### ONE PERFORMANCE INNOVATIVE TASK

**“Enhancing Local Community Waste Management from Traditional Methods to Digital Monitoring System”**

**

In Partial Fulfillment of the Requirements in Department of Information Technology

University of Science and Technology of Southern Philippines Oroquieta City Campus

**Aira Monique C. Precillas**

**Sheine Lacay**

**John Michael M. Clerigo**

**Ma. Julliana P. Cimafranca**

May 2025

### PROBLEM ANALYSIS REPORT

### Introduction

Proper waste management is essential to maintain a clean, safe & healthy community. Even local areas like barangays around Oroquieta City, the practices of waste monitoring are still done in the usual way like that is, pen and paper, verbal reports and manual checks. These traditional methods are slow and hard to manage, and often cause problems such as missed garbage report, unreported waste piles, and poor communication between the people living and the local government.

As the population continues to grow, the amount of waste also increases. This puts more pressure on barangay officials and sanitation workers, making their jobs more difficult. With limited tools and time, it's easy for them to get overwhelmed and miss important details about waste collection schedules, routes, or problems in the area.

Simultaneously, most residents in the community do not understand how waste is managed or how they can report waste management issues. Usually, they depend on photos plastered on the walls of barangay halls or whispers of mouth, which are often unreliable or out-of-date. In today’s digital age, there are better ways to manage and monitor waste. By moving from manual processes to a digital waste monitoring system, communities can solve many of these problems. A digital system can help keep track of waste collection more accurately, allow residents to report issues using their phones, and make the whole process faster and more organized.

This report will look closely at the current waste monitoring problems in the community, explain why they matter, and suggest how digital tools can improve the system for everyone.

### Community Problem Description

Waste management in the community faces many challenges that affect cleanliness and efficiency. There is no proper system to track, report, or manage waste properly. Below are some of the problems the community is currently experiencing.

1. Traditional and Manual Waste Monitoring. Current waste tracking is primarily paper-based, making it prone to data loss, human error, and limited accessibility. Without a centralized system, it becomes difficult to track waste collection schedules, frequency, volume, and area coverage efficiently.
2. Lack of Real-Time Data and Reporting. There is no structured way to gather or share real-time data regarding uncollected garbage, or areas with frequent waste overflow. Residents have no immediate platform to report issues, which delays action.
3. Limited Community Awareness and Involvement. Most community members are not regularly informed or engaged in local waste management efforts. Traditional bulletin boards and word of mouth communication are no longer effective for encouraging proper waste disposal and segregation practices.
4. Overworked Barangay Personnel. Barangay officials and sanitation workers face increasing pressure as population and waste volumes grow. Manual methods make it harder to efficiently allocate resources, respond to concerns, and enforce local ordinances.
5. Poor Waste Sorting and Rule Checking. Because there is no proper system, it’s hard to check if people are sorting their waste correctly. It’s also difficult to know who is following the rules and who is not. This makes it harder to teach and remind people about proper waste disposal.

### Relevance and Impact Assessment

### Proper waste management is needed to ensure public health, environmental protection, and community welfare. With increasing populations and worsening environmental conditions, traditional methods of waste monitoring often fall short. A digital waste monitoring solution addresses these gaps by improving transparency, accountability, and efficiency among stakeholders. The following are the various ways it has a comprehensive impact.

1. Environmental and Public Health Significance. Improper waste monitoring can lead to clogged drainage systems, pest infestations, and health hazards. A digital solution can improve sanitation and help prevent disease outbreaks.
2. Empowering Citizens through Digital Participation. A mobile-accessible waste monitoring platform can empower community members to report issues, access schedules, and stay updated, fostering a culture of shared responsibility.
3. Efficiency for Local Government Units (LGUs). Digital records can provide valuable data analytics to help LGUs monitor compliance, optimize routes, and allocate manpower more effectively.
4. Disaster Preparedness and Waste Surge Management. During calamities, unmonitored waste can pile up quickly. A digital system helps track surges and respond proactively to emergencies.

### Conclusion and Recommendations

The old way of checking and managing waste doesn’t work well anymore because more people live in the community now and there are more environmental problems. When waste is not handled properly, it can cause blocked drains, pests, and health problems. A digital waste monitoring system can help make the process easier and faster. It helps LGUs keep track of waste collection, plan better routes, and respond to problems quickly. It also lets people in the community take part by reporting issues and staying informed.

Recommendations:

1. Conduct a needs assessment survey among stakeholders.
2. Develop or adopt a Waste Monitoring Management System.
3. Pilot the system in the most affected barangay.
4. Train residents of the barangay and local government units for optimal use.

### TECHNOLOGICAL SOLUTION PROPOSAL

**"DigiWaste Console: A Python-Based Digital Waste Monitoring System for Local Community"**

1. **Overview of the Proposed Solution**

**DigiWaste Console** is a Python-based console application designed to provide a basic yet functional system for Monitoring Waste Management at Barangay’s in Oroquieta City. It is intended to serve as a lightweight tool for tracking, report, and manage waste properly.

