



# PROJECT REPORT

Project Title: *Warehouse Shelf Tracker*

Submitted By: Samrata

SAP ID: 590021799

Course: B.Tech. Computer Science &  
Engineering

Course Code: CSEG1032

Submitted to: Rahul Prasad

# INDEX

Sr. No.	Name	Pg. No.
1.	Abstract	3
2.	Problem Definition	4
3.	System Design	6
4.	Implementation Details	12
5.	Testing & Results	17
6.	Conclusion & Future Work	18
7.	Reference	20
8.	Appendix	21

# ABSTRACT

This document describes the design and implementation of a Warehouse Shelf Tracker system. The application is a console-based utility developed in C, designed to manage the occupancy status of a fixed number of storage units (slots). The core functionality is achieved using a static integer array of size 10, where each index represents a storage slot.

The system supports binary tracking of inventory (occupied or empty) and provides essential features such as adding and removing items from specific slots, viewing the current shelf status, finding the first available empty slot, and globally clearing all stored inventory. The system ensures robust operation through boundary checks and validation against array limits and current slot occupancy.

# PROBLEM DEFINITION

## 2.1. Goal

To create a simple, reliable console application that simulates the management of a fixed-size warehouse shelf, allowing a user to track the occupancy status of individual storage slots in real-time.

## 2.2. Functional Requirements

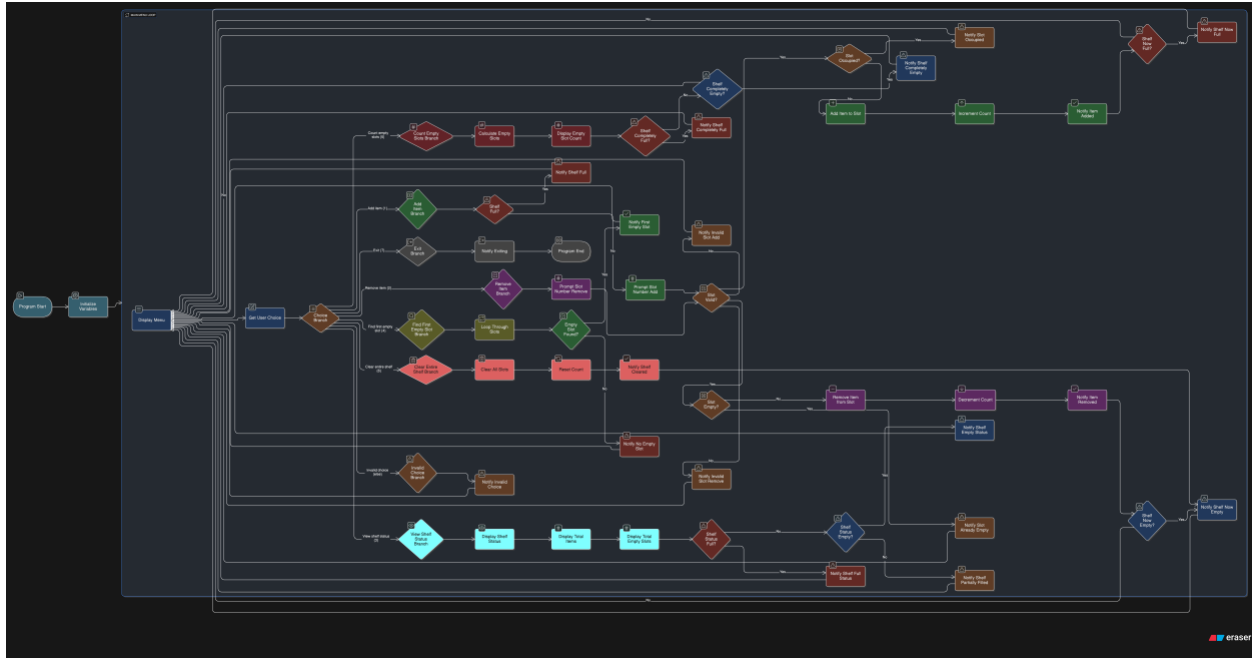
The system must satisfy the following user requirements:

1. **Add Item:** Mark a specific, currently empty slot as occupied, provided the shelf is not full.
2. **Remove Item:** Mark a specific, currently occupied slot as empty.
3. **View Status:** Display the occupancy status (Empty/Occupied) for every slot, along with an overall summary (total items, total empty slots).
4. **Find Empty Slot:** Identify and report the index of the first available empty slot, starting from index \$0\$.
5. **Count Empty Slots:** Calculate and display the total number of empty slots remaining.
6. **Clear Shelf:** Reset the status of all slots to empty.
7. **Exit:** Terminate the application gracefully.

