PROJECT REPORT

SHOPPING CART

21CSC201J – DATA STRUCTURES & ALGORITHM

Submitted By:

RISHIRAM B- RA2211026010553

AMAL RASHID – RA2211026010541

SURYA SATHYANARAYANAN - RA2211026010562

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this B.Tech project report titled **"SHOPPING CART USING DATA STRUCTURES**" is the bonafide work of Rishiram B [RA2211026010553] , Amal Rashid [RA2211026010541] and Surya Sathyanarayanan[RA2211026010562] who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

SIGNATURE

Dr. N. Meenakshi
Assistant Professor
Department of Computational Intelligence
SRMIST
Katankulathur

TABLE OF CONTENTS

1.a) PROBLEM DEFINITION

Online shopping carts are an essential part of e-commerce. They allow customers to browse and select items, add them to their cart, and then checkout and pay. One way to implement a shopping cart is to use a linked list.

A linked list is a data structure that consists of a sequence of nodes, where each node contains a value and a pointer to the next node in the list.

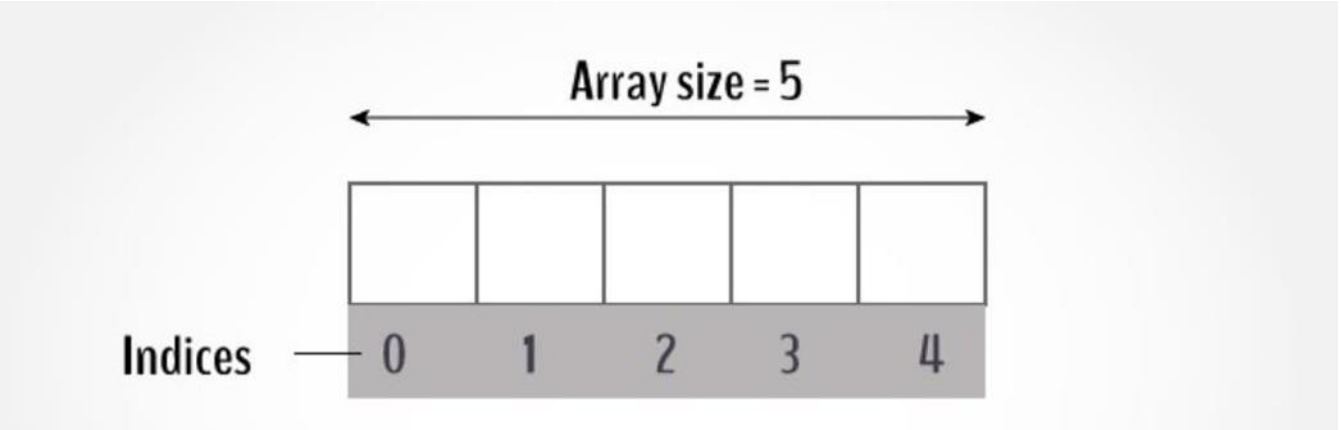
Linked lists are well-suited for implementing shopping carts because they are dynamic and can grow and shrink as needed. This is important because the number of items in a shopping cart can vary greatly from customer to customer.

1.b) PROBLEM EXPLANATION

1. Item Representation: Each item in the shopping cart has attributes such as name, price, quantity, and possibly other details like size or color. These attributes need to be stored for each item.
2. Add Item: Customers should be able to add items to their shopping cart. The system should support adding multiple instances of the same item (e.g., two identical t-shirts).
3. Remove Item: Customers should be able to remove items from their cart. When an item is removed, it should be completely taken out of the cart.
4. View Cart: Customers should be able to view the items in their shopping cart, including the item details and the total cost of the items in the cart.
5. Calculate Total: The system should calculate the total cost of all items in the cart, taking into account the quantity of each item.
6. Empty Cart: Customers should have the option to clear their shopping cart, removing all items.
7. Checkout: Once customers are ready to make a purchase, they can proceed to checkout, where they provide shipping and payment information. This step is not covered in the problem statement but is part of the larger e-commerce system.

DATA STRUCTURES

2.a) Using Arrays:



Advantages of using Arrays:

1. Sequential Storage: Arrays provide sequential storage, making them a natural choice for storing a collection of items in a shopping cart. Each element of the array can represent an item, making it easy to add, remove, or access items.