While limited to a text-based interface, the system will support key functionalities such as:

* Adding, Viewing, and deleting Waste
* Viewing and Updating scheduled Waste disposal in a list or by date
* Managing participant lists
* Tracking attendance manually
* Collecting basic post-event feedback

This console-based solution is especially suitable for educational use, particularly as One Performance Innovative Task (ONE-PIT) project for first year BSIT students, while still addressing the community’s real needs for a monitoring waste management system.

# Algorithms and Data Structures Used

Although the **DigiWaste Console** application is relatively simple compared to a full-stack web or mobile system, it incorporates fundamental programming logic and data structures to simulate real-world waste management operations.

### Core Features and Algorithms

* + 1. **Waste Report Submission**
       1. Residents enter their name, location, and description of the issue.
       2. The system adds the current date and time automatically.
       3. Reports are saved in a file (Reports\_File.txt) using a structured format.
       4. Each report is added as a new line for future tracking.

### Waste Collection Logging (LGU)

* + - 1. LGUs enter the resident’s name, area, type of waste (Biodegradable, Non - Biodegradable, Reusable), and weight.
      2. The system checks if the input is valid (waste type and weight).
      3. Entries are saved in Waste\_Collection\_File.txt with date and time.
      4. A summary function adds up total waste and shows the amount per type.

### Schedule Management

* + - 1. Collection schedules are stored in a dictionary with days as keys.
      2. Users can update the time for any day.
      3. The schedule is displayed using loops for clean formatting.

### Waste Report Review and Status Update

* + - 1. LGUs can view all reports in a table format.
      2. Status (e.g., “Resolved”) can be updated by editing the selected report line.

### Notice Posting and Viewing

### LGUs can post announcements, saved with time and date in Notice\_File.txt.

### Each notice is stored on a new line.

* + - 1. Residents can view all posted notices in a list format.

### Key Data Structures

* + 1. **Waste Report (String Record in Text File)**

Each report is stored as a line in Reports\_File.txt with this format:

**Format:**

"time|name|location|issue|status"

## Waste Collection Entry (String Record in Text File)

## Logged by the LGU in Waste\_Collection\_File.txt:

## Format:

## "time|resident\_name|area|waste\_type|weight"

## Collection Schedule (Dictionary)

schedules = {

"Monday": "6:00am - 9:00am",

"Tuesday": "6:00am - 9:00am",

...

}

## Notices (String Record in Text File)

## 

## Format:

## "time|announcement"

1. **Minimum Viable Product (MVP) prototype** of the **DigiWaste Console Application** written in Python.

Core functionalities:

* Submit Waste Report
* View and Update Waste Collection Logs
* Post and View Announcements
* Manage Collection Schedules
* View Report History and Update Status
* Generate Total Summary of Collected Waste

#Main\_Menu.py

import os

import msvcrt as m

import LGU\_MENU

import Residents\_Menu

def clear():

os.system('cls')

def border():

for \_ in range(178):

print('-',end='')

def border2():

clear()

border()

print("")

print('{:^170}'.format(" │ │ │┌──│ ┌──┐┌──┐┌──┬──┐┌── ──┬──┌──┐ │ │ │┌──┐┌── ──┬──┌── ┌──┬──┐┌──┐┌──┬┌──┐┌── ┌── ┌──┬──┐┌── ┌──┬ ──┬──"))

print('{:^170}'.format(" │ │ │├──│ │ │ ││ │ │├── │ │ │ │ │ │├──┤└──┐ │ ├── │ │ │├──┤│ │├──┤│ ─┐├── │ │ │├── │ │ │ "))

print('{:^170}'.format(" └─┴─┘└──└─┘└──┘└──┘┴ ┴ ┴└── ┴ └──┘ └─┴─┘┴ ┴ ──┘ ┴ └── ┴ ┴ ┴┴ ┴┴ ┴┴ ┴└──┘└── ┴ ┴ ┴└── ┴ ┴ ┴ "))

print('{:^170}'.format("┌── ┬ ┬ ┌── ──┬──┌── ┌──┬──┐"))

print('{:^170}'.format("└──┐└─┬─┘ └──┐ │ ├── │ │ │"))

print('{:^170}'.format(" ──┘ ┴ ──┘ ┴ └── ┴ ┴ ┴"))

border()

print("\n\n\n\n\n\n")

def select1():

while True:

clear()

border2()

print('{:^170}'.format("─" \* 50))

print("\n")

print('{:^170}'.format("│[A] Menu|"))

print(" ")

print('{:^170}'.format("│[B] Exit|"))

print("\n")

print('{:^170}'.format("─" \* 50))

print("")

select = input('{:>80}'.format("Select: ")).upper()

if select == 'A':

main()

elif select == 'B':

print("Exiting...")

m.getch()

clear()

exit()

else:

print("Invalid Choice...")

m.getch()

def main():

while True:

clear()

border2()

print('{:^170}'.format("Waste Management System"))

print('{:^170}'.format("─" \* 50))

print("\n")

print('{:>95}'.format("[A]. Local Government Unit (LGU)"))

print('{:>77}'.format("[B]. Residents"))

print('{:>72}'.format("[C]. Exit"))

print("\n\n")

print('{:^170}'.format("─" \* 50))

print("\n\n")

choice = input('{:>80}'.format("Enter your choice: ")).upper()

if choice == 'A':

