VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**JNANA SANGAMA,BELAGAVI – 590018**

**KARNATAKA**

**Assignment Report**

**On**

# "SHOPPING CART"

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**

**FOR THE DATA STRUCTURES AND APPLICATIONS (BCS304) COURSE OF III SEMESTER**

Submitted by


**T G DARSHAN**

**USN:1CG23CS109**

**Guide:**
**Prof. Asif Ulla Khan** .M. Tech.
Asst. Prof., Dept. of CSE
CIT, Gubbi.

**HOD:**
**Dr. Shantala C P** PhD.,
Head, Dept. of CSE
CIT, Gubbi.


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**


**Channabasaveshwara Institute of Technology**

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(**NAAC Accredited & ISO 9001:2015 Certified Institution)**

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka


**2024-25**

# Rubric – B.E. Mini-Project [BCS304]

| Course outcome | Rubric/Level | Excellent (91-100%) | Good (81-90%) | Average (61-80%) | Moderate (40-60%) | Score |
|---|---|---|---|---|---|---|
| CO1 | Identification of project proposal (05 Marks) | | | | | |
| CO2 | Design and Implementation (10 Marks) | | | | | |
| CO3 | Presentation skill (05 Marks) | | | | | |
| CO4 | Report (05 Marks) | | | | | |
| Total | | | | | | |

## Course outcome:

**CO 1: Identification of project proposal which is relevant to subject of engineering.**

**CO 2: Design and implement proposed project methodology.**

**CO 3: Effective communication skill to assimilate their project work.**

**CO 4: Understanding overall project progress and performance.**

**Student Signature** **Faculty signature**

# SHOPPING CART SYSTEM USING LINKEDLIST IN C

## ABSTRACT

A shopping cart system is an essential feature of any ecomerce platform, enabling users to select, update, and manage items before proceeding to checkout.

The implementation of such a system requires efficient data structures to handle dynamic operations.

This project utilizes a linked list to design and manage the shopping cart, providing flexibility and dynamic memory allocation.

The linked list's structure allows easy addition, deletion, and traversal of items in the cart, making it suitable for handling operations like modifying quantities, removing items, and displaying the cart's contents.

The objective is to demonstrate how linked lists can optimize the functionality of a shopping cart system, ensuring a seamless and user-friendly experience.

### Explanation:

### The Abstract summarizes:

1. **Purpose :** Focus on implementing a shopping cart system using a linked list.
2. **Choice of Data Structure :** A linked list is chosen for its dynamic nature and efficient handling of insertions and deletions.
3. **Advantages :** Linked lists are well-suited for applications requiring frequent updates and traversal.
4. **Goal :** To create an efficient and user-friendly shopping cart system.

# CHAPTER 1

# INTRODUCTION

In the realm of e-commerce, the shopping cart acts as a dynamic repository for users to add, update, or remove items before checkout. Implementing a shopping cart requires careful consideration of data management to ensure smooth user interactions.a linked list is an ideal choice for this application due to its ability to dynamically allocate memory and manage elements efficiently.

A linked list consists of nodes, where each node contains :

1. The data related to an item (e.g., product name, price, quantity, and ID).
2. A pointer to the next node in the list.

This structure offers several advantages :

- **Dynamic Size :** The cart's size can grow or shrink as items are added or removed, without predefining a fixed size.
- **Efficient Insertions and Deletions :** Adding or removing items involves updating pointers, making these operations efficient.
- **Traversal :** The cart's contents can be easily traversed for display or calculations.

By using a linked list, the shopping cart system can efficiently handle :

1. Addition of items at the beginning, end, or any specific position in the cart.
2. Deletion of items based on user actions or constraints (e.g., inventory shortages).
3. Dynamic updates, such as changing item quantities or applying discounts.

The goal of this project is to explore the application of linked lists in building a scalable and user-friendly shopping cart system, ensuring real-time updates and seamless performance.

# CHAPTER 2

# PROBLEM STATEMENT

The problem this program addresses is to create a shopping cart system for an e-commerce platform where users can manage items before proceeding to check-out. The system must support dynamic operations like adding items, updating their quantities, removing items, displaying the cart's contents, and calculating the total bill. These operations need to be handled efficiently, and for that, a **linked list** data structure is used.

The **linked list** allows for dynamic memory allocation, meaning items can be added or removed without size limitations. Each item in the cart is represented by a node containing its name, quantity, price, and a pointer to the next item. This dynamic nature of the linked list makes it well-suited for handling frequent changes in the cart.

The key challenges include ensuring **data integrity**, such as preventing duplicate items in the cart and updating the quantity of existing items rather than adding new ones. Additionally, the program must allow for efficient **real-time visualization** of the cart and **total bill calculation**. This requires traversing the linked list to display the items and compute the subtotal for each item based on quantity and price.

In summary, the program is designed to manage a dynamic shopping cart using a linked list, addressing the challenges of flexibility, data integrity, and efficiency in real-time operations.

