PROJECT SUBMITTED UNDER THE TITLE

"DONATION MANAGEMENT SYSTEM"

A Dissertation submitted to
Bharathidasan University, Tiruchirapalli.
For the partial fulfillment of the requirement for the
Award of the Degree of

BACHELOR OF COMPUTER SCIENCE

Ms. T. THARSHINI	**Reg. No: CB21S205414**
Ms. S. NIKITHA	**Reg. No: CB21S205371**
Ms. S. SIVASAKTHI	**Reg. No: CB21S205401**

Under the Guidance of
Dr. K. KALYANI,MCA.,M.Phil,Ph.D
Head of the Department,
Department of Computer Science

BON SECOURS COLLEGE FOR WOMEN
Accredited with 'A++' Grade by NAAC in Cycle II
Recognized 2(f) and 12(B) Institution
Affiliated to Bharathidasan University, Trichy.
Vilar Bypass, Thanjavur – 613 006.
APRIL-2024

DEPARTMENT OF COMPUTER SCIENCE
BON SECOURS COLLEGE FOR WOMEN
Accredited with 'A++' Grade by NAAC in Cycle II
Recognized 2(f) and 12(B) Institution
Affiliated to Bharathidasan University, Trichy.
Vilar Bypass, Thanjavur – 613 006.

BONAFIDE CERTIFICATE

This is to certify that the project work entitled **" DONATION MANAGEMENT SYSTEM"** is a Bonafide work done by **T. THARSHINI (Reg. No: CB21S205414)** in partial fulfilment of the requirement for the B.Sc., Degree Course in Computer Science during the year of 2023-2024.

Signature of the Internal guide	**Signature of Head of the Department**
Dr.K .Kalyani,MCA.,M.Phil,Ph.D	Dr.K .Kalyani,MCA.,M.Phil,Ph.D
Head of the Department,	Head of the Department,
Department of Computer Science.	Department of Computer Science.

Submitted for project Viva-voce examination held on…………………………………….at BON SECOURS COLLEGE FOR WOMEN, Thanjavur.

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

First of all, I would like to express my sincere thanks to God Almighty, for his constant love and grace that he has showered upon me. I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. Their guidance, inspiration suggestions, cooperation and criticisms always script me on the right track.

I am very grateful to our honorable Secretary **Rev. Sr. Dr. MARIAMMAL, FBS,** Bon Secours College for Women, Thanjavur for her support and encouragement during the course of my study.

I would like to express my Heartfelt gratitude and sincere thanks to our honourable Director **Rev. Sr. TERENCIA MARY, FBS**, Bon Secours College for Women, Thanjavur for her endless support and motivation during the course of my study.

I would like to express my sincere thanks and express my heartfelt gratitude to our honourable Principal **Dr. S. GAYATHRI,** Bon Secours College for Women, Thanjavur, for extending all resources that facilitated the conduct of the present study.

I would like to thank of Vice Principals **Dr.R.KALAIVANI, Dr.M.FLORENCE DAYANA** for encouraging and supporting through the project.

I am extremely thankful and pay my gratitude to my Head and Assistant Professor **Dr. K. KALYANI** for her valuable guidance and support on completion of this project in its presently.

I am highly indebted to my Project Guide **Dr. K. KALYANI** Head of the Department, Department of Computer Science for her valuable timely guidance and support to bring it out in the project in a successful way.

Last yet importantly, I would like to thank my parents and all my Well Wishers for their kind inspiration.

T. THARSHINI

-

CONTENT

INTRODUCTION

ABSTRACT

This project proposes the development of a donation management system using Python. The system aims to streamline the process of managing donations for organizations, charities, and non-profits. Key functionalities include donor registration, donation tracking, inventory management for donated items, financial tracking, reporting, and communication tools for donors and administrators.

The system will leverage Python's versatility, along with libraries such as Flask for web development, SQLAlchemy for database management, and Pandas for data analysis. Through this project, organizations can efficiently manage and track donations, enhancing transparency and accountability in their operations.

SYSTEM SPECIFICATION

HARDWARE REQUIREMENTS

Processor - i7

Hard Disk - 512 GB SSD

Memory - 16 GB RAM

SOFTWARE REQUIREMENTS

Front End - Python 3.12(64 bit)

Back End - SQLite 3

Operating System - Windows 11

Software - IDLE python

SOFTWARE SPECIFICATION

ABOUT SOTWARE

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, **https://www.python.org/,** and may be freely distributed. The same site also contains distributions of and pointers to many free third-party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

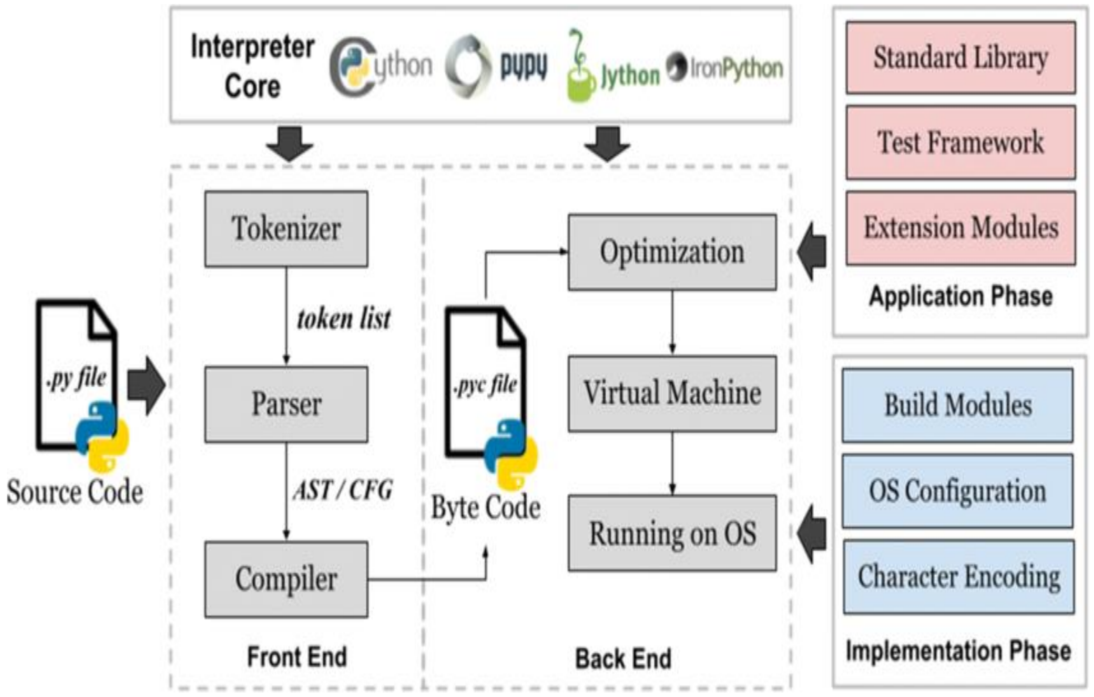
This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see **The Python Standard Library. The Python Language Reference** gives a more formal definition of the language. To write extensions in C or C++, read **Extending and Embedding the**

Python Interpreter and **Python/C API Reference Manual.** There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in **The Python Standard Library.**

Architecture



Most popular language

Python is the most popular and widely used computer programming language, particularly in the fields of Data Science and Machine Learning. What's more, Python is a cross-platform language that runs on a variety of operating systems, such as Windows, macOS and Linux, making it an ideal choice for developers working on different environments. Find out all you need to know about the Python language: origins, usage, tools, advantages, disadvantages, training.

Python is a general-purpose computer programming language. Unlike HTML, CSS or JavaScript, its use is not limited to web development. It can be used for all types of programming and software development.

In particular, it is used for the back-end development of web and mobile applications, and for the development of PC software and applications. It can also be used to write system scripts, to create instructions for a computer system.

Python is also the most popular computer language for Big Data processing, mathematical calculations and Machine Learning. Generally speaking, it is the language of choice for Data Science.

Python library

Python's libraries are one of the main reasons for its success. It's a vast ecosystem of software developed by third parties. This collection has been enriched and extended over the decades. Several standard libraries are available, offering modules adapted to the most common programming tasks: networking, asynchronous operations, threading, file access…

Some modules can also handle the high-level programming tasks required by modern applications. These may include reading and writing structured file formats such as JSON and XML, manipulating compressed files, or working with web protocols and data formats.

The default **Python distribution** also offers a cross-platform GUI library with Tkinter, and an integrated copy of the SQLite 3 database. In addition to these native libraries, thousands of third-party libraries are available via the Python Package Index (PyPI). It is these libraries that give the language its versatility.

There are many Python libraries that can be useful, depending on the field of application and specific needs. However, here are a few of the main Python libraries that are recommended to know:

- **NumPy:** a library for mathematical and numerical operations on arrays and matrices.
- Pandas: a library for array data manipulation and analysis.
- **Matplotlib:** a library for creating graphs and data visualizations.
- **Scikit-learn:** a library for machine learning and data ining.

- **TensorFlow:** a library for deep learning and neural network model development.
- **PyTorch:** a library for deep learning and the creation of neural network_models.

  Beautiful Soup: a library for parsing HTML and XML data.
- **Requests:** a library for sending HTTP requests.
- **Flask and Django:** frameworks for developing web services. These libraries are very popular and widely used in data science, machine learning, data analysis and web development.

Platform independence

Programming language Python is a binary platform independent. The same Python code may be executed on practically any platform or operating system. When working with Python, numerous measures must be taken to minimize compatibility issues, such as paying attention to case sensitivity and avoiding specific modules.

➡ Python programs can be written and run on a variety of operating systems.

➡ Python can be used on a variety of platforms, including Linux, Windows, Macintosh, Solaris, and others.

Type Safety

Type Safety in Python. Python is a semi type safe language. Python is a dynamically and strongly typed language that has a high degree of type safety control built-in. But the type checking is done only at the run time.

Python's readability and ease of understanding contribute to the security of applications, but it's important to note that this does not directly make Python more secure than other languages. Its clear syntax and coding practices make code easier to read, understand, and maintain.

Portability

Python is highly portable due to its cross-platform compatibility and extensive community support. It can run on various operating systems, making it accessible on different machines. **So, some c like languages is portable example is C,** python is portable because you don't have recompile, build. If the code is written well and in a generic form it will run cross platform. C code because the operating systems have some C or a lot of C's can do the same as well.

Interoperability

Python API interoperability is an important requirement for this project. While Swift is designed to integrate with other programming languages (and their runtimes), the nature of dynamic languages does not require the deep integration needed to support static languages.

Security

Python, as with any programming language, requires attention to security considerations to ensure the integrity and confidentiality of applications and data. One fundamental aspect is input validation, which guards against injection attacks such as SQL injection and code injection. Developers must adhere to secure coding practices, such as avoiding hardcoded credentials and utilizing trusted libraries for cryptographic operations, to mitigate risks like cross-site scripting and cross-site request forgery.

Robust authentication and authorization mechanisms are essential for controlling access to resources and preventing unauthorized actions. Additionally, implementing secure communication protocols like HTTPS and encrypting sensitive data during transmission are crucial to thwart eavesdropping and data manipulation attempts. Regularly updating and patching dependencies is vital to address known vulnerabilities. Rigorous code reviews, testing procedures encompassing static analysis, dynamic analysis, and penetration testing, and secure deployment configurations further bolster the overall security posture of Python applications. By integrating these measures and staying abreast of evolving threats and security best practices, developers can fortify Python applications against potential exploits and breaches.

