CHARITY MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted By

Sneha Sajeevan 220701281

Swetha R 220701297

In partial fulfillment for the award of the degree of

BACHELOR OF

ENGINEERING IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE

(AUTONOMOUS) THANDALAM

CHENNAI-602105

2023 - 2024

BONAFIDE CERTIFICATE

Certified that this project report "**CHARITY MANAGEMENT SYSTEM**" is the bonafide work of **"SNEHA SAJEEVAN (220701281), SWETHA R (220701297)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
DR SABITHA R	MS JANANEE V
Professor and II Year Academic Head,	Assistant Professor(SG),
Computer Science and Engineering,	Computer Science and Engineering,
Rajalakshmi Engineering College,	Rajalakshmi Engineering College,
(Autonomous),	(Autonomous),
Thandalam, Chennai - 602 105	Thandalam, Chennai - 602 105

Submitted for the Practical Examination held on __________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Charity management system that acts as an interface between the users who are looking for a channel to give the excess things without wasting it. It enables us to donate the excess by notifying nearby users with the details of the donation that is available. The required users claim the notification. The system allocates the items based on the priority. It will be a web-based system that can be accessible from everywhere. It provides an easy method so that anyone can easily use.

The main objectives of Charity Management System project are to reduce wastage and ensure basic human needs. Even if any food item remains in any function people can send request to charity. This project plays a vital role in economy as it reduces wastage as well reducing poverty. This project helps charities to increase operational efficiencies and reduce costs by eliminating much manual paperwork.

TABLE OF CONTENTS

CHAPTER 1

1.INTRODUCTION

1.1 INTRODUCTION

Charity is an act of kindness, in which financially stable people provide help to those people who are needy. Finding a sponsor was a difficult task, and it was a big challenge to deal with sponsors. This Charity Management System helps to find sponsors easily. However, with the rising population and development of this country, wastage has risen to new high. There are many people who wish to donate but unaware of how exactly they can execute that. Our application resolves around helping the needy by connecting donor and needy people. Our application aims to bring about transparency, charity and swiftness in the process of donation.

1.2 OBJECTIVES

The main objective of the Charity Management System is

● To ensure basic human rights.

● Enable easy interaction between donors and organization.

● Both the donor and the needy will be benefited.

● Make work faster and quicker.

● Man power required is very less.

● Data can be stored for a long period of time

1. User Experience (UX) and Interface Design

Intuitive Design: Create a user-friendly interface that is easy to navigate for both novice and experienced users.

Responsive Design:Ensure the application works seamlessly across various devices (mobile, tablet, desktop).

Engaging Visuals: Use high-quality graphics and animations to enhance user engagement.

Personalization: Offer personalized recommendations and features based on user preferences and betting history.

2. Functionality

Real-time Data: Provide live updates on horse races, including odds, results, and other relevant statistics.

Betting Options:Offer a variety of betting types (e.g., win, place, show, exacta, trifecta) to cater to different user preferences.

Account Management: Allow users to create and manage their accounts, including viewing betting history and managing funds.

Notifications: Implement push notifications for race updates, bet outcomes, promotions, and other relevant information.

3. Security

Data Protection:Ensure robust data encryption and secure user data storage to protect against breaches.

Secure Transactions: Implement secure payment gateways for deposits and withdrawals.

Authentication:Use multi-factor authentication to enhance account security.

Regulatory Compliance: Adhere to legal and regulatory requirements for online betting in different regions.

4. Performance and Reliability

Scalability: Design the application to handle high traffic volumes, especially during major racing events.

Low Latency: Ensure minimal delay in real-time updates and transactions.

Reliability: Maintain high uptime and quickly address any technical issues or outages.

5. Customer Support

Help Center: Provide a comprehensive help center with FAQs, tutorials, and guides.

Live Support: Offer live chat, email, and phone support to assist users with any issues.

Community Engagement: Foster a community through forums or social media integration where users can share tips and experiences.

6. Marketing and Growth

Promotions and Bonuses: Offer promotions, bonuses, and loyalty programs to attract and retain users.

User Acquisition: Implement effective marketing strategies to acquire new users, including social media campaigns, partnerships, and influencer marketing.

Feedback and Improvement: Collect user feedback regularly and use it to improve the application continuously.

7. Analytics and Reporting

User Analytics: Track user behavior to understand preferences and improve the user experience.

Betting Analytics: Provide users with access to detailed analytics and insights to help them make informed betting decisions.

Performance Reports: Generate reports on the application's performance, user engagement, and financial metrics.

By focusing on these objectives, you can develop a comprehensive and competitive horse race betting application that meets user needs and stands out in the market.

1.3 MODULES

- ❖ Add donor details
- ❖ Update donor details
- ❖ Delete donor details
- ❖ Add receiver details
- ❖ Update receiver details
- ❖ Delete receiver details

These modules form the core CRUD (Create, Read, Update, Delete) operations necessary for managing donor and receiver information in a charity management system. Each module ensures data integrity and consistency by carefully handling relationships between the main entity tables (Donor and Receiver) and their corresponding detail tables (Donation and Received Item).

Add Donor Details

Description:

This module is responsible for adding new donor information to the database. This involves collecting the necessary data from the user and inserting a new record into the Donor and Donation tables.

Steps:

i. Collect donor information (Donor Name, Contact No, Address, Email).
ii. Insert the collected information into the Donor table to generate a DonorID.
iii. Collect donation details (Date, Donation Item).
iv. Insert the collected donation details along with the generated DonorID into the Donation table.

Example:

User inputs: John Doe, 1234567890, Addr1,

john@example.com, 2023-05-01, Clothes

Insert into Donor: Generates DonorID 1
Insert into Donation: (1, 2023-05-01, Clothes)

Update Donor Details

Description:

This module updates existing donor information. The user can modify details such as contact number, address, email, or donation items.

Steps:

i. Identify the donor using DonorID.
ii. Collect updated information from the user.
iii. Update the relevant fields in the Donor table.
iv. If donation details are updated, update the relevant record in the Donation table.

Example:

User updates: DonorID 1, new contact number 0987654321
Update Donor table: Set Contact No to 0987654321 where DonorID is 1

Delete Donor Details

Description:

This module deletes a donor's information from the database. This involves removing the donor's record from both the Donor and Donation tables.

Steps:

i. Identify the donor using DonorID.
ii. Delete the corresponding records from the Donation table.
iii. Delete the donor's record from the Donor table.

Example:

User deletes: DonorID 1
Delete from Donation table where DonorID is 1
Delete from Donor table where DonorID is 1

Add Receiver Details

Description:

This module is responsible for adding new receiver information to the database. This involves collecting the necessary data from the user and inserting a new record into the Receiver and Received Item tables.

Steps:

i. Collect receiver information (Receiver Name, Contact No, Address, Email).
ii. Insert the collected information into the Receiver table to generate a ReceiverID.
iii. Collect received item details (Date, Received Item).
iv. Insert the collected received item details along with the generated ReceiverID into the Received Item table.

Example:

User inputs: Alice Brown, 5555555555, Addr11, alice@example.com, 2023-05-03, Clothes
Insert into Receiver: Generates ReceiverID 1
Insert into Received Item: (1, 2023-05-03, Clothes)

Update Receiver Details

Description:

This module updates existing receiver information. The user can modify details such as contact number, address, email, or received items.

Steps:

i. Identify the receiver using ReceiverID.
ii. Collect updated information from the user.
iii. Update the relevant fields in the Receiver table.
iv. If received item details are updated, update the relevant record in the Received Item table.

Example:

User updates: ReceiverID 1, new address Addr12

Update Receiver table: Set Address to Addr12 where ReceiverID is 1

Delete Receiver Details

Description:

This module deletes a receiver's information from the database. This involves removing the receiver's record from both the Receiver and Received Item tables.

Steps:

i. Identify the receiver using ReceiverID.
ii. Delete the corresponding records from the Received Item
iii. table.
iv. Delete the receiver's record from the Receiver table.

Example:

User deletes: ReceiverID 1

Delete from Received Item table where ReceiverID is 1
Delete from Receiver table where ReceiverID is 1

CHAPTER 2

2.SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

Visual studio Code Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas.

2.2 LANGUAGES

2.2.1 SQL

Many of the world's largest and fastest-growing organisations including Facebook, Google, Adobe, Alcatel Lucent and Zappos rely on MySQL to save time and money powering their high-volume Web sites, business-critical systems and packaged software. Since then, the performance & scalability, reliability, and ease of use of the world's most popular open source database, characteristics that made MySQL the #1 choice for web applications, have relentlessly been improved.

2.2.2 PYTHON

Python is widely used for a variety of projects due to its ease of learning and use. This makes it an excellent choice for both beginners and experienced developers, allowing them to write and understand code quickly. Additionally, Python boasts an extensive ecosystem of libraries and frameworks, such as NumPy and pandas for data analysis, TensorFlow and PyTorch for machine learning, and Django and Flask for web development, which streamline development processes.

CHAPTER 3

3.REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENTS SPECIFICATION

User Requirements

The user requirement in charity management focuses on the possibility of search the donors and the needy receivers.

