Supermarket Inventory Management System

Project Report

Struct 122

Ronil Kag
Rajdev Singh
Anand S Menon

November 25, 2025

Contents

1 Project Description

The Supermarket Management System is a terminal-based software application designed to digitize the inventory tracking process for retail stores. Traditional pen-and-paper methods are prone to human error and data loss; this project solves that problem by creating a digital database of products.

We have built a system that utilizes a **Linked List** as its core data structure. This allows the inventory to be dynamic—growing as new products are added and shrinking as they are deleted—without the fixed-size limitations of standard arrays. The system persists data between sessions by saving the inventory to a binary file (supermarket.dat) upon exit and reloading it upon startup.

2 Detailed Functionalities

The project offers two primary categories of functionality: Administrative and Operational.

2.1 Administrative Features

- **Inventory Creation:** Administrators can add new items to the database. The system captures the Product Name, Price, and Quantity.
- **Stock Visualization:** Users can view the entire list of products in a formatted table, showing ID, Name, Price, and current Quantity.
- **Data Maintenance:** The system allows for updating existing product details and deleting items from the inventory.

2.2 Customer Features

- **Purchase System:** A simulation of the checkout process. Customers select items by name and specify a quantity. The system calculates the total bill and automatically deducts the purchased amount from the inventory.

2.3 System Features

- **Data Persistence:** The system automatically loads data when the program starts and saves data when the user chooses to exit, ensuring no changes are lost.

3 Implementation Details

3.1 Adding a Product

1. **Allocation:** The system uses malloc to reserve memory for a new Product node.
2. **Linking:** The new node is inserted in the ascending order of its price.

3.2 Deleting a Product

1. **Search:** The system iterates through the linked list looking for the user-provided name.
2. **Unlinking:**
   - If the target is the head, the head pointer is moved to the second node.
   - If the target is in the middle, the previous node's next pointer is set to skip the target node.
3. **Deallocation:** The memory of the unlinked target node is released using free().

3.3 File Persistence (Save/Load)

1. **Saving:** The system opens supermarket.dat in binary write mode (wb). It iterates through the list, writing the raw memory content of each struct to the file.
2. **Loading:** The system opens the file in binary read mode (rb). It reads struct-sized chunks in a loop. For every chunk read, it allocates a new node in the linked list and gives it the file data.

4 Breakdown of Contributions

The development of this project was divided among the team members as follows:

Team Member	Contributions
Ronil Kag	Implemented the create and display functions. Created the project report.
Rajdev Singh	Implemented the customer function and free memory function along with writing main.c
Anand S Menon	Implemented the Update and Delete functions, and wrote data.h

5 Function Explanations

File: main.c

- menu()
  - *Role:* Displays the available options (1-5) to the user.
  - *Input/Output:* Takes no arguments; returns void. Prints text to stdout.
- main(void)
  - *Role:* The entry point of the program. It handles the main infinite loop, input validation, and switch-case logic to call other functions.

File: crud.c

- load_dbfile()
  - *Role:* Reads the binary database file on startup and reconstructs the linked list in memory.
  - Modifies the global product_lis_head
- create()
  - *Role:* Allocates memory for a new node, takes user input for details, and adds in ascending order of the pricing.
- display()
  - *Role:* Traverses the linked list and prints the details of every product in a tabular format.
- update_product()
  - *Role:* Finds a product by name and allows the user to overwrite its Name, Price, or Quantity.

- delete_product()
  - *Role:* Removes a node from the linked list and frees the associated memory to prevent leaks.

- customer()
  - *Role:* Handles sales. Reduces the quantity of a specific item based on user purchase amount.
  - *Input/Output:* Takes no arguments; returns void. Output is the final bill amount.
- save_dbfile()
  - *Role:* Writes the current state of the linked list to the hard drive for persistence.
  - *Input/Output:* Writes to supermarket.dat.
- free_memory()
  - *Role:* Iterates through the entire list and frees every node before the program exits.