Security in Python can be addressed through various measures such as:

Input validation: Ensuring that user inputs are properly validated to prevent injection attacks like SQL injection or code injection.

Secure coding practices: Following best practices such as avoiding hardcoded credentials, using secure libraries for cryptographic operations, and sanitizing inputs to prevent vulnerabilities like cross-site scripting (XSS) and cross-site request forgery (CSRF).

Authentication and authorization: Implementing strong authentication mechanisms and access control to ensure that only authorized users can access resources and perform actions.

Secure communication: Using secure communication protocols like HTTPS and implementing encryption for sensitive data transmission.

Dependency management: Regularly updating and patching dependencies to mitigate known vulnerabilities.

Code review and testing: Performing thorough code reviews and testing, including static analysis, dynamic analysis, and penetration testing, to identify and fix security flaws.

Deployment security: Securing the deployment environment by properly configuring servers, firewalls, and other infrastructure components.

By implementing these measures and staying informed about the latest security threats and best practices, you can enhance the security of Python applications.

Memory Management

Memory management in Python is primarily handled by the Python runtime through a mechanism called automatic memory management or garbage collection. Python's memory management system employs a combination of reference counting and a garbage collector to efficiently allocate and deallocate memory for objects. When an object is created, Python assigns memory to it and initializes its attributes. The reference count of the object is then set to 1.

As references to the object are created, for example, by assigning it to variables or passing it as arguments to functions, the reference count is incremented. Conversely, when references are deleted or go out of scope, the reference count is decremented. When the reference count of an object reaches zero, meaning there are no more references to it, Python's garbage collector automatically deallocates the memory associated with the object, freeing up resources for reuse. This automated memory management system relieves developers from manually managing memory allocation and deallocation, reducing the likelihood of memory leaks and making Python a convenient and efficient language for development. However, developers should still be mindful of certain scenarios, such as circular references, that can lead to memory leaks if not handled properly.

In Python's IDLE (Integrated Development and Learning Environment), memory management is handled similarly to standard Python runtime environments. IDLE uses the same underlying Python interpreter and therefore benefits from Python's automatic memory management mechanisms, including reference counting and garbage collection.

Developers using IDLE can rely on Python's built-in memory management features to handle memory allocation and deallocation transparently. This allows for efficient memory usage without the need for manual intervention, simplifying the development process and reducing the risk of memory-related errors. However, as with any Python environment, developers should still be aware of potential memory leaks and inefficiencies in their code and follow best practices to ensure optimal memory usage.

Performance

Python offers several tools and libraries to help measure and optimize performance, which can be valuable for documentation purposes. One commonly used tool is the timeit module, which allows developers to measure the execution time of small code snippets. By including timeit benchmarks in documentation, users can understand the performance implications of different approaches or optimizations.

Profiling tools like cProfile and line_profiler provide detailed insights into code execution, helping identify bottlenecks and areas for improvement. Including profiling results in documentation can help users understand where optimizations are needed and how to prioritize them effectively. For larger projects, tools like Py-Spy and SnakeViz offer visualization capabilities, making it easier to understand performance profiles and identify areas for optimization at a higher level.

Additionally, documenting performance considerations, such as algorithmic complexity and common pitfalls, can help users write efficient code from the outset.

Providing examples and benchmarks alongside explanations can further aid comprehension and implementation. Overall, integrating performance analysis and optimization guidance into documentation can empower users to write more efficient Python code and make informed decisions about trade-offs between performance and other factors.

Python Implementation

An "implementation" of Python should be taken to mean a program or environment which provides support for the execution of programs written in the Python language, as represented by the CPython reference implementation.

There have been and are several distinct software packages providing what we all recognize as Python, although some of those are more like distributions or variants of some existing implementation than a completely new implementation of the language.

It is important to keep in mind that alternative Python implementations exist. They may provide a speedup or other benefit depending on your specific usecase or environment. On the downside they do not always support all features of CPython and may not support all packages.

Code-behind model

In Python, a model typically refers to a class that represents a specific entity or concept within a software application, often corresponding to data stored in a database. A model class encapsulates the attributes and behaviors associated with that entity. For instance, a user model might have attributes like username, email, and password, along with methods for updating the user's information or performing authentication tasks.

Directives

In Python, a directive typically refers to a statement that provides instructions to the interpreter or compiler regarding how to handle certain aspects of the code. One common directive is the #!/usr/bin/env python shebang line at the beginning of a script, which tells the system which interpreter to use to execute the script. Another directive is # -- coding: utf-8 --, which specifies the encoding of the Python source code file. Directives are not Python code themselves but rather special comments or declarations that affect how the code is interpreted or executed.

User Controls

User controls to graphical user interface (GUI) elements that allow users to interact with a program. Python provides several libraries for creating GUI applications, including Tkinter, PyQt, PyGTK, and wxPython.

Common user controls in Python GUI applications include:

Buttons: Used to trigger actions when clicked.

Textboxes: Input fields for users to enter text.

Labels: Static text or images used for displaying information.

Checkboxes: Allow users to select or deselect options.

Radio buttons: Allow users to choose one option from a list.

Combo boxes: Dropdown lists for selecting options.Sliders: Used for selecting a value from a range by dragging a slider handle.

Menus: Dropdown menus for accessing different commands or options.

These controls can be created and customized using the respective libraries' classes and methods. For example, in Tkinter, you would create a button using the Button class, while in PyQt, you would use the QPushButton class. Each control can be configured with various properties and event handlers to define its appearance and behavior in the GUI application.

State Management

In Python IDLE, state management refers to the ability to inspect and modify the state of objects and variables within the current session. However, IDLE itself doesn't have built-in features specifically dedicated to state management or documentation generation.To document your code within Python IDLE, you can use comments and docstrings. Comments are denoted by the # symbol and are used to annotate code for human readers. Docstrings are multi-line strings enclosed in triple quotes (''' or """) and are used to document functions, classes, and modules.

To manage state within Python IDLE, you can use the interactive interpreter to inspect variables and objects. For example, after executing code that defines variables or functions, you can directly interact with those variables and functions in the interpreter.If you need more sophisticated state management, you might want to consider using Python scripts or interactive notebooks like Jupyter Notebook, which provide more advanced features for state management and documentation.

Application

Interactive Exploration and Documentation: IDLE provides an interactive environment where you can test code snippets, explore libraries, and document your findings directly within the interpreter. You can write comments to explain your thought process and the functionality of different code segments.

Docstring Support: IDLE allows you to write and view docstrings easily. You can document functions, classes, and modules using docstrings, and IDLE provides features to view these docstrings interactively. This helps in maintaining well-documented code.

Code Analysis and Inspection: IDLE's built-in features, such as code completion and call tips, can aid in documenting code. When you're writing or exploring code, IDLE can provide suggestions and information about function parameters, return types, and more, which can be useful for documentation purposes.

Creating Python Scripts for Documentation: While IDLE is primarily an interactive environment, you can also use it to write Python scripts. You can create scripts that generate documentation from source code comments or docstrings using tools like Sphinx or Doxygen.

Testing Documentation Examples: IDLE allows you to execute code snippets interactively. This feature can be handy for testing examples provided in documentation or tutorials. You can run the examples directly in IDLE and verify their functionality.

About Database

Database Software

Database software is used to create, edit, and maintain database files and records, enabling easier file and record creation, data entry, data editing, updating, and reporting. The software also handles data storage, backup and reporting, multi-access control, and security. Strong database security is especially important today, as data theft becomes more frequent. Database software is sometimes also referred to as a "database management system" (DBMS).

Database software makes data management simpler by enabling users to store data in a structured form and then access it. It typically has a graphical interface to help create and manage the data and, in some cases, users can construct their own databases by using database software.

Use case

Databases are used to support internal operations of organizations and to underpin online interactions with customers and suppliers (see Enterprise software).

Databases are used to hold administrative information and more specialized data, such as engineering data or economic models. Examples include computerized library systems, flight reservation systems, computerized parts inventory systems, and many content management systems that store websites as collections of webpages in a database.

Classification

One way to classify databases involves the type of their contents like bibliographic, document-text, statistical, or multimedia objects. Another way is by their application area, for example: accounting, music compositions, movies, banking, manufacturing, or insurance. A third way is by some technical aspect, such as the database structure or interface type. This section lists a few of the adjectives used to characterize different kinds of databases.

A cloud database relies on cloud technology. Both the database and most of its DBMS reside remotely, "in the cloud", while its applications are both developed by programmers and later maintained and used by end-users through a web browser and Open APIs.

Database management system

Connolly and Beg define database management system (DBMS) as a "software system that enables users to define, create, maintain and control access to the database". Examples of DBMS's include MySQL, Microsoft SQL Server, Oracle Database, and Microsoft Access.

The DBMS acronym is sometimes extended to indicate the underlying database model, with RDBMS for the relational, OODBMS for the object (oriented) and ORDBMS for the object-relational model.

The functionality provided by a DBMS can vary enormously. The core functionality is the storage, retrieval and update of data. Cod proposed the following functions and services a fully-fledged general purpose DBMS should provide.

- Data storage, retrieval and update
- User accessible catalos or data dictionary describing the metadata
- Support for transactions and concurrency
- Facilities for recovering the database should it become damaged
- Support for authorization of access and update of data
- Access support from remote locations
- Enforcing constraints to ensure data in the database abides by certain rules

It is also generally to be expected the DBMS will provide a set of utilities for such purposes as may be necessary to administer the database effectively, including import, export, Monitoring, and analysis utilities.

The core part of the DBMS interacting between the database and the application interface sometimes referred to as the database engine.

The large major enterprise DBMSs have tended to increase in size and functionality and have involved up to thousands of human years of development effort throughout their lifetime.

Database languages

Database languages are special-purpose languages, which allow one or more of the following tasks, sometimes distinguished as sublanguages:

- Data control language (DCL) – controls access to data;
- Data definition language (DDL) – defines data types such as creating, altering, or dropping tables and the relationships among them;
- Data manipulation language (DML) – performs tasks such as inserting, updating, or deleting data occurrences;
- Data query language (DQL) – allows searching for information and computing derived information.

OQL is an object model language standard (from the Object Data Management Group). It has influenced the design of some of the newer query languages like JDOQL and EJB QL.

XQuery is a standard XML query language implemented by XML database systems such as Mark Logic and exist, by relational databases with XML capability such as Oracle and Db2, and also by in-memory XML processors such as Saxon.

A database language may also incorporate features like:

- DBMS-specific configuration and storage engine management
- Computations to modify query results, like counting, summing, averaging, sorting, grouping, and cross-referencing
- Constraint enforcement
- Application programming interface version of the query language, for programmer convenience

Database application

External interaction with the database will be via an application program that interfaces with the DBMS. This can range from a database tool that allows users to execute SQL queries textually or graphically, to a website that happens to use a database to store and search information.

Application program interface:

A programmer will code interactions to the database (sometimes referred to as a data source) via an application program interface (API) or via a database language.

The particular API or language chosen will need to be supported by DBMS, possibly indirectly via processor or a bridging API. Some API's aim to be database independent, ODBC being a commonly known example. Other common API's include JDBC and ADO.NET.

Backup and restore

Sometimes it is desired to bring a database back to a previous state (for many reasons, e.g., cases when the database is found corrupted due to a software error, or if it has been updated with erroneous data).

To achieve this, a backup operation is done occasionally or continuously, where each desired database state (i.e., the values of its data and their embedding in database's data structures) is kept within dedicated backup files (many techniques exist to do this effectively). When it is decided by a database administrator to bring the database back to this state (e.g., by specifying this state by a desired point in time when the database was in this state), these files are used to restore that state.