2. Simple and Efficient: Arrays are simple to use and offer fast access to items. You can directly access an item by its index, which is often the product's unique identifier. This simplicity and efficiency are crucial in a shopping cart, where customers frequently add, remove, or modify items.

3. Fixed or Dynamic Size: Depending on your cart's design, you can use fixed-size or dynamic arrays. Fixed-size arrays work well if the cart has a maximum limit on the number of items. Dynamic arrays (e.g., ArrayList in Java) can resize themselves as needed, accommodating an arbitrary number of items.

4. Easy Iteration: You can easily iterate through the items in the cart using a simple for loop. This is helpful when you need to calculate the total price, apply discounts, or generate a list of items for display.

Disadvantages of using Arrays:

1. Fixed Size: Arrays have a fixed size, which can be problematic if your shopping cart needs to accommodate a variable number of items. In a fixed-size array, you must allocate a maximum capacity in advance, which can be inefficient if the cart size fluctuates widely.

2. Inefficient Inserts and Deletions: Inserting or deleting items in the middle of an array can be inefficient. If you frequently add or remove items from the middle of the cart, you may experience performance issues, as shifting elements to accommodate changes can be time-consuming.

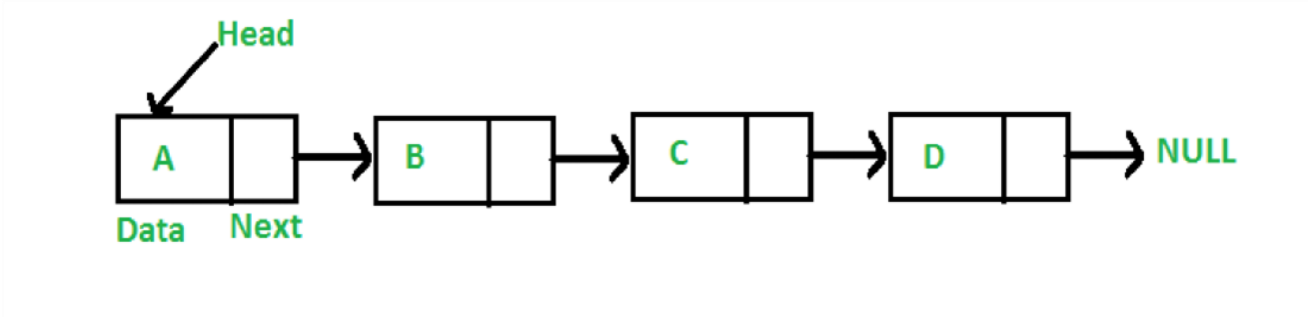
3. Wasted Memory: In a dynamic array, memory may be wasted because you allocate space for a maximum number of items even if the cart typically contains far fewer items. This can be inefficient, especially for large e-commerce platforms with many shopping carts.

4. No Built-in Key-Value Relationship: Arrays are essentially ordered lists of items, and there's no built-in key-value relationship. If you need to store additional information or metadata about items in the cart (e.g., quantity, price, product details), you may need to manage separate arrays or data structures, which can lead to code complexity.

USE CASES:

Arrays are well-suited for scenarios where the dataset size is known in advance and remains relatively static. Examples include maintaining a fixed collection of documents in a reference library or a dictionary.

2.b) Using linked list



Advantages of using linked list:

1. Dynamic Size: Linked lists can dynamically adjust in size, making them suitable for shopping carts that need to accommodate a variable number of items. As customers add or remove items, the linked list can expand or shrink as needed.

2. Efficient Inserts and Deletions: Linked lists are highly efficient for inserting and deleting items, especially when those operations are frequently performed in the middle of the list. There's no need to shift elements as you would with arrays, reducing the time complexity of these operations.

3. Memory Efficiency: Linked lists are memory-efficient, as they allocate memory for each item individually. This means memory is allocated only for the items present in the cart, minimizing wasted memory.

Disadvantages of using linked list

1. Lack of Random Access: Linked lists do not provide direct random access to items by index, unlike arrays. To access a specific item in a linked list, you typically need to traverse the list from the beginning, which can be less efficient for large carts.