LGU\_MENU.Menu()

m.getch()

elif choice == 'B':

Residents\_Menu.Menu()

m.getch()

elif choice == 'C':

print("Exiting...")

m.getch()

select1()

else:

print("Invalid choice. Please try again.")

m.getch()

if \_\_name\_\_ == "\_\_main\_\_":

select1()

#LGU\_MENU.py

import os

import msvcrt as m

import Main\_menu

import Waste\_Collect\_Funct

import Notice

import Show\_Sched

import Residents\_Menu

Reports\_File = "Reports\_File.txt"

# Clear screen

def clear():

os.system('cls')

# line (short)

def border():

for \_ in range(50):

print('-',end='')

# line (long)

def border1():

for \_ in range(178):

print('-',end='')

#l ogo

def border2():

clear()

border1()

print("")

print('{:^170}'.format(" │ │ │┌──│ ┌──┐┌──┐┌──┬──┐┌── ──┬──┌──┐ │ │ │┌──┐┌── ──┬──┌── ┌──┬──┐┌──┐┌──┬┌──┐┌── ┌── ┌──┬──┐┌── ┌──┬ ──┬──"))

print('{:^170}'.format(" │ │ │├──│ │ │ ││ │ │├── │ │ │ │ │ │├──┤└──┐ │ ├── │ │ │├──┤│ │├──┤│ ─┐├── │ │ │├── │ │ │ "))

print('{:^170}'.format(" └─┴─┘└──└─┘└──┘└──┘┴ ┴ ┴└── ┴ └──┘ └─┴─┘┴ ┴ ──┘ ┴ └── ┴ ┴ ┴┴ ┴┴ ┴┴ ┴└──┘└── ┴ ┴ ┴└── ┴ ┴ ┴ "))

print('{:^170}'.format("┌── ┬ ┬ ┌── ──┬──┌── ┌──┬──┐"))

print('{:^170}'.format("└──┐└─┬─┘ └──┐ │ ├── │ │ │"))

print('{:^170}'.format(" ──┘ ┴ ──┘ ┴ └── ┴ ┴ ┴"))

border1()

# Waste\_Collection Menu

def Waste\_Collection\_Menu():

while True:

clear()

Main\_menu.border2()

print('{:^170}'.format("[LGU] Waste Collection"))

print('{:^170}'.format("─" \* 50))

print("\n")

print('{:>88}'.format("[A] Collect Waste"))

print('{:>95}'.format("[B] View Collection List"))

print('{:>92}'.format("[C] Delete Collection"))

print('{:>92}'.format("[D] Back to Main Menu"))

print("\n\n")

print('{:^170}'.format("─" \* 50))

print("\n\n")

selection = input('{:>90}'.format("Enter your choice: ")).upper()

if selection == 'A':

clear()

Waste\_Collect\_Funct.Collect\_Waste()

m.getch()

elif selection == 'B':

clear()

Waste\_Collect\_Funct.View\_Collection\_List()

m.getch()

elif selection == 'C':

clear()

Waste\_Collect\_Funct.delete\_collection()

m.getch()

elif selection == 'D':

print("\nGoing back to main menu...")

m.getch()

break

else:

print("\nInvalid choice. Please try again.")

m.getch()

# Notice\_Menu

def Notice\_Menu():

while True:

Main\_menu.border2()

print('{:^170}'.format("Notice Post:"))

print('{:^170}'.format("─" \* 50))

print("\n")

print('{:>86}'.format("[A] Post Notice"))

print('{:>92}'.format("[B] View Notice Posts"))

print('{:>93}'.format("[C] Delete Notice Post"))

print('{:>92}'.format("[D] Back to Main Menu"))

print("\n\n")

print('{:^170}'.format("─" \* 50))

print("\n\n")

selection = input('{:>80}'.format("Enter your choice: ")).upper()

if selection == 'A':

clear()

Notice.Post\_Notice()

elif selection == 'B':

clear()

Notice.View\_Notice()

elif selection == 'C':

clear()

Notice.delete\_Notice()

m.getch()

elif selection == 'D':

print("Going back to main menu...")

m.getch()

break

else:

print("\nInvalid choice. Please try again.")

m.getch()

# Schedule\_menu

def schedule():

while True:

clear()

Main\_menu.border2()

print('{:^170}'.format("[LGU] Waste Collection Schedule"))

print('{:^170}'.format("─" \* 50))

print("\n ")

print('{:>90}'.format("[A] View Schedule"))

print('{:>92}'.format("[B] Update Schedule"))

print('{:>94}'.format("[C] Back to Main Menu"))

print("\n\n")

print('{:^170}'.format("─" \* 50))

print("\n")

selection = input('{:>70}'.format("Enter your choice: ")).upper()

if selection == 'A':

clear()

Show\_Sched.View\_Schedule()

m.getch()

elif selection == 'B':

clear()