System Requirements

There should be a database backup of the charity management system. Operating system should be WindowsXP or a higher version of windows.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements

• Operating System Windows 10

• Front End Python

• Back End SQL

Hardware Requirements

● Desktop PC or a Laptop

● Printer

● Operating System – Windows 10

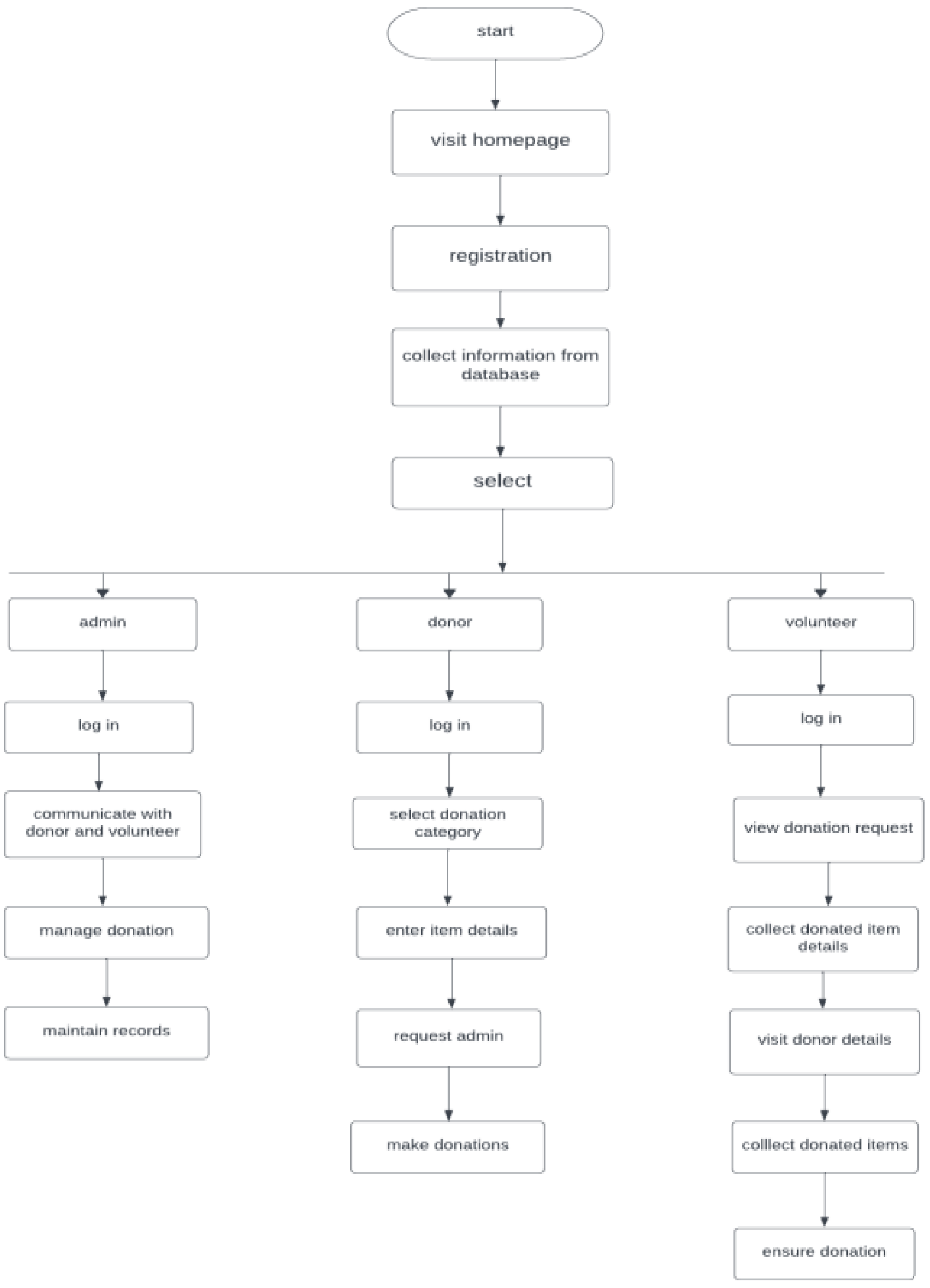
● Intel® CoreTM i3-6006U CPU @ 2.00GHz

● 4.00 GB RAM ● 64-bit operating system, x64 based processor

● 1024 x 768 monitor resolution

● Keyboard and Mouse

3.3 ARCHITECTURE DIAGRAM



3.4 ENTITY RELATIONSHIP DIAGRAM

Entities and Attributes:

<u>Donor</u>

- DonorID (Primary Key)
- Name
- Email
- Phone
- Address
- DonationItem

<u>Receiver</u>

- ReceiverID (Primary Key)
- Name
- Email
- Phone
- Address
- ReceivedItem

<u>Logout</u>

- A Donor can have multiple Logout records.
- A Receiver can also have multiple Logout records.

Actions:

Donor and Receiver tables have actions such as save, delete, modify, and view. These actions can be implemented as buttons in the user interface but are not part of the database schema directly.

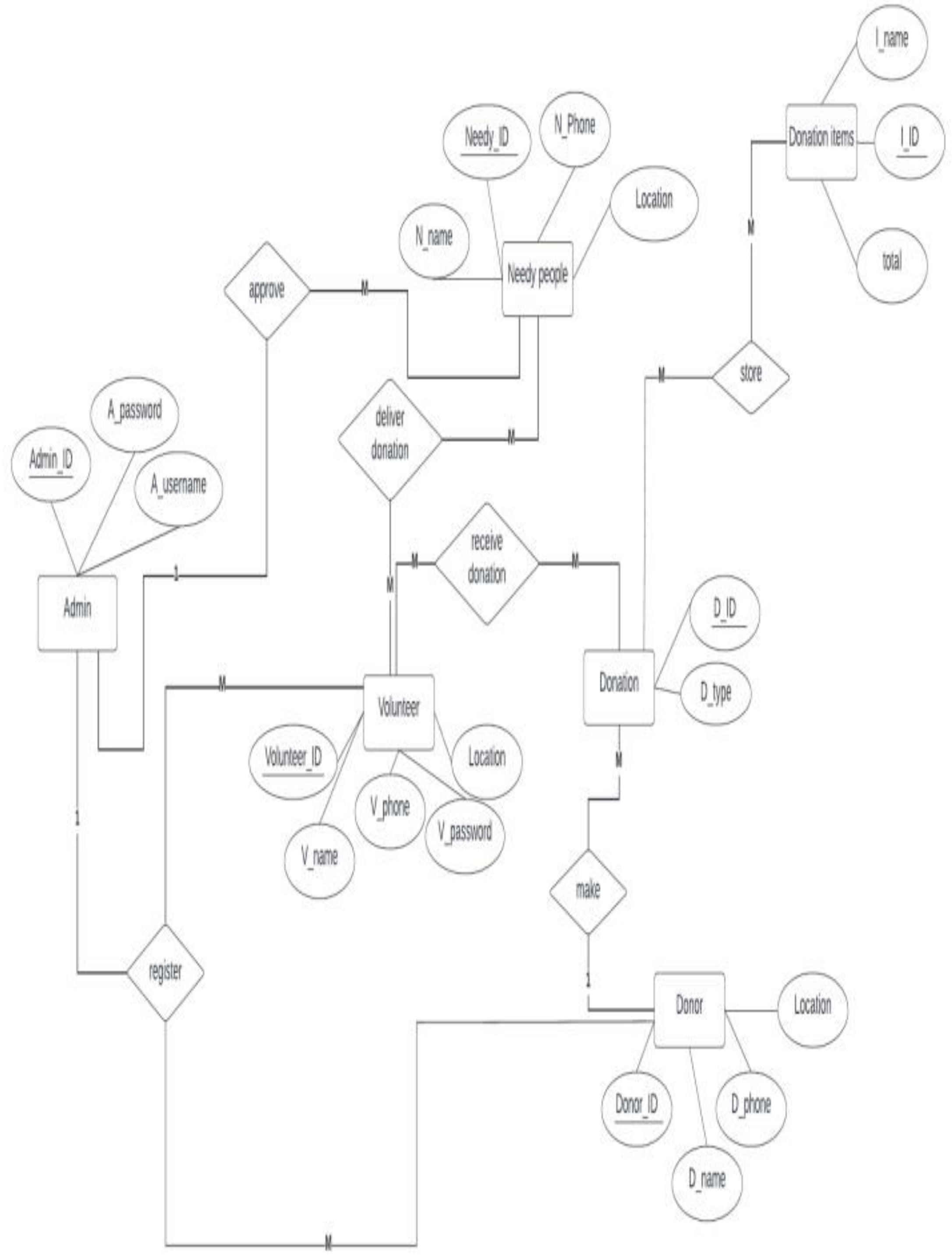
Donor Table: Stores information about the donors. Each donor has unique attributes like DonorID, Name, Email, Phone, Address, and DonationAmount. Actions (save, delete, modify, view) are for UI purposes.

Receiver Table: Stores information about the receivers. Each receiver has unique attributes like ReceiverID, Name, Email, Phone, Address, and ReceivedAmount. Actions (save, delete, modify, view) are for UI purposes.

Logout Table: Stores logout records. It includes LogoutID, UserID (which references DonorID or ReceiverID), and LogoutTime.

The relationships indicate that both donors and receivers can have multiple logout records, represented by the many-to-many relationship.

