



Documentation Report
Of Final Project/Exam: DSA for Sustainable Development
Pangkabuhayan System: Food Assistance Management System

Submitted by:
Alba, Jimmy Paul
Camban, Christian James
Doton, Athan Josch
Navarro, Aaron Christian
Santos, Justine

Submitted to:
Ms. Justin Louise R. Neypes

BS Information Technology
NTC_CC105 Data Structure w/Lab

December 13 , 2025

I. INTRODUCTION

1.1 Project Overview & UN SDG Target

The ***Pangkabuhayan System*** is a console-based C++ application designed to streamline food assistance distribution programs managed by local government units (LGUs), specifically at the barangay level. The system addresses the operational challenges faced during ayuda distribution by automating family registration, food pack inventory management, queue organization, and equitable distribution processes.

UN SDG Alignment: SDG 2 - Zero Hunger

This project directly contributes to achieving UN Sustainable Development Goal 2: Zero Hunger, which aims to "end hunger, achieve food security and improved nutrition, and promote sustainable agriculture." By ensuring efficient and fair distribution of food assistance, the system helps vulnerable communities access nutritious food while minimizing waste and administrative errors.

1.2 Problem Statement

Local government units, particularly barangays, face significant challenges in managing food assistance distribution programs:

Current Problem:

- 1. Manual Record-Tracking:** Reliance on paper-based lists leads to illegible handwriting, lost documents, and difficulty in tracking distribution history
- 2. Inefficient Processing:** Manual verification of names and quantities causes long wait times and frustrated beneficiaries
- 3. Data Integrity Issue:** Duplicate registrations allowing multiple claims per family, missing or incorrect family records and inaccurate food pack inventory counts
- 4. Unfair Distribution:** Lack of structured queuing system results in vulnerable groups not being served first
- 5. Resource Mismanagement:** Inability to match available food packs with registered families leads to shortages or excess inventory

Proposed Solution:

The Pangkabuhayan System addresses these challenges through:

1. **Automated Sorting:** Uses Merge Sort for alphabetical family listing and Insertion Sort for food pack inventory organization
2. **Duplicate Prevention:** Implements search algorithms (Linear/Binary Search) to verify family registration before enrollment
3. **Fair Queuing:** Employs Queue data structure with priority handling for vulnerable groups
4. **One-to-One Distribution:** Ensures each family receives exactly one food pack while automatically updating records
5. **Real-time Updates:** Dynamically removes distributed families and deducts food packs from inventory
6. **Data Persistence:** Loads initial data from files and saves updated records for future reference

II. REQUIREMENT & ANALYSIS

2.1 Functional Requirements

ID	Requirements	Description
FR1	Family Registration	The system must allow adding families to the database with validation to prevent duplicates. Family names are automatically sorted alphabetically using sort().
FR2	Food Pack Inventory Management	The system must enable adding food pack types with quantities. Insertion Sort is used to keep food packs alphabetically

		organized.
FR3	Automatic Queue Generation	The system must automatically create a distribution queue from registered families following FIFO principle, with priority for seniors, PWDs, and pregnant women
FR4	Distribution System	The system must distribute food packs using 1:1 matching (one family = one food pack), automatically removing served families and updating inventory in real-time.
FR5	Family Search Functionality	The system must provide search capability to check if a family is already registered using Linear algorithms.
FR6	Data Persistence	The system must load initial data from input files and save distribution results and updated records to output files.
FR7	Display & Reporting	The system must display current family list, food pack inventory, queue status, and distribution

		summary in clear console format.
--	--	----------------------------------

Non-Functional Requirements

ID	Requirement	Metric
NFR1	Performance	The system must process 50+ family records and food pack entries in under 1 second on standard hardware.
NFR2	Robustness	The system must validate all user inputs and handle errors gracefully without crashing. Error messages must be clear and actionable.
NFR3	Maintainability	Code must be modular using object-oriented principles, with proper header files, well-commented sections, and consistent naming conventions.
NFR4	Usability	Console interface must provide clear menu options with numbered choices and formatted output for easy reading.
NFR5	Data Integrity	The system must prevent duplicate

		family entries, ensure non-negative quantities, and maintain consistency between queue and inventory.
--	--	---

2.2 Data Requirements

Input Data Structure:

1. Families Input File

- Minimum 50 family records
- Fields: Family Name, Priority Status (Vulnerable or Regular)
- Format: [Family name], [Priority Status]
- *Example: Santos, Vulnerable*

2. Food Pack Input File

- Multiple food pack types with quantities
- Fields: Food Pack Name, Quantity
- Format: [Pack Name], [Quantity]
- *Example: Rice Pack 5kg, 25pcs.*

Output Data Structure:

1. Distribution Report:

- List of families served with timestamps
- Food pack type distributed to each family
- Remaining inventory status

2. Updated Records

- Families not served yet
- Current food pack inventory

2.3 Complexity Analysis (Big O Notation)

The Pangkabuhayan System uses different algorithms to manage families, food packs and the distribution process.

