Project Report

shopping management system

**Submitted by:**  
Tahira Hafeez (SP24-BCS-123)  
Sabiha Usman (SP24-BCS-109)  
Haleema Jawad (SP24-BCS-046)

**Contents**

[Introduction And Features Of The Project 3](#_Toc200795830)

[Key Features 3](#_Toc200795831)

[Modules: 3](#_Toc200795832)

[1. User Management Module 3](#_Toc200795833)

[2. Admin Functionalities 3](#_Toc200795834)

[3. Product Management Module (AVL Tree Based) 4](#_Toc200795835)

[4. Cart Management Module 4](#_Toc200795836)

[5. Wishlist Module 4](#_Toc200795837)

[6. File Handling Module 4](#_Toc200795838)

[7. Search Functionalities 4](#_Toc200795839)

[8. Stack Operations (Undo Feature) 4](#_Toc200795840)

[Data Structures Used 5](#_Toc200795841)

[AVL Tree (Self-Balancing Binary Search Tree) 5](#_Toc200795842)

[Singly Linked List 5](#_Toc200795843)

[Stack (Linked List Based) 6](#_Toc200795844)

[File I/O (Text File Handling) 6](#_Toc200795845)

[Objectives 6](#_Toc200795846)

[System Overview 7](#_Toc200795847)

[User Registration and Login 7](#_Toc200795848)

[Product Browsing 7](#_Toc200795849)

[Product Search 7](#_Toc200795850)

[Product Addition (Admin) 7](#_Toc200795851)

[Cart Management 7](#_Toc200795852)

[View Cart 8](#_Toc200795853)

[Checkout Process 8](#_Toc200795854)

[Purchase History 8](#_Toc200795855)

[Exit 8](#_Toc200795856)

[Functions: 8](#_Toc200795857)

[USER 8](#_Toc200795858)

[🔹 loginUser() 8](#_Toc200795859)

[🔹 registerUser() 9](#_Toc200795860)

[🔹 saveUsersToFile() 9](#_Toc200795861)

[🔹 loadUsersFromFile() 9](#_Toc200795862)

[🔹 deleteUser() 10](#_Toc200795863)

[PRODUCT 10](#_Toc200795864)

[🔹 insertion() (of product) 10](#_Toc200795865)

[🔹 loadFromFile() 10](#_Toc200795866)

[🔹 deleteProductById() 11](#_Toc200795867)

[CART 11](#_Toc200795868)

[🔹 addToCart() 11](#_Toc200795869)

[🔹 deleteFromCart() 12](#_Toc200795870)

[🔹 editCart() 12](#_Toc200795871)

[🔹 undoCart() 12](#_Toc200795872)

[🔹 displayCart() 13](#_Toc200795873)

[🔹 checkoutCart() 13](#_Toc200795874)

[🔹 clearCart() 14](#_Toc200795875)

[🔹 displayOrderHistory() 14](#_Toc200795876)

[WISHLIST 15](#_Toc200795877)

[🔹 writeWishlistToFile() 15](#_Toc200795878)

[🔹 addToWishlist() 15](#_Toc200795879)

[🔹 removeFromWishlist() 15](#_Toc200795880)

[🔹 displayWishlistFromFile() 15](#_Toc200795881)

[SEARCH 16](#_Toc200795882)

[🔹 searchById() 16](#_Toc200795883)

[🔹 searchByCategory() 16](#_Toc200795884)

[🔹 searchByName() 16](#_Toc200795885)

[Tools and Technologies 17](#_Toc200795886)

[Contribution of Each Group Member 17](#_Toc200795887)

[Tahira Hafeez (SP24-BCS-123): 17](#_Toc200795888)

[Sabiha Usman (SP24-BCS-109): 17](#_Toc200795889)

[Haleema Jawad (SP24-BCS-046): 17](#_Toc200795890)

[Challenges Faced 18](#_Toc200795891)

[Conclusion 18](#_Toc200795892)

[Future Enhancements 18](#_Toc200795893)

[References 18](#_Toc200795894)

**Shopping Management System**

# Introduction And Features Of The Project

Our semester project, titled **"Shopping Management System"**, is a console-based simulation of an e-commerce platform that enables users to browse, search, and purchase clothing products.