2. Increased Memory Overhead: Each item in a linked list is associated with extra memory overhead to store the reference to the next item, which can lead to higher memory consumption compared to some other data structures, especially for small items.

3. Search Complexity: While searching is efficient for finding specific items, linked lists are not well-suited for operations like finding the maximum or minimum item based on a specific attribute (e.g., finding the most expensive item in the cart).

USE CASE:

Linked lists are well-suited for scenarios where the dataset is dynamic, and the focus is on efficient insertions, deletions, and updates. Examples include maintaining a document collection in a collaborative document management system or a content version history.

3.PROGRAM FOR SHOPPING CART

3.a) IMPLEMENTED USING ARRAY

```c
#include <stdio.h>
#include <string.h>

#define MAX_PRODUCTS 100
#define MAX_CART_ITEMS 100

struct Product {
    int id;
    char name[20];
    int price;
    int qty;
};

struct CartItem {
    int id;
    int qty;
};

struct Product products[MAX_PRODUCTS];
struct CartItem cart[MAX_CART_ITEMS];

int productCount = 0;
int cartItemCount = 0;

```

```
    while (1) {
        printf("=========================================================\n\n");
        printf("\t\t WELCOME TO SHOPPING CART!!\n\n");
        printf("=========================================================\n\n");
        printf("1. Manage Product\n\n");
        printf("2. Purchase Product\n\n");
        printf("3. Generate Bill\n\n");
        printf("0. Exit\n\n");
        printf("=========================================================\n\n");
        printf("\nPlease enter your Choice: ");
        scanf("%d", &ch);

        switch (ch) {
            case 1: {
                manageProduct();
                break;
            }
            case 2: {
                // Implement the purchaseProduct function or remove this reference
                printf("Function purchaseProduct is not implemented.\n");
                break;
            }
            case 3: {
                // Implement the generateBill function or remove this reference
                printf("Function generateBill is not implemented.\n");
                break;
            }
            case 0:
                return 0;
            default:
                printf("Valid choice not entered!\n");
        }
    }
    return 0;
}

void manageProduct() {
    int ch;

    while (1) {
        printf("=========================================================\n\n");
        printf("\t\t WELCOME MANAGER!!\n\n");
        printf("=========================================================\n\n");
        printf("1. Add New Product\n\n");
        printf("2. Display All Products\n\n");
        printf("0. Back\n\n");
        printf("=========================================================\n\n");
        printf("\nPlease enter your Choice: ");
        scanf("%d", &ch);
        switch (ch) {
```