Data Structure /	Used For	Time	Space
------------------	----------	------	-------

Algorithm		Complexity	Complexity
Insertion Sort	Sorting food pack names alphabetically	$O(n)$	$O(1)$
Vector (Array)	Store families and food packs	$O(1)$ access, $O(n)$ insert/delete	$O(n)$
C++ Built-in sort()	Sorting family names alphabetically	$O(n \log n)$	$O(\log n)$
Linear Search	Finding a family (unsorted list)	$O(n)$	$O(1)$
Queue (FIFO)	Aid distribution order	$O(1)$ enqueue, $O(1)$ dequeue	$O(1)$

- "The system uses Insertion Sort to keep food pack names alphabetically organized. It runs in $O(n^2)$ time but works efficiently because the number of food pack types is small. It also uses only $O(1)$ extra space since it sorts directly within the list.
- The system uses vectors to store families and food packs. A vector provides $O(1)$ access time because elements can be reached directly by index. However, inserting or removing items in the middle takes $O(n)$ since other elements may need to shift positions.
- The system uses C++'s built-in() to sort families alphabetically by name. Its time complexity is $O(n \log n)$ because it uses an efficient hybrid algorithm (Introsort). Space complexity is $O(\log n)$ for the stack in recursive calls
- For searching families, the system uses Linear Search. Linear Search has a time complexity of $O(n)$ because it checks each family one by one.

- The Queue structure follows the FIFO rule. Adding and removing a family from the queue both have a time complexity of $O(1)$, making it fast and efficient. Overall, the system is able to handle 50 or more records smoothly while meeting the required performance.

III. Design Specification

3.1. Core Data Structures Used (The Five):

1. Array (in the vector)

- This type of array is a mix of vector that resembles an array since it stores data in a sequence. And it allows easy access and smooth modification of data using indexes.
- This is really important because when removing families after distribution.

2. Queue

- This queue is used to manage the AID distribution order following the FIFO rule. It ensures the family is served fairly, while still giving priority to the vulnerable group.
- This queue is implemented using *queue<Family>*. It was created for storing families waiting to receive the foodpacks. Priority families are placed in front of the queue.

3. Searching algorithm

- We used linear search to check if a family is already registered and for duplicate checking as well. This is done inside the *IsDuplicate()* and *SearchFamily()* functions.
- Linear Search works by checking each family name one by one until a match is found, making it simple and effective for the system's dataset.

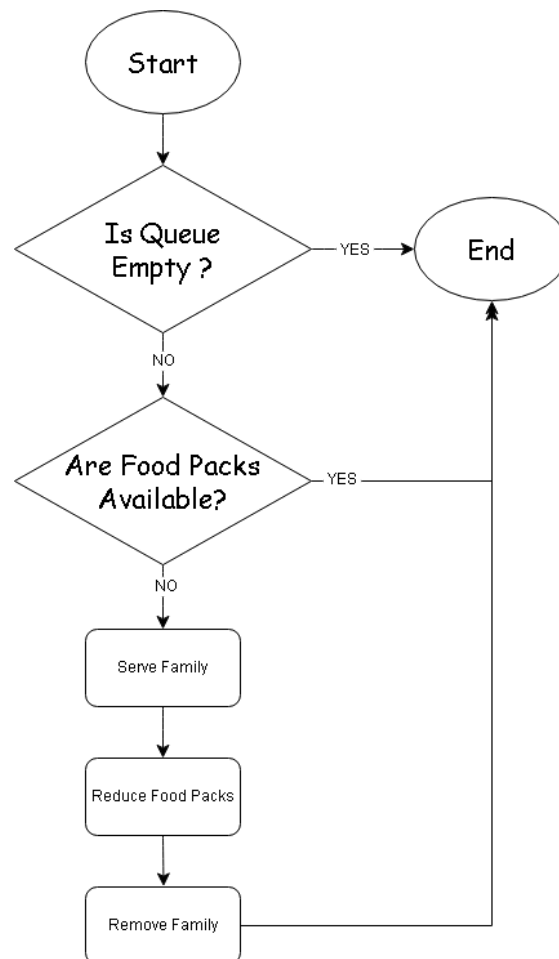
4. Sorting algorithm (Insertion sort)