It includes role-based access (admin/user), product management via AVL Tree, cart handling with undo support using Stack, and wish list operations. The system allows personal detail registration, secure login, invoice-based billing, and order history tracking.

# Key Features

## Modules:

### 1. User Management Module

* User Registration
* User Login & Authentication
* User Role Assignment (Admin/User)
* Displaying All Registered Users
* Saving and Loading User Data from File
* Deleting Users

### 2. Admin Functionalities

* Admin Menu Interface
* User Management by Admin

### 3. Product Management Module (AVL Tree Based)

* Product Insertion Using AVL Tree
* Height & Balance Calculation
* AVL Tree Rotations (Right & Left)
* Loading Products from File
* In-Order Display of Products
* Searching Products by:
  + Name
  + ID
  + Category

### 4. Cart Management Module

* Adding Products to Cart
* Deleting Products from Cart
* Editing Cart Items
* Displaying Cart Contents
* Clearing Entire Cart
* Undo Operation (Using Stack)
* Checkout and Bill Generation

### 5. Wishlist Module

* Adding Products to Wishlist
* Displaying Wishlist
* Saving Wishlist to File

### 6. File Handling Module

* Saving and Loading:
  + Users
  + Products
  + Wishlist
  + Bills/Checkout Info

### 7. Search Functionalities

* Product Search by Name
* Product Search by ID
* Product Search by Category

### 8. Stack Operations (Undo Feature)

* Push Operation (Before Cart Modification)
* Pop Operation (Undo Last Cart Action)

## Data Structures Used

This shopping management system utilizes a combination of advanced and fundamental data structures to ensure efficient storage, searching, and management of users, products, carts, and more.

Below is a detailed explanation of each data structure and its application within the system:

### AVL Tree (Self-Balancing Binary Search Tree)

**Purpose**: To store and manage all products in a balanced tree for fast insertion, deletion, and search operations.

**Why AVL Tree?**

* Guarantees O(log n) time complexity for search, insert, and delete.
* Ensures the tree remains balanced after every insertion/deletion using rotations.

**Used in Functions**:

* insertion, rightRotate, leftRotate, getHeight, getBalance, updateHeight
* searchByName, searchById, searchByCategory
* inOrder (for displaying sorted products)

### Singly Linked List

**Purpose**: To manage and store:

* Users (User\* head)
* Cart items (Cart\* cart)
* Wishlist (Wishlist\* wishlist)

**Why Linked List?**

* Dynamic size, ideal for scenarios where the number of elements changes frequently.
* Easy insertion and deletion without shifting elements.

**Used in Functions**:

* User Management: insertAtEnd, displayUsers, deleteUsers
* Cart Management: addToCart, editCart, deleteFromCart, clearCart, displayCart
* Wishlist: addToWishlist, displayWishlistFromFile

### Stack (Linked List Based)

**Purpose**: To implement undo functionality for cart operations.

**Why Stack?**

* Follows Last In First Out (LIFO) principle.
* Ideal for tracking the most recent cart state before changes.

**Used in Functions**:

* push (to save the previous state of the cart)
* pop (to restore the previous state of the cart)
* undoCart

### File I/O (Text File Handling)

**Purpose**: To persist and retrieve data such as users, wishlist, products, and checkout history.

**Why File Handling?**

* Maintains data across sessions.
* Allows admin/user data to be saved locally and reloaded when needed.

**Used in Functions**:

* loadUsersFromFile, saveusersToFile, loadFromFile, writeWishlistToFile, file, etc.

# Objectives

* To implement and demonstrate the application of core data structures in a real-world context.
* To design an interactive and intelligent shopping simulation system.
* To provide essential features of an e-commerce platform using data structures.
* To maintain data persistence through file handling mechanisms.

