ALLMART SHOPPING APPLICATION

PROJECT REPORT

Data Structures and Algorithms (CSE2003)

SLOT-G1

GROUP MEMBERS

Name	Registration No.
Aditya Gupta	16BCE0021
Sagar Udayan	16BCE0790
Vatsal Jain	16BCE0912

PROJECT SUPERVISOR

Prof. Murali S

APRIL,2017

CERTIFICATE

This is to certify that the project work entitled " Allmart Shopping Application" that is being submitted by "Sagar Udayan, Aditya Gupta, Vatsal Jain" for Engineering Physics (PHY1001) is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Place : Vellore

Date :

Signature of Students:

Sagar Udayan

Vatsal Jain

Aditya Gupta

Signature of Faculty:

Prof. Murali S

TABLE OF CONTENTS

Chapter 1: **Abstract**

Chapter 2: **Introduction**

Chapter 3: **Literature Survey**

Chapter 4: **Algorithm**

Chapter 5: **System Architecture**

Chapter 6: **Module Description**

----C based App Description

Chapter 7: **Output**

1-C code output

2-Proposed Website Screen Shots

Chapter 8: **Conclusion**

Chapter 9: **Result**

Chapter 10: **References**

Chapter 1: ABSTRACT

The objective of the project is to create a application for all the VITians so that they can easily shop for all the products they need for their daily usage easily by sitting at room rather than going physically to shop. Going to the Allmart for shopping after daily college is bit difficult for many, so we decided to create a online application from where all the people can buy any product of their choice and get a door step delivery with minimal charges. But due to our lack in knowledge of web development we created the dummy application of the above in C. This project is the output of our planning, schedule, programming skill and the hard work, and this report reflects our steps taken at various levels of programming skill, planning and schedule.We have learnt a lot during this project and liked the improvement in our testing skills and deep concept related to these kinds of projects.

Chapter 2: INTRODUCTION

In today's busy world, people don't have time for their personal needs. And the technology is so fast that anyone can do anything by just sitting in a room. The internet is the way that helps a person in all aspects. If someone wishes to buy and view things, he can buy online with the help of internet.

Today there are very least organizations which are manual. Everything is going to be computerized and online whether it is banking, advertising or shopping. We are trying to help people to make their life easier by proving online clothes shopping. So we decided to make dummy programme for online shopping at Allmart.

EXISTING SYSTEM:

The current system for shopping is to visit the Allmart shop manually and from the available product choose the item customer want and buying the item by payment of the price of the item .

1. It is less user-friendly.
2. User must go to shop and select products.
3. It is difficult to identify the required product.
4. Description of the product limited.
5. It is a time consuming process .
6. Not in reach of distant users.

PROPOSED SYSTEM:

In the proposed system customer need not go to the shop for buying the products. He can order the product he wishes to buy through the application. The shop owner will be admin of the system. Shop owner can appoint moderators who will help owner in managing the customers and product orders. The system also recommends a home delivery system for the purchased products.

Chapter 3: LITERATURE SURVEY

Online shopping is a form of electronic commerce which allows consumers to directly buy goods or services from a seller over the Internet using a web browser. Consumers find a product of interest by visiting the website of the retailer directly or by searching among alternative vendors using a shopping search engine, which displays the same product's availability and pricing at different e-retailers. As of 2016, customers can shop online using a range of different computers and devices, including desktop computers, laptops, tablet computers and smartphones.

History:

The growth of the internet as a secure shopping channel has developed since 1994, with the first sales of Sting album 'Ten Summoner's Tales'. Wine, chocolates and flowers soon followed and were among the pioneering retail categories which fuelled the growth of online shopping. Researchers found that having products that are appropriate for e-commerce was a key indicator of Internet success. Many of these products did well as they are generic products which shoppers didn't need to touch and feel in order to buy.