Show\_Sched.update\_schedule()

m.getch()

elif selection == 'C':

print("\n")

print('{:^170}'.format("Going back to main menu..."))

m.getch()

break

else:

print("")

print('{:^170}'.format("Invalid choice. Please try again."))

m.getch()

# Waste report

def Waste\_report():

Residents\_Menu.View\_Report()

print("─" \* 178)

print("\n\n")

print('{:>40}'.format("Select:"))

print(f"{'┌':>56}{'─'\*61}┐")

print(f"{'│':>56} {'':<10}{'[A] Update Status':<24} {'[B] Exit':<25}│")

print(f"{'└':>56}{'─'\*61}┘")

select = input('{:>47}'.format("Enter Choice: ")).upper()

if select == 'A':

clear()

Update\_Status\_Report()

Waste\_report()

m.getch()

elif select == 'B':

print("\n")

print('{:^170}'.format("Exiting..."))

return

m.getch()

else:

print("\n")

print("Invalid Choice../")

m.getch()

# Change the Status report of the resident

def Update\_Status\_Report():

Residents\_Menu.View\_Report()

if os.path.exists(Reports\_File):

with open(Reports\_File, 'r') as file:

reports = file.readlines()

print("─" \* 178)

print("\n\n")

print('{:^170}'.format("Update Report Status"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

try:

report\_number = int(input('{:>60}'.format("Enter the report number to update (0 to exit): "))) - 1

if report\_number == -1:

print('{:^170}'.format("Exiting update operation..."))

return

elif 0 <= report\_number < len(reports):

Time, Name\_Reporter, Location, Issue, Status = reports[report\_number].strip().split("|")

new\_status = input('{:>67}'.format("Enter new status (e.g.,[R] Resolved, [I] In Progress): ")).upper()

if new\_status == "R":

new\_status = "Resolved"

elif new\_status == "I":

new\_status = "In Progress"

else:

print("\n\n")

print('{:x^170}'.format("Invalid status. Please enter [R] for Resolved or [I] for In Progress."))

m.getch()

return

reports[report\_number] = f"{Time}|{Name\_Reporter}|{Location}|{Issue}|{new\_status}\n"

with open(Reports\_File, 'w') as file:

file.writelines(reports)

print("\n")

print('{:^170}'.format("Report status updated successfully!"))

m.getch()

else:

print("\n")

print('{:^170}'.format("Invalid report number."))

m.getch()

except ValueError:

print("\n")

print('{:^170}'.format("Invalid input. Please enter a valid number."))

m.getch()

# Main Menu LGU

def Menu():

while True:

os.system('cls')

Main\_menu.border2()

print('{:^170}'.format("[LGU] Waste - Management System"))

print('{:^170}'.format("─" \* 50))

print("\n")

print('{:>92}'.format("[A] Waste Collection"))

print('{:>82}'.format("[B] Notice"))

print('{:>84}'.format("[C] Schedule"))

print('{:>93}'.format("[D] View Waste Report"))

print('{:>82}'.format("[E] Logout"))

print("\n\n")

print('{:^170}'.format("─" \* 50))

print("\n\n")

choice = input('{:>80}'.format("Enter Choice: ")).upper()

if choice == 'A':

clear()

Waste\_Collection\_Menu()

m.getch()

elif choice == 'B':

clear()

Notice\_Menu()

m.getch()

elif choice == 'C':

clear()

schedule()

m.getch()

elif choice == 'D':

clear()

Waste\_report()

m.getch()

elif choice == 'E':

print("Logging out...")

m.getch()

Main\_menu.main()

else:

print("Invalid choice. Please try again.")

m.getch()

#Resident\_Menu.py

import os

import msvcrt as m

import Main\_menu

import LGU\_MENU

import datetime

import Show\_Sched

import Notice

import Waste\_Collect\_Funct

Reports\_File = "Reports\_File.txt"

#clear screen

def clear():

clear = os.system('cls')

#line

def border():

for \_ in range(120):

print('-',end='')

# input waste Report or issue

def Waste\_Report():

Main\_menu.border2()

print('{:^170}'.format("Waste - Report"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

print('{:^170}'.format("─" \* 120))

print("\n")

Name\_Reporter = input('{:>97}'.format("Enter your name : "))

Location = input('{:>97}'.format(f"Enter the location of the problem : "))

Issue = input('{:>97}'.format(f"Describe the issue (e.g., Overflow, No Collection) : "))

Time = datetime.datetime.now().strftime("%d-%m-%Y, %H:%M:%S")

Status = "In Progress"

print('{:>117}'.format(f"Date & Time : {Time}"))

print('{:>108}'.format(f"Status : {Status}"))

Report = f"{Time}|{Name\_Reporter}|{Location}|{Issue}|{Status}\n"

with open(Reports\_File, 'a') as file:

file.write(Report)

print("\n\n")

print('{:^170}'.format("─" \* 120))

print("\n\n")

print('{:^170}'.format("Report submitted successfully!"))