# System Overview

## User Registration and Login

* **Functionality:** Allows new users to register and existing users to log in.
* **Data Structure:** Linked List
* **Code Snippet:** Used insertAtEnd() to store user nodes; file handling used for persistence.

## Product Browsing

* **Functionality:** View products categorized and sorted.
* **Data Structure:** AVL Tree
* **Code Snippet:** AVL insertion ensures balanced viewing tree; in-order traversal used for display.

## Product Search

* **Functionality:** Search by name or category; view recommendations.
* **Data Structures:** AVL Tree and Graph
* **Code Snippet:** Graph connections used to suggest similar products; search traverses AVL nodes.

## Product Addition (Admin)

* **Functionality:** Add products temporarily or in batch.
* **Data Structures:** Stack and Queue
* **Code Snippet:** Temporary undo uses Stack; Queue supports batch loading.

## Cart Management

* **Functionality:** Add/remove products from cart.
* **Data Structures:** Linked List and Stack
* **Code Snippet:** Cart is a linked list; delete/undo handled by pushing last node into a Stack.

## View Cart

* **Functionality:** Show cart contents and total cost.
* **Data Structure:** Linked List
* **Code Snippet:** Traverse linked list to calculate total price and list items.

## Checkout Process

* **Functionality:** VIP and small carts prioritized.
* **Data Structures:** Min Heap (Priority Queue), File Handling
* **Code Snippet:** Heap stores carts based on priority; data written to invoice file.

## Purchase History

* **Functionality:** View previous purchases.
* **Data Structure:** Linked List with File Handling
* **Code Snippet:** Order info stored in separate files; linked list helps manage in memory.

## Exit

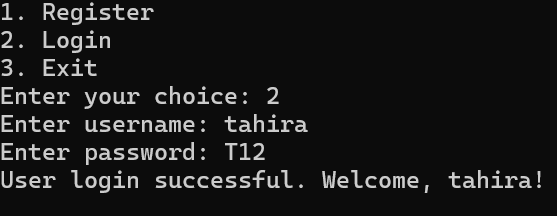
* **Functionality:** Save data and close system.
* **Data Structure:** File Handling
* **Code Snippet:** All updates written to file before program ends.

# Functions:

## USER

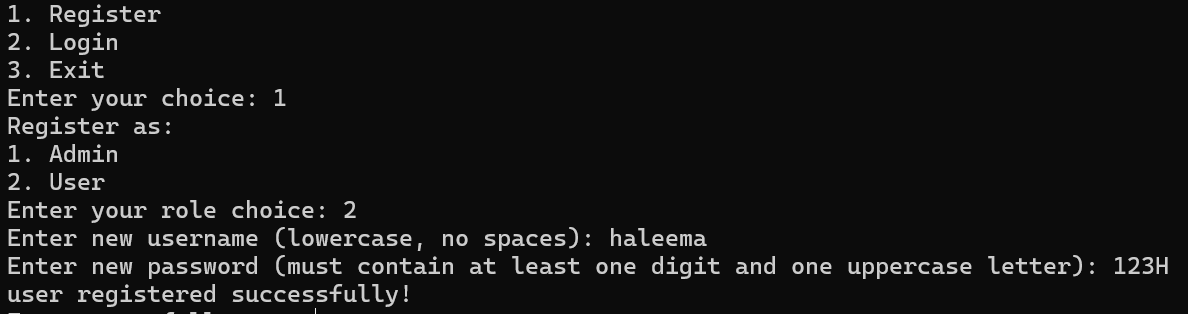
### 🔹 loginUser()

Checks the linked list to verify if the given username and password match an existing user. If found, it logs the user in and returns their role.  
**DSA Concept:** Uses **Linked List Traversal** to search for a match in the user records.



### 🔹 registerUser()

Handles new user registration by validating input, checking for duplicates, saving the new user in the linked list, and writing it to a file.  
**DSA Concept:** Uses **Linked List Insertion**, **Traversal**, and **File Handling** to manage dynamic user data.