Advantages:

- ensures data integrity
- security
- consistency
- reduces data redundancy
- improves data access
- sharing
- integration

SYSTEM ANALYSIS

Existing System

Donation management system software that is ideal for churches or charities as well as non-profit organizations to track donations and management of data. It provides unlimited contributions, unlimited donations, and excellent online donations tools to increase donation collection, offer immediate acknowledgment, keep track of every donation, and produce regular reports to allow you to concentrate on your mission. Control your team, multiple organizations, and seamlessly integrate customized software features according to your requirements.

The advanced features offered by several donor management software are attractive to organisations looking for a single solution that will manage the entire process of donating. We use the idle software for making this Donation management system.

Donor communication communicating to donors crucial to keep solid donor relationships. There are several ways that users can stay in touch with donors. Certain solutions contain mass messaging. It's not like you've never been informed concerning "Donation management" systems. But, the thing is that you don't really need all the features they offer. Prices are astronomical and it takes weeks to set up and understand.

It doesn't include anything in the software that's not mission-critical. It's precisely what you require and absolutely nothing more. Everything is simple to understand. There's no complicated installation, and no need for training.

Creating a donation management system using IDLE Python involves leveraging Python's capabilities for scripting and creating a simple command-line interface.

Donation, and provides basic functions for adding donors, recording donations, retrieving donor information, and generating a donation report. Keep in mind that this is a rudimentary system, and for a production environment, additional features, error handling, and security considerations should be implemented.

Proposed system

Proposing system in donation management system, emphasizing its importance in efficiently handling and tracking contributions to support a cause or organization. Create a centralized platform for managing and tracking donations. In a donation management system, we use the database as a backend for data storing.

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

- User friendliness is provided in the application with various controls.
- The system makes the overall project management much easier and flexible.
- There is no risk of data mismanagement at any level while the project development is under process.
- It provides high level of security with different level of authentication.

Donor Module details

- Add Donation Detail
- View Donation History
- Edit Profile

Volunteer Module

- View Donation Collection Request
- Update Donation Status and Remark
- View Donation History
- Delete the Donor details

Improve transparency by providing donors with real-time information about their contributions. Streamline the process of receiving, recording, and acknowledging donations. Enhance reporting capabilities for better analysis and decision-making. We should build the application for the donation management system to help the charity people. Use the idle software to build the application

Features and Functionalities:

1. Donor Registration
2. Donation Recording
3. Real-time Tracking
4. Reporting and Analytics
5. Donor Communication
6. Security Measures
7.

Use Cases:

Use Case 1: Add Donor
Actor: User
Description: User inputs donor details (name, contact) to add a new donor to the system.

Use Case 2: Record Donation
Actor: User
Description: User inputs donor ID and donation amount to record a new donation.

Use Case 3: View Donor Information

Actor: User

Description: User inputs donor ID and retrieves information such as name, contact, and total donations.

Use Case 4: Generate Donation Report

Actor: User

Description: User triggers a command to generate a report summarizing donor information and total donations.

Project description

Project Overview:

The Donation Management System with IDLE Python is a simple yet effective software solution developed to assist non-profit organizations in managing their donations efficiently. Leveraging the Python programming language and the IDLE integrated development environment, this system provides a user-friendly interface for organizations to track, record, and manage their donations seamlessly.

Key Features:

1. **Donor Management**:
    - Capture and store donor information such as name, contact details, and donation history.
    - Implement a simple database system for organizing donor data.

2. **Donation Tracking:**
    - Create functions to record and track incoming donations in real-time.
    - Assign unique identifiers to each donation for easy reference.

3. **Payment Information:**
    - Include functionality for manual entry of donation amounts and payment methods.
    - Support both monetary and in-kind donations.

4. **Campaign Management:**
    - Design modules for creating and managing fundraising campaigns.
    - Allow for the setting of campaign goals and tracking progress.

5. **Basic Reporting:**
    - Generate basic reports on total donations received, campaign performance, and donor statistics.
    - Display reports within the IDLE environment.

6. **User Interface:**
    - Develop a straightforward and user-friendly interface using the Tkinter library for Python.
    - Implement easy navigation for data entry, retrieval, and reporting.

Benefits:

- Simplified donation tracking and management.
- Easy-to-use interface, suitable for users with basic Python knowledge.
- Quick and straightforward deployment without the need for complex setups.

Target Users:

- Small to medium-sized non-profit organizations.
- Individuals or groups managing small-scale fundraising activities.

Technology Stack:

- Python (3.x)
- Tkinter (for GUI)
- SQLite (for database)
- Version Control: Git

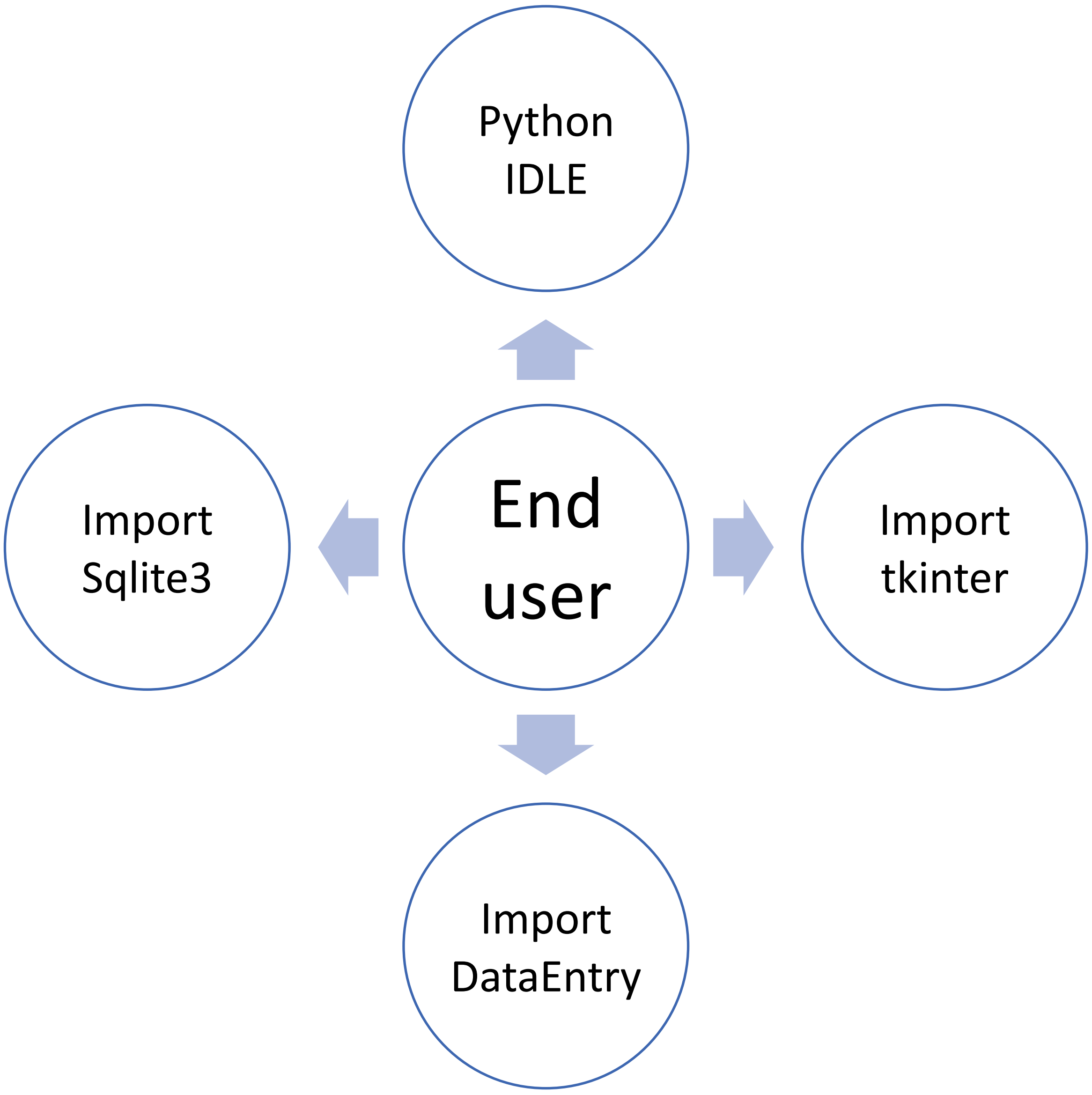
Project Timeline:

- Planning and Design: 2 weeks
- Development: 4 weeks
- Testing and Debugging: 2 weeks
- Deployment and Training: 1 week

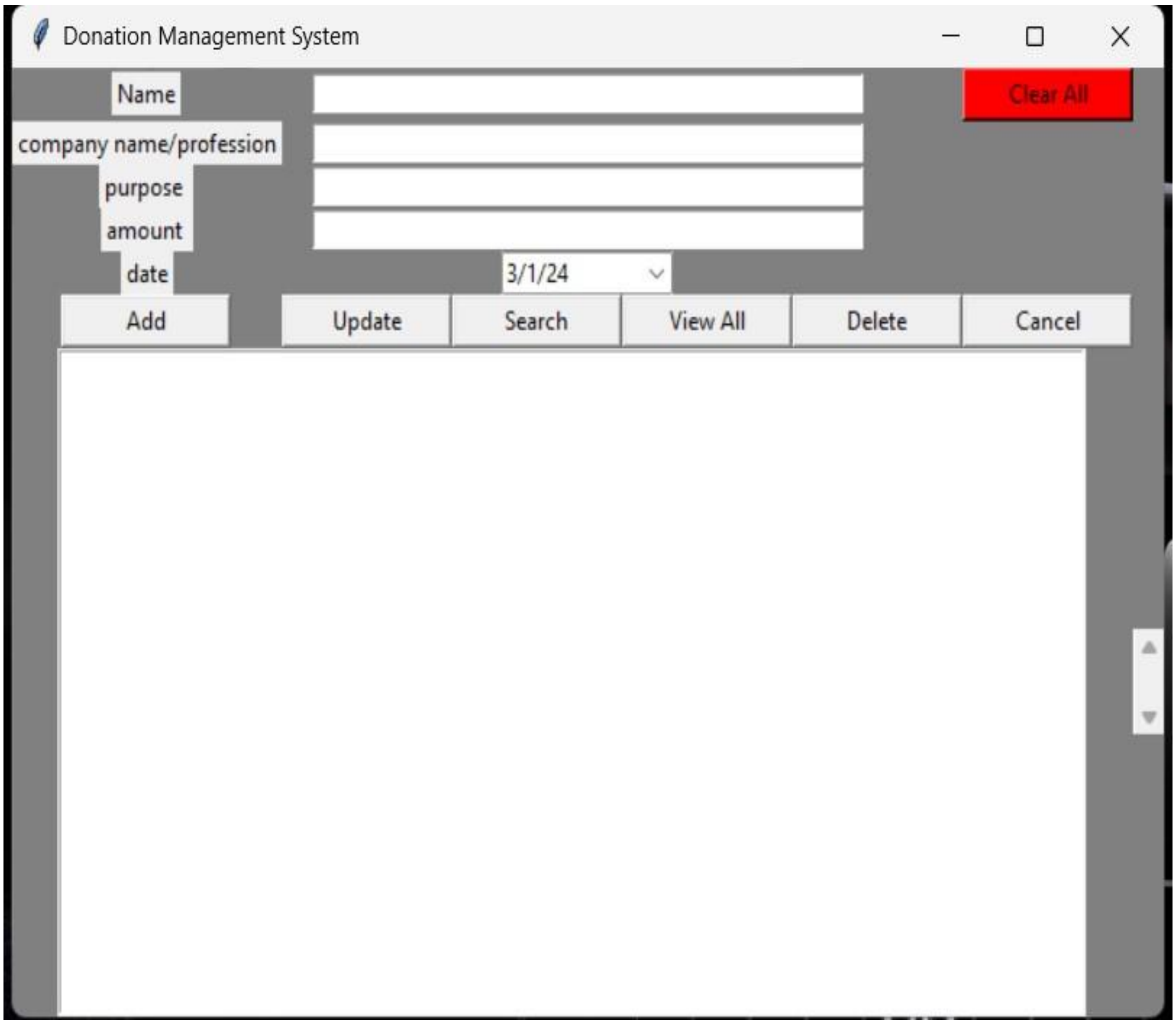
Note: This project description focuses on simplicity and ease of use, making it suitable for organizations with limited technical resources. For more advanced features and scalability, consider using additional frameworks and libraries or moving to a more sophisticate;