## 2.3. Constraints

- **Fixed Capacity:** The shelf must have a fixed capacity, defined by MAX\_SLOTS (currently 10).
- **Binary State:** Each slot can only exist in one of two states: occupied or empty (1 or 0).
- **Data Structure:** The storage mechanism must be a standard C array.
- **User Interface:** The interface must be command-line (console) based.

# SYSTEM DESIGN



## Algorithm: Warehouse Shelf Tracker

### Global Definitions

- **Define Constant MAX\_SLOTS = 10**  
Represents the total number of shelf slots available.
- **Define Array shelf[MAX\_SLOTS]**  
Stores 0 (empty) or 1 (occupied) for each slot.
- **Define Integer count = 0**  
Stores the number of occupied slots.

### Main Function (Control Loop)

#### Step 1: Start.

## **Step 2: Initialize variables**

- Declare integer choice.
- Declare integer slot.
- Initialize array shelf with 0 (empty).

## **Step 3 (Start of Loop): Display the Main Menu:**

1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit

## **Step 4: Ask user to enter choice. Read input into choice.**

## **Step 5: Check user choice using Switch Case.**

### **Case 1: Add Item**

1. Check if count == MAX\_SLOTS
  - If TRUE: Display "Shelf Full" and Return.
2. Ask user for slot number.
3. Validate slot number (0–9).
4. Check if slot is already occupied.

5. If empty:

- Set shelf[slot] = 1
- Increment count
- Display success message

6. If count becomes 10, display "Shelf Full Warning".

7. Return to Main Menu.

---

## **Case 2: Remove Item**

1. Ask user for slot number.

2. Validate slot number (0–9).

3. Check if slot is already empty:

- If TRUE: Display "Already Empty".

4. Else:

- Set shelf[slot] = 0
- Decrement count
- Display "Item Removed".

5. If count becomes 0, display "Shelf Empty".

6. Return to Main Menu.

---

## **Case 3: View Shelf Status**



1. Print table-like slot status for all slots.
  2. Display total items stored.
  3. Display total empty slots.
  4. Display Shelf Status:
    - FULL (if count == 10)
    - EMPTY (if count == 0)
    - PARTIALLY FILLED otherwise
  5. Return.
- 

#### **Case 4: Find First Empty Slot**

1. Loop from i = 0 to 9
  2. If shelf[i] == 0:
    - Display that slot i is the first empty
    - Break
  3. If no slot found, display "Shelf Full".
  4. Return.
- 

#### **Case 5: Count Empty Slots**

1. Compute empty = MAX\_SLOTS - count.
2. Display number of empty slots.

3. If `empty == 0`, display "Shelf Full".
  4. If `empty == 10`, display "Shelf Empty".
  5. Return.
- 

### **Case 6: Clear Entire Shelf**

1. Loop from `i = 0` to `9`:
    - Set `shelf[i] = 0`
  2. Set `count = 0`
  3. Display "All slots cleared".
  4. Return.
- 

### **Case 7: Exit**

1. Display "Exiting Program..."
  2. Stop the program.
- 

### **Default Case**

1. Display "Invalid Choice".
2. Return to Main Menu.

# IMPLEMENTATION DETAILS

## 1. Initializing Shelf System

A static integer array `shelf[MAX_SLOTS]` is used to store the status of 10 shelf slots, where

0 = empty and 1 = occupied.

A counter variable `count` tracks the total number of occupied slots.