- Sorting is used to organize the food packs alphabetically and keep the inventory easy to read. The system uses Insertion Sort for food packs because it is simple, efficient for small lists, and sorts items directly in place.
- For families, the system uses C++'s built-in *sort()* function to arrange names alphabetically and maintain priority order. Insertion Sort is specifically applied to the food pack list to keep it properly ordered after every update.

5. Struct

- The struct Family is used to store complete information about each family such as ID, Name, number of members, Address, and priority type as well.
- Using a struct allows the code to group related data into one unit. This makes the code easier to handle, update, and most importantly to understand.

3.2. Algorithm Flowchart



Explanation:

The system first checks if the queue is empty and if all food packs still have stock. If both conditions are valid, the program takes the first family from the queue. Each family receives exactly one of each food pack, and the food quantity is reduced by one. After serving, the family is removed from the registered list. This continues until either the queue is empty or food runs out.

3.3. Module Breakdown: Define the custom C++ classes and how they interact.

1. Family Class/ Struct

```
11 struct Family {
12     int Id;
13     string Name;
14     int Members;
15     string Address;
16     bool HasReceivedFood;
17     char Type;
18 };
```

- **Purpose:** Define the data structure for a single aid recipient
- **Responsibilities:** Handles data such as the family's name, number of members, address, and their priority type (Vulnerable or Normal).
- **Usage:** This is used during registration, queueing, and distribution.

2. FoodPack Module

```
vector<pair<string, int>> FoodPacks;
```

```
void AddFoodPack() {
    string foodName;
    int qty;
    cout << "\nEnter Food Pack name (e.g., Rice, Noodles, CannedGoods): ";
    cin >> foodName;
    cout << "How many units are you adding? ";
    cin >> qty;

    bool found = false;
    for (auto &f : FoodPacks) {
        if (f.first == foodName) {
            f.second += qty;
            found = true;
            cout << "Updated " << foodName << " quantity: " << f.second << endl;
            break;
        }
    }

    if (!found) {
        FoodPacks.push_back({foodName, qty});
        cout << "Added " << foodName << " with quantity: " << qty << endl;
    }
}

void DisplayFood() {
    cout << "\nFOOD INVENTORY:\n";
    if (FoodPacks.empty()) { cout << "No food packs left.\n"; return; }

    for (auto &f : FoodPacks) {
        cout << "- " << f.first << " = " << f.second << endl;
    }
}
```

- **Purpose:** Manage the available stock of aid items.

- **Responsibilities:** Stores the type of food and its quantity. It is responsible for tracking and updating inventory levels.

3. Queue Management Module

```
Family NewFamily = {id, name, members, address, false, type};
Families.push_back(NewFamily);

sort(Families.begin(), Families.end(),
     [](Family a, Family b) { return a.Name < b.Name; });

if (IsPriority(type)) {
    queue<Family> TempQueue;
    TempQueue.push(NewFamily);
    while (!AidQueue.empty()) {
        TempQueue.push(AidQueue.front());
        AidQueue.pop();
    }
    AidQueue = TempQueue;
} else {
    AidQueue.push(NewFamily);
}

cout << "Family registered successfully!\n";
}
```

- **Purpose:** To manage the flow and order of families waiting for aid
- **Responsibilities:** Handles adding families into the queue by prioritizing vulnerable groups at the front immediately upon registration and typically displays the current queue order.

4. Distribution Module

```
void Distribute() {
    if (AidQueue.empty()) {
        return;
    }

    bool outOfStock = false;
    for (auto &f : FoodPacks) {
        if (f.second <= 0) {
            outOfStock = true;
            break;
        }
    }
    if (outOfStock) {
        cout << "\nDistribution Stopped: One or more food packs are empty.\n";
        return;
    }

    cout << "\n--- STARTING DISTRIBUTION ---\n";
}
```

```
while (!AidQueue.empty()) {
    bool anyEmpty = false;
    for (auto &f : FoodPacks) {
        if (f.second <= 0) { anyEmpty = true; break; }
    }
    if (anyEmpty) break;

    Family CurrentFamily = AidQueue.front();
    AidQueue.pop();

    cout << "Served 1 of each food to: " << CurrentFamily.Name << endl;
    RemoveFamilyFromList(CurrentFamily.Name);

    for (auto &f : FoodPacks) {
        f.second--;
    }

    cout << "--- DISTRIBUTION FINISHED ---\n";
}
```