# CHAPTER 3

# IMPLEMENTATION

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

// Define a structure for each item in the cart
typedef struct Item {
    char name[50];
    int quantity;
    float price;
    struct Item* next;
} Item;
// Function prototypes
Item* addItem(Item* head, char name[], int quantity, float price);
Item* removeItem(Item* head, char name[]);
void updateQuantity(Item* head, char name[], int quantity);
void displayCart(Item* head);
void generateBill(Item* head);
int main() {
    Item* cart = NULL;
    int choice;
    char name[50];
    int quantity;
    float price;
    while (1) {
        printf("\n--- Shopping Cart Menu ---\n");
```

```
printf("1. Add Item\n");
printf("2. Remove Item\n");
printf("3. Update Quantity\n");
printf("4. Display Cart\n");
printf("5. Generate Bill\n");
printf("6. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
    printf("Enter item name: ");
    scanf("%s", name);
    printf("Enter quantity: ");
    scanf("%d", &quantity);
    printf("Enter price: ");
    scanf("%f", &price);
    cart = addItem(cart, name, quantity, price);
    break;
case 2:
    printf("Enter the name of the item to remove: ");
    scanf("%s", name);
    cart = removeItem(cart, name);
    break;
case 3:
    printf("Enter the name of the item to update: ");
    scanf("%s", name);
    printf("Enter new quantity: ");
    scanf("%d", &quantity);
    updateQuantity(cart, name, quantity);
```

```
            break;
        case 4:
            displayCart(cart);
            break;
        case 5:
            generateBill(cart);
            break;
        case 6:
            printf("Exiting...\n");
            return 0;
        default:
            printf("Invalid choice! Please try again.\n");
        }
    }
}
// Function to add an item to the cart
    Item* addItem(Item* head, char name[], int quantity, float price) {
    Item* newItem = (Item*)malloc(sizeof(Item));
    strcpy(newItem->name, name);
    newItem->quantity = quantity;
    newItem->price = price;
    newItem->next = NULL;
    if (head == NULL) {
        return newItem;
    }
    Item* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
```

```
    temp->next = newItem;
    printf("Item added to cart successfully.\n");
    return head;
}
// Function to remove an item from the cart
 Item* removeItem(Item* head, char name[]) {
    if (head == NULL) {
        printf("Cart is empty. No items to remove.\n");
        return NULL;
    }
    Item* temp = head;
    Item* prev = NULL;
    while (temp != NULL && strcmp(temp->name, name) != 0) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Item not found in cart.\n");
        return head;
    }
    if (prev == NULL) {
        head = temp->next;
    } else {
        prev->next = temp->next;
    }
    free(temp);
    printf("Item removed from cart successfully.\n");
    return head;
}
```

```
// Function to update the quantity of an item in the cart
void updateQuantity(Item* head, char name[], int quantity) {
    Item* temp = head;
    while (temp != NULL) {
        if (strcmp(temp->name, name) == 0) {
            temp->quantity = quantity;
            printf("Item quantity updated successfully.\n");
            return;
        }
        temp = temp->next;
    }
    printf("Item not found in cart.\n");
}
// Function to display the items in the cart
void displayCart(Item* head) {
    if (head == NULL) {
        printf("Cart is empty.\n");
        return;
    }
    printf("\n--- Shopping Cart Contents ---\n");
    printf("%-20s %-10s %-10s\n", "Name", "Quantity", "Price");
    printf("-----------------------------------\n");
    Item* temp = head;
    while (temp != NULL) {
        printf("%-20s %-10d $%-10.2f\n", temp->name, temp->quantity, temp-
>price);
        temp = temp->next;
    }
}
```

```
// Function to generate the total bill
void generateBill(Item* head) {
    if (head == NULL) {
        printf("Cart is empty. No bill to generate.\n");
        return;
    }
    float total = 0;
    printf("\n--- Bill ---\n");
    printf("%-20s %-10s %-10s %-10s\n", "Name", "Quantity", "Price", "Subto-
tal");
    printf("-----------------------------------------------------\n");
    Item* temp = head;
    while (temp != NULL) {
        float subtotal = temp->quantity * temp->price;
        total += subtotal;
        printf("%-20s %-10d $%-10.2f $%-10.2f\n", temp->name, temp->quantity,
temp->price, subtotal);
        temp = temp->next;
    }
    printf("-----------------------------------------------------\n");
    printf("Total: $%.2f\n", total);
}
```

# CHAPTER 4

## RESULT/SCREEN SHOT

```
--- Shopping Cart Menu ---
1. Add Item
2. Remove Item
3. Update Quantity
4. Display Cart
5. Generate Bill
6. Exit
Enter your choice: 1
Enter item name: Laptop
Enter quantity: 1
Enter price: 80000
Item added to cart successfully.

--- Shopping Cart Menu ---
1. Add Item
2. Remove Item
3. Update Quantity
4. Display Cart
5. Generate Bill
6. Exit
Enter your choice: 1
Enter item name: mouse
Enter quantity: 1
Enter price: 5000
Item added to cart successfully.
```

```
--- Shopping Cart Menu ---
1. Add Item
2. Remove Item
3. Update Quantity
4. Display Cart
5. Generate Bill
6. Exit
Enter your choice: 4

--- Shopping Cart Contents ---
Name                 Quantity    Price
-------------------------------------
Laptop               1           80000.00
mouse                1           5000.00

--- Shopping Cart Menu ---
1. Add Item
2. Remove Item
3. Update Quantity
4. Display Cart
5. Generate Bill
6. Exit
Enter your choice: 5

--- Bill ---
Name                 Quantity    Price       Subto-tal
-----------------------------------------------------
Laptop               1           80000.00    80000.00
mouse                1           5000.00     5000.00
-----------------------------------------------------
Total: 85000.00
```

# CHAPTER 4

# CONCLUSION

The above program simulates a basic shopping cart system using a linked list. It allows users to :

1. Add items to the cart with a name, quantity, and price.
2. Remove items from the cart by specifying the item name.
3. Update the quantity of an existing item.
4. Display the contents of the cart.
5. Generate a bill showing the total cost based on the quantity and price of each item.

The program uses a simple menu-driven interface to interact with the user, offering a good example of linked list manipulation in C for managing dynamic data like a shopping cart.

# REFERENCES


www.google.com

www.chatgpt.com

https://github.com/Darshantg56/shoppingcart.git