**Implemented using:** arrays and variable initialization.

```
C sam.c > main()
1  #include <stdio.h>
2
3  #define MAX_SLOTS 10
4
5  int main() {
6
7      int shelf[MAX_SLOTS] = {0};
8      int choice, slot;
9      int count = 0;
10     int i;
11
12     while (1) {
13         printf("\n=====\\n");
14         printf("        WAREHOUSE SHELF TRACKER\\n");
15         printf("=====\\n");
16         printf("1. Add item to a slot\\n");
17         printf("2. Remove item from a slot\\n");
18         printf("3. View shelf status\\n");
19         printf("4. Find first empty slot\\n");
20         printf("5. Count empty slots\\n");
21         printf("6. Clear entire shelf\\n");
22         printf("7. Exit\\n");
23         printf("=====\\n");
24         printf("Enter your choice: ");
25         scanf("%d", &choice);
26         printf("=====\\n");
27
28     }
```

## 2. Adding an Item

The user enters a slot number.

The program checks if:

- the slot number is between 1–10
- the selected slot is empty

If valid, the program marks the slot as occupied and increments the count.

**Implemented using:** array indexing, validation, and conditionals.

```

27
28
29     if (choice == 1) {
30         if (count == MAX_SLOTS) {
31             printf("Shelf is FULL! Cannot add more items.\n");
32         } else {
33             printf("Enter slot number to add (0-9): ");
34             scanf("%d", &slot);
35
36             if (slot < 0 || slot >= MAX_SLOTS) {
37                 printf("Invalid slot! Choose between 0 and 9.\n");
38             }
39             else if (shelf[slot] == 1) {
40                 printf("That slot is already occupied!\n");
41             }
42             else {
43                 shelf[slot] = 1;
44                 count++;
45                 printf("Item successfully added to slot %d.\n", slot);
46
47                 if (count == MAX_SLOTS) {
48                     printf("WARNING: Shelf is now completely full!\n");
49                 }
50             }
51         }
52     }
53

```

### 3. Removing an Item

The user chooses a slot to clear.

The program checks if:

- the slot number is valid
- the slot contains an item

If true, the slot is set to empty and the count decreases.

**Implemented using:** conditional checks and array updates.

```

54     else if (choice == 2) {
55         printf("Enter slot number to remove (0-9): ");
56         scanf("%d", &slot);
57
58         if (slot < 0 || slot >= MAX_SLOTS) {
59             printf("Invalid slot! Choose between 0 and 9.\n");
60         }
61         else if (shelf[slot] == 0) {
62             printf("Slot %d is already empty!\n", slot);
63         }
64         else {
65             shelf[slot] = 0;
66             count--;
67             printf("Item removed from slot %d.\n", slot);
68
69             if (count == 0) {
70                 printf("Shelf is now completely empty.\n");
71             }
72         }
73     }
74 }

```

---

## 4. Viewing Shelf Status

Displays all 10 slots with their status (Empty/Occupied).

Loop is used to iterate through the array and print values.

**Implemented using:** for loops and formatted output.

```

75
76     else if (choice == 3) {
77         printf("Current Shelf Status:\n");
78         printf("-----\n");
79
80         for (i = 0; i < MAX_SLOTS; i++) {
81             if (shelf[i] == 1)
82                 printf("Slot %d : Occupied\n", i);
83             else
84                 printf("Slot %d : Empty\n", i);
85         }
86
87         printf("-----\n");
88         printf("Total items stored: %d\n", count);
89         printf("Total empty slots: %d\n", MAX_SLOTS - count);
90
91         if (count == MAX_SLOTS)
92             printf("SHELF STATUS: FULL\n");
93         else if (count == 0)
94             printf("SHELF STATUS: EMPTY\n");
95         else
96             printf("SHELF STATUS: PARTIALLY FILLED\n");
97     }
98 }

```

---

## 5. Viewing Total Items

Simply prints the value of the counter variable count, which stores the number of occupied slots.

**Implemented using:** integer counter tracking.

```
100     else if (choice == 4) {
101         int found = 0;
102
103         for (i = 0; i < MAX_SLOTS; i++) {
104             if (shelf[i] == 0) {
105                 printf("First empty slot is: %d\n", i);
106                 found = 1;
107                 break;
108             }
109         }
110
111         if (!found)
112             printf("No empty slot found. Shelf is full!\n");
113     }
114 }
```

---

## 6. Resetting All Slots

This option clears all shelf slots by setting the entire array back to zero and resetting count to 0.