# delete a waste report

def delete\_report():

Main\_menu.border2()

print('{:^170}'.format("Waste Report History"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

print(f"{'':<12}{'No.':<5} {'TIME':<23} {'NAME':<32} {'AREA':<35} {'ISSUE':<45} {'STATUS'}")

print(f"─" \* 178)

if not os.path.exists(Reports\_File):

print("\n\n\n\n\n")

print('{:^170}'.format("No reports found."))

print("\n\n\n\n")

print("─" \* 178)

m.getch()

return

with open(Reports\_File, 'r') as file:

reports = file.readlines()

if not reports:

print("\n\n\n\n\n")

print('{:^170}'.format("No reports found."))

print("\n\n\n\n")

print("─" \* 178)

m.getch()

return

for i, line in enumerate(reports):

Time, Name\_Reporter, Location, Issue, Status = line.strip().split("|")

print(f"{'':<12} {i + 1:<5} {Time:<23} {Name\_Reporter:<32} {Location:<35} {Issue:<45} {Status:}")

print("\n\n")

try:

report\_number = int(input('{:>60}'.format("\nEnter the report number to delete (0 to exit): "))) - 1

print("\n\n")

if report\_number == -1:

print('{:^170}'.format("Exiting delete operation..."))

return

elif 0 <= report\_number < len(reports):

del reports[report\_number]

with open(Reports\_File, 'w') as file:

file.writelines(reports)

print('{:^170}'.format("Report deleted successfully!"))

else:

print('{:^170}'.format("Invalid report number."))

except ValueError:

print('{:^170}'.format("Invalid input. Please enter a valid number."))

print("")

print('{:^170}'.format("Press enter key to return to the menu..."))

m.getch()

# View history report or issue

def View\_Report():

Main\_menu.border2()

print('{:^170}'.format("Waste Report History"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

print(f"{'':<12}{'No.':<5} {'TIME':<23} {'NAME':<32} {'AREA':<35} {'ISSUE':<45} {'STATUS'}")

print(f"─" \* 178)

if not os.path.exists(Reports\_File):

print("\n\n\n\n\n")

print('{:^170}'.format("No reports found."))

print("\n\n\n\n")

print("─" \* 178)

m.getch()

return

with open(Reports\_File, 'r') as file:

reports = file.readlines()

if not reports:

print("\n\n\n\n\n")

print('{:^170}'.format("No reports found."))

print("\n\n\n\n")

print("─" \* 178)

m.getch()

return

for i, line in enumerate(reports):

Time, Name\_Reporter, Location, Issue, Status = line.strip().split("|")

print(f"{'':<12} {i + 1:<5} {Time:<23} {Name\_Reporter:<32} {Location:<35} {Issue:<45} {Status:}")

print("\n")

print('{:^170}'.format("Press enter key to return to the menu..."))

m.getch()

# Waste report Menu

def Waste\_Report\_Menu():

while True:

clear()

Main\_menu.border2()

print('{:^170}'.format("Waste - Report"))

print('{:^170}'.format("─" \* 50))

print("\n")

print('{:>92}'.format("[A] Report an issue"))

print('{:>89}'.format("[B] View reports"))

print('{:>92}'.format("[C] Delete a report"))

print('{:>94}'.format("[D] Back to main menu"))

print("\n\n")

print('{:^170}'.format("─" \* 50))

print("\n\n")

selection = input('{:>80}'.format("Enter your choice: ")).upper()

if selection == 'A':

clear()

Waste\_Report()

m.getch()

elif selection == 'B':

clear()

View\_Report()

m.getch()

elif selection == 'C':

clear()

delete\_report()

m.getch()

elif selection == 'D':

print("Going back to main menu...")

m.getch()

break

else:

print("Invalid choice. Please try again.")

m.getch()

# Residents Main menu

def Menu():

while True:

os.system('cls')

Main\_menu.border2()

print('{:^170}'.format("Waste - Management System"))

print('{:^170}'.format("─" \* 50))

print("\n")

print('{:>89}'.format("[A] Waste Report"))

print('{:>90}'.format("[B] View Schedule"))

print('{:>94}'.format("[C] View Announcement"))

print('{:>100}'.format("[D] View Collection History"))

print('{:>83}'.format("[E] Logout"))

print("\n\n")

print('{:^170}'.format("─" \* 50))

print("\n\n")

choice = input('{:>80}'.format("Enter Choice: ")).upper()

if choice == 'A':

clear()

Waste\_Report\_Menu()

m.getch()

elif choice == 'B':

clear()

Show\_Sched.View\_Schedule()

m.getch()

elif choice == 'C':

clear()

Notice.View\_Notice()

m.getch()

elif choice == 'D':

clear()

Waste\_Collect\_Funct.View\_Collection\_List()

m.getch()

elif choice == 'E':

print("Logging out...")

m.getch()

Main\_menu.main()

else:

print("Invalid choice. Please try again.")

m.getch()

#Notice.py

import os

import msvcrt as m

import Main\_menu

import LGU\_MENU

import datetime

# Post notice

def Post\_Notice():

LGU\_MENU.border2()

print("\n")

print('{:^170}'.format("Post Notice:"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

print('{:^165}'.format("─" \* 120))

print("\n")

announcement = input('{:>46}'.format("Enter the Notice: "))