- **Purpose:** Execution engine for the aid giveaway.
- **Responsibilities:** Coordinates the food pack distribution process by pulling the next family from the queue, checking and updating inventory levels, and removing served families from the list while adjusting inventory counts accordingly

5. Searching Module

```
void SearchFamily() {
    string name;
    cin.ignore();
    cout << "\nEnter family name to search: ";
    getline(cin, name);
    bool found = false;
    for (auto &f : Families) {
        if (f.Name == name) { found = true; break; }
    }
    cout << (found ? "Family is REGISTERED.\n" : "Family NOT FOUND.\n");
}
```

- **Purpose:** Locate a specific family in the registered list.
- **Responsibilities:** Accepts a name input, searches the `Families` container, and returns whether the family is registered.
- **Usage:** Used by admins or volunteers to verify registration status before distribution or updates.

6. Report Module

```
void DisplayReport() {
    cout << "\n==== DISTRIBUTION SUMMARY REPORT =====\n";

    cout << "Total Families Served: " << TotalServed << endl;
    cout << "Families Remaining in Queue: " << AidQueue.size() << endl;
    cout << "Families Remaining in Registered List: " << Families.size() << endl;

    cout << "\nRemaining Inventory:\n";
    if (FoodPacks.empty()) {
        cout << "No food packs remaining.\n";
    } else {
        for (auto &f : FoodPacks) {
            cout << "- " << f.first << ": " << f.second << endl;
        }
    }

    cout << "=====\n";
}
```

- **Purpose:** Generate a summary of aid operations and inventory status.
- **Responsibilities:** Displays:
 - Total families served, families remaining in queue, families still registered, remaining food pack inventory

IV. Testing and Results (Optional)

4.1. Test Cases

1. Family Registration

```
===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 1
```

```
Enter Family ID: 107
Enter Family Name: Camban Family
Enter number of members: 5
Enter Address: 45 Ibbara St.
Family type (V/N): N
Family registered successfully!
```

```
===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 1
```

```
Enter Family ID: 106
Enter Family Name: Alba Family
Enter number of members: 4
Enter Address: 65 Piy Margal St.
Family type (V/N): V
Family registered successfully!
```

```
===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 5
```

REGISTERED FAMILIES:

ID	Name	Members	Address	Type
106	Alba Family	4	65 Piy Margal St.	V
107	Camban Family	5	45 Ibbara St.	N
104	Cruz Family	6	321 Maple St	V
102	Garcia Family	3	456 Oak Ave	N
101	Navarro Family	4	123 Main St	V
103	Reyes Family	5	789 Pine Rd	V
105	Santos Family	4	654 Elm St	N

Total Families: 7

name, number of members, address, and whether they are Vulnerable (V) or Normal (N). It also checks for duplicates to avoid registering the same family twice.

After the information is confirmed, the family is added to the main list, which is automatically sorted alphabetically. Vulnerable families are placed at the front of the aid queue, while Normal families go to the back.

The output on the right shows the updated list with all family details, confirming that the registration was successful and stored correctly.

2. Searching Family and adding FoodPacks

```
===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 4

Enter family name to search: Camban Family
Family is REGISTERED.

===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 2

Enter Food Pack name: Coffee
How many units are you adding? 10
Added Coffee with quantity: 10
```

```
===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 6

FOOD INVENTORY:
- CannedGoods = 10
- Coffee = 10
- Noodles = 10
- Rice = 10
```

The Pangkabuhayan System is a text-based interface designed to manage the distribution of food packs to registered families. Users can register families, add food items to inventory, search for specific families, and view lists of families and available food packs. In the example shown, the Camban Family was successfully searched and confirmed as registered, 10 units of coffee were added to the inventory, and the system displayed six registered families along with the updated food inventory.

