DONATION MANAGEMENT SYSTEM

Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of  
BACHELOR OF TECHNOLOGY  
In  
COMPUTER SCIENCE AND ENGINEERING

By  
  
P.Amarnadh reddy(24KB1A05CW)

P.Gowtham(24KB1A05DK)

P.Manoj(24KB1A05CT)

S.Sai Kaushal(24KB1A05GL)

Under the Guidance of  
  
SivanrajSamy

  
  
  
N.B.K.R INSTITUTE OF SCIENCE AND TECHNOLOGY

VIDYANAGAR-524413-Tirupathi Dt., A.P

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



# CERTIFICATE

This is to certify that the project report entitled DONATION MANAGEMENT SYSTEM being submitted by

P.Amarnadh reddy(24KB1A05CW)

P.Gowtham(24KB1A05DK)

P.Manoj(24KB1A05CT)

S. Sai Koushal(24KB1A05GL)

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technological University, Anantapoor is a record of bonafide work carried out under my guidance and supervision.

Mrs.Sruthi Reddy Head of the department

Designation:

DECLARATION

I hereby declare that the dissertation entitled DONATION MANAGEMENT SYSTEM submitted for the B.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Place: Vidyanagar by

Date: 04-05-2025 P. Amarnadh Reddy

P. Gowtham

P. Manoj

S. Sai Koushal

# 

# **ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to all those who supported me throughout the course of this project.

First and foremost, I would like to thank my project guide, **SivanrajSamy**, for their valuable guidance, encouragement, and constant support during the development of the project titled **"Donation Management System"**. Their insights and suggestions were instrumental in the successful completion of this work.

I am also thankful to Head of the Department of **Computer Science and Engineering,** for providing me with the necessary resources and a conducive environment to carry out this project.

I extend my appreciation to all the faculty members and staff of the department for their cooperation and assistance.

Lastly, I would like to thank my family and friends for their constant motivation and support throughout this endeavor.

**ABSTRACT**

The Donation Management System is a software application designed to streamline the collection, organization, and management of donations across three main categories: food, clothes, and money. This system aims to provide an efficient and transparent platform for donors to contribute, and for administrators to record and track donations. Implemented in the C programming language, the system offers features such as donor registration, category-wise donation logging, and listing of donor details. All data is stored in local files for future reference and accountability. The project demonstrates key concepts such as file handling, structures, dynamic memory

**Contents**

[CERTIFICATE 2](#_Toc197269804)

[ACKNOWLEDGEMENT 3](#_Toc197269805)

[ABSTRACT 3](#_Toc197269806)

[**Chapter 1: Introduction** 5](#_Toc197269807)

[**Chapter 2: Literature Review** 5](#_Toc197269808)

[**Chapter 3: Requirement Analysis** 6](#_Toc197269809)

[3.1 Software Requirements 6](#_Toc197269810)

[3.2 Hardware Requirements 6](#_Toc197269811)

[3.3 Functional Requirements 6](#_Toc197269812)

[3.4 Non-Functional Requirements 6](#_Toc197269813)

[**Chapter 4: System Design** 6](#_Toc197269814)

[**Chapter 5: Implementation** 7](#_Toc197269815)

[**Chapter 6: Testing** 8](#_Toc197269816)

[**Chapter 7: Output Screens / Results** 8](#_Toc197269817)

[7.1 Out put Screen: 9](#_Toc197269818)

[7.2 Saved file screen 9](#_Toc197269819)

[**Chapter 9: source code** 10](#_Toc197269820)

[**Chapter 8: Conclusion** 13](#_Toc197269821)

# **Chapter 1: Introduction**

The need for efficient donation management has become more important than ever, especially in times of humanitarian crises, natural disasters, or widespread social campaigns. Managing contributions in an organized and transparent manner ensures that the right resources reach the right people. This project aims to develop a **Donation Management System** that allows the tracking and categorization of donations made in the form of **food**, **clothes**, and **money**.

Developed using the C programming language, this system provides a **console-based user interface** for adding donor details and displaying all registered contributions. It stores each donation with the donor’s name, contact number, donation quantity, and category. Furthermore, it ensures that records are not lost by saving all data to a file on the system.

The goal of this project is to implement a **simple, lightweight, and functional system** for managing basic donation information using core data structures like **linked lists** and essential file handling techniques. This application can serve as a base model for more complex donation management solutions in real-world NGOs, welfare organizations, or community support initiatives.

# **Chapter 2: Literature Review**

Several large-scale donation systems exist in the market today, such as Give India, Goonj, and UNICEF platforms. These systems usually integrate web portals, databases, payment gateways, and logistics support. While such solutions are powerful and scalable, they require internet access, professional development, and significant infrastructure.