ER DIAGRAM


I_name
Donation items
I_ID
total
Needy_ID
N_Phone
Location
N_name
Needy people
approve
M
M
store
M
deliver donation
M
A_password
Admin_ID
A_username
receive donation
M
M
M
1
Admin
D_ID
Donation
D_type
M
Volunteer
Volunteer_ID
Location
V_phone
V_password
V_name
M
1
make
register
1
Donor
Location
Donor_ID
D_phone
D_name
M

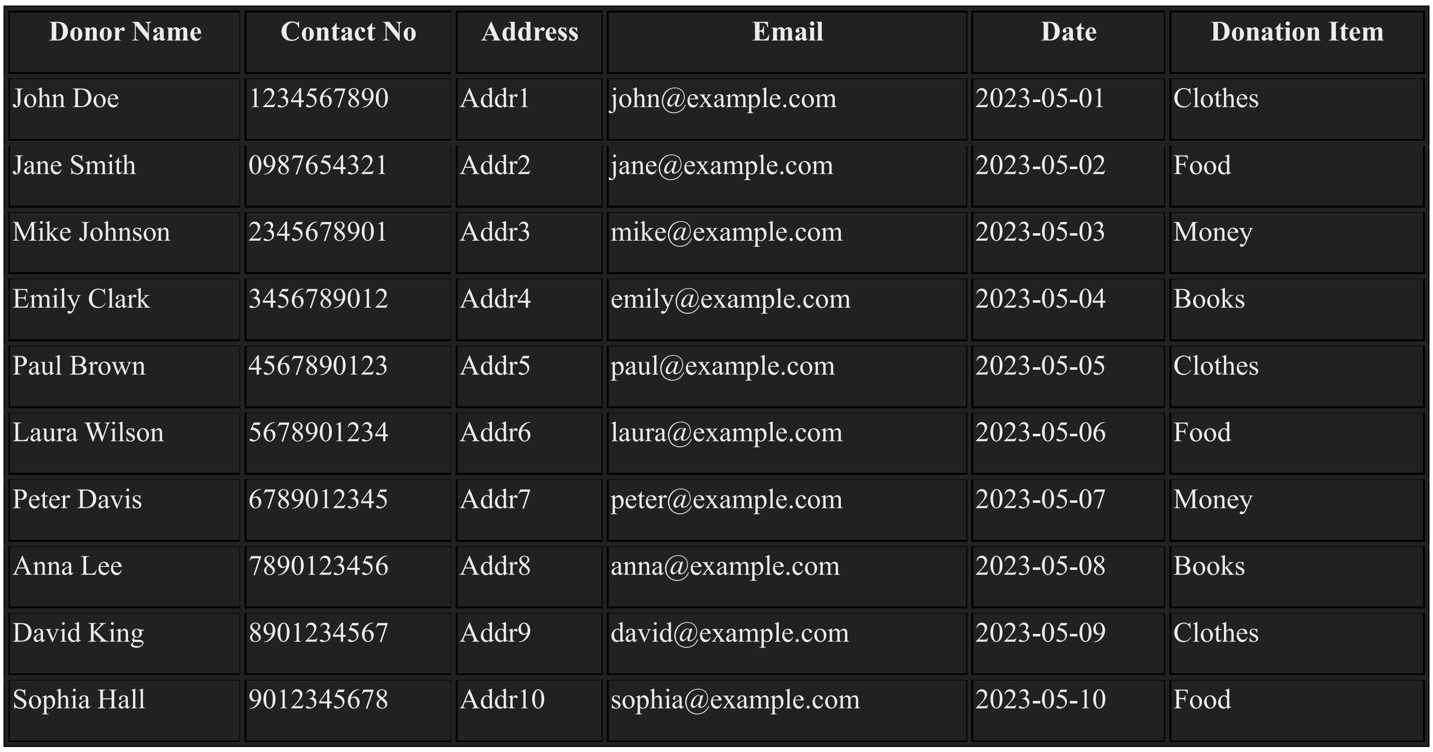
3.5 NORMALIZATION

Donor Detail Table

1NF (First Normal Form)

In 1NF, each column contains atomic values, and each column contains values of a single type.

Original Donor Detail Table:



Donor Name	Contact No	Address	Email	Date	Donation Item
John Doe	1234567890	Addr1	john@example.com	2023-05-01	Clothes
Jane Smith	0987654321	Addr2	jane@example.com	2023-05-02	Food
Mike Johnson	2345678901	Addr3	mike@example.com	2023-05-03	Money
Emily Clark	3456789012	Addr4	emily@example.com	2023-05-04	Books
Paul Brown	4567890123	Addr5	paul@example.com	2023-05-05	Clothes
Laura Wilson	5678901234	Addr6	laura@example.com	2023-05-06	Food
Peter Davis	6789012345	Addr7	peter@example.com	2023-05-07	Money
Anna Lee	7890123456	Addr8	anna@example.com	2023-05-08	Books
David King	8901234567	Addr9	david@example.com	2023-05-09	Clothes
Sophia Hall	9012345678	Addr10	sophia@example.com	2023-05-10	Food

The table is in 1NF since all values are atomic.

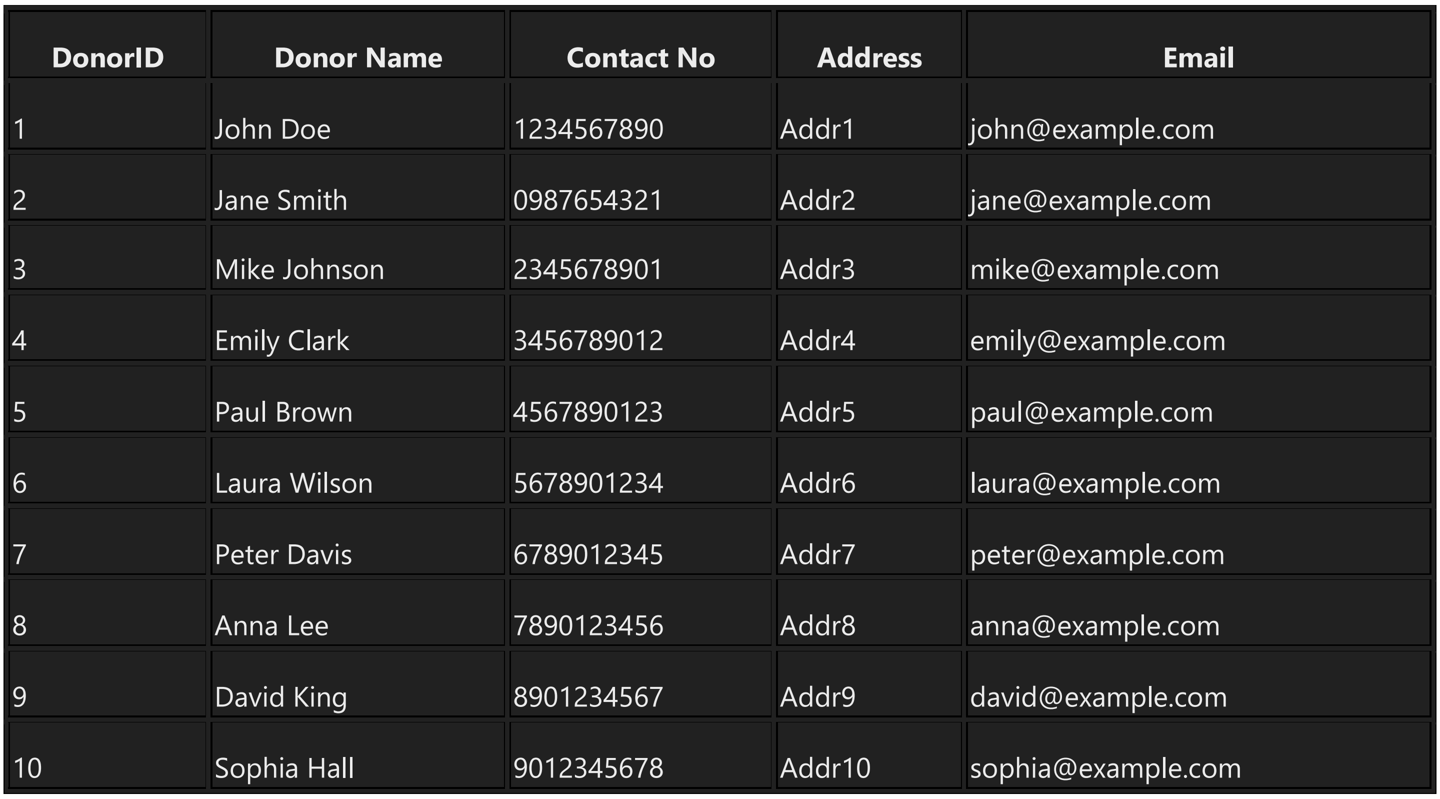
2NF (Second Normal Form)

To move to 2NF, we must ensure that all non-key attributes are fully functional dependent on the primary key. We'll introduce surrogate keys and split the table to remove partial dependencies.

Donor Detail Table Normalized into Two Tables:

<u>Donor Table</u>:

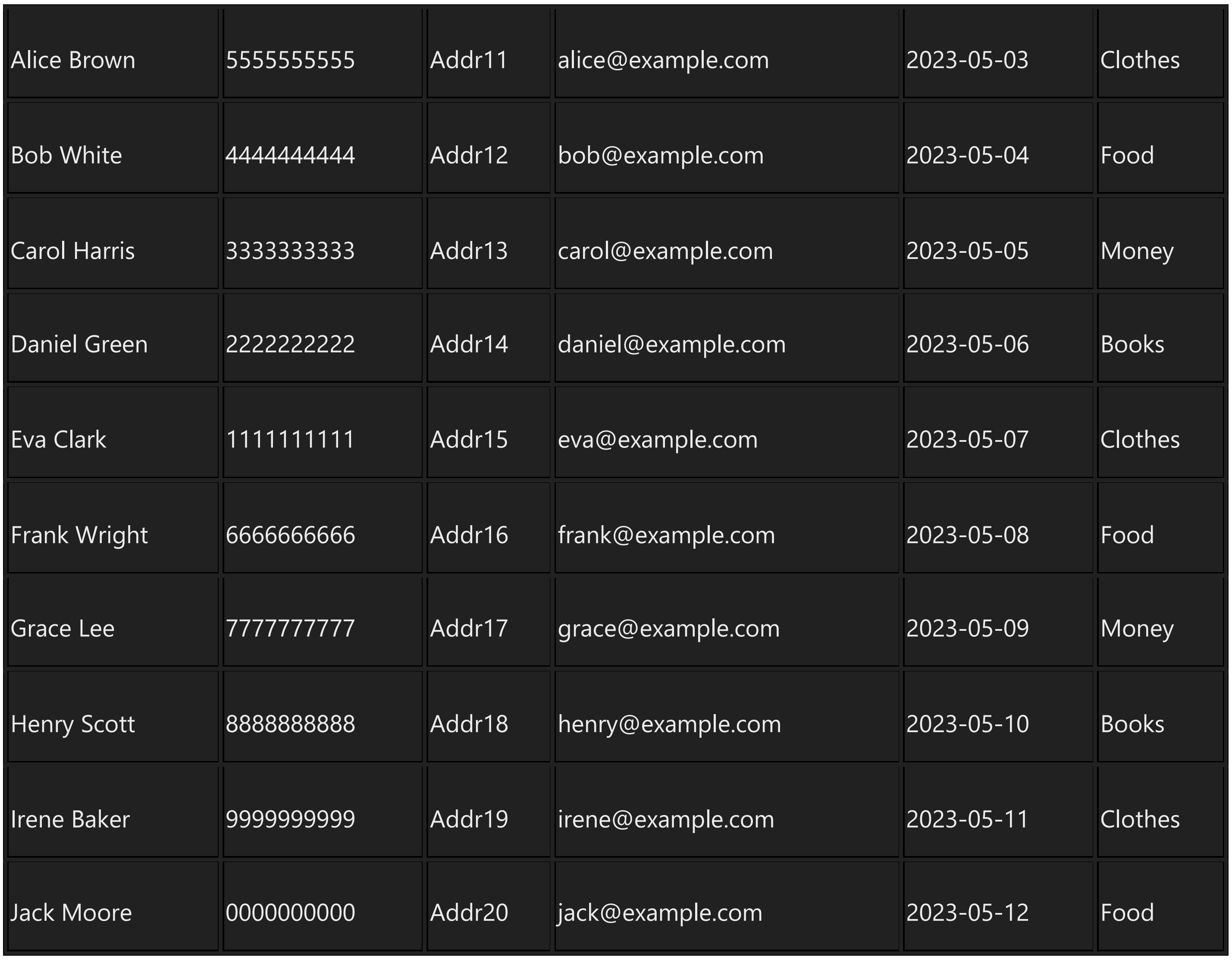
- ❖ DonorID (Primary Key)
- ❖ Donor Name
- ❖ Contact No
- ❖ Address
- ❖ Email



DonorID	Donor Name	Contact No	Address	Email
1	John Doe	1234567890	Addr1	john@example.com
2	Jane Smith	0987654321	Addr2	jane@example.com
3	Mike Johnson	2345678901	Addr3	mike@example.com
4	Emily Clark	3456789012	Addr4	emily@example.com
5	Paul Brown	4567890123	Addr5	paul@example.com
6	Laura Wilson	5678901234	Addr6	laura@example.com
7	Peter Davis	6789012345	Addr7	peter@example.com
8	Anna Lee	7890123456	Addr8	anna@example.com
9	David King	8901234567	Addr9	david@example.com
10	Sophia Hall	9012345678	Addr10	sophia@example.com

Donation Table:

- ❖ DonationID (Primary Key)
- ❖ DonorID (Foreign Key referencing DonorID in Donor Table)
- ❖ Date
- ❖ Donation Item



Alice Brown	5555555555	Addr11	alice@example.com	2023-05-03	Clothes
Bob White	4444444444	Addr12	bob@example.com	2023-05-04	Food
Carol Harris	3333333333	Addr13	carol@example.com	2023-05-05	Money
Daniel Green	2222222222	Addr14	daniel@example.com	2023-05-06	Books
Eva Clark	1111111111	Addr15	eva@example.com	2023-05-07	Clothes
Frank Wright	6666666666	Addr16	frank@example.com	2023-05-08	Food
Grace Lee	7777777777	Addr17	grace@example.com	2023-05-09	Money
Henry Scott	8888888888	Addr18	henry@example.com	2023-05-10	Books
Irene Baker	9999999999	Addr19	irene@example.com	2023-05-11	Clothes
Jack Moore	0000000000	Addr20	jack@example.com	2023-05-12	Food

Receiver Detail Table

1NF (First Normal Form)

Similarly, in 1NF, each column contains atomic values.

Original Receiver Detail Table:

Receiver Name	Contact No	Address	Email	Date	Received Item
Alice Brown	5555555555	Addr11	alice@example.com	2023-05-03	Clothes
Bob White	4444444444	Addr12	bob@example.com	2023-05-04	Food
Carol Harris	3333333333	Addr13	carol@example.com	2023-05-05	Money
Daniel Green	2222222222	Addr14	daniel@example.com	2023-05-06	Books
Eva Clark	1111111111	Addr15	eva@example.com	2023-05-07	Clothes
Frank Wright	6666666666	Addr16	frank@example.com	2023-05-08	Food
Grace Lee	7777777777	Addr17	grace@example.com	2023-05-09	Money
Henry Scott	8888888888	Addr18	henry@example.com	2023-05-10	Books
Irene Baker	9999999999	Addr19	irene@example.com	2023-05-11	Clothes
Jack Moore	0000000000	Addr20	jack@example.com	2023-05-12	Food

The table is in 1NF since all values are atomic.

2NF (Second Normal Form)

Again, to move to 2NF, we ensure that all non-key attributes are fully functional dependent on the primary key by introducing surrogate keys and splitting the table.

Receiver Detail Table Normalized into Two Tables:

<u>Receiver Table:</u>

- ❖ ReceiverID (Primary Key)
- ❖ Receiver Name
- ❖ Contact No
- ❖ Address
- ❖ Email

ReceiverID	Receiver Name	Contact No	Address	Email
1	Alice Brown	5555555555	Addr11	alice@example.com
2	Bob White	4444444444	Addr12	bob@example.com
3	Carol Harris	3333333333	Addr13	carol@example.com
4	Daniel Green	2222222222	Addr14	daniel@example.com
5	Eva Clark	1111111111	Addr15	eva@example.com
6	Frank Wright	6666666666	Addr16	frank@example.com
7	Grace Lee	7777777777	Addr17	grace@example.com
8	Henry Scott	8888888888	Addr18	henry@example.com
9	Irene Baker	9999999999	Addr19	irene@example.com
10	Jack Moore	0000000000	Addr20	jack@example.com

Received Item Table:

- ❖ ReceivedItemID (Primary Key)
- ❖ ReceiverID (Foreign Key referencing ReceiverID in Receiver Table)
- ❖ Date
- ❖ Received Item

ReceivedItemID	ReceiverID	Date	Received Item
1	1	2023-05-03	Clothes
2	2	2023-05-04	Food
3	3	2023-05-05	Money
4	4	2023-05-06	Books
5	5	2023-05-07	Clothes
6	6	2023-05-08	Food
7	7	2023-05-09	Money
8	8	2023-05-10	Books
9	9	2023-05-11	Clothes
10	10	2023-05-12	Food