SYSTEM DESIGN

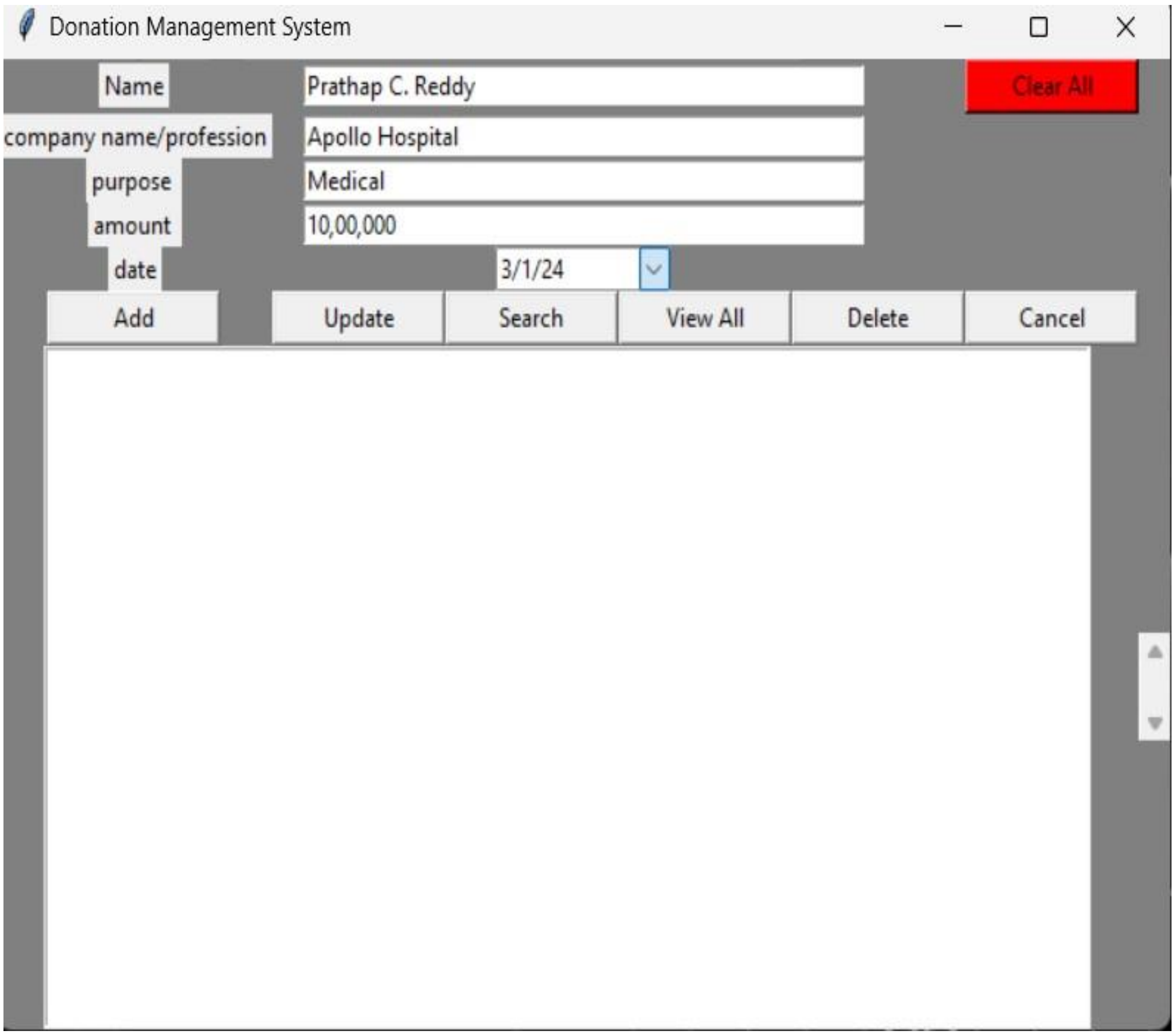
DATA FLOW CHART



REGISTRATION FORM PAGE



FINAL FORM PAGE


Donation Management System

Name: Prathap C. Reddy

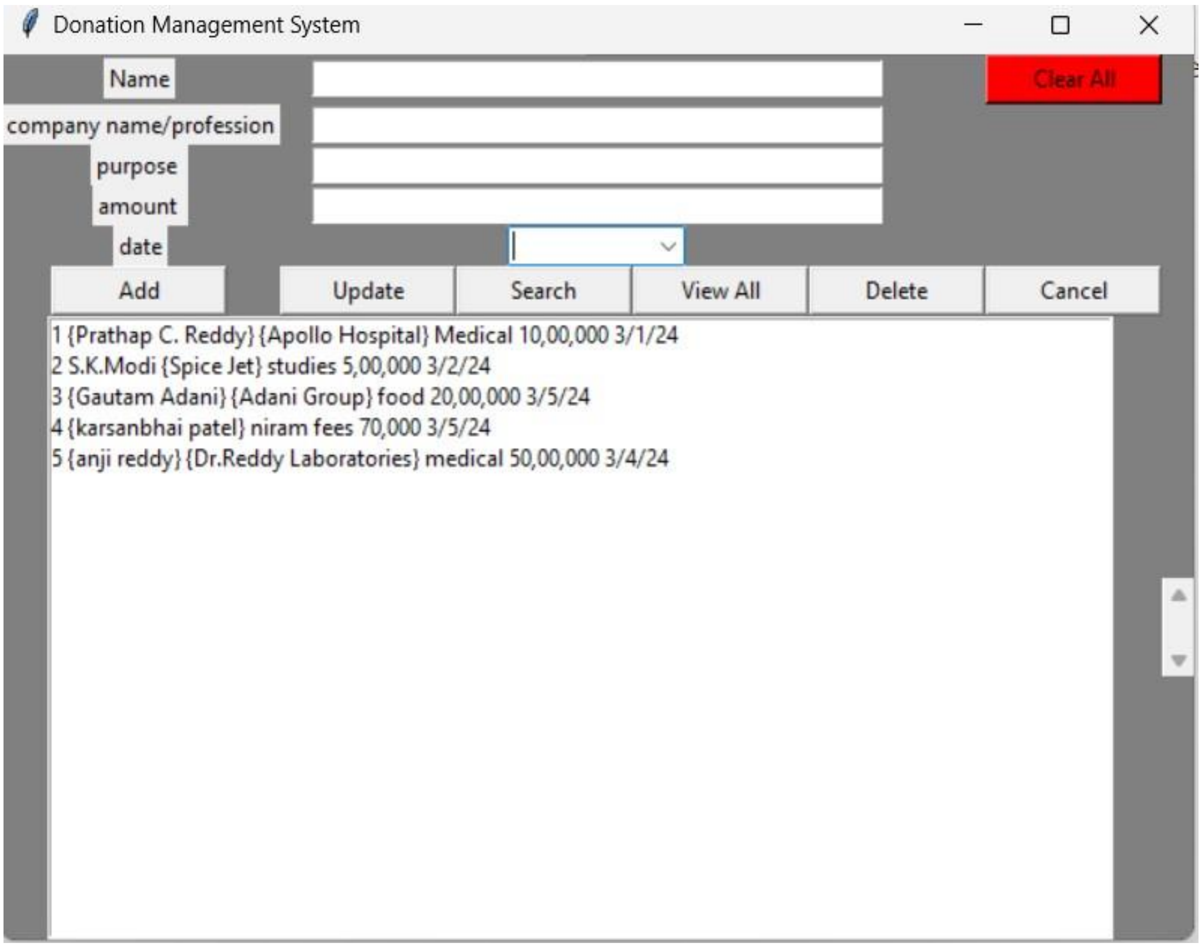
company name/profession: Apollo Hospital

purpose: Medical

amount: 10,00,000

date: 3/1/24

REPORT FORM



```
1 {Prathap C. Reddy} {Apollo Hospital} Medical 10,00,000 3/1/24
2 S.K.Modi {Spice Jet} studies 5,00,000 3/2/24
3 {Gautam Adani} {Adani Group} food 20,00,000 3/5/24
4 {karsanbhai patel} niram fees 70,000 3/5/24
5 {anji reddy} {Dr.Reddy Laboratories} medical 50,00,000 3/4/24
```

Table Structure

NAME	**COMPANY NAME/PROFESSION**	**PURPOSE**	**AMOUNT**	**DATE**
PRATHAP C.REDDY	**A**POLLO HOSPITAL	MEDICAL	10,00,000	3/1/2024
S.K.MODI	SPICE JET	STUDIES	5,00,000	3/2/2024
GAUTAM ADANI	ADANI GROUP	FOOD	20,00,000	3/5/2024
KARSANBHAI PATEL	NIRAM	FEES	70,000	3/5/2024
ANJI REDDY	DR.REDDY LABORATORIES	MEDICAL	50,00,000	3/4/2024

IMPLEMENTATION

SOURCE CODE

LABEL SOURCE CODE

```
from tkinter import *

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

l1 = Label(window,text="Name", fg="black")

l1.grid(row=0,column=0)

l2 = Label(window,text="company name/profession", fg="black")

l2.grid(row=1,column=0)

l3 = Label(window,text="purpose ", fg="black")

l3.grid(row=2,column=0)

l4 = Label(window,text="amount ", fg="black")

l4.grid(row=3,column=0)

l5 = Label(window,text="date", fg="black")

l5.grid(row=4,column=0)

window.mainloop()
```