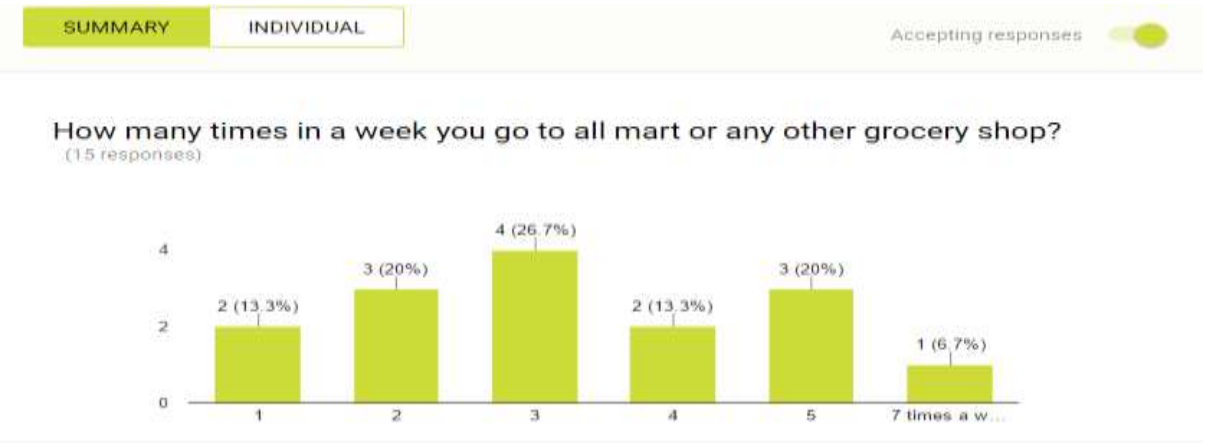


Fig 3.1

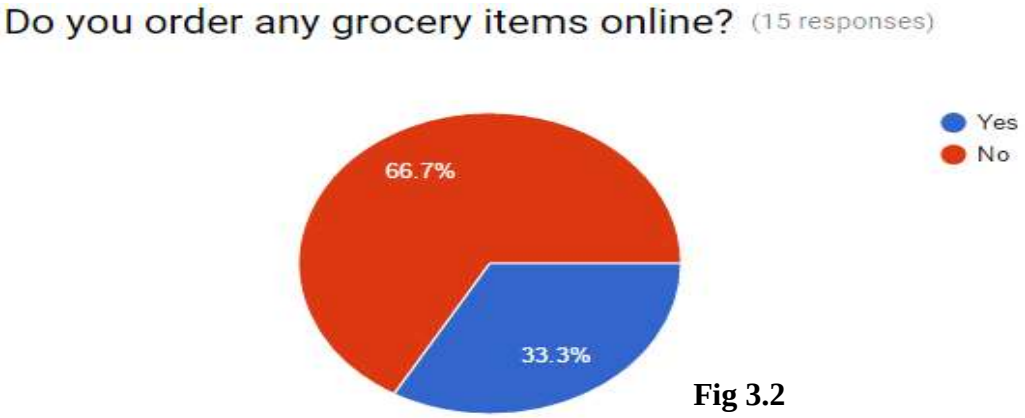


Fig 3.2

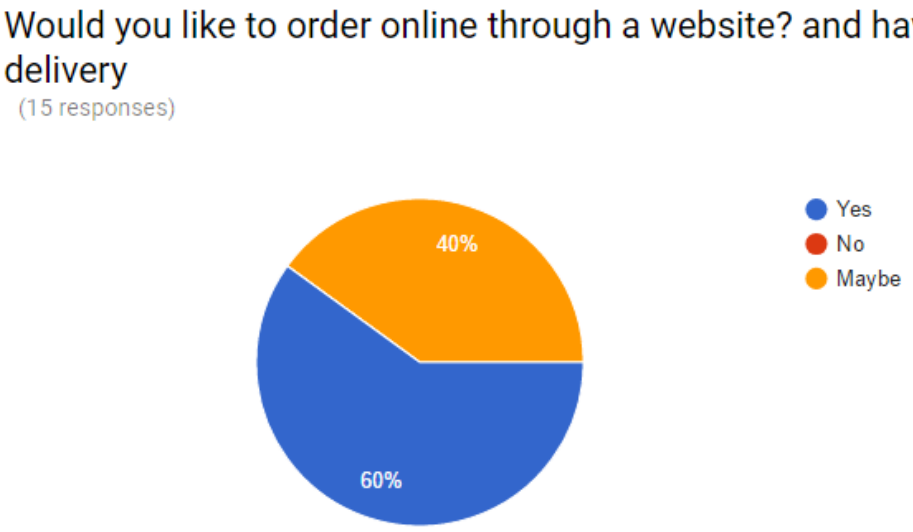


Fig 3.3

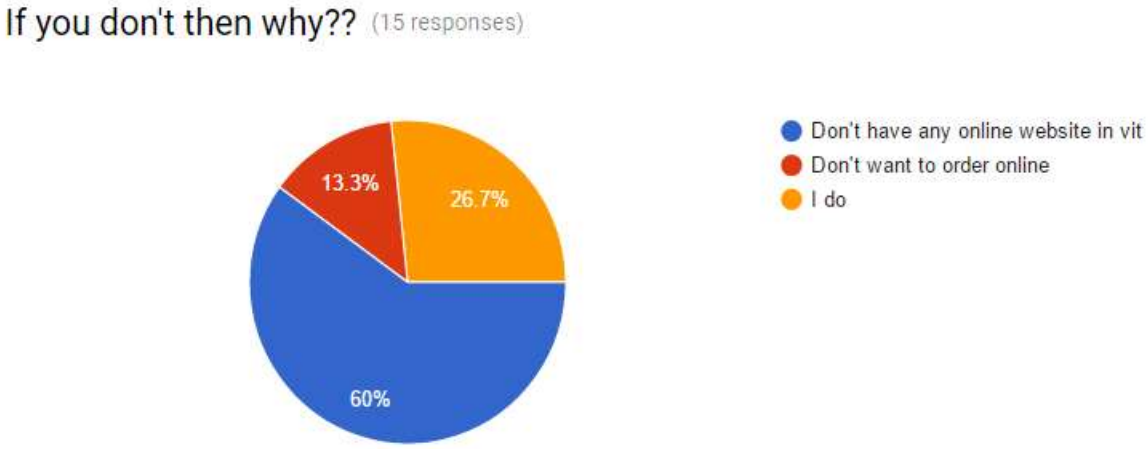


Fig 3.4

Fig 3.5

Most of the surveyed people agreed that there should be a online shopping facility and a door step Delivery option to facilitate shopping rather than physically going and shopping we can easily shop the product of our need and get the product delivered at our doorsgteps

Chapter 4: ALGORITHM

This program makes use of the linked list data structure to store data and perform various functions

We create a structure called item which stores the details of id, name, cost and quantity for each item, and the address for the next item in the list.

```
struct item
{
        int id;
        char name[20];
        int price;
        int qty;
        struct item *next;
}
```

We initialize *shopList of type item as NULL

A function shopInsert() is created to insert items into the list of items in the shop

```
        struct item *newnode;
        newnode = (struct item*) malloc (sizeof(struct item));
        GET newnode->name, newnode->price, newnode->qty
        An ID is assigned to each item in the shop
        if(shopList == NULL)
        {
                newnode->next = NULL;
                newnode->id = 10000;
        }
```

```
else
{
        newnode->id = shopList->id + 1;
        newnode->next = shopList;
}
shopList = newnode;
```

A function shopRemove(int id) is created to remove items from the shop. It returns 1 if the item is removed, else it returns 0

```
int flag = 0;
struct item *t1 = shopList, *t2;
while(t1->id != id)
{
        if(t1->next == NULL)
        {
                flag = 1;
                break;
        }
        t2 = t1;
        t1 = t1->next;
}
if(flag == 1)
{
        //id is not in the list
}
else
{
        if(t1 == shopList)
```

```
                shopList = shopList->next;
        else
                t2->next = t1->next;
        free(t1);
        return 1;
    }
    return 0;
```

A function shopDisplay() is created to display all the items and their details in the shop

```
    if(shopList == NULL)
        //Shop is empty
    else
    {
        struct item *t = shopList;
        while(t != NULL)
        {
            DISPLAY t->id, t->name, t->price, t->qty
            t = t->next;
        }
    }
```

A function shopModify() is created to edit the details of any item in the shop

```
    READ id
    int flag = 0;
    struct item *t = shopList;
    while(t->id != id)
    {
        if(t == NULL)
        {
```

```
                flag = 1;
                break;
            }
            t = t->next;
        }
        if(flag == 1)
        {
            //id is not in the list
        }
        else
        {
            GET t->name, t->price, t->qty
        }
```