Time = datetime.datetime.now().strftime("%d-%m-%Y, %H:%M:%S")

with open("Notice\_File.txt", 'a') as file:

file.write(f"{Time}|{announcement}\n")

print("\n\n")

print('{:^165}'.format("─" \* 120))

print("\n\n")

print('{:>60}'.format("Notice posted successfully!"))

print("\n")

print('{:>67}'.format("Press enter key to return to the menu..."))

m.getch()

#View Notice

def View\_Notice():

LGU\_MENU.border2()

print("\n\n")

print('{:^170}'.format("[LGU] Notice Board"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

print(f"{'┌':>15}{"─" \* 147}┐")

print(f"{'':<15} {'No.':<5} {'TIME':<25} {' NOTICE ':<100}")

print(f"{'├':>15}{"─" \* 147}┤")

if not os.path.exists("Notice\_File.txt"):

print("\n\n\n\n")

print('{:^170}'.format("No Notice found."))

print('{:^170}'.format("Press enter key to return to the menu..."))

print("\n\n\n\n")

print(f"{'├':>15}{"─" \* 147}┤")

m.getch()

return

with open("Notice\_File.txt", 'r') as file:

notices = file.readlines()

if not notices:

print("\n\n\n\n")

print('{:^170}'.format("No Notice found."))

print('{:^170}'.format("Press enter key to return to the menu..."))

print("\n\n\n\n")

print(f"{'├':>15}{"─" \* 147}┤")

m.getch()

return

for i, line in enumerate(notices):

Time, Notice = line.strip().split("|")

print("")

print(f"{'':>15} {i + 1:<5} {Time:<25} {Notice:<100}")

print("")

print(f"{'├':>15}{"─" \* 147}┤")

print("\n\n")

print('{:^170}'.format("Press enter key to return to the menu..."))

m.getch()

#delete Notice

def delete\_Notice():

LGU\_MENU.border2()

print("\n\n")

print('{:^170}'.format("[LGU] Delete Notice"))

print('{:^170}'.format("─" \* 59))

print("\n\n")

View\_Notice()

print("\n")

if os.path.exists("Notice\_File.txt"):

with open("Notice\_File.txt", 'r') as file:

notices = file.readlines()

try:

print("\n")

Notice\_number = int(input('{:>60}'.format("\nEnter the Notice number to delete (0 to exit): "))) - 1

if Notice\_number == -1:

print('{:^170}'.format("Exiting delete operation..."))

return

elif 0 <= Notice\_number < len(notices):

del notices[Notice\_number]

with open("Notice\_File.txt", 'w') as file:

file.writelines(notices)

print("\n")

print('{:^170}'.format("Notice deleted successfully!"))

m.getch()

else:

print("\n")

print('{:^170}'.format("Invalid Notice number."))

m.getch()

except ValueError:

print("\n")

print('{:^170}'.format("Invalid input. Please enter a valid number."))

m.getch()

print('{:^170}'.format("Press enter key to return to the menu..."))

#Show\_Sched.py

import os

import LGU\_MENU

import msvcrt as m

# defualt sched

schedules = {"Sunday": "6:00am - 9:00am",

"Monday": "6:00am - 9:00am",

"Tuesday": "6:00am - 9:00am",

"Wednesday": "6:00am - 9:00am",

"Thursday": "6:00am - 9:00am",

"Friday": "6:00am - 9:00am",

"Saturday": "6:00am - 9:00am"}

# Shoe Schedule

def View\_Schedule():

LGU\_MENU.border2()

print("\n\n\n")

print('{:^170}'.format("Waste Collection Schedule"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

print(f"{'┌':>62}{'─'\*46}┐" )

print(f"{'':>62} {'DAY':<27} {'TIME':>11} ")

print(f"{'├':>62}{'─'\*46}┤")

for day, time in schedules.items():

print(f"{"":>62} {day:<24} : {time:>11} ")

print(f"{'├':>62}{'─'\*46}┤")

print('{:^170}'.format("─" \* 50))

print("\n\n")

print('{:^170}'.format("Press enter key to return to the menu..."))

m.getch()

# Update the schedule

def update\_schedule():

LGU\_MENU.border2()

print("\n\n\n")

print('{:^170}'.format("Update Schedule:"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

print('{:^68}'.format("Current Schedule:"))

View\_Schedule()

print("\n")

day = input('{:>62}'.format("Enter the day to update (e.g., Monday): ")).capitalize()

if day in schedules:

new\_time = input('{:>62}'.format(f"Enter new time for {day} (e.g., 6:00am - 9:00am): "))

schedules[day] = new\_time

print("\n")

print('{:^170}'.format(f"Schedule for {day} updated successfully!"))

else:

print("\n\n")

print('{:^170}'.format("Invalid day. Please try again."))

print('{:^170}'.format("Press enter key to return to the menu..."))

m.getch()

#Waste\_Collect\_Funct.py

import os

import msvcrt as m

import Main\_menu

import datetime

import LGU\_MENU

# Collect\_Waste function

def Collect\_Waste():