**Implemented using:** loops to reset array values.

```

114
115
116     else if (choice == 5) {
117         int empty = MAX_SLOTS - count;
118
119         printf("Total empty slots: %d\n", empty);
120
121         if (empty == 0)
122             printf("Shelf is completely full.\n");
123         else if (empty == MAX_SLOTS)
124             printf("Shelf is completely empty.\n");
125     }
126
127
128     else if (choice == 6) {
129         for (i = 0; i < MAX_SLOTS; i++)
130             shelf[i] = 0;
131
132         count = 0;
133
134         printf("All slots have been cleared.\n");
135         printf("Shelf is now completely empty.\n");
136     }
137

```

---

## 7. Exiting the Program

Selecting this option terminates the infinite loop and exits the program.

**Implemented using:** return statement from main().

```

138     else if (choice == 7) {
139         printf("Exiting program... Goodbye!\n");
140         break;
141     }
142
143     else {
144         printf("Invalid choice! Please select a valid option.\n");
145     }
146 }
147
148 return 0;
149 }
150

```



# Testing and Results

The code was tested through various scenarios, inputting both valid and invalid entries.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS R:\C language\project> cd "r:\C language\project\" ; if ($?) { gcc sam.c -o sam } ; if ($?) { .\sam }

=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 1
=====
Enter slot number to add (0-9): 3
Item successfully added to slot 3.

=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 1
=====
Enter slot number to add (0-9): 1
Item successfully added to slot 1.
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 1
=====
Enter slot number to add (0-9): 2
Item successfully added to slot 2.

=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 4
=====
First empty slot is: 0
```

```
=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 5
=====
Total empty slots: 7

=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 6
=====
All slots have been cleared.
Shelf is now completely empty.
```

```
=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 5
=====
Total empty slots: 10
Shelf is completely empty.

=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 7
=====
Exiting program... Goodbye!
PS R:\C language\project> █
```

```
Exiting program... Goodbye!cd "r:\C language\project\" ; if ($?) { gcc sam.c -o
sam } ; if ($?) { .\sam }

=====
WAREHOUSE SHELF TRACKER
=====
1. Add item to a slot
2. Remove item from a slot
3. View shelf status
4. Find first empty slot
5. Count empty slots
6. Clear entire shelf
7. Exit
=====
Enter your choice: 8
=====
Invalid choice! Please select a valid option.
```

# CONCLUSION & FUTURE WORK

## Conclusion

The *Warehouse Shelf Tracker* successfully demonstrates how arrays and menu-driven programming can be used to manage storage slots efficiently. The program allows users to add, remove, search, count, and clear shelf items in real time. It ensures error-free operation through proper input validation and provides clear system status updates. This project strengthens understanding of conditional logic, loops, array manipulation, and user-interactive C applications.

---

## Future Work

To improve and expand this project, the following enhancements can be implemented:

### **1. Dynamic Shelf Size**

Allow users to choose the number of slots instead of using a fixed size (10).

### **2. Item Details**

Store more information about each item (name, ID, weight, category) instead of only 0 and 1.

### **3. File Handling**

Save shelf data to a file and load it on program start, ensuring data is not lost when the program exits.

### **4. Search Feature**

Add functionality to search for specific items or item types.

### **5. GUI Version**

Upgrade the program from a console interface to a graphical interface using libraries like GTK or a simple web-based frontend.

### **6. Multiple Shelves / Sections**

Extend support for multiple racks, shelves, or warehouse zones.

### **7. Automatic Slot Allocation**

Introduce algorithms to automatically find optimal slots based on size, weight, or item type.

### **8. Error Logging**

Add logs to track incorrect inputs, removals, or system warnings.

# REFERENCES

1. Mr. Rahul Prasad
2. Let us C by Yashwant Kanetkar
3. <https://www.programiz.com/c-programming/online-compiler/>
4. <https://www.geeksforgeeks.org/c/c-programming-language/>

# APPENDIX

## SOURCE CODE

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