A function owner() is created which performs the list operations for shop depending on inputs

```
        do
        {
            int ch
            GET ch
            switch(ch)
            {
                case 1: shopInsert();
                        break;
                case 2: GET id
                        if(shopRemove(id) == 1)
                            printf("Item with ID #%d has been removed\n",
id);
                        else
```

```
                                        printf("ID #%d is not in the list\n", id);
                                break;
        case 3: shopDisplay();
                                break;
        case 4: shopModify();
                                break;
        case 5: //Exits the function
                                printf("Thank you\n");
                                break;
        default:printf("Invalid choice\n");
    }
}while(ch != 5);
```

We initialize *cartList of type item as NULL

A function cartInsert() is created to take an ID and quantity from the user and add that particular item to their cart. Depending on the quantity entered by the user, the corresponding quantity of that item in the shopList is altered as well. If its quantity becomes 0, it is removed from the shop.

```
struct item *newnode;
newnode = (struct item*) malloc (sizeof(struct item));
int id, flag = 0;
GET id
struct item *t = shopList;
while(t != NULL)
{
    if(t->id == id)
    {
        flag = 1;
```

```
int cartflag = 0;
struct item *temp = cartList;
while(temp != NULL)
{
        if(temp->id == id)
        {
                cartflag = 1;
                break;
        }
        temp = temp->next;
}
if(cartflag == 0)
{
        newnode->id = id;
        newnode->name = t->name;
        newnode->price = t->price;
        int qty;
        while(1)
        {
                printf("Enter the quantity: ");
                scanf("%d", &qty);
                if(qty <= t->qty)
                {
                        newnode->qty = qty;
                        break;
                }
                else
```

```
                printf("Enter valid quantity\n");
        }
        t->qty = t->qty - qty;
        if(t->qty == 0)
                shopRemove(id);
        if(cartList == NULL)
        {
                newnode->next = NULL;
        }
        else
        {
                newnode->next = cartList;
        }
        cartList = newnode;
}
else
{
        t->qty += temp->qty;
        int qty;
        while(1)
        {
                printf("Enter the quantity: ");
                scanf("%d", &qty);
                if(qty <= t->qty)
                {
                        newnode->qty = qty;
                        break;
```

```
                    }
                    else
                        printf("Enter valid quantity\n");
                }
                t->qty = t->qty - qty;
                if(t->qty == 0)
                    shopRemove(id);
            }
            break;
        }
        t = t->next;
    }
    if(flag == 0)
        //id is not in the shop
```

A function cartRemove() is created to remove an item from the cart. The quantity of items is returned back.

```
    int id;
    GET id
    int flag = 0;
    struct item *t1 = cartList, *t2;
    while(t1->id != id)
    {
        if(t1->next == NULL)
        {
            flag = 1;
            break;
        }
```

```
            t2 = t1;
            t1 = t1->next;
        }
        if(flag == 1)
        {
            //id is not in the list
        }
        else
        {
            DISPLAY t1->id, t1->name before removing them
            struct item *t = shopList;
            while(t != NULL)
            {
                if(t->id == id)
                {
                    t->qty = t->qty + t1->qty;
                    break;
                }
                t = t->next;
            }
            if(t1 == cartList)
                cartList = cartList->next;
            else
                t2->next = t1->next;
            free(t1);
        }
```

A function cartModify is created to modify the quantity of an item in the cart

```
int id;
GET id
struct item *cart_t = cartList;
struct item *shop_t = shopList;
int flag = 0;
while(cart_t->id != id)
{
	if(cart_t == NULL)
	{
		flag = 1;
		break;
	}
	cart_t = cart_t->next;
}
if(flag == 1)
{
	printf("Item with ID #%d is not in the list!\n", id);
}
else
{
	int qty;
	while(shop_t->id != id)
	{
		shop_t = shop_t->next;
	}
	shop_t->qty += cart_t->qty;
	cart_t->qty = 0;
```

```
		while(1)
		{
			printf("Enter the quantity: ");
			scanf("%d", &qty);
			if(qty <= shop_t->qty)
			{
				cart_t->qty = qty;
				break;
			}
			else
				printf("Enter valid quantity\n");
		}
		shop_t->qty -= qty;
		if(shop_t->qty == 0)
			shopRemove(id);
	}
```

A function cartDisplay() is created to display all the items in the cart

```
	if(cartList == NULL)
		/Cart is empty
	else
	{
		int total = 0;
		struct item *t = cartList;
		while(t != NULL)
		{
			DISPLAY t->id, t->name, t->price, t->qty, Total price of that item (t->price)*(t->qty)
```

```
            total += (t->price)*(t->qty);
            t = t->next;
        }
        DISPLAY total
    }
```

A function checkout(char name[], char phno[]) is created to check out all the items in the cart and display the bill. After this, it empties the entire cart back to its original state.

```
    if(cartList != NULL)
    {
        printf("\t\t\t\t\tBILL\n\n");
        printf("Customer Name: %s\nContact Number: %s\n\n", name, phno);
        printf("ID\tNAME\tPRICE\tQUANTITY\tTOTAL PRICE\n");
        int total = 0;
        struct item *t = cartList;
        while(t != NULL)
        {
            printf("#%d\t", t->id);
            printf("%s\t", t->name);
            printf("%d\t", t->price);
            printf("%d\t", t->qty);
            printf("\t%d\n", (t->price)*(t->qty));
            total += (t->price)*(t->qty);
            t = t->next;
        }
        printf("\nNET COST = %d\n", total);
        if(total >= 1000)
        {
```

```
                printf("Your total expenditure is over 1000. Do you want to avail of free home delivery? (Y/N)\n");
            }
            else
            {
                printf("Do you want to avail of home delivery (Y/N)? Delivery charges are 10% of your total cost.\n");
            }
            GET answer
            while(cartList != NULL)
            {if(shopList != NULL)
        {
            printf("\n\t\t\t\t\t\tWELCOME!\n");
            char name[20], phno[11];
            printf("Enter your name: ");
            scanf("%s", name);
            printf("Enter your number: ");
            scanf("%s", phno);
            int ch;
            do
            {
                printf("\n1. See all items in the shop\n2. Add an item to item\n3. Remove an item from item\n4. Modify quantity\n5. See your item\n6. Check Out\n");
                printf("Enter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                    case 1: shopDisplay();
```

```
                    break;
                case 2: cartInsert();
                    break;
                case 3: cartRemove();
                    break;
                case 4: cartModify();
                    break;
                case 5: cartDisplay();
                    break;
                case 6: checkOut(name, phno);
                    break;
                default:printf("Invalid choice\n");
            }
        }while(ch != 6);
    }
    else
        printf("Shop is closed. Please come back later!\n");
            struct item *t = cartList;
            cartList = cartList->next;
            free(t);
        }
    }
    else
        printf("THANK YOU! PLEASE VISIT AGAIN!\n");
```

