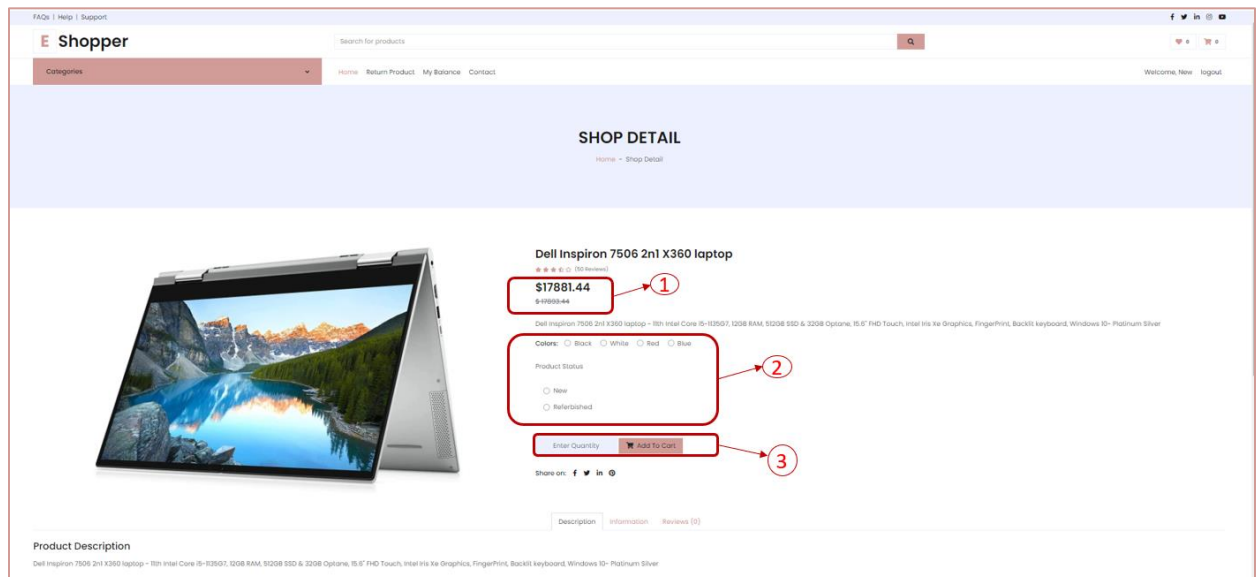


Figure 36 Product Details Page

1. See the price after and before discount
2. Choose details you want in your product (Refurbished products are sold with 10% discount from original price)
3. Add to your cart the quantity you want

5.5 View your favorites list









FAVOURITES		
Home - Favourites		
Products	Price	Remove
 Samsung Galaxy 10	\$ 152	
 Dell Vostro 3500 laptop	\$ 12775	

Figure 37 Favorites Page

5.6 View your cart and edit it

SHOPPING CART					
Home - Shopping Cart					
Products	Price	Quantity	Product Status	Total	Remove
 Samsung Galaxy 10	\$ 152	1	New	\$ 152	
 Dell Vostro 3500 laptop	\$ 12775	2	Referbished	\$ 25550	
 Epson LQ-590 Impact Printer	\$ 190	1	New	\$ 190	

Cart Summary

Subtotal \$ 25892

Shipping \$10

Total \$ 25902

Figure 38 Shopping Cart Page

1. Apply your discount coupon if you have
2. checkout

5.7 Easy checkout

The screenshot shows the 'CHECKOUT' page with a breadcrumb 'Home - Checkout'. It is divided into three main sections: Billing Address, Order Total, and Payment. The Billing Address section shows 'First Name: Ahmed' and 'Last Name: Fouad'. The Shipping Address is 'West of Somid, 6th October City, Egypt', which is highlighted with a red box and a circled '1' with an arrow. The Order Total section lists products: Samsung Galaxy 10 (\$152), Dell Vostro 3500 laptop (\$25550), and Epson LQ-590 Impact Printer (\$190). The Subtotal is \$25892, Shipping is \$10, and the Total is \$25902. The Payment section has two options: 'Cash on Delivery' (selected) and 'Installment', which is highlighted with a red box and a circled '2' with an arrow.