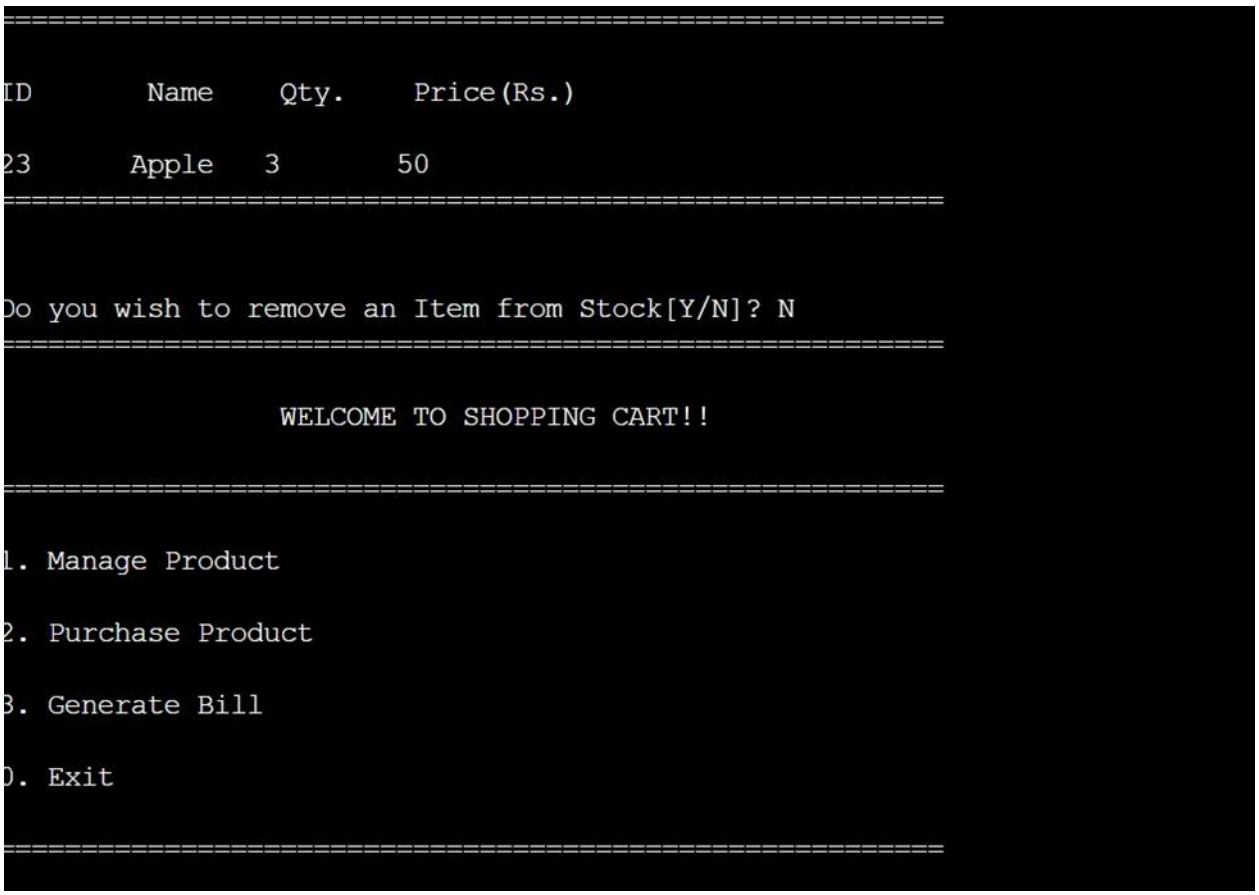
```
        switch (ch) {
            case 1: {
                addProduct();
                break;
            }
            case 2: {
                displayAllProduct();
                printf("\nDo you wish to remove an Item from Stock[Y/N]? ");
                char ch2;
                scanf(" %c", &ch2);
                if (ch2 == 'Y' || ch2 == 'y') {
                    removeProduct();
                }
                if (ch2 == 'N' || ch2 == 'n') {
                    return;
                }
                break;
            }
            case 0: {
                return;
            }
            default:
                printf("Valid choice not entered!\n");
        }
    }
}
```

OUTPUT

```
============================================================

                WELCOME TO SHOPPING CART!!

============================================================

1. Manage Product

2. Purchase Product

3. Generate Bill

0. Exit

============================================================


Please enter your Choice: 1
============================================================

                WELCOME MANAGER!!

============================================================
```

```
1. Add New Product

2. Display All Products

0. Back

=========================================================


Please enter your Choice: 1
=========================================================

                ADD PRODUCTS!!

=========================================================


Enter the ID of the product: 23

Enter the name of the product: Apple

Enter the price of the product: 50

Enter the quantity of the product: 3
```

```
Product Added Successfully!!
Do you want to add another product[Y/N]? N
=========================================================

                WELCOME MANAGER!!

=========================================================

1. Add New Product

2. Display All Products

0. Back

=========================================================


Please enter your Choice: 2
=========================================================

                Product Details

=========================================================
```


ID Name Qty. Price(Rs.)
23 Apple 3 50
Do you wish to remove an Item from Stock[Y/N]? N
WELCOME TO SHOPPING CART!!
1. Manage Product
2. Purchase Product
3. Generate Bill
0. Exit

