Report

Smart Waste Management System

Project Id: 7

Students Name

Pushpesh Pant (2461395)

Rishita Nainwal (2461270)

Rohit Kumar Rathour (2461399)

Tanushree Joshi (2461324)

(Section-C)

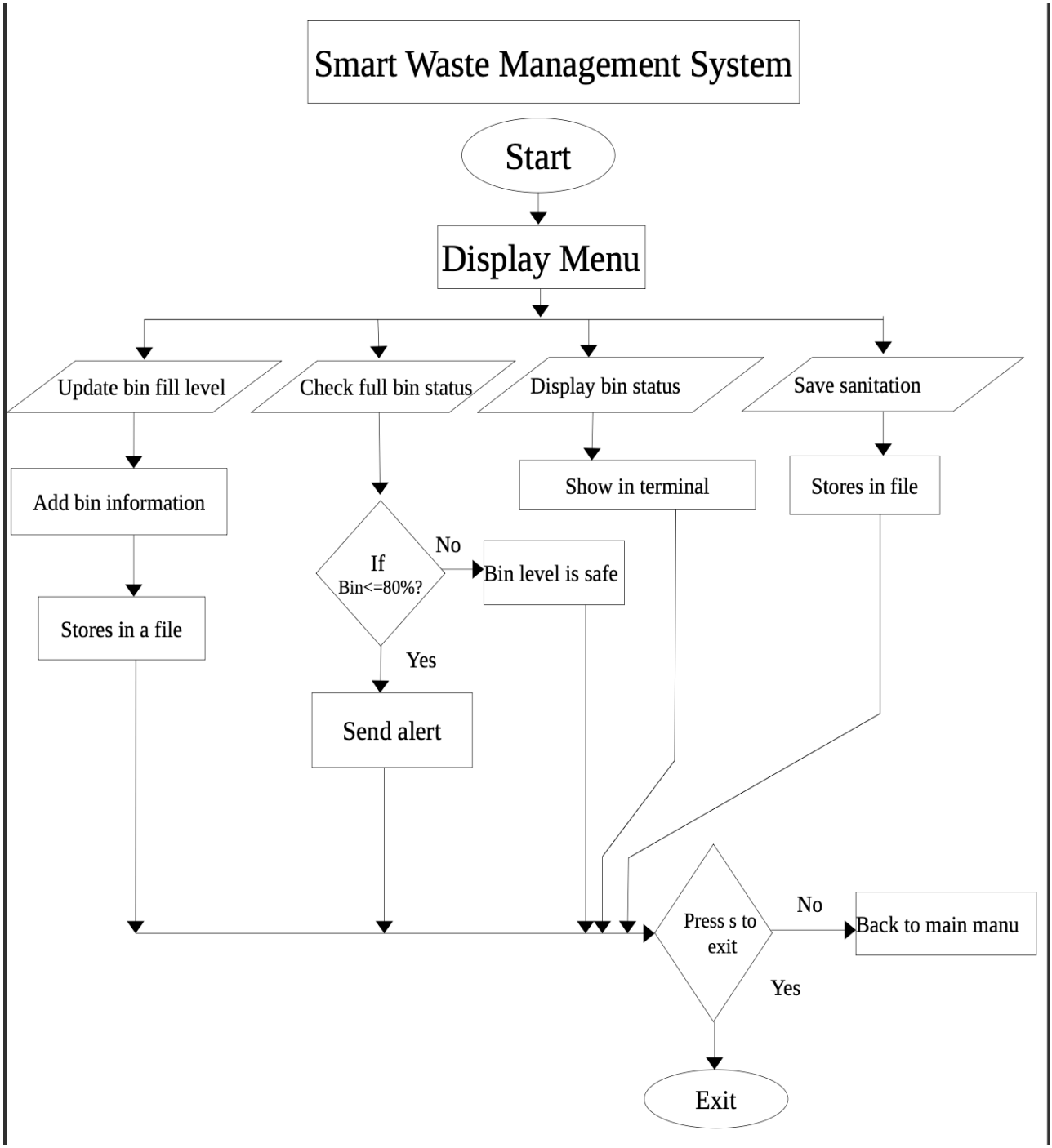
Table of contents

Problem Description

Managing waste collection manually in urban and rural areas often leads to inefficiencies such as overflowing bins, missed pickups, and poor hygiene conditions. Traditional waste management systems rely heavily on fixed schedules and lack real-time monitoring, which results in unnecessary resource usage and environmental concerns. These systems are not responsive to actual waste levels and can lead to increased operational costs and dissatisfaction among residents.