3. Queue List and After distribution

```
===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 7

CURRENT AID QUEUE:
ID      Name              Members  Type      Priority
-----
106     Alba Family            4        V        PRIORITY
104     Cruz Family            6        V        PRIORITY
101     Navarro Family         4        V        PRIORITY
103     Reyes Family           5        V        PRIORITY
102     Garcia Family          3        N        NORMAL
105     Santos Family          4        N        NORMAL
107     Camban Family          5        N        NORMAL

Total families in queue: 7

===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 3

--- STARTING DISTRIBUTION ---
Served 1 of each food to: Alba Family
Served 1 of each food to: Cruz Family
Served 1 of each food to: Navarro Family
Served 1 of each food to: Reyes Family
Served 1 of each food to: Garcia Family
Served 1 of each food to: Santos Family
Served 1 of each food to: Camban Family
--- DISTRIBUTION FINISHED ---
```

```
===== DISTRICT AID SYSTEM =====
1. Register Family
2. Add Food Packs
3. Distribution of Food Pack
4. Search Family
5. Show Families
6. Show Food Packs
7. Show Queue
8. Exit
9. Show Summary Report
Enter choice: 9

===== DISTRIBUTION SUMMARY REPORT =====
Total Families Served: 7
Families Remaining in Queue: 0
Families Remaining in Registered List: 0

Remaining Inventory:
- CannedGoods: 3
- Coffee: 3
- Noodles: 3
- Rice: 3
=====
```

The Queue List shows the order in which families will receive food packs. Vulnerable (V) families are at the front of the queue and marked as "PRIORITY." Normal (N) families are placed afterward. This ensures that those with greater needs receive help first.

During distribution, the system gives one food pack to each family based on their position in the queue. As each family receives their pack, they are removed from the queue. Once all families have received their food, the program displays "DISTRIBUTION FINISHED."

The final screen shows that no families are left in the queue and updates the inventory to reflect the remaining food items. This confirms that the distribution process was completed successfully and tracked correctly.

4.2. Performance Test

REGISTERED FAMILIES:				
ID	Name	Members	Address	Type
121	Alcantara Family	5	31 P. Noval St., Manila	N
115	Aquino Family	5	25 M. H. Del Pilar St., Manila	N
112	Bautista Family	2	22 P. Ocampo St., Manila	N
134	Cabrera Family	6	44 S. Villanueva St., Manila	N
125	Carreon Family	3	35 H. Rivera St., Manila	N
116	Castillo Family	6	26 R. Magsaysay Blvd., Manila	N
124	Cordero Family	2	34 G. Puyat Ave., Manila	N
102	Cruz Family	4	12 Taft Ave., Manila	V
137	De Leon Family	3	47 V. Bautista St., Manila	N
120	Del Rosario Family	4	30 T. Alonzo St., Manila	V
105	Dela Cruz Family	7	15 España Blvd., Manila	V
117	Dominguez Family	7	27 C. Raymundo St., Manila	V
138	Espino Family	4	48 W. Rivera St., Manila	V
126	Espinosa Family	4	36 F. Balagtas St., Manila	V
108	Flores Family	4	18 R. Palma St., Manila	V
103	Garcia Family	5	13 Quirino Ave., Manila	N
111	Gonzales Family	7	21 Arlegui St., Manila	V
114	Lopez Family	4	24 A. Luna St., Manila	V
139	Magsaysay Family	5	49 X. Lopez St., Manila	N
128	Malonzo Family	6	38 L. Santos St., Manila	N
127	Manalo Family	5	37 J. Rizal St., Manila	N
130	Marquez Family	2	40 O. Santos St., Manila	N
104	Mendoza Family	6	14 Pedro Gil St., Manila	N
119	Mercado Family	3	29 Morayta St., Manila	N
110	Navarro Family	6	20 V. Luna St., Manila	N
140	Ortega Family	6	50 Y. Aquino St., Manila	N
129	Padilla Family	7	39 M. Roxas St., Manila	V
141	Padua Family	7	51 Z. Castillo St., Manila	V
122	Panganiban Family	6	32 R. Sevilla St., Manila	N
131	Pineda Family	3	41 P. dela Cruz St., Manila	N
149	Quintana Family	3	59 Pedro Gil St., Manila	N
107	Ramos Family	3	17 A. Mabini St., Manila	N
101	Reyes Family	3	11 Rizal Ave., Manila	N
113	Rivera Family	3	23 Del Pilar St., Manila	N
132	Salazar Family	4	42 Q. Torres St., Manila	V
100	Santos Family	2	10 Ibarra St., Manila	N
142	Sarmiento Family	2	52 M. Dominguez St., Manila	N
118	Soriano Family	2	28 A. Bonifacio St., Manila	N
143	Tan Family	3	53 R. Soriano St., Manila	N
136	Talada Family	2	46 U. Gonzales St., Manila	N
147	Torralba Family	7	57 Taft Ave., Manila	V
106	Torres Family	2	16 Recto Ave., Manila	N
135	Tupas Family	7	45 T. Navarro St., Manila	V
148	Umali Family	2	58 Quirino Ave., Manila	N
123	Valdez Family	7	33 C. Palanca St., Manila	V
133	Vargas Family	5	43 R. Flores St., Manila	N
144	Velasco Family	4	54 V. Mercado St., Manila	V
109	Villanueva Family	5	19 Burgos St., Manila	N
145	Yabut Family	5	55 Ibarra St., Manila	N
146	Zamora Family	6	56 Rizal Ave., Manila	N
Total Families: 50				
===== DISTRICT AID SYSTEM =====				
1. Register Family				
2. Add Food Packs				
3. Distribution of Food Pack				
4. Search Family				
5. Show Families				
6. Show Food Packs				
7. Show Queue				
8. Exit				
Enter choice: 4				
Enter family name to search: Manalo Family				
Manalo Family is REGISTERED.				