3.b) IMPLEMENTED USING LINKED LIST

```c
#include <stdio.h>
#include <malloc.h>
#include <conio.h>
#include <string.h>
#define ISEMPTY printf("\nEMPTY LIST:");

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

struct node2
{
     int id;
     int qty;
     struct node2 *next2;
};

typedef struct node snode;
typedef struct node2 snode2;
snode *newnode, *ptr, *prev,*temp;
snode *first = NULL, *last = NULL;
snode2 *newnode2, *ptr2, *prev2,*temp2;
void main()
{
   int ch;

   while (1)
   {
       system("cls");
       printf("============================================================\n\n");
       printf("\t\t WELCOME TO SHOPPING CART!!\n\n");
       printf("============================================================\n\n");
       printf("1. Manage Product\n\n");
       printf("2. Purchase Product\n\n");
       printf("3. Generate Bill\n\n");
       printf("0. Exit\n\n");
       printf("============================================================\n\n");
       printf("\nPlease enter your Choice: ");
       scanf("%d",&ch);

       switch (ch)
       {
           case 1: {
               manageProduct();
               break;
           }
           case 2:{
               purchaseProduct();
               break;
```

```
while (1)
{
    system("cls");
    printf("============================================================\n\n");
    printf("\t\t WELCOME MANAGER!!\n\n");
    printf("============================================================\n\n");
    printf("1. Add New Product\n\n");
    printf("2. Display All Products\n\n");
    printf("0. Back\n\n");
    printf("============================================================\n\n");
    printf("\nPlease enter your Choice: ");
    scanf("%d",&ch);
    switch (ch)
    {
        case 1: {
            addProduct();
            break;
        }
        case 2:{
            displayAllProduct();
            printf("\n Do you wish to remove an Item from Stock[Y/N]? ");
            scanf("%s",&ch2);
            if (ch2 == 'Y'||ch2 == 'y')
            {
                removeProduct();
                getch();
```

```
void addProduct()
{
    system("cls");
    int id, price, qty,pos,cnt=0,i;
    printf("============================================================\n\n");
    printf("\t\t ADD PRODUCTS!!\n\n");
    printf("============================================================\n\n");
    char name[20],ch;
    printf("\nEnter the ID of the product: ");
    scanf("%d", &id);
    for (ptr = first;ptr != NULL;ptr = ptr->next)
    {
        if (ptr->id==id)
        {
            printf("Product ID already in use.");
            getch();
            return manageProduct();
        }

    }
    printf("\nEnter the name of the product: ");
    scanf("%s", name);
    printf("\nEnter the price of the product: ");
    scanf("%d", &price);
    printf("\nEnter the quantity of the product: ");
    scanf("%d", &qty);
```