A function customer() is created where the customer can perform all the list operations on the cart depending on their choices. It exits when the customer checks out.

```
    if(shopList != NULL)
```

```
{
        char name[20], phno[11];
        GET name, phno
        int ch;
        do
        {
                printf("\n1. See all items in the shop\n2. Add an item to item\n3.
Remove an item from item\n4. Modify quantity\n5. See your item\n6. Check Out\n");
                READ ch
                switch(ch)
                {
                        case 1: shopDisplay();
                                        break;
                        case 2: cartInsert();
                                        break;
                        case 3: cartRemove();
                                        break;
                        case 4: cartModify();
                                        break;
                        case 5: cartDisplay();
                                        break;
                        case 6: checkOut(name, phno);
                                        break;
                        default:printf("Invalid choice\n");
                }
        }while(ch != 6);
}
```

```
else
        //No item in shopList, hence shop is closed/empty
```

In main() function, we ask whether the user is admin or customer. If they are an admin, they gain admin access once they enter the correct password. They can then carry out functions of owner. If they are a customer, they can carry out the functions of a customer.

```
int ch;
char password[20] = "DSAProject";
do
{
        char pwd[20];
        printf("\nEnter 1 if you are the admin, 2 if you are a user. Enter 3 to exit.\n");
        READ ch
        switch(ch)
        {
                case 1: GET pwd
                                if(password == pwd)
                                        owner();
                                else
                                        printf("Incorrect Password!\n");
                                break;
                case 2: customer();
                                break;
                case 3: printf("Thank you\n");
                                break;
                default:printf("Invalid choice\n");
                                break;

        }
}while(ch != 3);
```

Chapter 5: SYSTEM ARCHITECTURE

In this context diagram, the information provided to and received from the 'Online Shopping' is identified. The arrows represent the information received or generated by the application. The closed boxes represent the set of sources and sinks of information. In the system, we can observe that the user interacts with the application through a graphical user interface. The inputs to the system are the Search and Filter criteria provided by the user and a new review written by the user. Also, the output is in the form of Repeater and grid views which present the users with list of Products available. The users can view complete specification, view Images and reviews by other users in our dummy website

In our C based Application we are limited to administrator feeding all the available stock to system and user can search for the product he wants by filtering all the available resources and adding the item to be shopped in the cart and he can modify his cart and checkout.

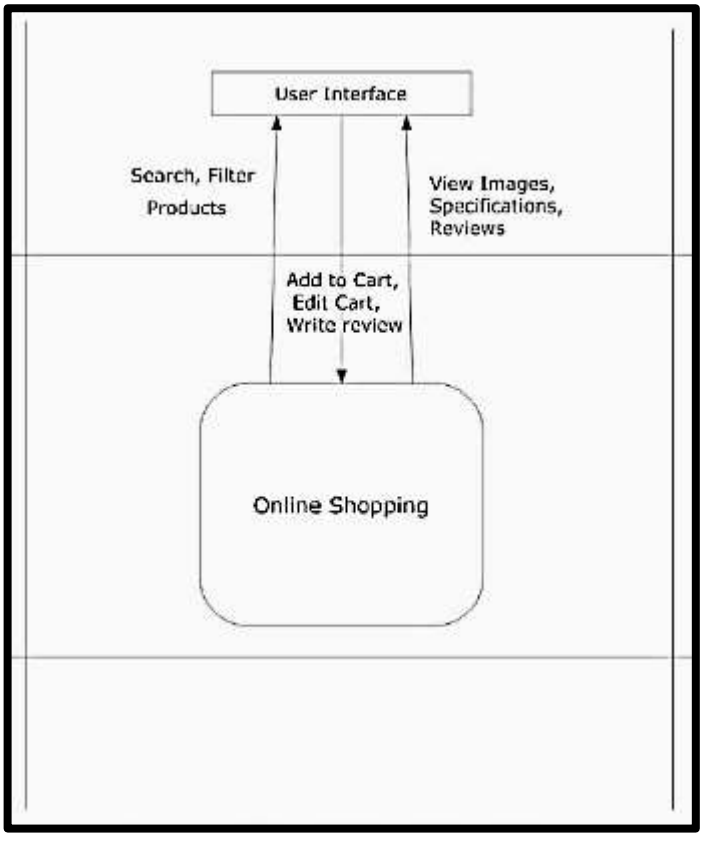


Fig5. 1: Architectural Context Diagram

System Architecture Analysis:

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is viewed as a whole, the inputs are identified and the system is subjected to close study to identify the problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

EXISTING SYSTEM:

The current system for shopping is to visit the Allmart shop manually and from the available product choose the item customer want and buying the item by payment of the price of the item .

7. It is less user-friendly.
8. User must go to shop and select products.
9. It is difficult to identify the required product.
10. Description of the product limited.
11. It is a time consuming process .
12. Not in reach of distant users.

PROPOSED SYSTEM:

In the proposed system customer need not go to the shop for buying the products. He can order the product he wishes to buy through the application. The shop owner will be admin of the system. Shop owner can appoint moderators who will help owner in managing the customers and product orders. The system also recommends a home delivery system for the purchased products.