### 🔹 saveUsersToFile()

Saves all user data from the linked list into a text file by writing each node’s contents.  
**DSA Concept:** Combines **Linked List Traversal** with **File Handling (ofstream)**.

A black screen with white text

AI-generated content may be incorrect.

### 🔹 loadUsersFromFile()

Reads user data from a file and inserts each record into the linked list dynamically.  
**DSA Concept:** Uses **File Handling (ifstream)** and **Linked List Insertion**.

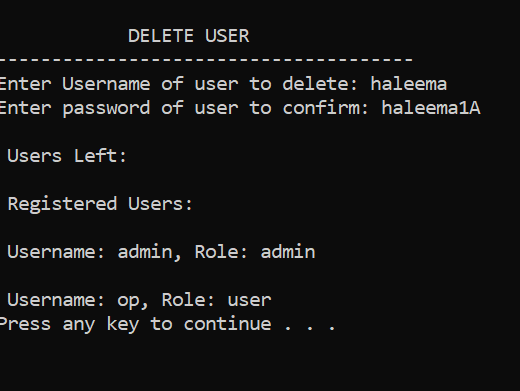
A screenshot of a phone

AI-generated content may be incorrect.

### 🔹 deleteUser()

Deletes a user node from a singly linked list if the username, password, and role match.

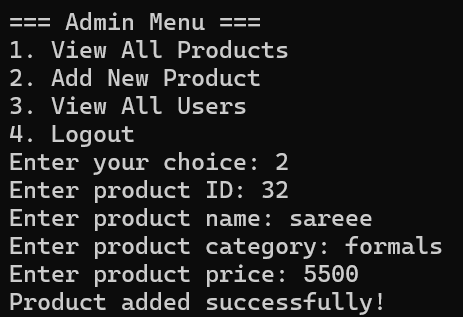
**DSA Concept:**  
Uses **Linked List Traversal** to locate the node and **Pointer Manipulation** to remove it safely.



## PRODUCT

### 🔹 insertion() (of product)

**Description:**  
Inserts a new product into the AVL tree based on product ID and maintains balance using rotations.  
**DSA Concept:** Implements **AVL Tree Insertion**, including **Balancing** and **Rotations** to ensure height balance.



### 🔹 loadFromFile()

Reads product data from a file, creates the product list, and inserts them into the AVL tree using the insertion() function.  
**DSA Concept:** Uses **File Handling** to read data and **AVL Tree Construction** for organizing products by ID.



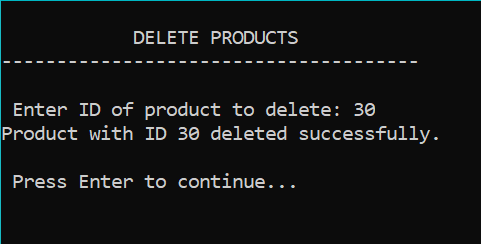
### 🔹 deleteProductById()

Deletes a product from the AVL tree by ID, updates the tree structure, and writes the updated data to file. Displays a confirmation message on successful deletion.

**DSA Concept:**  
Integrates **AVL Tree deletion**, **File Handling**, and **modular programming**. Enhances usability by combining logic for deletion and data persistence.

**Sub functions:-**

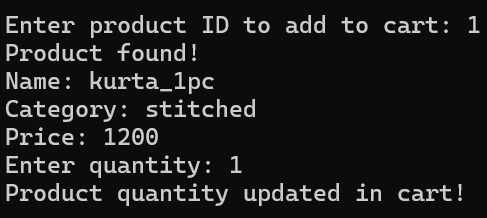
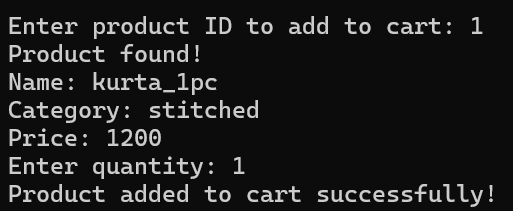
* deleteProduct()
* updateItemsFile()
* saveToFile()
* minValueNode()
* updateHeight()
* getBalance()
* leftRotate()
* rightRotate()