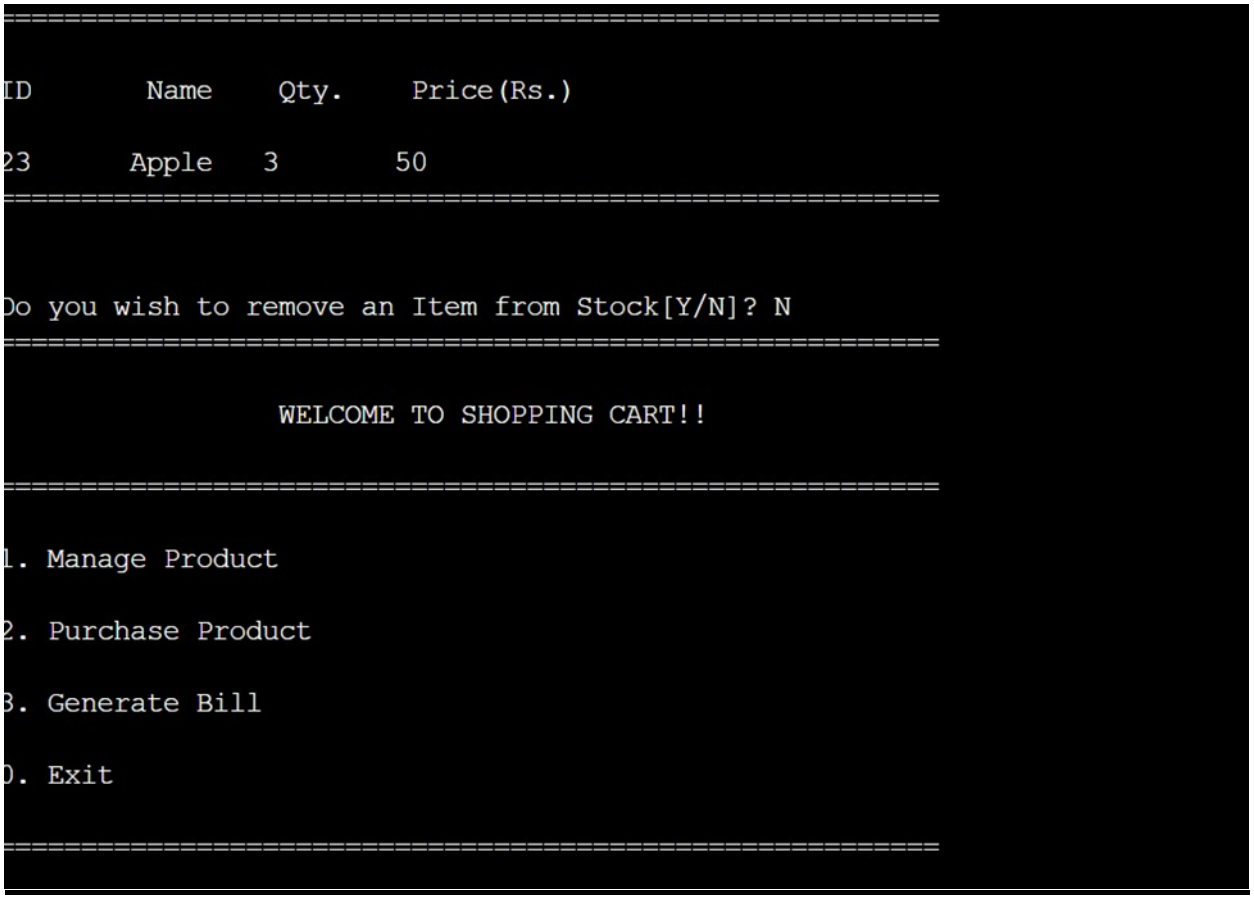
```
        switch (ch) {
            case 1: {
                addProduct();
                break;
            }
            case 2: {
                displayAllProduct();
                printf("\nDo you wish to remove an Item from Stock[Y/N]? ");
                char ch2;
                scanf(" %c", &ch2);
                if (ch2 == 'Y' || ch2 == 'y') {
                    removeProduct();
                }
                if (ch2 == 'N' || ch2 == 'n') {
                    return;
                }
                break;
            }
            case 0: {
                return;
            }
            default:
                printf("Valid choice not entered!\n");
        }
    }
}
```

OUTPUT

```
============================================================

                WELCOME TO SHOPPING CART!!

============================================================

1. Manage Product

2. Purchase Product

3. Generate Bill

0. Exit

============================================================


Please enter your Choice: 1
============================================================

                WELCOME MANAGER!!

============================================================
```

```
1. Add New Product

2. Display All Products

0. Back

=========================================================


Please enter your Choice: 1
=========================================================

                ADD PRODUCTS!!

=========================================================


Enter the ID of the product: 23

Enter the name of the product: Apple

Enter the price of the product: 50

Enter the quantity of the product: 3
```

```
Product Added Successfully!!
Do you want to add another product[Y/N]? N
=========================================================

                WELCOME MANAGER!!

=========================================================

1. Add New Product

2. Display All Products

0. Back

=========================================================


Please enter your Choice: 2
=========================================================

                Product Details

=========================================================
```


ID Name Qty. Price(Rs.)
23 Apple 3 50
Do you wish to remove an Item from Stock[Y/N]? N
WELCOME TO SHOPPING CART!!
1. Manage Product
2. Purchase Product
3. Generate Bill
0. Exit

4.COMPARISON BETWEEN USE OF ARRAYS AND LINKED LIST FOR SHOPPING CART

Cart Size: For small to medium-sized shopping carts with a relatively stable number of items, arrays may be a suitable and memory-efficient choice. For carts that can grow or shrink dynamically, linked lists are more appropriate.

Insertion and Deletion Frequency: If your shopping cart involves frequent insertions and deletions of items, linked lists offer better performance. For carts with infrequent changes, arrays may be more efficient.

Complex Data Structures: If your cart requires custom data structures for items, linked lists allow you to accommodate additional metadata and complex structures.

5.CONCLUSION

Linked lists are a good choice for implementing shopping carts because they are dynamic and can grow and shrink as needed.

They are also relatively easy to implement in C.

The project along with all attributes implemented in it is capable and has potential to penetrate in market and get its usage in day to day work space and can be scaled.