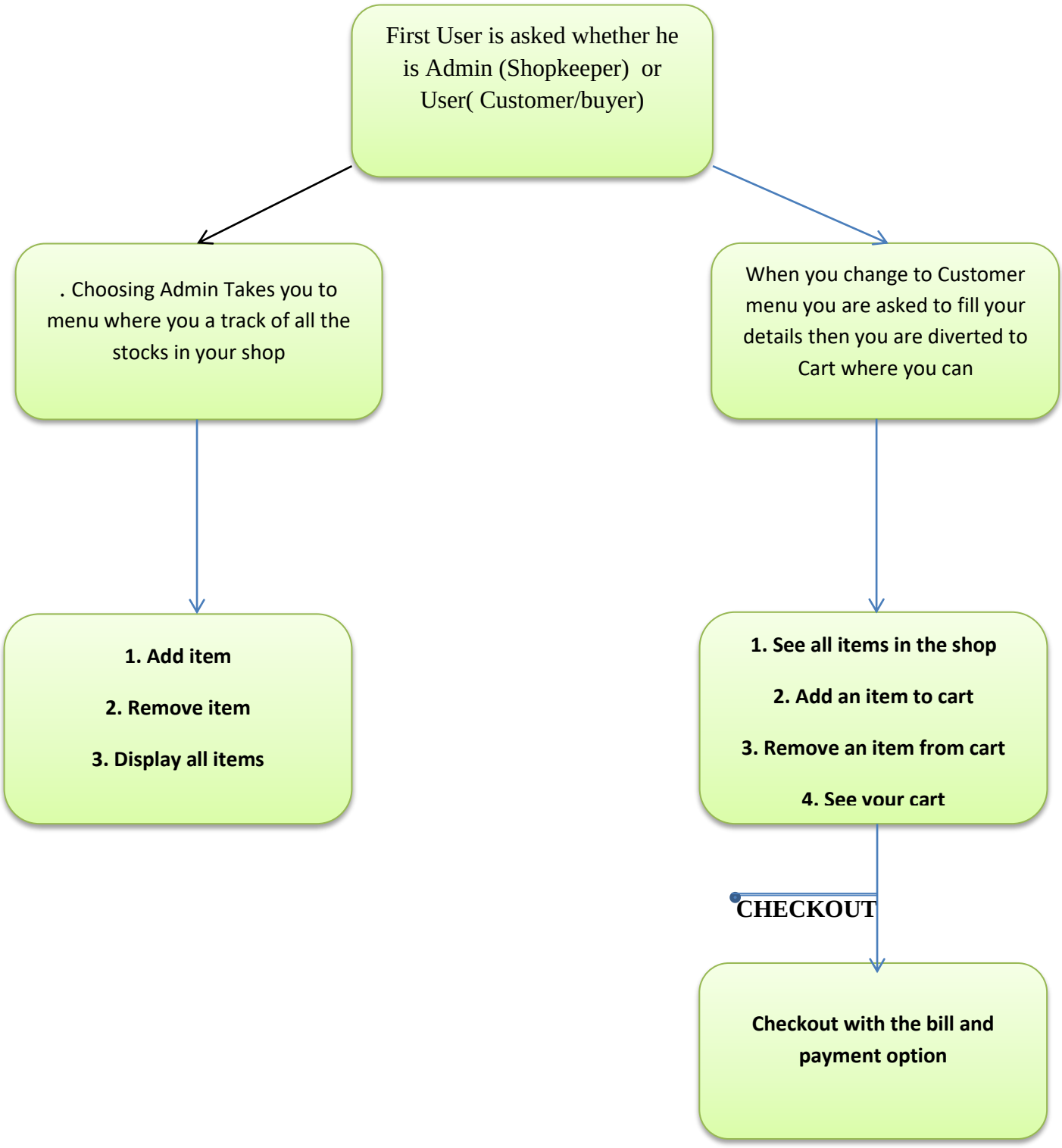


Fig5.2- BASIC ARCHITECTURE OF THE PROGRAM

SYSTEM REQUIREMENTS

1: Hardware Requirements

Number	Description
1	Intel core ,WIN xp/7/vista/8/10
2	320 MB RAM

2: Software Requirements

Number	Description
1	Windows XP –10
2	Codeblocks/DevC++/Turbo C++ Or any other c compiler

Chapter 6: MODULE DISCRIPTION

The system after careful analysis has been identified to be presented with the following modules and roles. The modules involved are:

- Administrator
- Moderators
- Users

1-Asministrator Module:

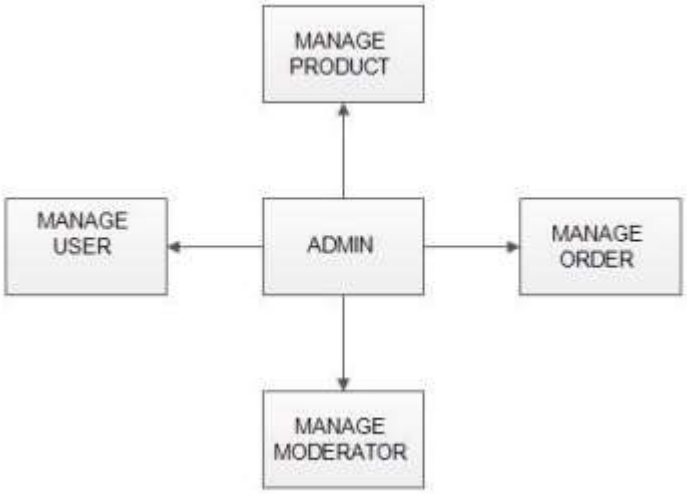


Fig: 6.1

The administrator is the super user of this application. Only admin have access into this admin page. Admin may be the owner of the shop. The administrator has all the information about all the users and about all products.

This module is divided into different sub-modules:

1. Manage Moderators
2. Manage Products
3. Manage User
4. Manage Orders

Mange Moderator:

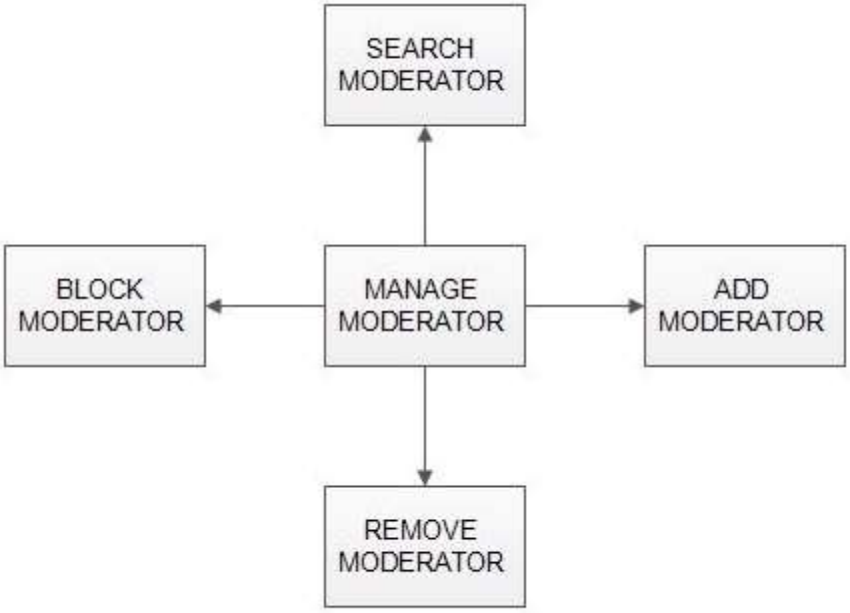


Fig: 6.1.1

Only admin is having the privilege to add a moderator. A moderator can be considered as a staff who manages the orders or owner of a group of products.