This project, in contrast, is inspired by the need for a **lightweight and offline-capable donation tracker**, particularly useful for small communities, schools, or rural areas. Previous academic projects and tutorials have showcased basic inventory or contact management systems using C. This Donation Management System expands on such foundational work by implementing multiple linked lists, file I/O for data persistence, and a structured category-based approach.

In particular, it draws from traditional file-based record-keeping methods and introduces a simple but effective way to handle **categorical segmentation** of donors using a modular approach. This improves both the readability and traceability of data and makes it a suitable case study for beginner and intermediate programmers.

# **Chapter 3: Requirement Analysis**

## 3.1 Software Requirements

* Operating System: Windows/Linux
* Compiler: GCC / Turbo C / any standard C compiler
* Editor: Code Blocks, Dev C++, or Visual Studio Code

## 3.2 Hardware Requirements

* Processor: Intel Pentium or later
* RAM: Minimum 2 GB
* Storage: 10 MB free space
* Input: Keyboard

## 3.3 Functional Requirements

* The system must allow users to:
  + Add a new donor with name, contact, and quantity.
  + Select donation category (Food, Clothes, Money).
  + Display all donors by category.
  + Automatically store donation records into a file.
  + Display total monetary donations separately.

## 3.4 Non-Functional Requirements

* The system should have a **user-friendly command-line interface**.
* Must handle **invalid inputs gracefully**.
* Data must persist even after the program is closed (via file handling).
* Memory must be efficiently managed (using dynamic allocation).

# **Chapter 4: System Design**

The design of the Donation Management System is structured around three main donation categories: **Food**, **Clothes**, and **Money**. For each category, a **separate linked list** is maintained using a global array of pointers. This modular approach ensures that donations in one category do not interfere with others. Each donor is represented by a structure containing fields for the **donor's name**, **contact number**, and **donation quantity or amount**.

The system features a **menu-driven interface**, prompting the user to add a donor, display all donors, or exit. When a new donor is added, their data is appended to the appropriate linked list and simultaneously saved to a text file (donors.txt) using file handling. This file stores each donor’s details along with the donation category and its appropriate unit (e.g., kg for food, pairs for clothes, amount for money).

Error checking is implemented during category selection to ensure valid input. The system also calculates and displays the total donation amount specifically for the Money category when displaying all donors. The design emphasizes **simplicity**, **modularity**, and **data persistence**, making it ideal for small-scale donation tracking with minimal resources.

# **Chapter 5: Implementation**

The system is implemented in **C programming language**, leveraging core features such as **structures**, **linked lists**, and **file I/O**. Each donor is created dynamically using malloc and stored in the corresponding linked list (donorLists) based on the donation category selected. The donor structure contains the fields name, contact, quantity, and a pointer to the next donor node, forming the basis of the linked list.

When a new donor is added, the program:

1. Prompts the user to select a donation category.
2. Collects the donor's name, contact, and donation quantity.
3. Inserts the donor at the head of the linked list for the selected category.
4. Saves the record in a formatted string to donors.txt, including the unit type ("kg", "pairs", or "amount").

The displayDonors() function iterates through all three linked lists, showing each donor’s details clearly and calculating the total amount for the Money category. The main function manages the user interface, using a do-while loop and switch-case structure for option control. The implementation focuses on clarity, data safety, and user interaction.

# **Chapter 6: Testing**

The system was tested using several manual test cases to ensure functionality and reliability:

* **Valid Category Input**: Selecting each valid category and adding donors successfully.
* **Invalid Input Handling**: Entering invalid category numbers triggers an error without crashing the program.
* **Data Persistence**: After adding donors, the contents of donors.txt were checked and confirmed accurate.
* **Multiple Donors**: Added multiple donors to each category to verify linked list traversal and file output.
* **Display Functionality**: The display function outputs donors neatly grouped by category, with proper units and formatted contact details.

Boundary cases such as long names, empty input fields, and repeated category entries were handled gracefully. The application remains stable throughout all tests, and the final total donation amount for money is calculated and shown correctly.

# **Chapter 7: Output Screens / Results**

The output is entirely **console-based** and text-driven. The application presents a clean interface with prompts for user input and categorized data display. Sample outputs include:

* When a donor is added:  
  ✅ Donor added to Food category and saved to file.
* When displaying donors:

📂 Category: Clothes

Name: John | Contact: 1234567890 | Quantity: 5 pairs

📂 Category: Money

Name: Ramu | Contact: 9876543210 | Amount: 1000

💰 Total Donations in Money Category: 1000

Data stored in donors.txt appears as:  
John,1234567890,5, Clothes pairs  
Ramu,9876543210,1000, Money amount

## 7.1 Out put Screen:

A screenshot of a computer

AI-generated content may be incorrect.