To overcome these issues, a need arises for an intelligent and automated solution that can monitor, manage, and optimize the waste collection process. The **Smart Waste Management System** is proposed to address these challenges. This system utilizes sensors to detect the fill levels of waste bins and sends real-time data to a central system. Based on this data, the system can schedule pickups efficiently, reduce fuel consumption, and prevent bin overflows. It ensures cleaner surroundings and contributes to sustainable urban living. This project can be implemented as a C-based console application or integrated with IoT devices for real-time monitoring and optimized waste collection routing.

Flow Diagram



Modules For Project Implementation

This project is organized into multiple logical modules, each responsible for handling a specific functionality of the Waste Management System. The **Waste Level Monitoring Module** plays a crucial role in tracking the fill levels of waste bins in real time. Using sensors like ultrasonic or infrared, this module collects data about the amount of waste accumulated in each bin. The collected data is then transmitted to a central server where it is processed to determine which bins are full and need immediate attention. This module helps in optimizing collection routes by informing the system about high-priority bins, thereby reducing fuel consumption and improving operational efficiency. Alerts and notifications can also be generated when a bin crosses a certain threshold, ensuring timely waste disposal and preventing overflows.

Module 1: Bin Initialization (MAX_BINS)

- **Purpose**: Predefines and initializes the properties of bins.
- **Data Structure Used**: `struct WasteBin`
- **Functionality**:
  - Stores the bin ID, location, current fill level, and hours since last cleaned.
  - Initializes `MAX_BINS` (5 bins) at fixed campus locations.

CODE SNIPPET

```
 5   #define MAX_BINS 5
 6   #define FULL_THRESHOLD 80
 7   #define MAX_HOURS_BEFORE_CLEANING 48
 8
 9 > typedef struct {…
14   } WasteBin;
15
16   // Predefined bins within campus
17   WasteBin bins[MAX_BINS] = {
18       {1, "Library", 0, 0},
19       {2, "Hostel", 0, 0},
20       {3, "Canteen", 0, 0},
21       {4, "Admin Block", 0, 0},
22       {5, "Playground", 0, 0}
23   };
```

MODULE 2: User Interaction Module (Main Menu)

- **Purpose**: Provides a user-friendly interface for administrators.
- **Functionality**:
  - Displays options to update bins, check alerts, view statuses, save logs, or exit.
  - Takes user input and triggers corresponding modules.

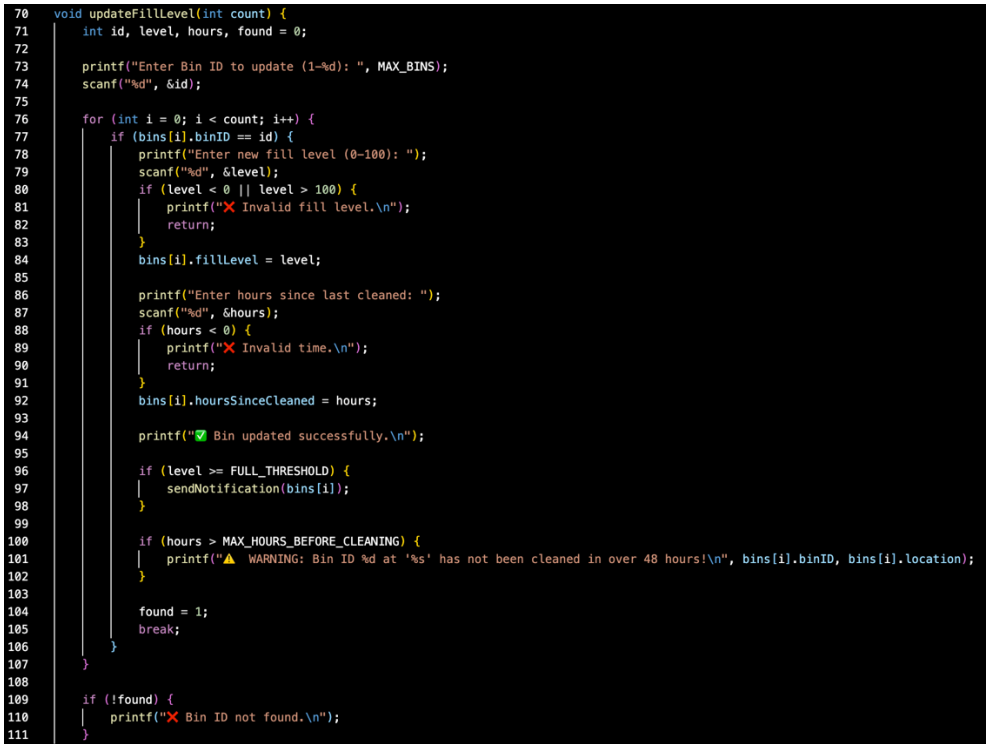
CODE SNIPPET

```c
int main() {
    int choice;

    while (1) {
        printf("\n===== Smart Waste Management System =====\n");
        printf("1. Update Bin Fill Level & Cleaning Time\n");
        printf("2. Check Full Bins & Time Alerts\n");
        printf("3. Display All Bin Status\n");
        printf("4. Save Sanitation Log to File\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar(); // clear newline

        switch (choice) {
            case 1:
                updateFillLevel(MAX_BINS);
                break;
            case 2:
                checkFullBins(MAX_BINS);
                break;
            case 3:
                displayBins(MAX_BINS);
                break;
            case 4:
                saveSanitationLog(MAX_BINS);
                break;
            case 5:
                printf("Exiting the system. Thank you!\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
```