4.PROGRAM CODE

```
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk
# Function to check login credentials
def check_login():
    username = username_entry.get()
    password = password_entry.get()
    # Simulate login check (replace with actual validation)
    if username == "admin" and password == "secret":
        # Login successful, open home window
        login_window.destroy()  # Close login window
        open_home_page()
    else:
        error_label.config(text="Invalid username or password!",
fg="red")

# Function to open the home page
def open_home_page():
    global home_window
    home_window = tk.Tk()
home_window.title("Charity Management System - Home")

    # Load background image for home screen
    home_background_image = Image.open("home5.jpg")

 # Replace with your image path
    resized_home_image=home_background_image.resize((1400,
650))

home_background_image=ImageTk.PhotoImage(resized_home_imag
e)

    # Keep a reference to the image object

home_window.home_background_image=home_background_image

    # Create a label for the home screen background image
    home_background_label=tk.Label(home_window,
image=home_background_image)
    home_background_label.place(relwidth=1, relheight=1)

  # Create buttons for donor, receiver, and logout
    donor_button=tk.Button(home_window,text="Donor",
command=open_donor_page, font=("Helvetica", 16), width=15,
height=2)
    donor_button.place(x=150, y=150)

    receiver_button=tk.Button(home_window,text="Receiver",
command=open_receiver_page, font=("Helvetica", 16), width=15,
height=2)
```

```
receiver_button.place(x=150,y=230)logout_button=tk.Button(home_
window,text="Logout",command=logout_function, font=("Helvetica",
16), width=15, height=2)
logout_button.place(x=150, y=310)
 home_window.mainloop()

# Function to open the donor detail page
def open_donor_page():
    donor_window = tk.Toplevel()
    donor_window.title("Donor Details")
# Load background image for donor detail page
 donor_background_image = Image.open("charity.jpg")  # Replace
with your image path
 resized_donor_image = donor_background_image.resize((1400,
650))

donor_background_image=ImageTk.PhotoImage(resized_donor_ima
ge)

 # Keep a reference to the image object
    donor_window.donor_background_image                              =
donor_background_image

# Create a label for the donor detail page background image
donor_background_label=tk.Label(donor_window,
image=donor_background_image)
    donor_background_label.place(relwidth=1, relheight=1)
 # Donor name label and entry
    donor_name_label = tk.Label(donor_window, text="Donor Name:")
    donor_name_label.grid(row=0, column=0, padx=10, pady=10)
    donor_name_entry = tk.Entry(donor_window)
    donor_name_entry.grid(row=0, column=1, padx=10, pady=10)

    # Contact number label and entry
    donor_contact_label = tk.Label(donor_window, text="Contact
Number:")
    donor_contact_label.grid(row=1, column=0, padx=10, pady=10)
    donor_contact_entry = tk.Entry(donor_window)
    donor_contact_entry.grid(row=1, column=1, padx=10, pady=10)

    # Address label and entry
    address_label = tk.Label(donor_window, text="Address:")
    address_label.grid(row=2, column=0, padx=10, pady=10)
    address_entry = tk.Entry(donor_window)
    address_entry.grid(row=2, column=1, padx=10, pady=10)

    # Donation item label and entry
    donation_item_label=tk.Label(donor_window, text="Donating
Item:")
    donation_item_label.grid(row=3, column=0, padx=10, pady=10)
    donation_item_entry = tk.Entry(donor_window)
    donation_item_entry.grid(row=3, column=1, padx=10, pady=10)

    # Donated cash amount label and entry
    cash_amount_label = tk.Label(donor_window, text="Donated Cash 
Amount:")
    cash_amount_label.grid(row=4, column=0, padx=10, pady=10)
    cash_amount_entry = tk.Entry(donor_window)
    cash_amount_entry.grid(row=4, column=1, padx=10, pady=10)

    # Save button
    save_button=tk.Button(donor_window,text="Save",command=
lambda:save_details("Donor",donor_name_entry.get(),
donor_contact_entry.get(),address_entry.get(),
donation_item_entry.get(), cash_amount_entry.get()))
    save_button.grid(row=5, columnspan=2, padx=10, pady=10)

# Function to open the receiver detail page
def open_receiver_page():
    receiver_window = tk.Toplevel()
    receiver_window.title("Receiver Details")

    # Load background image for receiver detail page
    receiver_background_image = Image.open("charity.jpg")  # Replace 
with your image path
    resized_receiver_image= receiver_background_image.resize((1400, 
650))
```

```
receiver_background_image=ImageTk.PhotoImage(resized_receiver_
image)

    # Keep a reference to the image object
receiver_window.receiver_background_image=receiver_background_
image

    # Create a label for the receiver detail page background image
    receiver_background_label=tk.Label(receiver_window,
image=receiver_background_image)
    receiver_background_label.place(relwidth=1, relheight=1)

    # Receiver name label and entry
    receiver_name_label=   tk.Label(receiver_window,   text="Receiver
Name:")
    receiver_name_label.grid(row=0, column=0, padx=10, pady=10)
    receiver_name_entry = tk.Entry(receiver_window)
    receiver_name_entry.grid(row=0, column=1, padx=10, pady=10)

    # Contact number label and entry
    receiver_contact_label  =  tk.Label(receiver_window,  text="Contact
Number:")
    receiver_contact_label.grid(row=1, column=0, padx=10, pady=10)
    receiver_contact_entry = tk.Entry(receiver_window)
    receiver_contact_entry.grid(row=1, column=1, padx=10, pady=10)

    # Address label and entry
    address_label = tk.Label(receiver_window, text="Address:")
    address_label.grid(row=2, column=0, padx=10, pady=10)
    address_entry = tk.Entry(receiver_window)
    address_entry.grid(row=2, column=1, padx=10, pady=10)

    # Received item label and entry
    received_item_label = tk.Label(receiver_window, text="Received
Item:")
    received_item_label.grid(row=3, column=0, padx=10, pady=10)
    received_item_entry = tk.Entry(receiver_window)
    received_item_entry.grid(row=3, column=1, padx=10, pady=10)

    # Received cash amount label and entry
    cash_amount_label = tk.Label(receiver_window, text="Received
Cash Amount:")
    cash_amount_label.grid(row=4, column=0, padx=10, pady=10)
    cash_amount_entry = tk.Entry(receiver_window)
    cash_amount_entry.grid(row=4, column=1, padx=10, pady=10)

# Save button

save_button=tk.Button(receiver_window,text="Save",command=lamb
da:save_details("Receiver",receiver_name_entry.get(),receiver_contac
t_entry.get(),address_entry.get(),received_item_entry.get(),
cash_amount_entry.get()))
    save_button.grid(row=5, columnspan=2, padx=10, pady=10)
```

```
# Function to save details
def save_details(role, name, contact, address, item, cash_amount):
   messagebox.showinfo("Saved", f"{role} Details Saved:\nName: {name}\nContact:{contact}\nAddress:{address}\nItem: {item}\nCash Amount: {cash_amount}")

# Function to logout
def logout_function():
   home_window.destroy()
 # Close home window
   login_window.deiconify()
# Show login window again

# Create the main window for login
login_window = tk.Tk()
login_window.title("Charity Management System - Login")

# Load background image for login screen
background_image = Image.open("charity.jpg")
# Replace with your image path
resized_image = background_image.resize((1400, 650))
background_image = ImageTk.PhotoImage(resized_image)

# Create a label for the login screen background image
background_label=tk.Label(login_window,image=background_image
)
```

```
background_label.place(relwidth=1, relheight=1)

# Create login frame
login_frame = tk.Frame(login_window, bg="white", padx=20, pady=20)
login_frame.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

# Username label and entry
username_label = tk.Label(login_frame, text="Username:")
username_label.pack()
username_entry = tk.Entry(login_frame)
username_entry.pack()

# Password label and entry
password_label = tk.Label(login_frame, text="Password:")
password_label.pack()
password_entry = tk.Entry(login_frame, show="*")
password_entry.pack()

# Error message label (initially empty)
error_label = tk.Label(login_frame, text="")
error_label.pack()

# Login button
```

```
login_button=tk.Button(login_frame,text="Login",command=check_l
ogin)
login_button.pack()

# Run the main loop
login_window.mainloop()
```

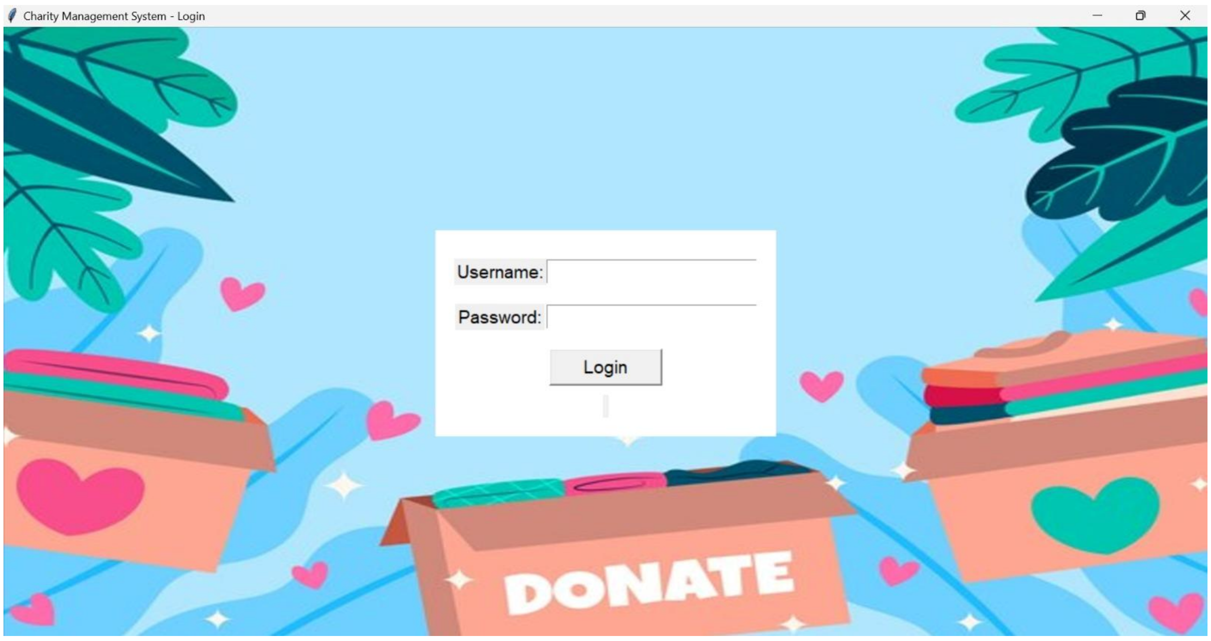
SQL CODE:

```
CREATE DATABASE charity_management2;
USE charity_management2;
CREATE TABLE donors (id INT AUTO_INCREMENT PRIMARY
KEY,
   name VARCHAR(100),
   contact VARCHAR(15),
   email VARCHAR(100),
   address VARCHAR(255),
   donation_item VARCHAR(100),
   date DATE
);

CREATE TABLE receivers (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100),
    contact VARCHAR(15),
    email VARCHAR(100),
    address VARCHAR(255),
    received_item VARCHAR(100),
    date DATE
);

DELIMITER //
CREATE TRIGGER before_donor_insert
BEFORE INSERT ON donors
FOR EACH ROW
BEGIN
    SET NEW.date = CURDATE();
END//
DELIMITER ;


DELIMITER //
CREATE TRIGGER before_receiver_insert
BEFORE INSERT ON receivers
FOR EACH ROW
BEGIN
    SET NEW.date = CURDATE();
END//
DELIMITER ;
```