```
#define MAX_SLOTS 10
```

```
int main() {
```

```
    int shelf[MAX_SLOTS] = {0};
```

```
    int choice, slot;
```

```
    int count = 0;
```

```
    int i;
```

```
    while (1) {
```

```
        printf("\n=====\\n");
```

```
        printf("    WAREHOUSE SHELF TRACKER\\n");
```

```
        printf("=====\\n");
```

```
        printf("1. Add item to a slot\\n");
```

```
        printf("2. Remove item from a slot\\n");
```

```
        printf("3. View shelf status\\n");
```

```
        printf("4. Find first empty slot\\n");
```

```
printf("5. Count empty slots\n");
printf("6. Clear entire shelf\n");
printf("7. Exit\n");
printf("=====\n");
printf("Enter your choice: ");
scanf("%d", &choice);
printf("=====\n");

if (choice == 1) {
    if (count == MAX_SLOTS) {
        printf("Shelf is FULL! Cannot add more items.\n");
    } else {
        printf("Enter slot number to add (0-9): ");
        scanf("%d", &slot);

        if (slot < 0 || slot >= MAX_SLOTS) {
            printf("Invalid slot! Choose between 0 and 9.\n");
        }

        else if (shelf[slot] == 1) {
            printf("That slot is already occupied!\n");
        }

        else {
            shelf[slot] = 1;
            count++;
            printf("Item successfully added to slot %d.\n", slot);
        }
    }
}
```

```
        if (count == MAX_SLOTS) {  
            printf("WARNING: Shelf is now completely full!\n");  
        }  
    }  
}
```

```
else if (choice == 2) {  
    printf("Enter slot number to remove (0-9): ");  
    scanf("%d", &slot);  
  
    if (slot < 0 || slot >= MAX_SLOTS) {  
        printf("Invalid slot! Choose between 0 and 9.\n");  
    }  
    else if (shelf[slot] == 0) {  
        printf("Slot %d is already empty!\n", slot);  
    }  
    else {  
        shelf[slot] = 0;  
        count--;  
        printf("Item removed from slot %d.\n", slot);  
  
        if (count == 0) {  
            printf("Shelf is now completely empty.\n");  
        }  
    }  
}
```

```
}
```

```
else if (choice == 3) {
```

```
    printf("Current Shelf Status:\n");
```

```
    printf("-----\n");
```

```
    for (i = 0; i < MAX_SLOTS; i++) {
```

```
        if (shelf[i] == 1)
```

```
            printf("Slot %d : Occupied\n", i);
```

```
        else
```

```
            printf("Slot %d : Empty\n", i);
```

```
    }
```

```
    printf("-----\n");
```

```
    printf("Total items stored: %d\n", count);
```

```
    printf("Total empty slots: %d\n", MAX_SLOTS - count);
```

```
    if (count == MAX_SLOTS)
```

```
        printf("SHELF STATUS: FULL\n");
```

```
    else if (count == 0)
```

```
        printf("SHELF STATUS: EMPTY\n");
```

```
    else
```

```
        printf("SHELF STATUS: PARTIALLY FILLED\n");
```

```
}
```

```
else if (choice == 4) {
```



```

int found = 0;

for (i = 0; i < MAX_SLOTS; i++) {
    if (shelf[i] == 0) {
        printf("First empty slot is: %d\n", i);
        found = 1;
        break;
    }
}

if (!found)
    printf("No empty slot found. Shelf is full!\n");
}

else if (choice == 5) {
    int empty = MAX_SLOTS - count;

    printf("Total empty slots: %d\n", empty);

    if (empty == 0)
        printf("Shelf is completely full.\n");
    else if (empty == MAX_SLOTS)
        printf("Shelf is completely empty.\n");
}

else if (choice == 6) {

```

```
    for (i = 0; i < MAX_SLOTS; i++)  
        shelf[i] = 0;  
  
    count = 0;  
  
    printf("All slots have been cleared.\n");  
    printf("Shelf is now completely empty.\n");  
}  
  
else if (choice == 7) {  
    printf("Exiting program... Goodbye!\n");  
    break;  
}  
  
else {  
    printf("Invalid choice! Please select a valid option.\n");  
}  
}  
  
return 0;  
}
```