¬ Block moderator: Admin can restrict a moderator from managing the orders by blocking them. Admin can unblock a blocked user if needed.

¬ Remove Moderator: Admin has privilege to delete a moderator who was added.

¬ Search moderator: All existing moderators can be viewed by the administrator as a list. If there is number of moderators and admin need to find one of them, the admin can search for a moderator by name.

Manage products:

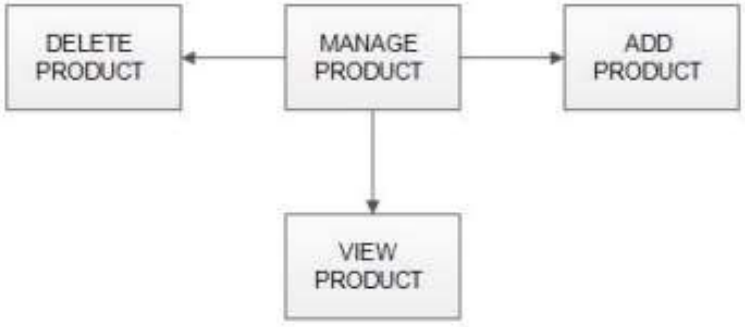


Fig: 6.1.2

¬ Add Products: The shopping cart project contains different kind of products. The products can be classified into different categories by name. Admin can add new products into the existing system with all its details including an image.

¬ Delete Products: Administrator can delete the products based on the stock of that particular product.

¬ Search products: Admin will have a list view of all the existing products. He can also search for a particular product by name.

Manage Users:

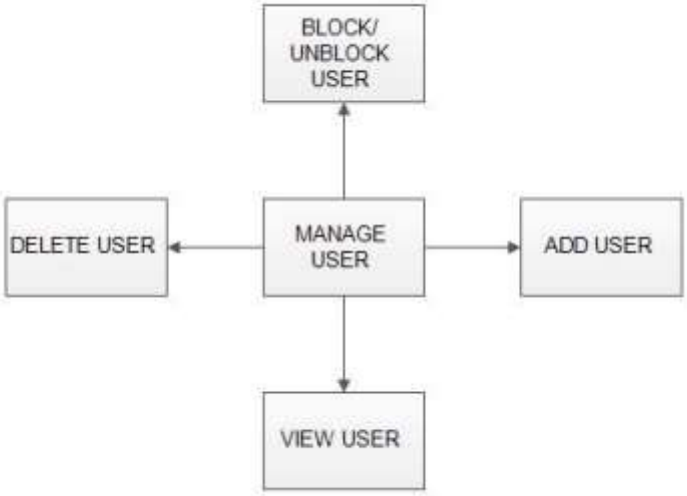


Fig 6.1.3

¬ View Users: The admin will have a list view of all the users registered in the system. Admin can view all the details of each user in the list except password.

¬ Add Users : Admin has privileges to add a user directly by providing the details

. ¬ Delete &Block Users : Administrator has a right to delete or block a user. The default status of a new user registered is set as blocked. The admin must accept the new user by unblocking him.

Manage Orders:

Fig: 6.1.4

¬ View Order : Administrator can view the Orders which is generated by the users. He can verify the details of the purchase

. ¬ Delete order: Admin can delete order from the orders list when the product is taken for delivery.

2.Moderators Module (Optional):

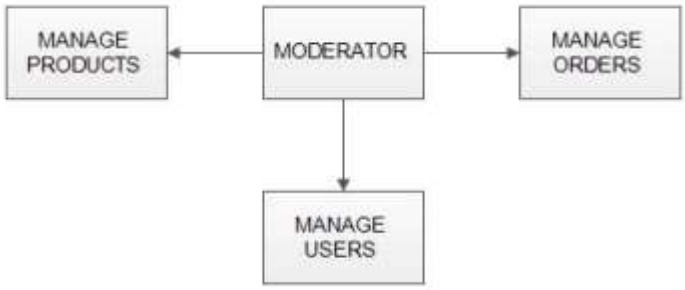


Fig: 6.2

A moderator is considered as a staff who can manage orders for the time being. As a future update moderator may give facility to add and manage his own products . Moderators can reduce the work load of admin. Now moderator has all the privilege an admin having except managing other moderators. He can add products and users. He can also check the orders and edit his profile

. ¬ Manage products

¬ Manage users

¬ Manage orders

3.Users Modules:

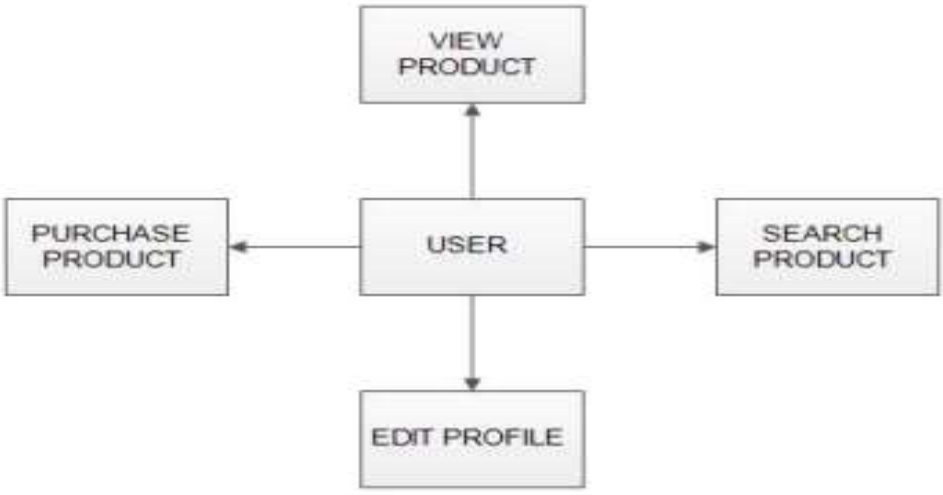


Fig: 6.3

¬ Registration: A new user will have to register in the system by providing essential details in order to view the products in the system. The admin must accept a new user by unblocking him

¬ View Products User can view the list of products based on their names after successful login. A detailed description of a particular product with product name, products details, product image, price can be viewed by users

. ¬ Search Product Users can search for a particular product in the list by name.