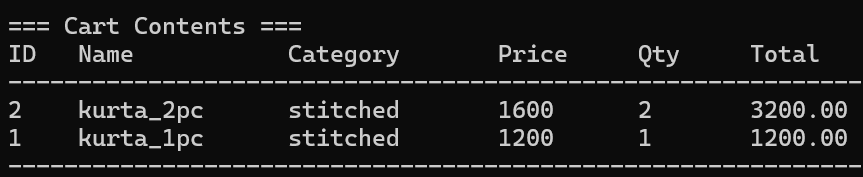


## CART

### 🔹 addToCart()

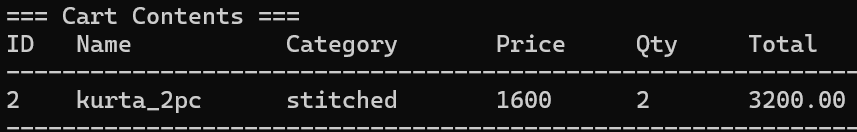
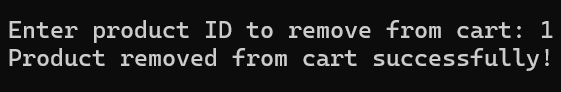
Adds a product to the cart or updates its quantity if it already exists. Uses a linked list to track cart items.  
**DSA Concept:** Implements a **Singly Linked List** for dynamic cart storage and **Linear Search** to detect duplicates.





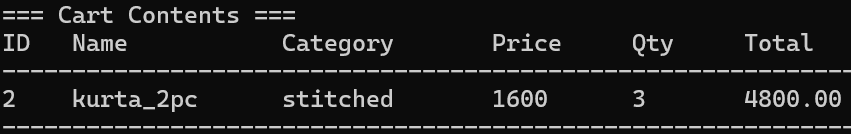
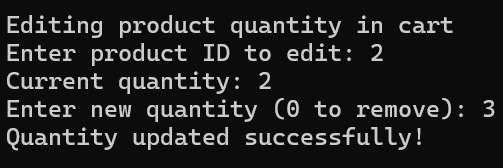
### 🔹 deleteFromCart()

Removes a product from the cart based on the product ID by updating pointers and deallocating memory.  
**DSA Concept:** Uses **Linked List Deletion** with **Pointer Manipulation** for memory management.



### 🔹 editCart()

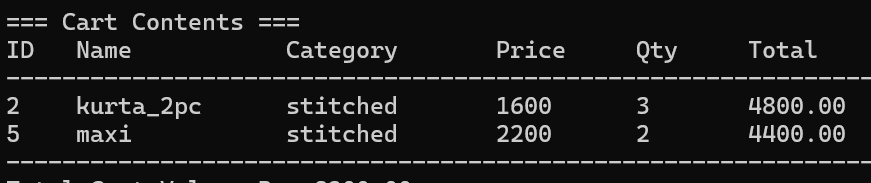
Allows the user to update the quantity of a product in the cart. If the new quantity is 0, the product is removed.  
**DSA Concept:** Uses **Linked List Traversal** to find and update nodes, and **Conditional Deletion** for quantity handling.

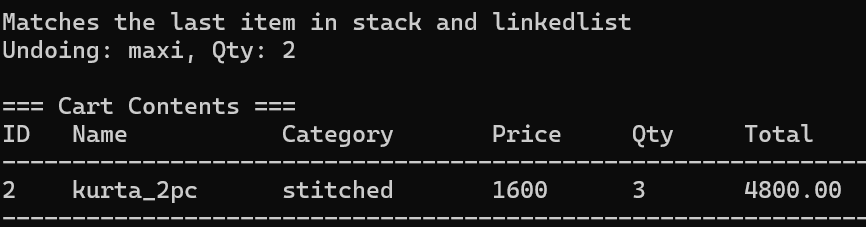


### 🔹 undoCart()

Removes the most recently added product from the cart, acting like an "undo" operation.  
**DSA Concept:** Utilizes a **Stack (LIFO)** operation via a **pop()** function to reverse the last cart action.