    Main\_menu.border2()

    print('{:^60}'.format("Collecting waste:"))

    print("\n\n")

    print('{:^165}'.format("─" \* 121))

    print("\n")

    Name = input('{:>117}'.format ("Enter Name of the Resident                                                                 : "))

    Area = input ('{:>117}'.format("Enter Area                                                                                 : "))

    Waste\_Type = input('{:>117}'.format("Enter Type of Waste (e.g., [B] Biodegradable, [N] Non-Biodegradable, [R] Reusable)         : ")).upper()

    if Waste\_Type == "B":

        Waste\_Type = "Biodegradable"

    elif Waste\_Type == "N":

        Waste\_Type = "Non-Biodegradable"

    elif Waste\_Type == "R":

        Waste\_Type = "Reusable"

    else:

        print("\n\n")

        print('{:^165}'.format("─" \* 121))

        print("\n\n")

        print('{:^130}'.format("Invalid waste type. Please enter [B], [N], or [R]."))

        m.getch()

        return

    try:

        Waste\_Weight = float(input('{:>117}'.format("Enter Weight of Waste (in kg)                                                              : ")))

        print("\n\n")

        print('{:^165}'.format("─" \* 121))

    except ValueError:

        print("\n\n")

        print('{:^130}'.format("Invalid weight. Please enter a numeric value."))

        return

    Time = datetime.datetime.now().strftime("%d-%m-%Y, %H:%M:%S")

    with open("Waste\_Collection\_File.txt", 'a') as file:

        file.write(f"{Time}|{Name}|{Area}|{Waste\_Type}|{Waste\_Weight}\n")

    print("\n\n\n\n")

    print('{:^130}'.format("Waste collection recorded successfully!"))

    print('{:^130}'.format("Press enter key to return to the menu..."))

    m.getch()

# View Waste that been collected

def View\_Collection\_List():

LGU\_MENU.border2()

print("\n")

print('{:^170}'.format("[LGU] Waste Collection List"))

print('{:^170}'.format("─" \* 50))

print("\n\n")

print(f"{'┌':>15}{"─" \* 147}┐")

print(f"{'':<20}{'No.':<5} {'TIME':<25} {'NAME':<40} {'AREA':<40} {'TYPE':<15} {'WEIGHT (kg)':<10}")

print(f"{'├':>15}{"─" \* 147}┤")

# Check if the file exists and is not empty

if not os.path.exists("Waste\_Collection\_File.txt"):

print("\n\n\n\n\n")

print('{:^170}'.format("No waste collection records found."))

print("")

print('{:^170}'.format("Press enter key to return to the menu..."))

print("\n\n\n\n")

print(f"{'├':>15}{"─" \* 147}┤")

m.getch()

return

# if the file exist Read the file and display the contents

with open("Waste\_Collection\_File.txt", 'r') as file:

collections = file.readlines()

# if file exist but empty

if not collections:

print("\n\n\n\n\n")

print("\n\n")

print('{:^170}'.format("No waste collection records found."))

print("")

print('{:^170}'.format("Press enter key to return to the menu..."))

print("\n\n\n\n")

print(f"{'├':>15}{"─" \* 147}┤")

m.getch()

return

# calucate the total weight and waste summary

total\_weight = 0

waste\_summary = {}

for i, line in enumerate(collections):

Time, Name, Area, Waste\_Type, Waste\_Weights = line.strip().split("|")

Waste\_Weight = float(Waste\_Weights)

total\_weight += Waste\_Weight

if Waste\_Type in waste\_summary:

waste\_summary[Waste\_Type] += Waste\_Weight

else:

waste\_summary[Waste\_Type] = Waste\_Weight

print(f"{'':<20}{i + 1:<5} {Time:<25} {Name:<40} {Area:<40} {Waste\_Type:<15} {Waste\_Weight:<10.2f}")

print(f"{'├':>15}{"─" \* 147}┤")

# Print the summary of the waste collection

print("\n\n")

print('{:^50}'.format("Summary:"))

print('{:>62}'.format(f"Total Waste Collected: {total\_weight:.2f} kg"))

print("")

print('{:^70}'.format("Waste Types Breakdown:"))

for waste\_type, weight in waste\_summary.items():

print(f" {"":<30} - {waste\_type:<15}: {weight:.2f} kg")

print("\n")

print('{:^170}'.format("Press enter key to return to the menu..."))

m.getch()

# Delete Waste Collection

def delete\_collection():

LGU\_MENU.clear()

LGU\_MENU.border2()

print("\n")

print('{:^170}'.format("[LGU] Delete Waste Collection List"))

print('{:^170}'.format("─" \* 53))

print("\n\n")

# Call the function

View\_Collection\_List()

print("\n")

if os.path.exists("Waste\_Collection\_File.txt"):

with open("Waste\_Collection\_File.txt", 'r') as file:

collections = file.readlines()

try:

collection\_number = int(input('{:>60}'.format("Enter the collection number to delete (0 to exit): "))) - 1

if collection\_number == -1:

print('{:^170}'.format("Exiting delete operation..."))