## 7.2 Saved file screen

A black background with white text and blue numbers

AI-generated content may be incorrect.

These results confirm that the application fulfills its purpose and records are persistent and human-readable.

# **Chapter 9: source code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define CATEGORY\_COUNT 3

char \*categories[CATEGORY\_COUNT] = {"Food", "Clothes", "Money"};

typedef struct Donor {

char name[50];

char contact[20];

int quantity;

struct Donor \*next;

} Donor;

Donor \*donorLists[CATEGORY\_COUNT] = {NULL};

// Create a new donor

Donor\* createDonor(char \*name, char \*contact, int quantity) {

Donor \*newDonor = (Donor\*)malloc(sizeof(Donor));

strcpy(newDonor->name, name);

strcpy(newDonor->contact, contact);

newDonor->quantity = quantity;

newDonor->next = NULL;

return newDonor;

}

// Save donor to file

void saveDonorToFile(char \*name, char \*contact, int quantity, char \*category, char \*unit) {

FILE \*file = fopen("donors.txt", "a");

if (file == NULL) {

printf("❌ Error opening file for writing.\n");

return;

}

fprintf(file, "%s,%s,%d,%s %s\n", name, contact, quantity, category, unit);

fclose(file);

}

// Add donor

void addDonor(int categoryIndex, char \*name, char \*contact, int quantity) {

Donor \*newDonor = createDonor(name, contact, quantity);

newDonor->next = donorLists[categoryIndex];

donorLists[categoryIndex] = newDonor;

// Determine unit

char unit[10] = "";

if (categoryIndex == 0) strcpy(unit, "kg");

else if (categoryIndex == 1) strcpy(unit, "pairs");

else strcpy(unit, "amount");

// Save to file

saveDonorToFile(name, contact, quantity, categories[categoryIndex], unit);

printf("✅ Donor added to %s category and saved to file.\n", categories[categoryIndex]);

}

// Display donors

void displayDonors() {

int totalMoneyDonations = 0;

for (int i = 0; i < CATEGORY\_COUNT; i++) {

printf("\n📂 Category: %s\n", categories[i]);

Donor \*current = donorLists[i];

if (current == NULL) {

printf(" No donors yet.\n");

} else {

while (current != NULL) {

char unit[10] = "";

if (i == 0) strcpy(unit, "kg");

else if (i == 1) strcpy(unit, "pairs");

if (i == 2) {

printf(" Name: %s | Contact: %s | Amount: %d\n",

current->name, current->contact, current->quantity);

totalMoneyDonations += current->quantity;

} else {

printf(" Name: %s | Contact: %s | Quantity: %d %s\n",

current->name, current->contact, current->quantity, unit);

}

current = current->next;

}

}

}

printf("\n💰 Total Donations in Money Category: %d\n", totalMoneyDonations);

}

// Main menu

int main() {

int choice;

char name[50], contact[20];

int quantity, categoryIndex;

do {

printf("\n=== Donation Management System ===\n");

printf("1. Add Donor\n");

printf("2. Display All Donors\n");

printf("3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // clear buffer

switch (choice) {

case 1:

printf("\nSelect Category:\n");

for (int i = 0; i < CATEGORY\_COUNT; i++) {

printf("%d. %s\n", i + 1, categories[i]);

}

printf("Enter category number: ");

scanf("%d", &categoryIndex);

getchar();

if (categoryIndex < 1 || categoryIndex > CATEGORY\_COUNT) {

printf("❌ Invalid category.\n");

break;

}

printf("Enter donor name: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = 0;

printf("Enter contact number: ");

fgets(contact, sizeof(contact), stdin);

contact[strcspn(contact, "\n")] = 0;

printf("Enter quantity donated: ");

scanf("%d", &quantity);

addDonor(categoryIndex - 1, name, contact, quantity);

break;

case 2:

displayDonors();

break;

case 3:

printf("👋 Exiting Donation Management System. Goodbye!\n");

break;

default:

printf("❗ Invalid choice. Try again.\n");

}

} while (choice != 3);

return 0;

}

# **Chapter 8: Conclusion**

The Donation Management System successfully demonstrates the use of C programming to manage and organize donor data based on categories. By leveraging linked lists for dynamic data storage and file handling for record persistence, the system provides a functional and user-friendly interface for small-scale donation tracking. It lays a solid foundation for future improvements, such as integrating a database, adding user authentication, or building a GUI. Overall, the project achieves its objective of creating an offline, category-based donation record system for food, clothes, and money.