¬ Add to cart: The user can add the desired product into his cart by clicking add to cart option on the product.

¬ Submit Car/Checkoutt: After confirming the items in the cart the user can submit the cart by providing a delivery address. On successful submitting the cart will become empty

Chapter 7: OUTPUT

1: C code output

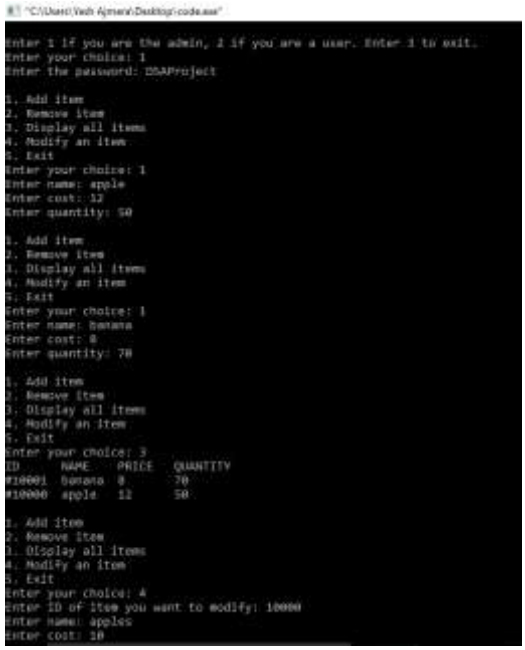


Fig: 7.1.1

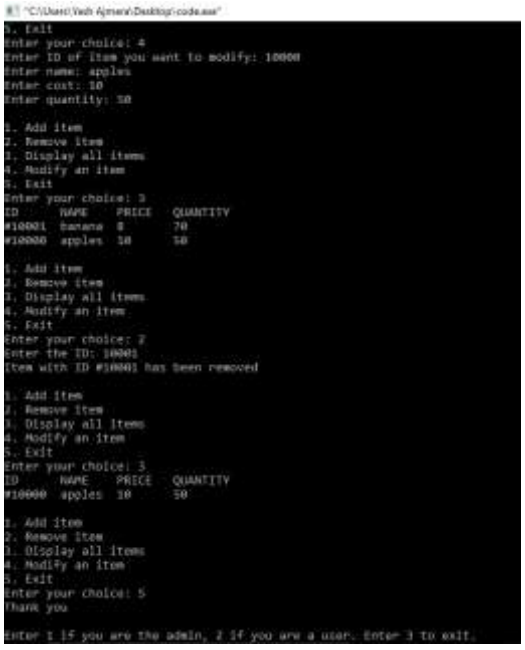
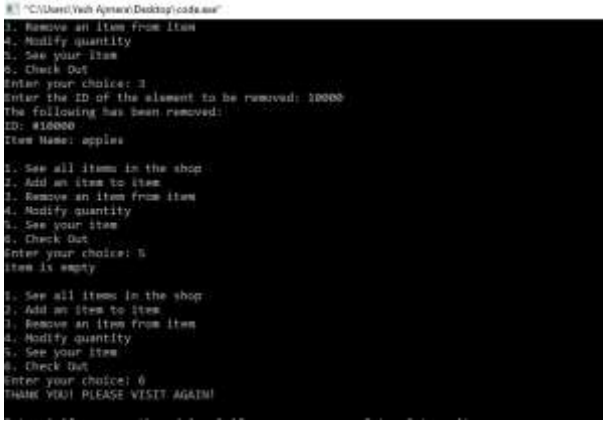


Fig 7.1.2

Fig 7.1.3

Fig7.1.4



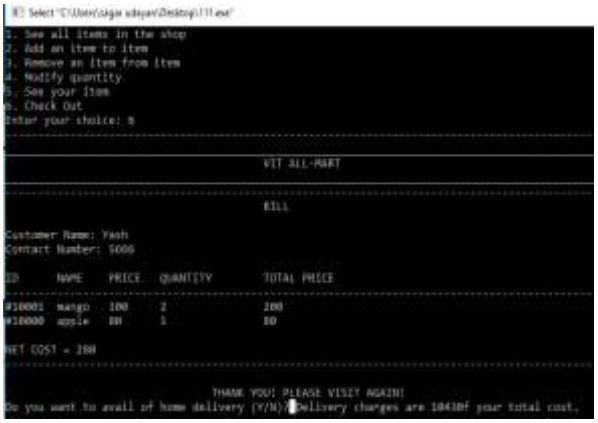


Fig 7.1.4 **Fig: 7.1.5**

2.Website Description(Screen shots)

1.Home Page:

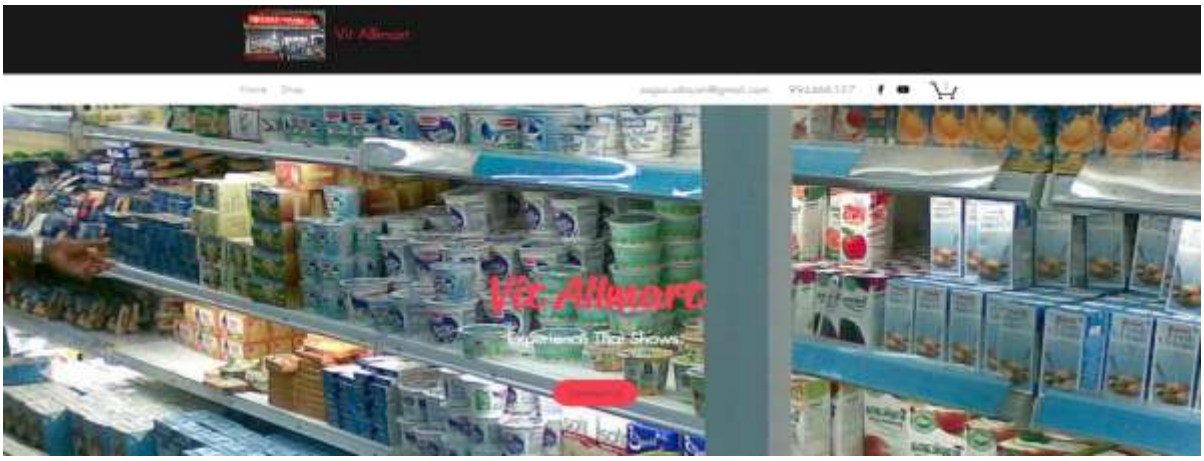


Fig 7.2.1

Fig 7.2.2

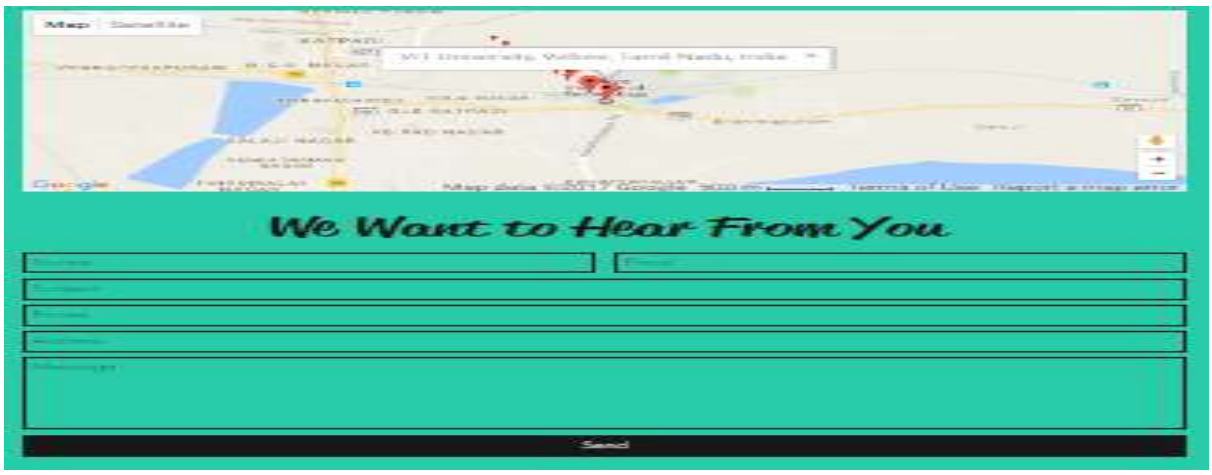


Fig 7.2.3

2.Shopping page

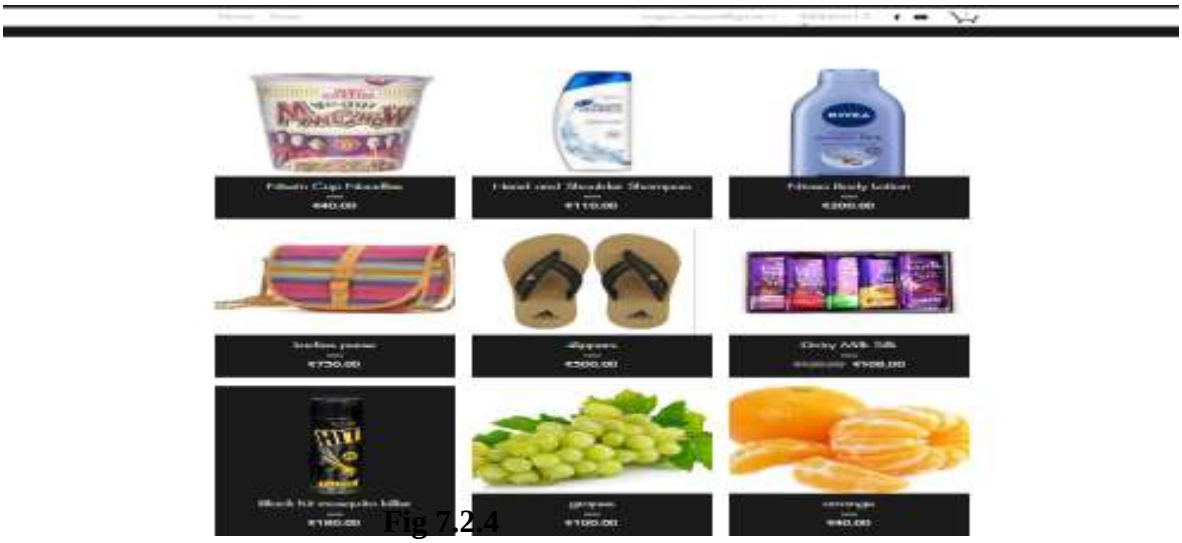
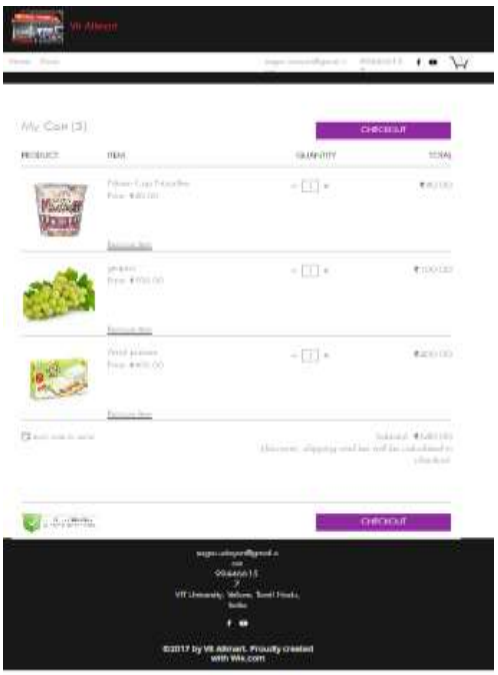


Fig 7.2.4

3.Shopping Kart:

Fig 7.2.5 & 7.2.6



Chapter 8: RESULT:

1. We have successfully developed a program for a shopping general store.
2. We have got the required inputs and printed the invoice with the details printed on it.
3. The outcomes of the program are displayed on the computer screen.
4. More number of people purchased when online shopping was installed than physically

Chapter 9: CONCLUSION

:

1. We have concluded that we can make a system of online shopping
2. Online shopping is far better than shopping physically.
3. Cheats and thief can be minimized with the help of online shopping
4. Online shop gives the platform for grow the business among more number of people

Chapter 10: REFFERANCES

1- VIT Library
2- W3 School
3- Wikipedia for literature Survey