return

elif 0 <= collection\_number < len(collections):

del collections[collection\_number]

with open("Waste\_Collection\_File.txt", 'w') as file:

file.writelines(collections)

print("\n")

print('{:^170}'.format("Collection deleted successfully!"))

m.getch()

else:

print("\n")

print('{:^170}'.format("Invalid collection number."))

m.getch()

except ValueError:

print("\n")

print('{:^170}'.format("Invalid input. Please enter a valid number."))

m.getch()

# Implementation Feasibility

### Technical Feasibility

* **Programming Language**: Python 3.x
* **Environment**: Runs on any system with a Python interpreter (Windows, MacOS, Linux).
* **Libraries:** No third-party libraries are required. The system relies only on Python's built-in libraries, such as os and datetime.
* **Data Storage:** Uses plain text files (.txt). No need for databases or internet access.
* **Offline Capability**: Fully functional in offline environments, making it suitable for barangays without stable internet access.

### Educational and Operational Feasibility

* **Target Developers:** Ideal for first- to second-year BSIT students who are learning programming logic, file handling, and data structures.
* **Ease of Use** Simple text-based interface with menu navigation, making it accessible to users with minimal technical background.
* **Training and Support** No advanced skills are required. A user manual or basic tutorial can support system use by residents and LGU personnel.
* **Integration**: Although standalone, future versions can be enhanced into a GUI app or web-based system based on the same logic.

### Limitations

* No real-time updates or multi-user access
* Manual input for all data (including waste collection and issue reports)
* Limited data visualization (text-only summaries)
* Reports and summaries are shown only in text only (No Charts)

Despite these limitations, the **DigiWaste Console** provides a practical and educational foundation for digitizing waste monitoring processes at the local level, and can evolve into a more advanced system over time.

# IV. Conclusion

The **DigiWaste Console** is a simple but useful system that helps barangays manage waste better by using a computer program instead of paper. It allows residents to report problems, lets LGU staff keep records of waste collection, and helps organize schedules and announcements.

This project is great for students who are learning Python because it teaches real skills while also solving a real problem in the community. Even though it is basic and runs in the command line, it can be improved later by adding graphics or making it work online. It’s a good first step toward cleaner, more organized waste management using technology.

### STS IMPACT ASSESSMENT REPORT

**Introduction**

As communities and schools move toward using technology, there is a greater need for systems that help manage everyday tasks in a faster and more organized way. In many barangays of Oroquieta City, waste is still managed using manual methods like writing on paper or relying on word of mouth. This causes delays, missed collections, and lack of communication between residents and the local government. The DigiWaste Console is a simple text-based system that helps both residents and LGUs track, report, and manage waste more easily. This report looks at the social, ethical, and environmental effects of using the system and explains why this kind of digital tool is helpful, especially from a Science, Technology, and Society (STS) point of view.

.

### Societal, Ethical, and Environmental Implications

In deploying and implementing this waste management system, it is essential to consider the broader societal, ethical, and environmental implications to ensure that it execute its purpose not only effectively but also responsible and sustainable.

1. Societal Implications
2. Improved Public Health. Digital monitoring can result in more effective waste collection, which lowers the chance of disease transmission and fosters environmental cleanliness.
3. Increased Community Engagement. By teaching locals about appropriate waste management techniques, digital platforms can promote a sense of civic duty.

II. Ethical Implications

* + - 1. Data Privacy. The digital system may collect sensitive data on waste generation patterns, requiring robust data protection measures to ensure resident privacy.
      2. Equitable Access. The system should be designed to accommodate diverse community needs, ensuring equal access to waste management services for all residents.
      3. Transparency and Accountability. Digital monitoring can promote transparency in waste management practices, holding authorities accountable for their actions.

III. Environmental Implications

1. Reduced Waste. Digital monitoring can help identify areas of high waste generation, enabling targeted interventions to reduce waste production.
2. Increased Recycling. The system can optimize waste collection routes and promote recycling practices, reducing the amount of waste sent to landfills.
3. Environmental Sustainability. By reducing waste and promoting sustainable practices, the digital system can contribute to a more environmentally sustainable community.

### Justification of the Technological Solution’s Benefits

The implementation of a Waste Monitoring Management System aligns with both institutional modernization goals and national efforts toward digital transformation. The benefits include:

1. Tasks like reporting problems, viewing schedules, and recording waste collection are easier and quicker using the system.
2. Greater accuracy and fewer errors compared to manual processes.
3. Residents and LGUs can share information more easily through notices and reports.
4. Staff can track data in one place, making it easier to respond to issues and manage schedules.
5. Alignment with ethical standards by ensuring transparency and data security.
6. Residents are more aware and can take part in keeping their area clean by reporting waste problems directly.

In the end, the DigiWaste Console is not just a new computer program—it is a helpful tool that makes the community cleaner, helps people work together, and takes care of the environment.

### Conclusion

From a Science, Technology, and Society (STS) point of view, using the DigiWaste Console is a good way to help the community. It makes waste management easier, safer, and better for the environment. It also helps people work together and take care of their surroundings. This project is not just about using computers it’s about helping the community become cleaner, more organized, and ready for the future.