MODULE 3: Update Bin Data

- **Function**: `updateFillLevel(int count)`
- **Purpose**: Allows the user to update the fill level and last cleaned time for a selected bin.
- **Features**:
  - Validates input data.
  - Automatically checks for critical conditions:
    - If fill level ≥ 80%, sends an alert.
    - If not cleaned for over 48 hours, issues a warning.

CODE SNIPPET

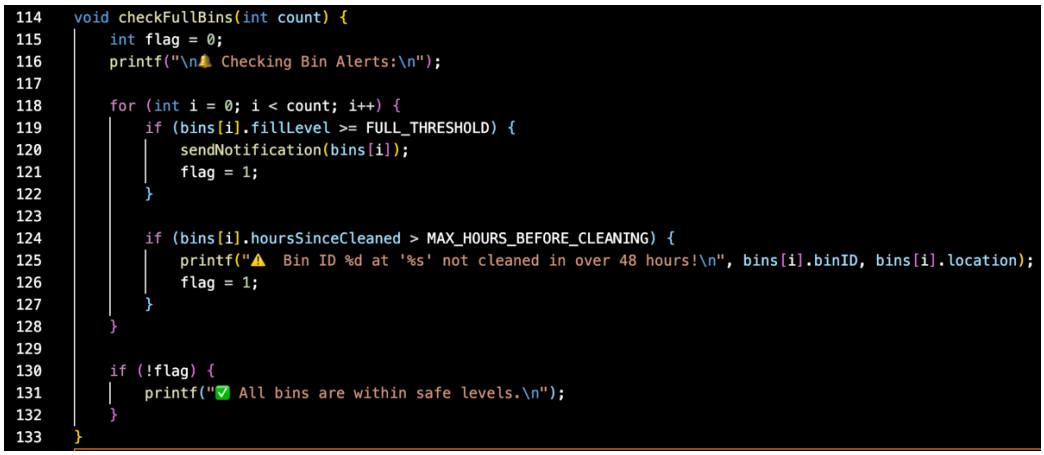


```c
void updateFillLevel(int count) {
    int id, level, hours, found = 0;

    printf("Enter Bin ID to update (1-%d): ", MAX_BINS);
    scanf("%d", &id);

    for (int i = 0; i < count; i++) {
        if (bins[i].binID == id) {
            printf("Enter new fill level (0-100): ");
            scanf("%d", &level);
            if (level < 0 || level > 100) {
                printf("❌ Invalid fill level.\n");
                return;
            }
            bins[i].fillLevel = level;

            printf("Enter hours since last cleaned: ");
            scanf("%d", &hours);
            if (hours < 0) {
                printf("❌ Invalid time.\n");
                return;
            }
            bins[i].hoursSinceCleaned = hours;

            printf("✅ Bin updated successfully.\n");

            if (level >= FULL_THRESHOLD) {
                sendNotification(bins[i]);
            }

            if (hours > MAX_HOURS_BEFORE_CLEANING) {
                printf("⚠️  WARNING: Bin ID %d at '%s' has not been cleaned in over 48 hours!\n", bins[i].binID, bins[i].location);
            }

            found = 1;
            break;
        }
    }

    if (!found) {
        printf("❌ Bin ID not found.\n");
    }
```

MODULE 4: Alert Monitoring

- **Function**: `checkFullBins(int count)`
- **Purpose**: Scans all bins for overflow or cleaning delays.
- **Features**:
    - Sends notifications if any bin is 80% full or hasn't been cleaned in over 48 hours.
    - Displays alerts for administrative action.