CHECKOUT
Home - Checkout

Billing Address

First Name
Ahmed

Last Name
Fouad

E-mail
a.fouad@gmail.com

Shipping Address
West of Somid, 6th October City, Egypt

Order Total

Products

Samsung Galaxy 10	\$ 152
Dell Vostro 3500 laptop	\$ 25550
Epson LQ-590 Impact Printer	\$ 190
Subtotal	\$ 25892
Shipping	\$10
Total	\$ 25902

Payment

☒ Cash on Delivery

☐ Installment

Figure 39 Checkout Page

1. Add your shipping address
2. Choose your payment method (Installment option is monthly payments (for one year) and would cost 20% extra for interest rate)
3. Press to place your order

5.8 Want to pay your balance? We got you

The screenshot shows the 'BALANCE PAGE' with a breadcrumb 'Home - Balance Page'. It features a 'My Balance' section with 'Due Amount: \$ 21,200' and 'Amount to Pay'. There is an input field labeled 'Enter Desired Amount' highlighted with a red box and a circled '1' with an arrow. Below it is a 'Pay Now' button highlighted with a red box and a circled '2' with an arrow.

FAQs | Help | Support

E Shopper Search for products

Categories Home Return Product My Balance Contact Welcome, Ahmed logout

BALANCE PAGE
Home - Balance Page

My Balance

Due Amount \$ 21,200

Amount to Pay

Enter Desired Amount

Pay Now

Figure 40 My Balance Page

1. Enter the amount you want to pay
2. Press Pay Now

5.9 Contact us whenever you want

Categories

HomeReturn ProductMy BalanceContact

Welcome, Newlogout

CONTACT US

Home - Contact US

— Contact For Any Queries —

New

neuser@gmail.com

I love you guys

Best website ever ❤️❤️❤️

Send Message

Get In Touch

Justo sed diam ut sed amet duo amet lorem amet stet sea ipsum, sed duo amet et. Est elit dolor elit erat sit sit. Dolor diam et erat clita ipsum justo sed.

Store 1

123 Street, New York, USA

info@example.com

+012 345 67890

Store 2

123 Street, New York, USA

info@example.com

+012 345 67890

5.10 Don't like it anymore? Ok, we return it back

Categories

HomeReturn ProductMy BalanceContact

Welcome, Newlogout

RETURN & REFUND

Home - Return Product


— Enter the Product Data —

Invoice Number

Product Name

Quantity

Submit



SECTION VI: Admin/ Manager's Manual

6.1 Manager's Dashboard







Function	Description
 1	Search customer by ID to check payments, balance and fees
 2	Add, Delete, or Update product info
 3	Update quantity of new/ refurbished products in inventory
 4	Add, Remove, or Update Admins info
 5	Add, Remove, or Update Call Center Representatives info and their assignments
 6	View Call Center history

Figure 41 Manager's Dashboard

1. View customer report (Search customer by ID to check payments, balance and fees)
2. Manage products (Add, Delete, or Update product info)
3. Manage your Inventory (Manage products (new/ refurbished) amounts available in the inventory)
4. Manage your employees (a function exclusive for managers, and allows them to change users' roles in the website which in turn changes their access permissions)
5. Manage the call center representatives (assign new call assignments & update call center representatives' info)
6. View call center archive

6.2 Generate customer reports

FAQs | Help | Support

E Shopper

Search for products

Categories

Home Return Product My Balance Contact

Welcome, Sara logout

Search by customer ID

1

CusID	CusFirstName	CusBalance	InvNumber	InvDate	InvTotal	InvStatus	ProdID	ProdName	PIQuantity
-------	--------------	------------	-----------	---------	----------	-----------	--------	----------	------------

E Shopper

Dolore erat dolor sit lorem vero amet. Sed sit lorem magna, ipsum no sit erat lorem et magna ipsum dolore amet erat.

123 Street, New York, USA
info@example.com
+012 345 67890

Quick Links

- Home
- Our Shop
- Shop Detail
- Shopping Cart
- Checkout
- Contact Us

Newsletter

Your Name

Your Email

Subscribe Now

© EShopper. All Rights Reserved.

Figure 42 Customer Reports I

1. Add the customer ID to generate the report

E Shopper

2

Categories

Home Return Product My Balance Contact

Welcome, Sara logout

2

CusID	CusFirstName	CusBalance	InvNumber	InvDate	InvTotal	InvStatus	ProdID	ProdName	PIQuantity
2	Abou Bakr	550	2	2021-01-03	500	Processing	1	Samsung Galaxy 10	5
2	Abou Bakr	550	3	2021-01-30	250	Processing	20	Huawei Matepad T10S	1

Figure 43 Customer Reports II

6.3 Manage products

Products

Product name:
write here

Category name:
write here

Product image link:
write here

Manufacturer name:
write here

Technical spes:
write here

Price double:
write here

Quantity Available New INT:
write here

Quantity Available Ref INT:
write here

Discount Double:
write here

Select if the product is new or inserted before: --Make a choice--

Submit

Figure 44 Manage Products