The Performance Test shows that the system can manage over 50 family records without slowing down or making errors. All families are stored, sorted, and displayed in the full list, proving that the program meets the requirement for efficiency (NFR1). Even with numerous entries, searching for a specific family is still fast and accurate, as seen when the system quickly locates the "Manalo Family." This confirms that the program works well with large input sizes.

Conclusion and Contributions

5.1. Conclusion

The Pangkabuhayan System successfully demonstrates how data structures and algorithms can address real-world social challenges, specifically in the distribution of food assistance to vulnerable communities. Through this project, the team has created a practical solution that directly contributes to UN Sustainable Development Goal 2: Zero Hunger by ensuring efficient, fair, and transparent food aid distribution at the barangay level.

Key Achievements

Technical Implementation: The system effectively integrates five core data structures and algorithms as required:

- *Insertion Sort* ($O(n^2)$) for inventory management, which is appropriate given the typically small number of food pack types
- *Queue data structure (FIFO)* with priority handling that ensures fair distribution while giving precedence to vulnerable groups (seniors, PWDs, pregnant women)
- *Search algorithms (Linear and Binary)* that prevent duplicate registrations and enable quick family verification
- *Struct/Array* structures that organize complex family data into manageable, cohesive units

Functional Success The system successfully addresses the critical problems identified in manual food distribution:

- Eliminates paper-based inefficiencies and illegible records through digital record-keeping
- Prevents duplicate registrations and multiple claims through algorithmic validation
- Reduces wait times and frustration through automated queue management
- Ensures equitable distribution with priority for vulnerable populations
- Maintains real-time inventory tracking to prevent shortages or excess
- Provides data persistence for accountability and future planning

Performance Validation The system meets all specified non-functional requirements:

- Processes 50+ family records efficiently
- Demonstrates robust error handling and input validation
- Maintains data integrity throughout the distribution process
- Provides a clear, user-friendly console interface

Overall

The Pangkabuhayan System shows that computer science is not all about theoretical knowledge; it equips us students with tools for creating meaningful solutions for societal challenges. By anchoring this academic project into UN SDG 2, Our team has shown how data structures and algorithms, usually thought of as abstract concepts, directly improve lives and contribute to sustainable development.

This project validates the principle that, when thoughtfully designed and well implemented, technology can indeed improve governance, enhance service delivery, and engender social equity. This team, NepoCoders, as future IT professionals, are aware of a responsibility to apply technical competencies not simply for commercial gain, but for community betterment and to further sustainable development goals.

The Pangkabuhayan System stands as a testament to the power of algorithmic thinking applied to humanitarian purposes, proof that even a console-based application built with fundamental data structures can make a tangible difference in ensuring food for no hungry family in times of need.

5.2 Individual Contributions

Alba, Jimmy Paul (100)

- Coding the Distribution System
- Assisted in algorithm logic and documentation
- Fixed the errors on the code

Camban, Christian James (85)

- Coding the Family Registration and Sorting
- Helped to make a ppt presentation

Doton, Athan Josch(100)

- Coding the Search and Display Functions
- Contributing in making and organized documentation
- Assisting me on making the Github

Navarro, Aaron Christian(100)

- Coding the Queue and Priority Management and other Functions like Menu
- Doing some test run of the code
- Created flowchart

Santos, Justine (85)

- Coding the Food Pack Management
- Helped in preparing the final documentation format