ENTRY SOURCE CODE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

l1 = Label(window,text="Name", fg="black")

l1.grid(row=0,column=0)

l2 = Label(window,text="company name/profession", fg="black")

l2.grid(row=1,column=0)

l3 = Label(window,text="purpose ", fg="black")

l3.grid(row=2,column=0)

l4 = Label(window,text="amount ", fg="black")

l4.grid(row=3,column=0)

l5 = Label(window,text="date", fg="black")

l5.grid(row=4,column=0)

name=StringVar()

e1 = Entry(window,textvariable=name,width=50)

e1.grid(row=0,column=0,columnspan=10)
```

```
user=StringVar()

e2 = Entry(window,textvariable=user,width=50)

e2.grid(row=1,column=0,columnspan=10)

password=StringVar()

e3 = Entry(window,textvariable=password,width=50)

e3.grid(row=2,column=0,columnspan=10)

category=StringVar()

e4 = Entry(window,textvariable=category,width=50)

e4.grid(row=3,column=0,columnspan=10)

cdate=StringVar()

e5 =DateEntry(window,selectmode='day',textvariable=cdate)

e5.grid(row=4,column=0,columnspan=10

window.mainloop()
```

FRONT END SOURCE CODE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

window.configure(bg="gray")

l1 = Label(window,text="Name", fg="black")

l1.grid(row=0,column=0)

l2 = Label(window,text="company name/profession", fg="black")

l2.grid(row=1,column=0)

l3 = Label(window,text="purpose ", fg="black")

l3.grid(row=2,column=0)

l4 = Label(window,text="amount ", fg="black")

l4.grid(row=3,column=0)

l5 = Label(window,text="date", fg="black")

l5.grid(row=4,column=0)

name=StringVar()

e1 = Entry(window,textvariable=name,width=50)

e1.grid(row=0,column=0,columnspan=10)
```

```
user=StringVar()

e2 = Entry(window,textvariable=user,width=50)

e2.grid(row=1,column=0,columnspan=10)

password=StringVar()

e3 = Entry(window,textvariable=password,width=50)

e3.grid(row=2,column=0,columnspan=10)

category=StringVar()

e4 = Entry(window,textvariable=category,width=50)

e4.grid(row=3,column=0,columnspan=10)

cdate=StringVar()

e5 =DateEntry(window,selectmode='day',textvariable=cdate)

e5.grid(row=4,column=0,columnspan=10)

b1 = Button(window,text="Add",width=12,command=add_command)

b1.grid(row=5,column=0)

b2=Button(window,text="Update",width=12,command=update_comman
d)

b2.grid(row=5,column=1)

b3=Button(window,text="Search",width=12,command=search_command
)

b3.grid(row=5,column=2)
```

```
b4 = Button(window,text="View
All",width=12,command=view_command)
b4.grid(row=5,column=3)
b5
=Button(window,text="Delete",width=12,command=delete_command)
b5.grid(row=5,column=4)
b6 =
Button(window,text="Cancel",width=12,command=window.destroy)
b6.grid(row=5,column=5)
b7 = Button(window,text="Clear
All",bg="red",width=12,command=clear_command)
b7.grid(row=0,column=5)
lb=Listbox(window,height=20,width=94)
lb.grid(row=6,column=0,columnspan=6)
sb=Scrollbar(window)
sb.grid(row=6,column=6,rowspan=6)
lb.configure(yscrollcommand=sb.set)
sb.configure(command=lb.yview)
lb.bind('<<ListboxSelect>>',get_selected_row)
window.mainloop()
```

BACK END SOURCE CODE:

```
import sqlite3

def create():

    con = sqlite3.connect("Donation.db")

    cur = con.cursor()

    cur.execute("CREATE TABLE IF NOT EXISTS Donation(id
INTEGER PRIMARY KEY,name TEXT,user TEXT, password
TEXT,category TEXT,cdate TEXT)")

    con.commit()

    con.close()

def viewall():

    con = sqlite3.connect("Donation.db")

    cur = con.cursor()

    cur.execute("SELECT * FROM Donation")

    rows = cur.fetchall()

    con.close()

    return rows

def search(name="",user="",password="",category=""):

    con = sqlite3.connect("Donation.db")

    cur = con.cursor()

    cur.execute("SELECT * FROM Donation WHERE name=? OR user=?
OR password=? OR category=?",(name,user,password,category))
```

```
    rows = cur.fetchall()

    con.close()

    return rows

def add(name,user,password,category,cdate):

    con = sqlite3.connect("Donation.db")

    cur = con.cursor()

    cur.execute("INSERT INTO Donation
VALUES(NULL,?,?,?,?,?)",(name,user,password,category,cdate))

    con.commit()

    con.close()

def update(id,name,user,password,category,cdate):

    con = sqlite3.connect("Donation.db")

    cur = con.cursor()

    cur.execute("UPDATE Donation SET
name=?,user=?,password=?,category=?,cdate=? WHERE
id=?",(name,user,password,category,cdate,id))

    con.commit()

    con.close()


def delete(id):

    con = sqlite3.connect("Donation.db")

    cur = con.cursor()
    cur.execute("DELETE FROM Donation WHERE id=?",(id,))
    con.commit()
    con.close()
create()
```

ADD SOURCE CODE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

window.configure(bg="gray")

def add_command():

   function.add(name.get(),user.get(),password.get(),category.get(),cdate.
   get())

   lb.delete(0,END)

   lb.insert(END,name.get(),user.get(),password.get(),category.get(),cdate
.get())

b1 = Button(window,text="Add",width=12,command=add_command)

b1.grid(row=5,column=0)

window.mainloop()
```

SEARCH SOURCE CODE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

window.configure(bg="gray")

def search_command():

   lb.delete(0,END)

   for row in
function.search(name=name.get(),user=user.get(),password=password.get
(),category=category.get()):

      lb.insert(END,row)

b3
=Button(window,text="Search",width=12,command=search_command)

b3.grid(row=5,column=2)

window.mainloop()
```

UPDATE SOURCE CODE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

window.configure(bg="gray")

def update_command():

function.update(selected_tuple[0],name.get(),user.get(),password.get(),cat
egory.get(),cdate.get())

    view_command()

b2 =
Button(window,text="Update",width=12,command=update_command)

b2.grid(row=5,column=1)

window.mainloop()
```

DELETE SOURCE CODE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

window.configure(bg="gray")

def delete_command():

    function.delete(selected_tuple[0])

    view_command()

   b5 =
Button(window,text="Delete",width=12,command=delete_command)

b5.grid(row=5,column=4)

window.mainloop()
```

CLEAR SOURCE CODE:

```
from tkinter import *
from tkcalendar import DateEntry
from tkinter import ttk
import function
window = Tk()
window.title("Donation Management System")
window.configure(bg="gray")
def clear_command():
   lb.delete(0,END)
   e1.delete(0,END)
   e2.delete(0,END)
   e3.delete(0,END)
   e4.delete(0,END)
   e5.delete(0,END)
   b7 = Button(window,text="Clear
All",bg="red",width=12,command=clear_command)
   b7.grid(row=0,column=5)
window.mainloop()
```

CANCEL SOURCE CODE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

window.configure(bg="gray")

b6 =
Button(window,text="Cancel",width=12,command=window.destroy)

b6.grid(row=5,column=5)

window.mainloop()
```

VIEW ALL SOURCE CODE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

window.configure(bg="gray")

def view_command():

    lb.delete(0,END)

    for row in function.viewall():

        lb.insert(END,row)

b4 = Button(windo

w,text="View All",width=12,command=view_command)

b4.grid(row=5,column=3)

window.mainloop()
```

FINAL PAGE:

```
from tkinter import *

from tkcalendar import DateEntry

from tkinter import ttk

import function

window = Tk()

window.title("Donation Management System")

window.configure(bg="gray")

def view_command():

   lb.delete(0,END)

   for row in function.viewall():

      lb.insert(END,row)

def search_command():

   lb.delete(0,END)

   for row in
function.search(name=name.get(),user=user.get(),password=password.get
(),category=category.get()):

      lb.insert(END,row)

def add_command():


function.add(name.get(),user.get(),password.get(),category.get(),cdate.get
())

   lb.delete(0,END)
lb.insert(END,name.get(),user.get(),password.get(),category.get(),cdate.g
et())
def get_selected_row(event):
   try:
      global selected_tuple
      index=lb.curselection()[0]
      selected_tuple = lb.get(index)
      e1.delete(0,END)
      e1.insert(END,selected_tuple[1])
      e2.delete(0,END)
      e2.insert(END,selected_tuple[2])
      e3.delete(0,END)
      e3.insert(END,selected_tuple[3])
      e4.delete(0,END)
      e4.insert(END,selected_tuple[4])
      e5.delete(0,END)
      e5.insert(END,selected_tuple[5])
   except IndexError:
      pass
```

```
def update_command():
function.update(selected_tuple[0],name.get(),user.get(),password.get(),cat
egory.get(),cdate.get())
    view_command()
def delete_command():
    function.delete(selected_tuple[0])
    view_command()
def clear_command():
    lb.delete(0,END)
    e1.delete(0,END)
    e2.delete(0,END)
    e3.delete(0,END)
    e4.delete(0,END)
    e5.delete(0,END)
l1 = Label(window,text="Name", fg="black")
l1.grid(row=0,column=0)
l2 = Label(window,text="company name/profession", fg="black")
l2.grid(row=1,column=0)
l3 = Label(window,text="purpose ", fg="black")
l3.grid(row=2,column=0)
l4 = Label(window,text="amount ", fg="black")
l4.grid(row=3,column=0)
```

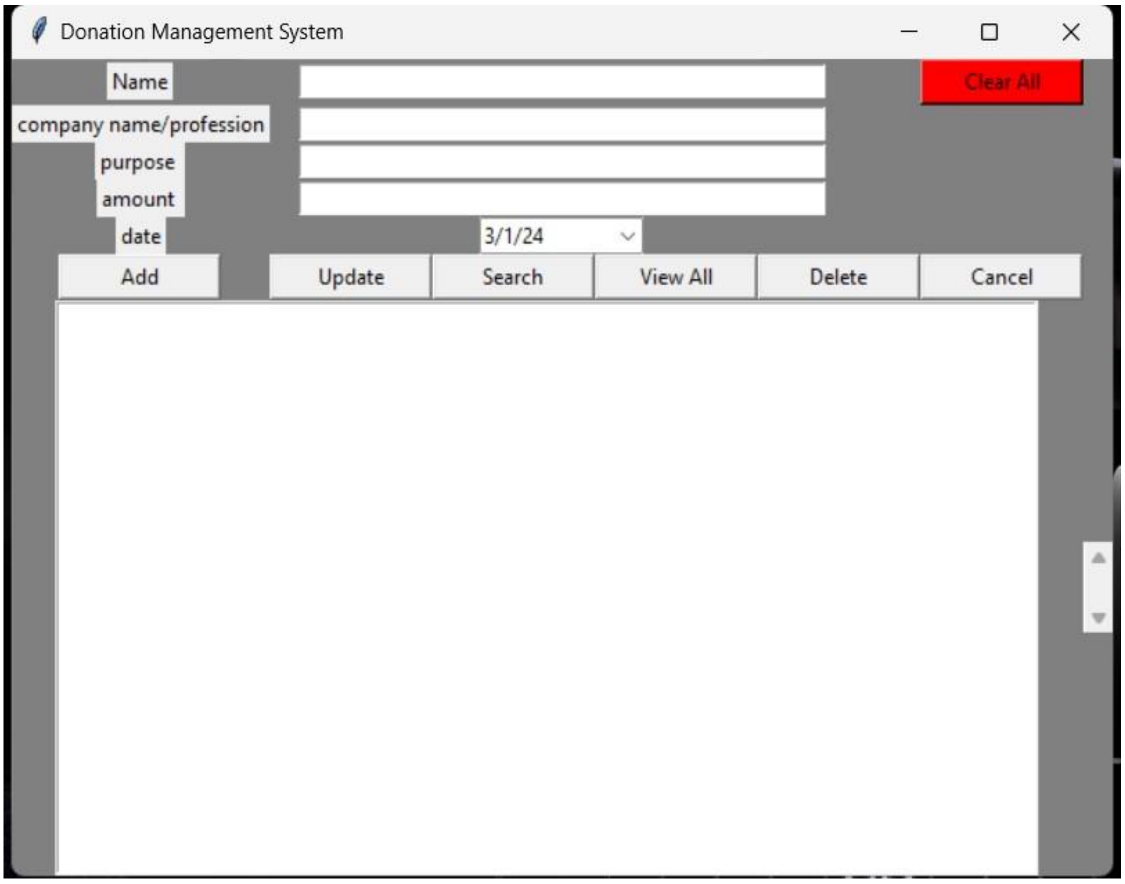
```
l5 = Label(window,text="date", fg="black")

l5.grid(row=4,column=0)

name=StringVar()

e1 = Entry(window,textvariable=name,width=50)

e1.grid(row=0,column=0,columnspan=10)

user=StringVar()

e2 = Entry(window,textvariable=user,width=50)

e2.grid(row=1,column=0,columnspan=10)

password=StringVar()

e3 = Entry(window,textvariable=password,width=50)

e3.grid(row=2,column=0,columnspan=10)

category=StringVar()

e4 = Entry(window,textvariable=category,width=50)

e4.grid(row=3,column=0,columnspan=10)

cdate=StringVar()

e5 =DateEntry(window,selectmode='day',textvariable=cdate)

e5.grid(row=4,column=0,columnspan=10)

b1 = Button(window,text="Add",width=12,command=add_command)

b1.grid(row=5,column=0)
```

```
b2 =
Button(window,text="Update",width=12,command=update_command)

b2.grid(row=5,column=1)

b3 =
Button(window,text="Search",width=12,command=search_command)

b3.grid(row=5,column=2)

b4 = Button(window,text="View
All",width=12,command=view_command)

b4.grid(row=5,column=3)

b5 =
Button(window,text="Delete",width=12,command=delete_command)

b5.grid(row=5,column=4)

b6 =
Button(window,text="Cancel",width=12,command=window.destroy)

b6.grid(row=5,column=5)

b7 = Button(window,text="Clear
All",bg="red",width=12,command=clear_command)

b7.grid(row=0,column=5)

lb=Listbox(window,height=20,width=94)

lb.grid(row=6,column=0,columnspan=6)

sb=Scrollbar(window)

sb.grid(row=6,column=6,rowspan=6)
```