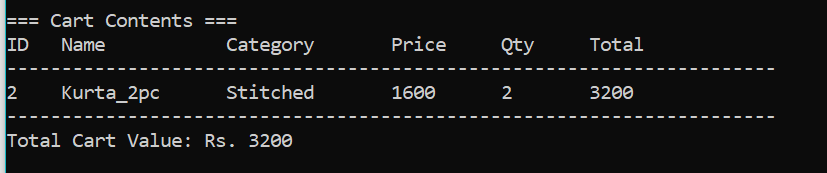
Subfunctions: push(), pop()





### 🔹 displayCart()

**Description:**  
Displays all products currently in the cart along with their details and computes the total cart value.  
**DSA Concept:** Uses **Linked List Traversal** to access and display node data, and **Basic Arithmetic** for total calculation.



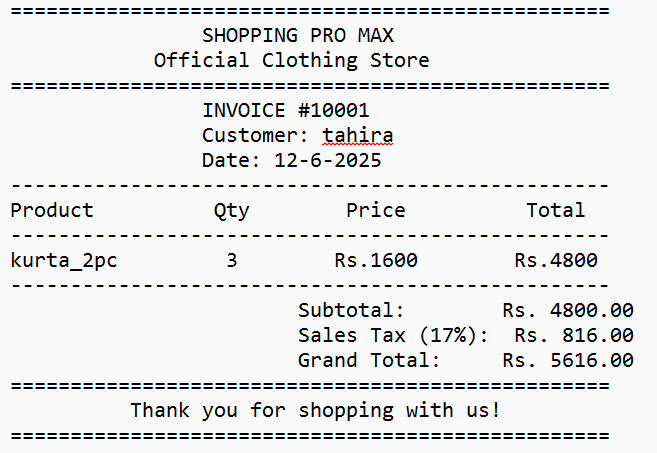
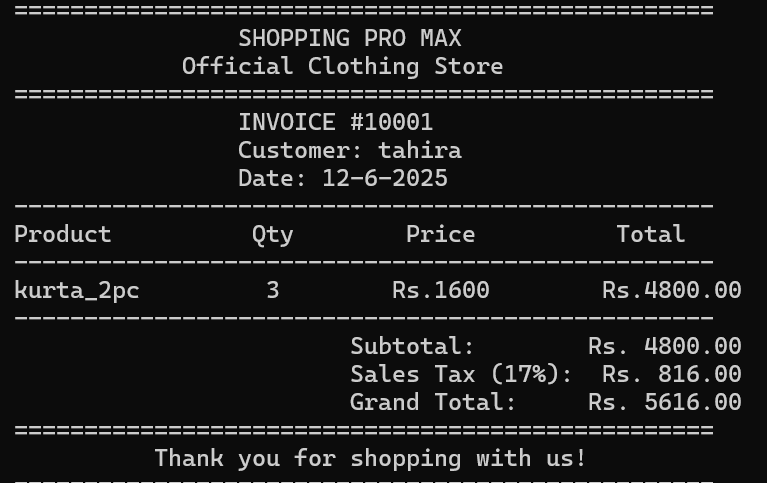
### 🔹 checkoutCart()

Generates an invoice for the cart, calculates total with 17% tax, saves it to a file, and clears the cart.

* Uses linked list to process cart items.
* Displays and saves bill with subtotal, tax, and grand total.
* Invoice number is managed using a file.
* Cart is cleared after checkout.