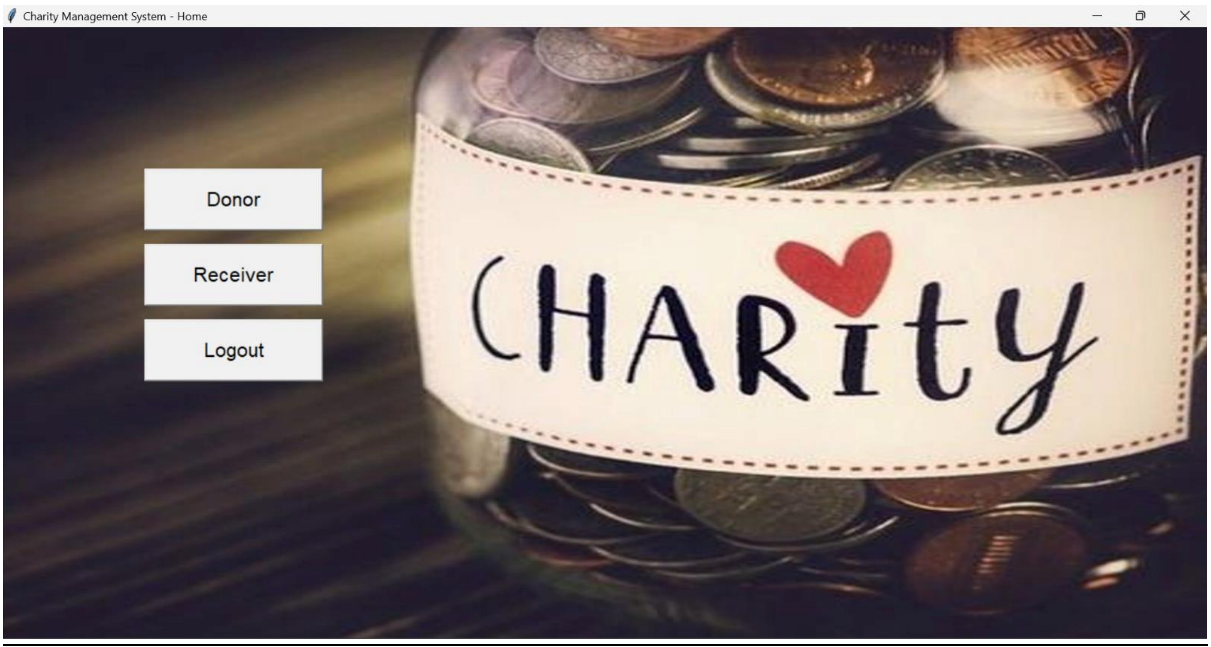
5. RESULTS AND DISCUSSION

Results

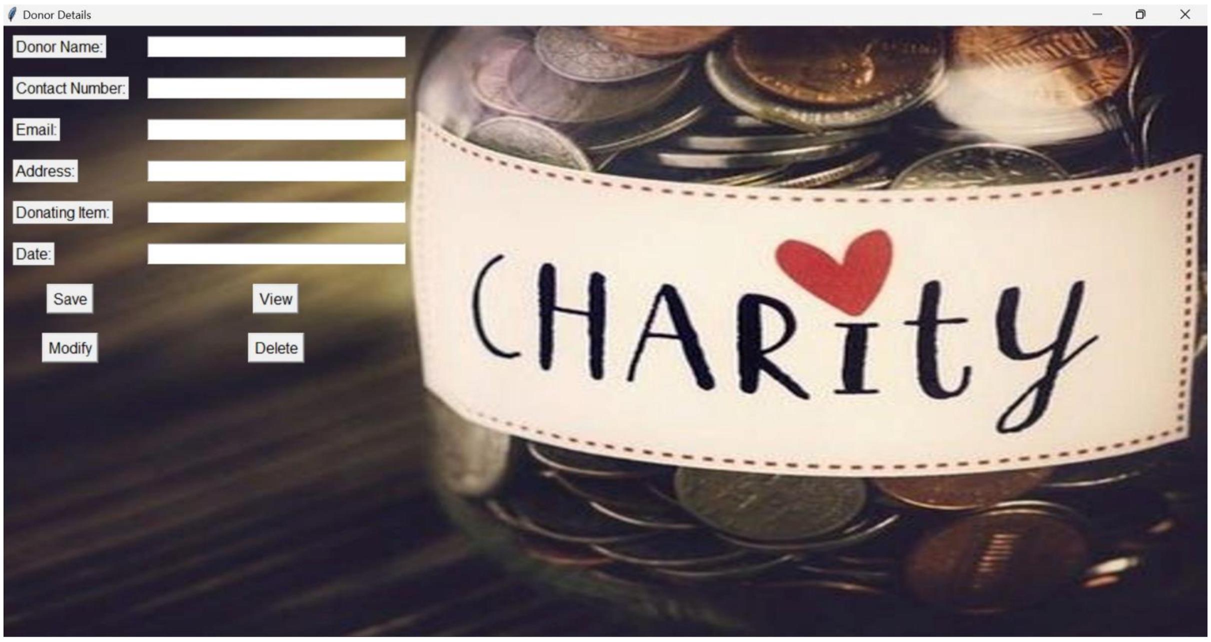
LOGIN PAGE:



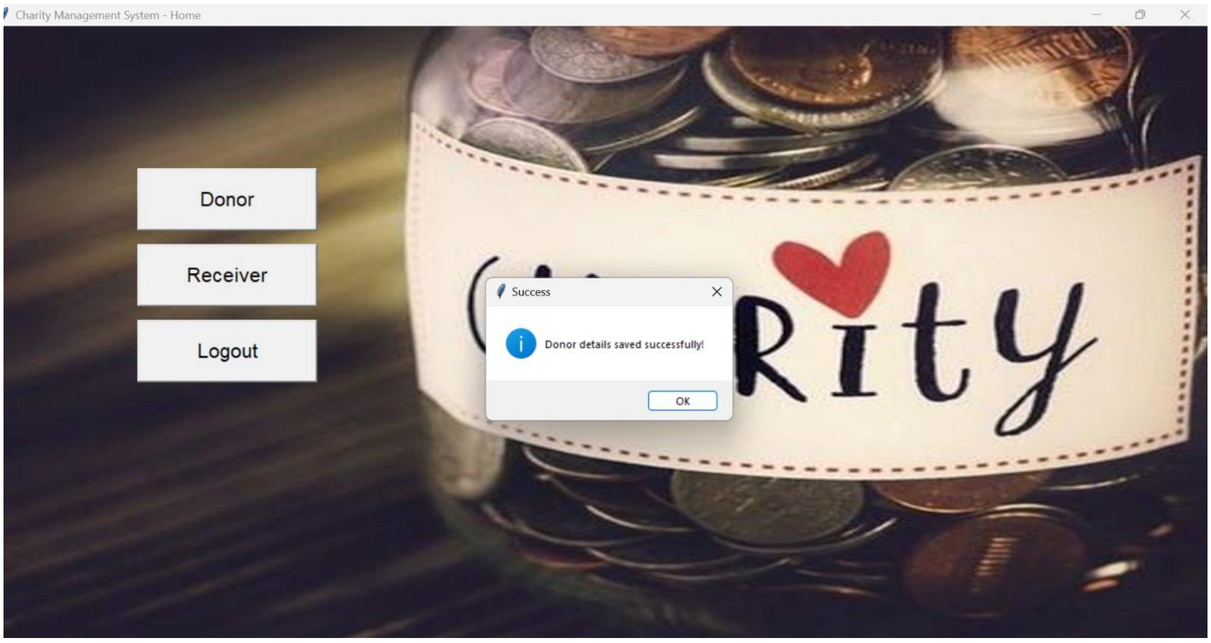
HOME PAGE:



DONOR PAGE:

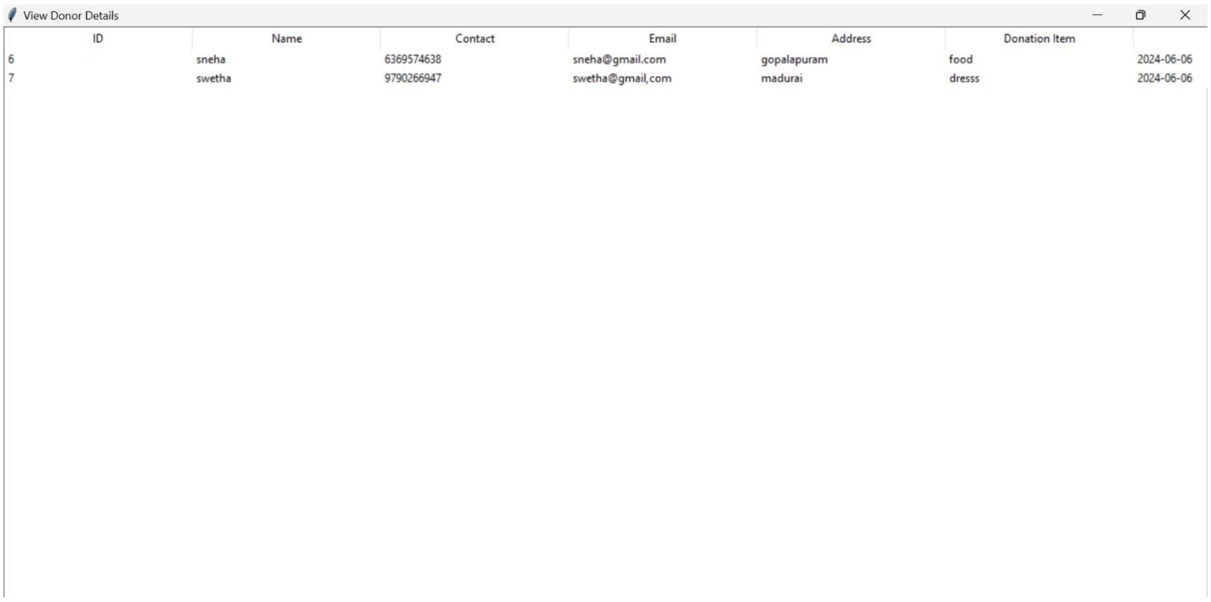


DONOR DETAILS:



Donor details saved successfully!