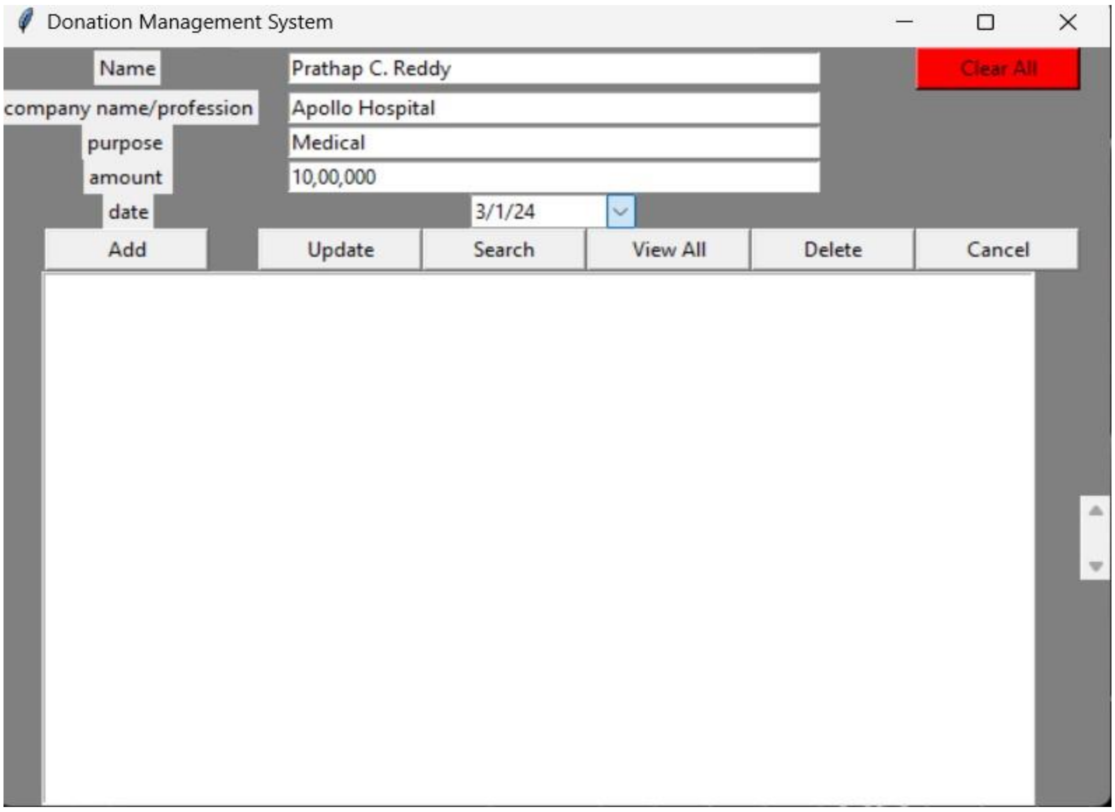
```
lb.configure(yscrollcommand=sb.set)

sb.configure(command=lb.yview)

lb.bind('<<ListboxSelect>>',get_selected_row)

window.mainloop()
```

SCREEN LAYOUT

REGISTRATION FORM:

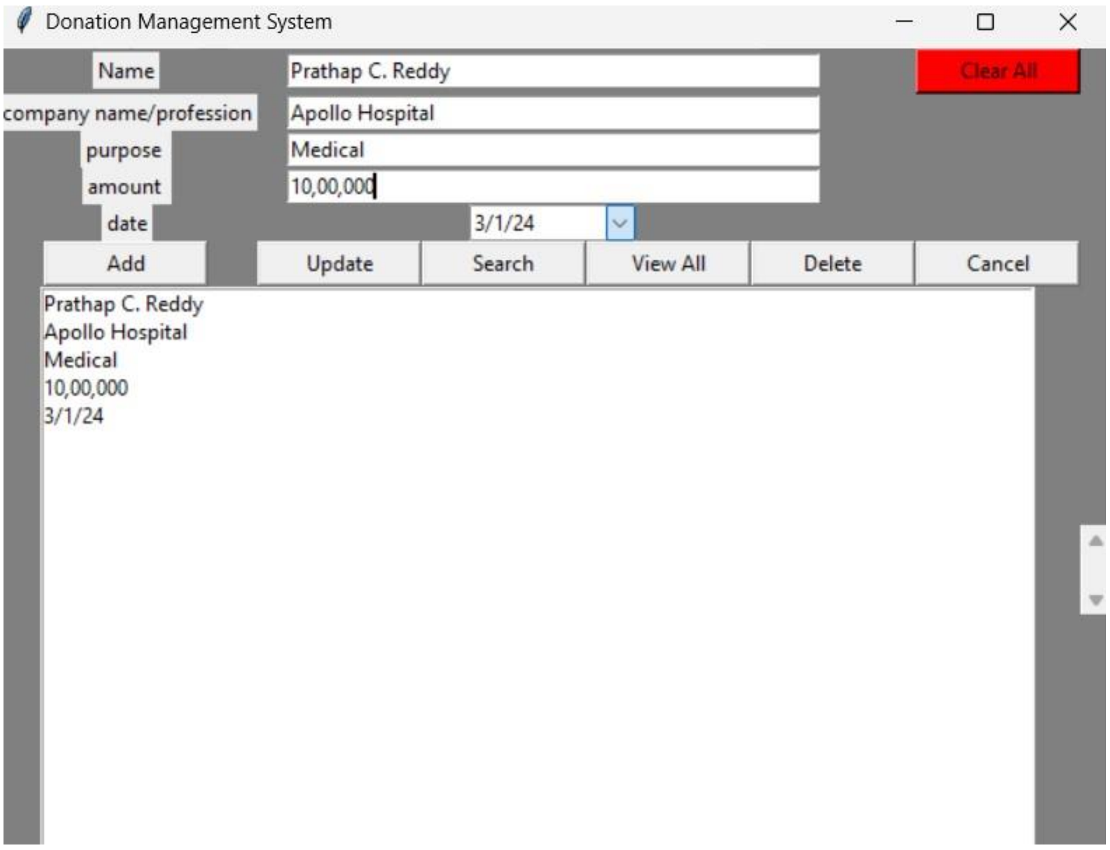


ENTRY FORM:


Donation Management System

Name	Prathap C. Reddy
company name/profession	Apollo Hospital
purpose	Medical
amount	10,00,000
date	3/1/24

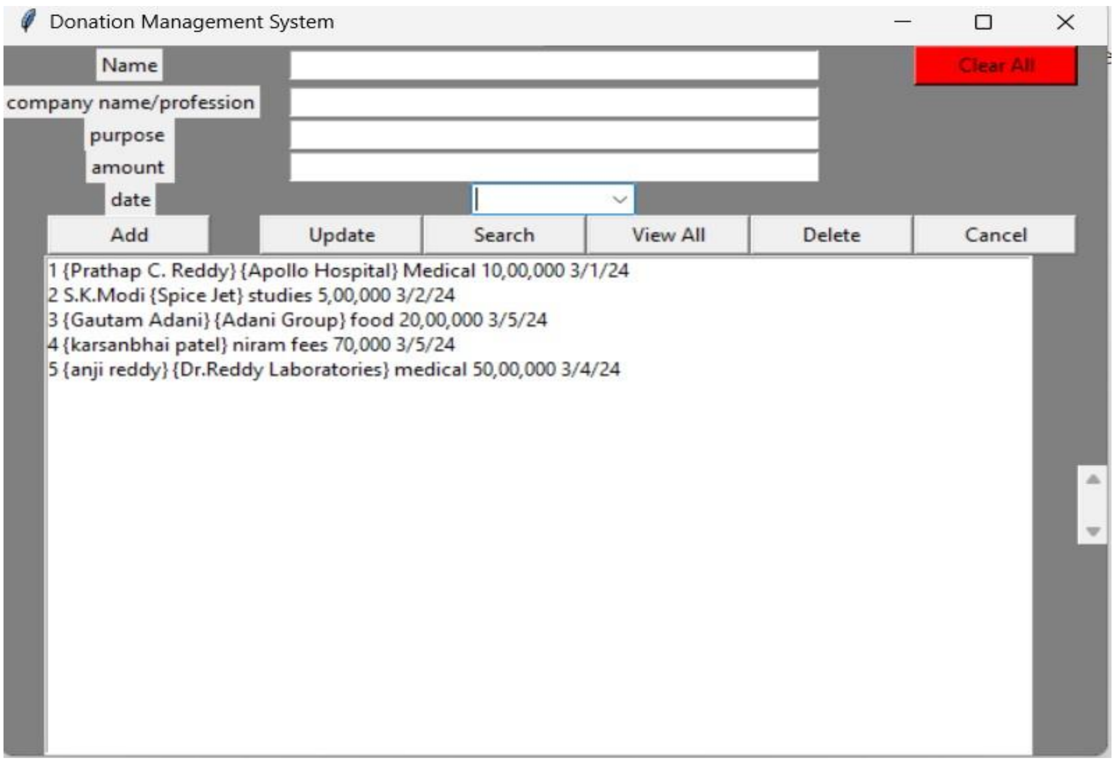
ADD FORM:



Prathap C. Reddy
Apollo Hospital
Medical
10,00,000
3/1/24

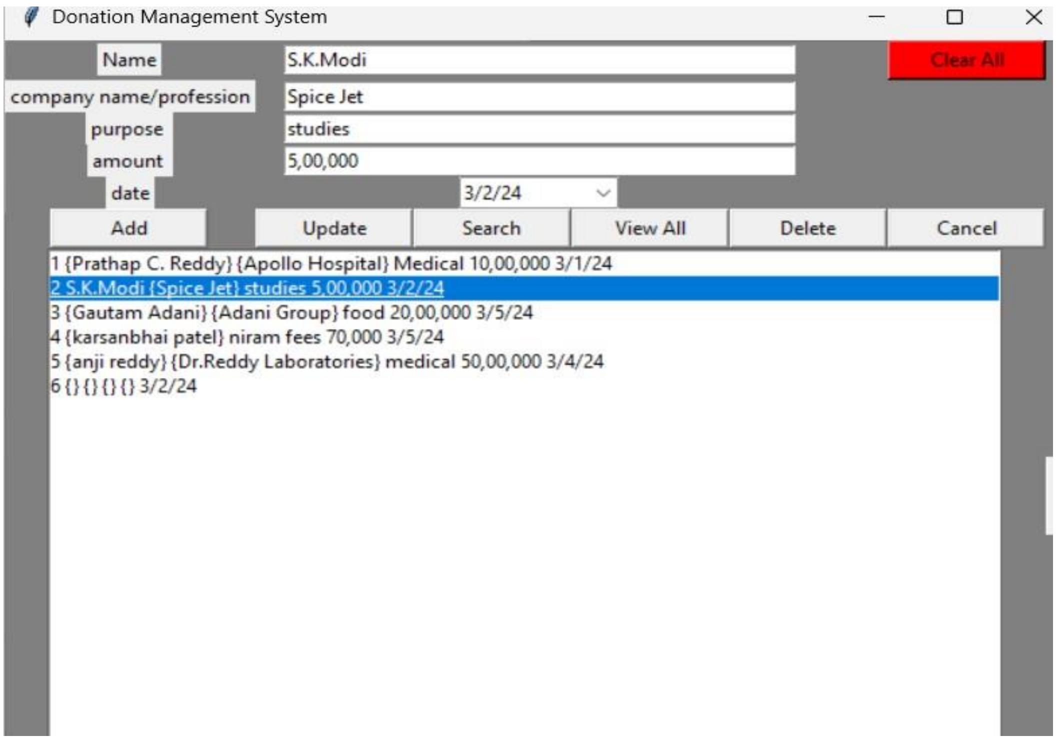
Prathap C. Reddy
Apollo Hospital
Medical
10,00,000
3/1/24

VIEW ALL FORM:



```
1 {Prathap C. Reddy} {Apollo Hospital} Medical 10,00,000 3/1/24
2 S.K.Modi {Spice Jet} studies 5,00,000 3/2/24
3 {Gautam Adani} {Adani Group} food 20,00,000 3/5/24
4 {karsanbhai patel} niram fees 70,000 3/5/24
5 {anji reddy} {Dr.Reddy Laboratories} medical 50,00,000 3/4/24
```

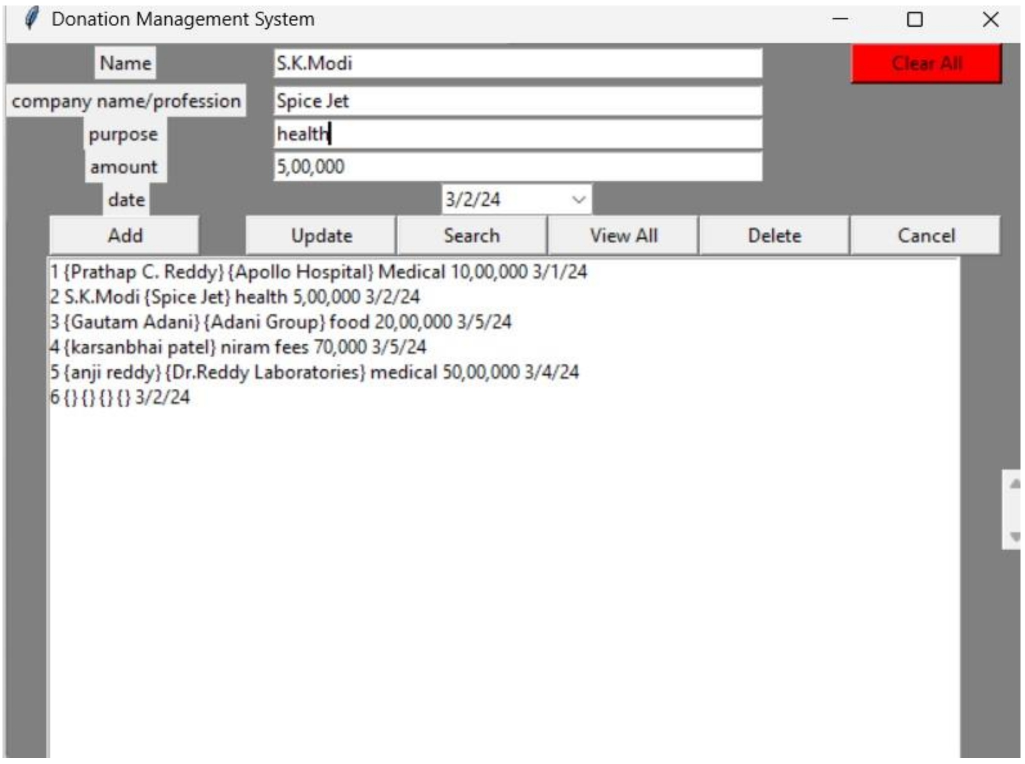
SELECT FORM:



Name	S.K.Modi
company name/profession	Spice Jet
purpose	studies
amount	5,00,000
date	3/2/24

```
1 {Prathap C. Reddy} {Apollo Hospital} Medical 10,00,000 3/1/24
2 S.K.Modi {Spice Jet} studies 5,00,000 3/2/24
3 {Gautam Adani} {Adani Group} food 20,00,000 3/5/24
4 {karsanbhai patel} niram fees 70,000 3/5/24
5 {anji reddy} {Dr.Reddy Laboratories} medical 50,00,000 3/4/24
6 {} {} {} {} 3/2/24
```

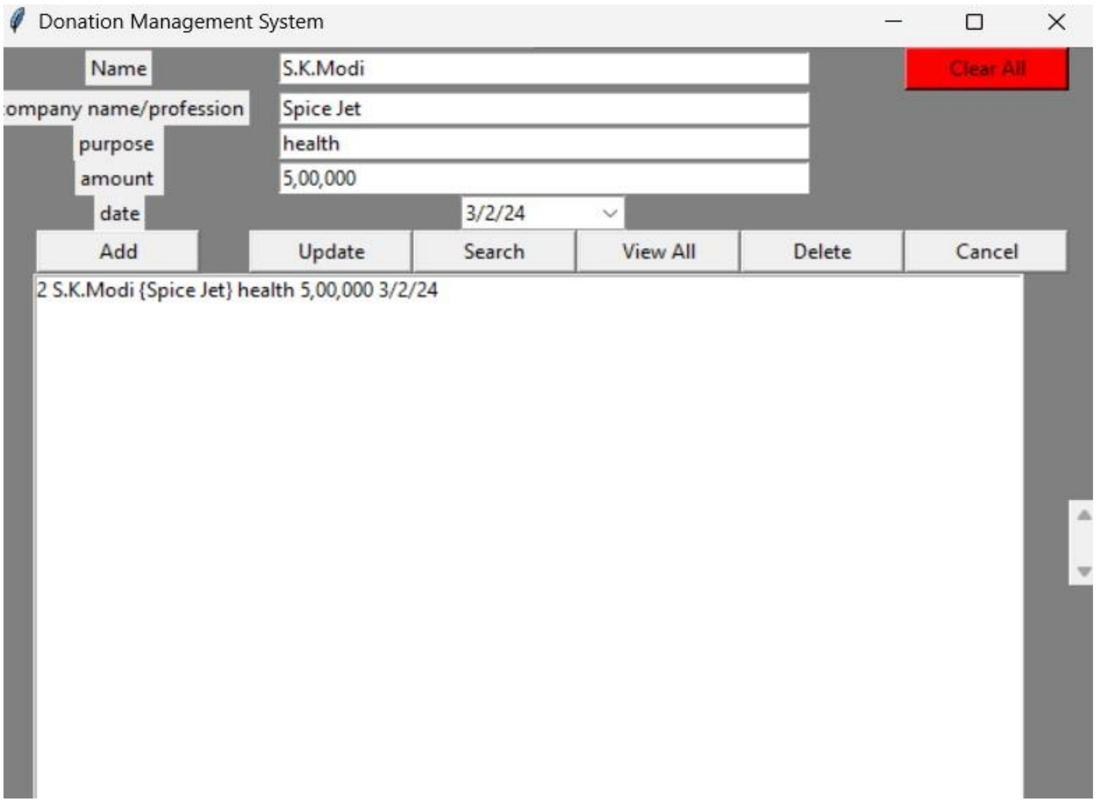
UPDATE PAGE:



Name: S.K.Modi
company name/profession: Spice Jet
purpose: health
amount: 5,00,000
date: 3/2/24

```
1 {Prathap C. Reddy} {Apollo Hospital} Medical 10,00,000 3/1/24
2 S.K.Modi {Spice Jet} health 5,00,000 3/2/24
3 {Gautam Adani} {Adani Group} food 20,00,000 3/5/24
4 {karsanbhai patel} niram fees 70,000 3/5/24
5 {anji reddy} {Dr.Reddy Laboratories} medical 50,00,000 3/4/24
6 {} {} {} {} 3/2/24
```

SEARCH PAGE:

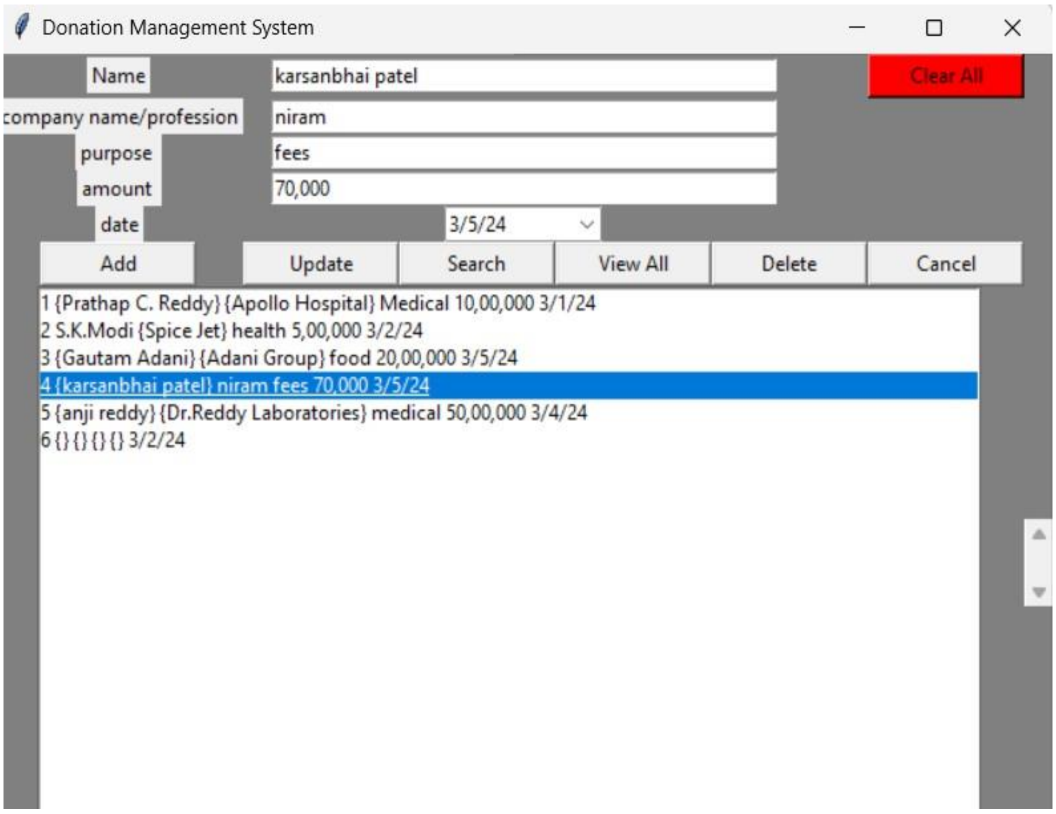


Name: S.K.Modi
ompany name/profession: Spice Jet
purpose: health
amount: 5,00,000
date: 3/2/24

```
2 S.K.Modi {Spice Jet} health 5,00,000 3/2/24
```