**DSA Concept:** Linked list, file handling, arithmetic operations.

**CONSOLE DISPLAY FILE DISPLAY**



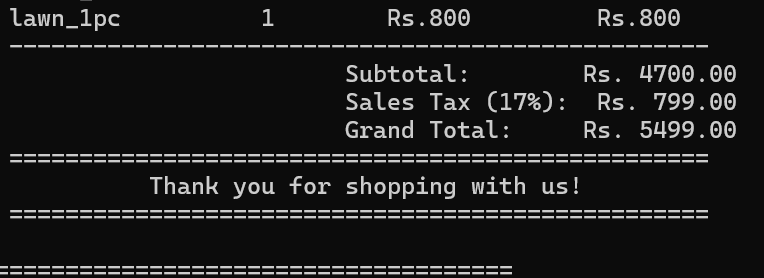
### 🔹 clearCart()

Deletes all items from the cart by deallocating each node, effectively resetting the cart.  
**DSA Concept:** Implements **Linked List Deletion** with memory cleanup using a loop to delete nodes.

### 🔹 **displayOrderHistory()**

Displays previous orders placed by user.  
**DSA Concept:** Uses Linked List Traversal to iterate through the cart, File Handling to save bill details persistently, and Basic Arithmetic for subtotal, tax (17%), and grand total calculation.

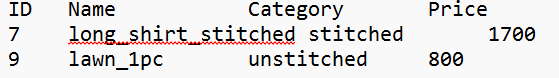




## WISHLIST

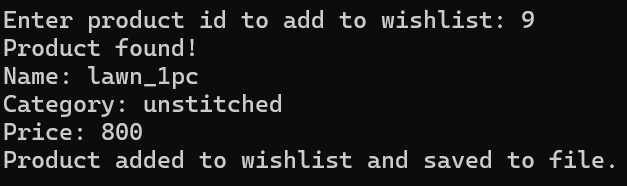
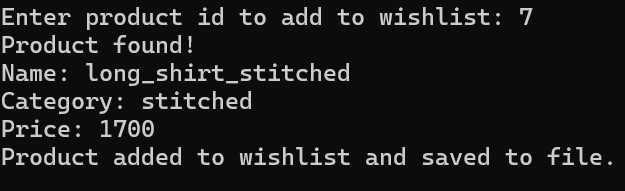
### 🔹 writeWishlistToFile()

Saves all wishlist items to a file (wishlist.txt) by overwriting the file each time. It writes a header and then each product’s ID, name, category, and price.  
**DSA Concept:**  
Uses **Linked List Traversal** to iterate over the Wishlist and **File Handling** to write structured data to a text file.



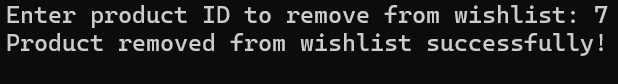
### 🔹 addToWishlist()

Adds a new product to the beginning of the wishlist and then rewrites the entire wishlist to the file (wishlist.txt) for persistence.  
**DSA Concept:**  
Uses **Linked List Insertion at Head** to quickly add the item, and **File Handling** to ensure changes are saved immediately.



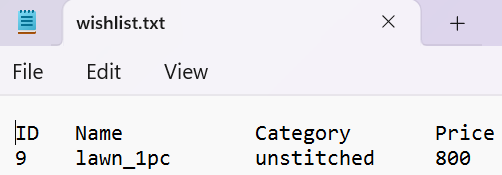
### 🔹 removeFromWishlist()

Removes an item with a specific product ID from the wishlist. Updates the file to reflect the change after deletion.  
**DSA Concept:**  
Implements **Linked List Deletion** by managing node pointers and memory deallocation. Also uses **File Handling** to maintain persistence after changes.



### 🔹 displayWishlistFromFile()

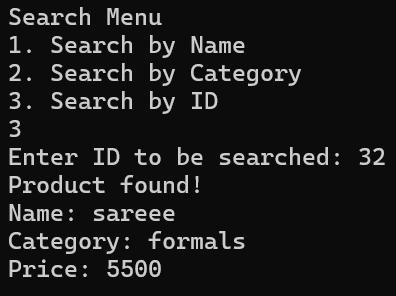
Reads the wishlist data from the wishlist.txt file and displays it line by line on the screen. Ensures the wishlist is accessible even after restarting the program.  
**DSA Concept:** Applies **File Handling (Read Operation)** to retrieve and display persisted wishlist data.