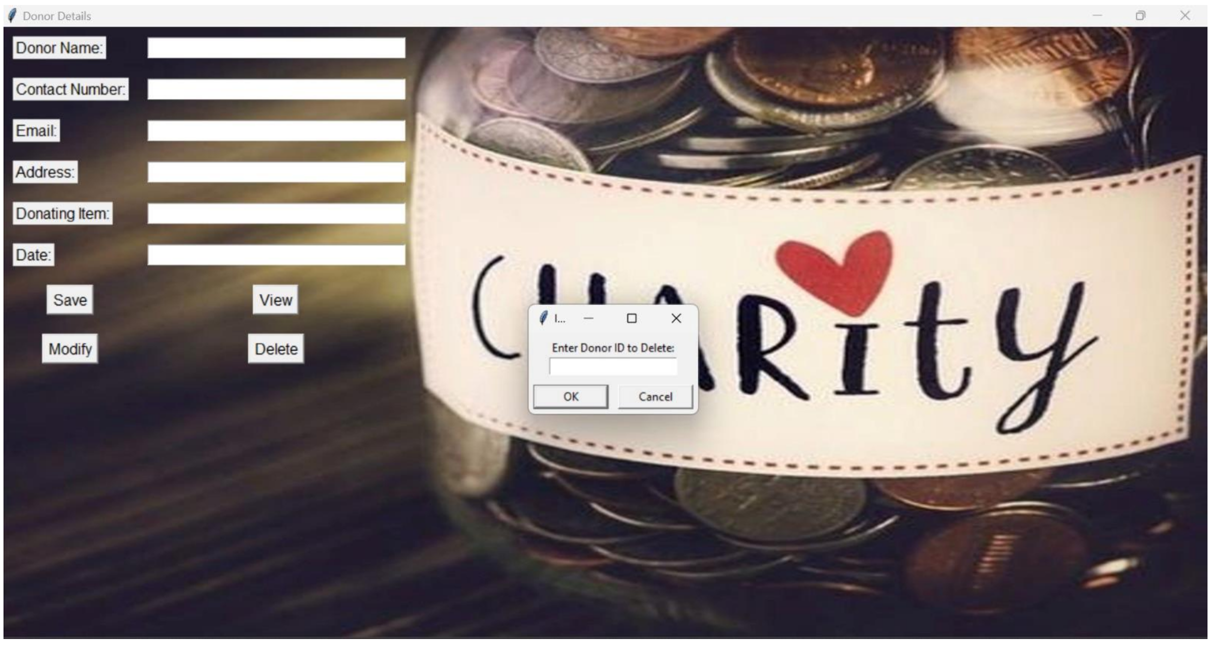
VIEW DONOR DETAILS:



View Donor Details

ID	Name	Contact	Email	Address	Donation Item	
6	sneha	6369574638	sneha@gmail.com	gopalapuram	food	2024-06-06
7	swetha	9790266947	swetha@gmail,com	madurai	dresss	2024-06-06

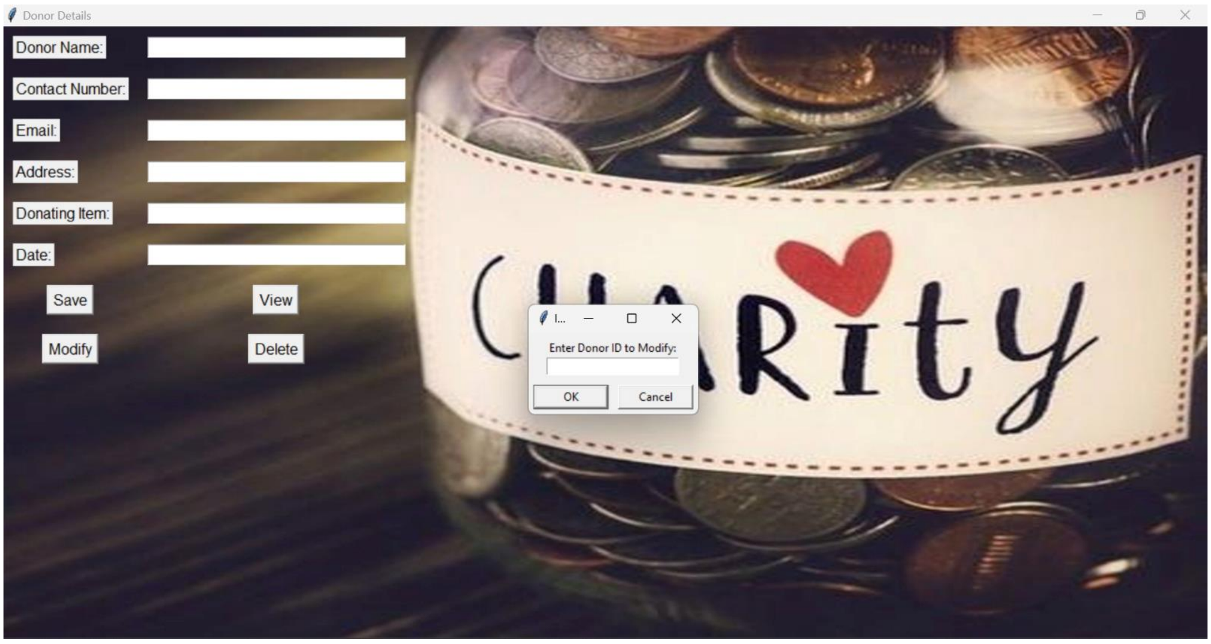
DONOR DELETE:



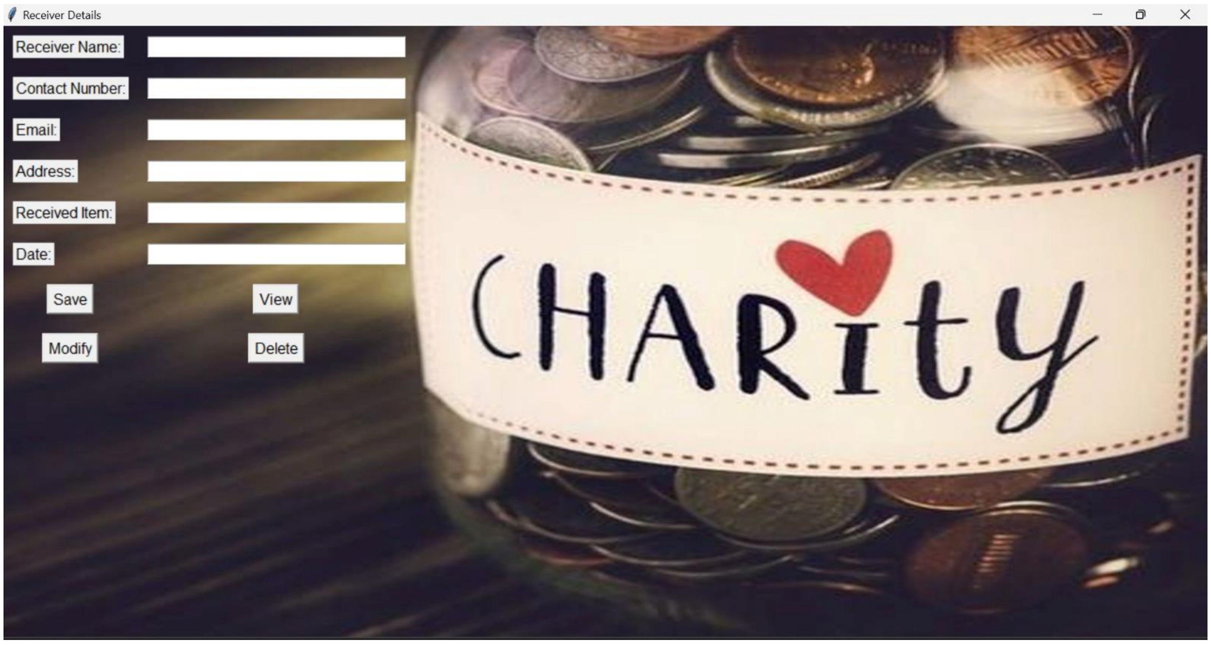
Donor Details

Donor Name:	
Contact Number:	
Email:	
Address:	
Donating Item:	
Date:	

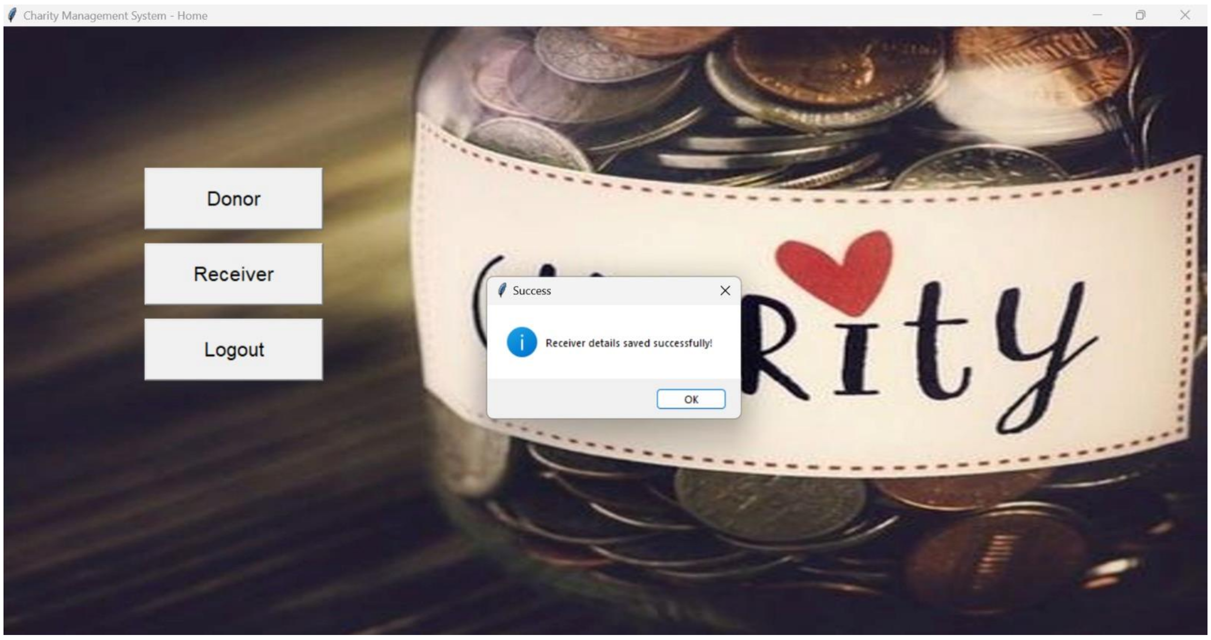
DONOR MODIFY:



RECEIVER PAGE:

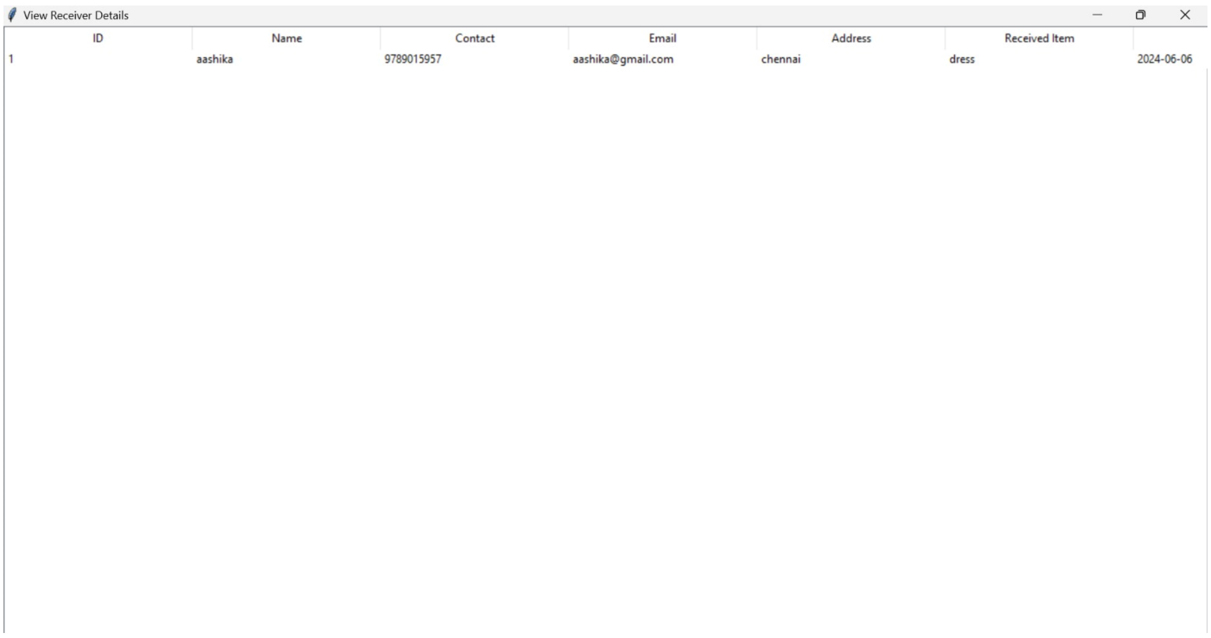


RECEIVER DETAILS:



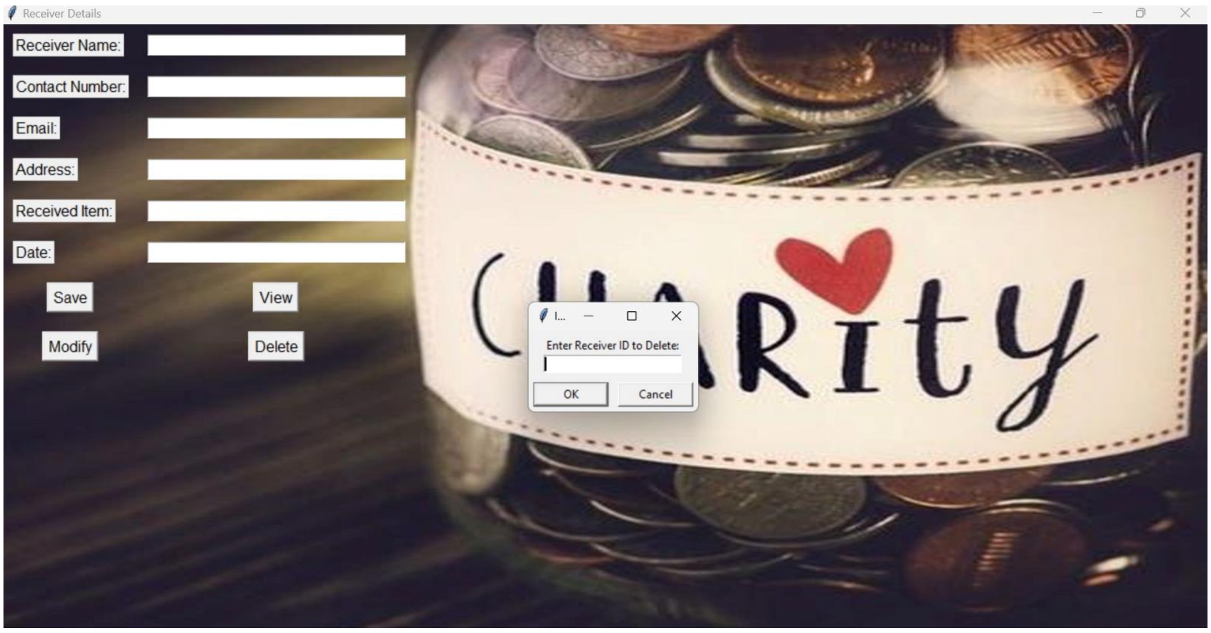
Receiver details saved successfully!

VIEW RECEIVER DETAILS :

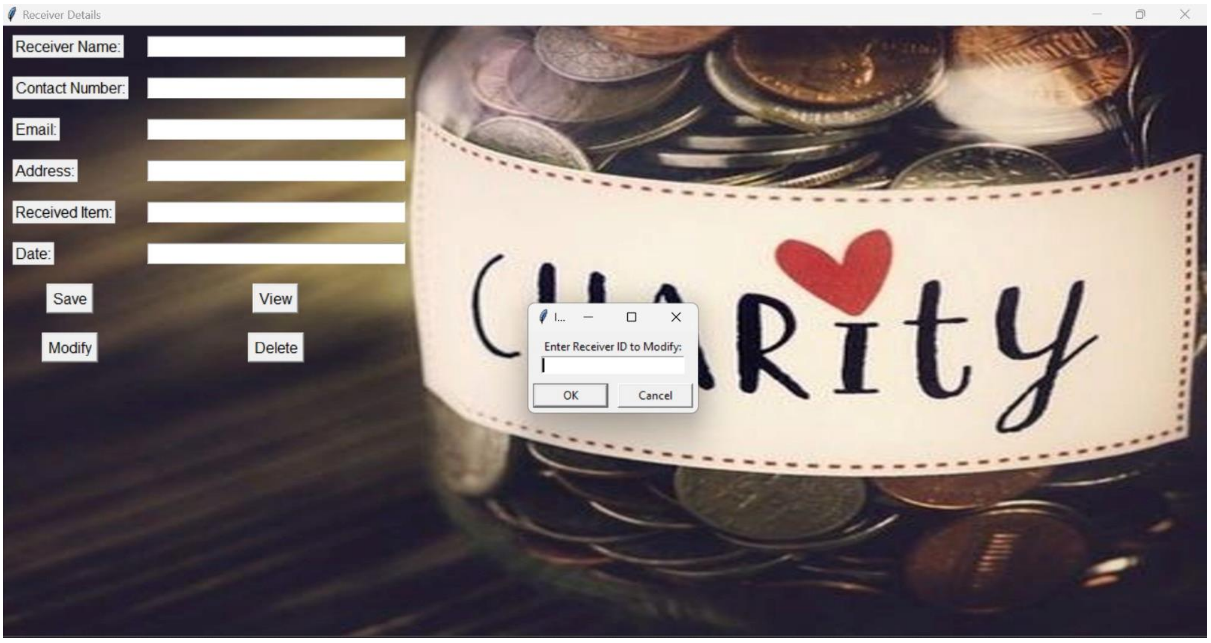


ID	Name	Contact	Email	Address	Received Item	
1	aashika	9789015957	aashika@gmail.com	chennai	dress	2024-06-06

RECEIVER DELETE:



RECEIVER MODIFY:



6.CONCLUSION

The Charity Management System effectively addresses the needs of managing donor and receiver information for charitable organizations. The system's robust architecture, combined with its user-friendly interface, ensures efficient and secure management of charity operations. Ongoing improvements and expansions will continue to enhance the system's functionality and adaptability, supporting the charity's mission and growth.

The Charity Management System has proven to be a valuable tool for charitable organizations, providing a structured and efficient way to manage donor and receiver information. By addressing the needs of data integrity, ease of use, and scalability, the system lays a strong foundation for supporting the mission of charitable organizations. Continuous improvements and updates will ensure that the system remains relevant and effective in meeting the evolving needs of its users.

In conclusion, the successful implementation of this project demonstrates the potential for technology to significantly enhance the operational capabilities of charitable organizations, thereby contributing to their overall mission and impact.

7.REFERENCES

1 Manish Kumar Srivastava, A.K Tiwari, “A Study of Behavior of Maruti SX4 and Honda City Customers in Jaipur”, Pacific Business Review- Quarterly Referred Journal, Zenith International Journal of Multi disciplinary Research Vol.4, Issue 4, pp. 77-90, Apr2011.

2 M.Prasanna Mohan Raj, Jishnu Sasikumar, S.Sriram , “A Study of Customers Brand Preference in SUVS and MUVS: Effect on Marketing Mix Variables”, International Referred Research Journal Vol.- IV, Issue-1, pp. 48-58, Jan2013.

3 Nikhil Monga, Bhuvender Chaudhary, “Car Market and Buying behavior - study on Consumer Perception”, IJRMEC Vol.2, Issue-2, pp. 44-63, Feb2012.