DELETE PAGE:

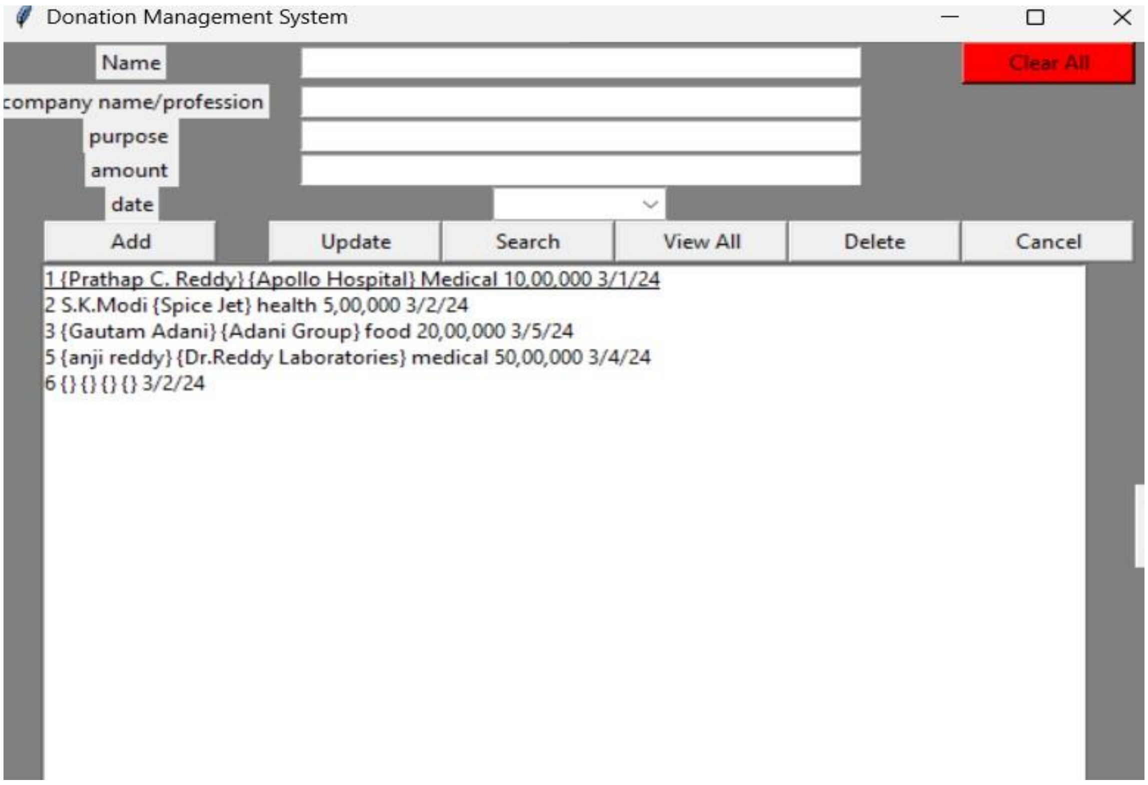
BEFORE DELETE:



Name: karsanbhai patel
company name/profession: niram
purpose: fees
amount: 70,000
date: 3/5/24

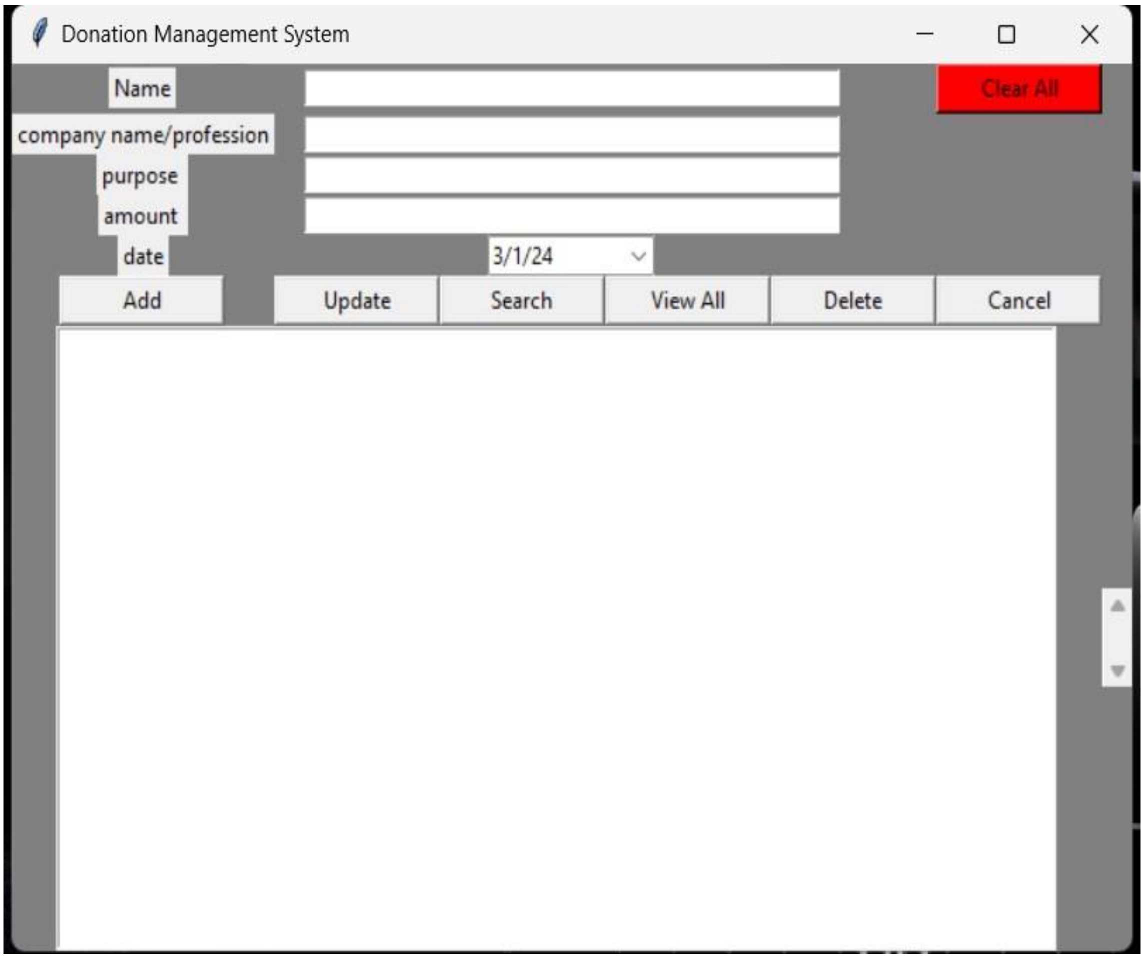
```
1 {Prathap C. Reddy} {Apollo Hospital} Medical 10,00,000 3/1/24
2 S.K.Modi {Spice Jet} health 5,00,000 3/2/24
3 {Gautam Adani} {Adani Group} food 20,00,000 3/5/24
4 {karsanbhai patel} niram fees 70,000 3/5/24
5 {anji reddy} {Dr.Reddy Laboratories} medical 50,00,000 3/4/24
6 {} {} {} {} 3/2/24
```

AFTER DELETE:



```
1 {Prathap C. Reddy} {Apollo Hospital} Medical 10,00,000 3/1/24
2 S.K.Modi {Spice Jet} health 5,00,000 3/2/24
3 {Gautam Adani} {Adani Group} food 20,00,000 3/5/24
5 {anji reddy} {Dr.Reddy Laboratories} medical 50,00,000 3/4/24
6 {} {} {} {} 3/2/24
```

CLEAR ALL PAGE:



SYSTEM STUDY

SYSTEM STUDY:

System testing in a donation management system involves evaluating the entire system to ensure that it meets specified requirements and functions as expected. The process can include testing various aspects such as functionality, performance, security, and user acceptance. In. this context, I'll provide you with a simple example of how you might perform system testing using Python

Let's assume you have a donation management system with functionalities such as adding a donation, viewing donation history, and generating reports. Here's a basic example using the unit test module in Python:

Python

```
Import unit test

From Donation_Management_system import Donation Manager # Import your
Donation Manager Class or module


Class TestDonationManagementSystem (unittest.TestCase):


   Def. setup (self):

      # set up the initial state or instantiate objects as needed

      self.donation_manager = Donation Manager ()


   def. test_add_donation (self):

      # Test adding a donation and check if the donation list is updated

      initial_donation_count = Len (self.donation_manager.get_donations ())


      # assuming add donation method adds a donation to the system
```

```
        self.donation_manager.add_donation (amount=100, donor="John Doe")

        # Check if the donation count has increased
        updated_donation_count = Len (self.donation_manager.get_donations ())
        self.assertEqual (updated_donation_count, initial_donation_count + 1)

    def. test_view_donation_history (self):
        # Test viewing the donation history and check if it's not empty
        # assuming view_donation_history method returns a list of donations
        Donation history = self.donation_manager. View_donation_history ()

        # Check if the donation history is not empty
        self.assertTrue (donation history)

    def. test_generate_reports (self):
        # Test generating reports and check if the report is created
        # assuming generate reports method creates a report file
        Report generated = self.donation_manager.generate_reports ()

        # Check if the report file is created successfully
        self.assertTrue (report generated)

If __name__ == '__main__':
    unittest.main ()
```

In this example, you need to replace donation_management_system and Donation Manager with the actual names and structure of your code. The setup method is used to set up any initial conditions needed for the tests, and each test method focuses on a specific aspect of the system.

Remember to adapt these tests according to the actual implementation of your donation management system. Additionally, you might want to consider using a testing framework like pits for more advanced and flexible testing scenarios.

CONCLUSION

Conclusion:

In conclusion, the donation management system developed using Python efficiently organizes and tracks donations, streamlining the process. Its user-friendly interface promotes ease of use, enhancing overall transparency and accountability in the donation workflow. The system's flexibility allows for seamless integration with existing platforms, making it a valuable tool for effective donation management.

Future Enhancement:

Future Enhancements

Future developments may include expanding reporting functionalities, incorporating additional security measures, and exploring opportunities for cloud-based deployment.

BIBLIOGRAPHY

Bibliography:

Creating a bibliography for a donation management system using IDLE (Python's integrated development environment) would typically involve citing relevant sources and references related to the development, design, and best practices in donation management systems. However, as of my last knowledge update in January 2022, I don't have specific sources or publications related to this topic. Therefore, I recommend consulting recent articles, books, or online resources related to donation management systems, Python programming, and software development. You can use academic databases, online libraries, and reputable websites to find relevant information. Be sure to follow the citation style (APA, MLA, Chicago, etc.) appropriate for your project or institution.