CODE SNIPPET



```c
void checkFullBins(int count) {
    int flag = 0;
    printf("\n🔔 Checking Bin Alerts:\n");

    for (int i = 0; i < count; i++) {
        if (bins[i].fillLevel >= FULL_THRESHOLD) {
            sendNotification(bins[i]);
            flag = 1;
        }

        if (bins[i].hoursSinceCleaned > MAX_HOURS_BEFORE_CLEANING) {
            printf("⚠️  Bin ID %d at '%s' not cleaned in over 48 hours!\n", bins[i].binID, bins[i].location);
            flag = 1;
        }
    }

    if (!flag) {
        printf("✅ All bins are within safe levels.\n");
    }
}
```

MODULE 5: Bin Status Display

- **Function**: `displayBins(int count)`
- **Purpose**: Displays a report of all bins' current status.
- **Output**:
  - Bin ID
  - Location
  - Fill Level
  - Hours Since Last Cleaned

CODE SNIPPET

```c
void displayBins(int count) {
    printf("\n🗑️ Bin Status Report:\n");
    for (int i = 0; i < count; i++) {
        printf("Bin ID: %d | Location: %s | Fill Level: %d%% | Hours Since Cleaned: %d\n",
               bins[i].binID, bins[i].location, bins[i].fillLevel, bins[i].hoursSinceCleaned);
    }
}
```

MODULE 6: Sanitation Log

- **Function**: `saveSanitationLog(int count)`
- **Purpose**: Saves the current bin status to a text file (`sanitation_log.txt`).
- **Importance**:
  - Keeps historical data for record-keeping and analysis.

CODE SNIPPET

```c
void saveSanitationLog(int count) {
    FILE *file = fopen("sanitation_log.txt", "w");
    if (file == NULL) {
        printf("❌ Could not open file.\n");
        return;
    }

    for (int i = 0; i < count; i++) {
        fprintf(file, "Bin ID: %d | Location: %s | Fill Level: %d%% | Hours Since Cleaned: %d\n",
                bins[i].binID, bins[i].location, bins[i].fillLevel, bins[i].hoursSinceCleaned);
    }

    fclose(file);
    printf("✅ Sanitation log saved to 'sanitation_log.txt'.\n");
}
```

MODULE 7: Notification System

- **Function**: `sendNotification(WasteBin bin)`
- **Purpose**: Notifies the user when a bin is critically full.
- **Implementation**:
    - Simple console alert with bin details.

CODE SNIPPET

```
159  void sendNotification(WasteBin bin) {
160      printf("🚨 ALERT: Bin ID %d at '%s' is %d%% full. Immediate collection required!\n",
161          bin.binID, bin.location, bin.fillLevel);
162  }
```

Sample output

```
🚨 ALERT: Bin ID 2 at 'Hostel' is 85% full. Immediate collection required!
⚠️ WARNING: Bin ID 4 at 'Admin Block' has not been cleaned in over 48 hours!
✅ Bin updated successfully.
📋 Bin Status Report:
Bin ID: 1 | Location: Library | Fill Level: 40% | Hours Since Cleaned: 12
Bin ID: 2 | Location: Hostel | Fill Level: 85% | Hours Since Cleaned: 10
Bin ID: 3 | Location: Canteen | Fill Level: 30% | Hours Since Cleaned: 5
Bin ID: 4 | Location: Admin Block | Fill Level: 50% | Hours Since Cleaned: 50
Bin ID: 5 | Location: Playground | Fill Level: 20% | Hours Since Cleaned: 8
```

Conclusion

The **Smart Waste Management System** effectively demonstrates how technology can be used to improve the efficiency and hygiene of public spaces, particularly in a campus environment. Through this system, we achieved real-time monitoring of waste bin fill levels and sanitation schedules. The project emphasizes the importance of timely waste disposal and maintenance, helping prevent overflow, unpleasant conditions, and health hazards.

Future scopes

The current Smart Waste Management System is a functional prototype based on manual inputs. However, there is vast potential for enhancement and real-world deployment. The future scope of this project includes:

1. Integration with IoT Devices

- Use smart sensors to automatically detect bin fill levels and cleaning status.
- Enable real-time data transmission using Wi-Fi, LoRa, or GSM modules.

---

2. Mobile and Web Applications

- Develop a user-friendly app for municipal workers and administrators.
- Allow remote access to bin status, alerts, and sanitation logs.

---

3. AI-based Route Optimization

- Use AI algorithms to optimize garbage collection routes.
- Save fuel, reduce labour, and improve response times to full bins.

---

4. Data Analytics and Reporting

- Generate trends and graphs from bin usage data.

- Predict peak usage times and plan bin placements more efficiently.

---

5. Public Participation Module

- Introduce a feedback system where users can report overflowing bins.
- Encourage responsible behavior and community involvement.

---

6. Solar-Powered Smart Bins

- Install bins with solar panels to power sensors and transmitters.
- Make the system more energy-efficient and eco-friendly.

---

7. Integration with Municipal Systems

- Connect the system to city municipal dashboards.
- Enable real-time monitoring and centralized control of city-wide waste management.

---

8. Multilingual and Voice Interface

- Add support for multiple languages and voice commands.
- Increase accessibility for workers with limited literacy.

Platform Used

Visual Studio Code

References

Geeks for Geeks, C in depth (Book)