## SEARCH

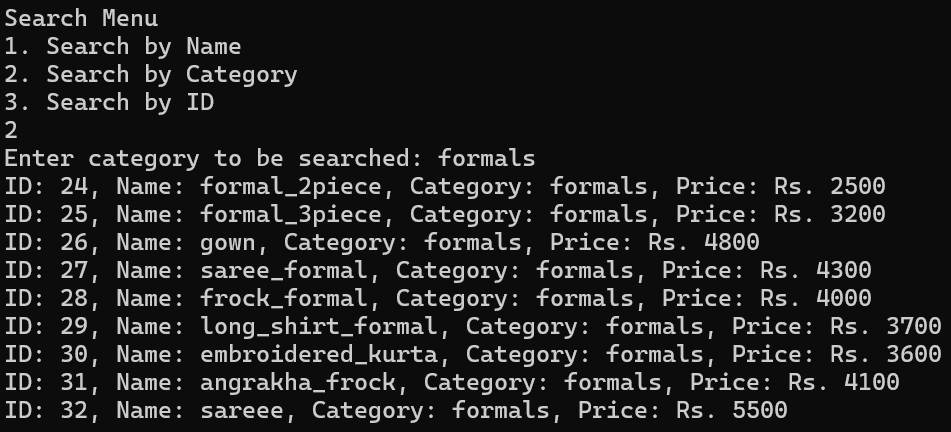
### 🔹 searchById()

Searches for a product in the BST by its ID and displays its details if found.  
**DSA Concept:**  
Implements **Binary Search Tree Traversal (Recursive)** using ID-based comparison.



### 🔹 searchByCategory()

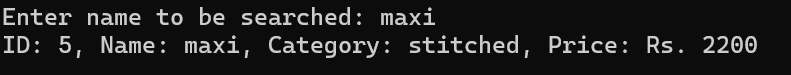
Searches and displays all products matching a given category in the BST.  
**DSA Concept:**  
Implements **Binary Search Tree Traversal (Recursive)** using catergory-based comparison.



### 🔹 searchByName()

Searches and displays all products that match a given name from the BST.

**DSA Concept:** Uses **In-order Traversal** of a Binary Search Tree with conditional matching on product name.



# Tools and Technologies

* **Programming Language:** C++
* **IDE:** Code::Blocks
* **Data Persistence:** File I/O (Text Format)

# Contribution of Each Group Member

## Tahira Hafeez (SP24-BCS-123):

* Handled product addition, removal and saving to file.
* Developed search modules and wish list using linked list.
* Connected wish list logic with file updates.

## Sabiha Usman (SP24-BCS-109):

* Implemented User Registration, Login and Profile system
* Handled file validation, structure design, and error handling.

## Haleema Jawad (SP24-BCS-046):

* Created cart system using linked list and undo using stack
* Developed checkout process and invoice/bill generation
* Built order history retrieval and managed file naming logic.
* Integration of entire code.

All group members worked jointly on integration, testing, and validation to ensure a smooth, logical, and complete end-to-end system.

# Challenges Faced

* Merging registration logic with file and memory operations
* Ensuring validation logic worked under all input conditions
* Structuring the AVL tree and search operations efficiently
* Building undo logic without breaking cart state
* Keeping user session data synced across modules

# Conclusion

The Shopping Management System successfully demonstrates how fundamental data structures can be utilized in building a practical e-commerce simulation. This system showcases the integration of multiple data structures into a fully functional shopping application. The cart system, AVL tree for product storage, stack for undo, and file operations simulate real-world e-commerce logic effectively.

# Future Enhancements

* Develop GUI using a framework like Qt or SFML
* Add proper user session timeouts and security layers
* Integrate dynamic memory and product update modules
* Add admin dashboard for product stats and inventory
* Enable live database connectivity (MySQL or SQLite)

# References

* Class lecture notes and DSA textbook
* GeeksforGeeks and CPlusPlus.com for code logic
* Stack Overflow for troubleshooting