

Project Report
On
NGO MANAGEMENT

Submitted in Partial fulfillment of the Requirement for the
Award of the Degree

BACHELOR OF COMPUTER APPLICATION
MAHATMA GANDHI UNIVERSITY-KOTTAYAM

By
AHAMMED SHAJMEER (Reg No: 200021091591)



UNDER THE GUIDANCE OF
Mrs. SIGU K PAUL

DEPARTMENT OF COMPUTER SCIENCE
JAI BHARATH ARTS AND SCIENCE COLLEGE,
ARACKAPPADY, VENGOLA P.O,
PERUMBAVOOR 2020-23

JAI BHARATH ARTS AND SCIENCE COLLEGE

(Affiliated to Mahatma Gandhi University, Kottayam)

VENGOLA P.O ARACKAPPADY

DEPARTMENT OF COMPUTER SCIENCE



CERTIFICATE

This is to certify that the project work “**NGO MANAGEMENT**” was done by **AHAMMED SHAJMEER** with **Reg No: 200021091591** under the guidance and supervision and was submitted in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF COMPUTER APPLICATION** during the academic year 2020-23.

Mrs. SIGU K PAUL

Head of the Department

Dept. of Computer Science

Mrs. SIGU K PAUL

Project Guide

Submitted for viva-voice held on

Internal Examiner

External Examiner

ABOUT ORGANIZATION

VISION AND MISSION

Vision

The Jai Bharath aims to create an institution committed to academic excellence, producing students with the ability to think critically, creatively, become technology literate and to communicate effectively.

Mission

Empowering students with innovative and techno savvy, learning, teaching, mentoring, peer and co-interaction, experiential learning, industrial visit, yoga, meditation, and variety of personal development programs for the realization of the abundant potential of individuals.

Jai Bharath emirates to create a teaching and research-oriented institution of higher-level learning. Jai Bharath provides the need of the students to pursue the education goals, community and society. Jai Bharath aims to explore, develop and apply human and technological capabilities for the benefit of the regional, national and international community.

DECLARATION

I hereby to declare that the project report “**NGO MANAGEMENT**” is the bonafide work done by **AHAMMED SHAJMEER** (Reg. No: 200021091591) toward the partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Application of Mahatma Gandhi University, Kottayam, during the academic year 2020-23.

Date:

Arackappady

ACKNOWLEDGEMENT

It has been said that gratitude is the memory of heart. Hence, take this opportunity to express my gratitude to all those, whose contribution in this project cannot be forgotten.

First and fore, I thank The Almighty for his showers of blessings on this work and for endowing me the will to complete it on time.

I would like to thank **Dr. NITESH K N** (Principal, Jai Bharath Arts and Science College, Arackappady) for providing me all the necessary facilities. I would like to thank **Mrs. SIGU K PAUL** Head of the Department of Computer Science, and our project guide for their constant and sincere efforts to help bring out the best in me.

I am also grateful for their valuable and most helpful guidance all through the course of the project. With immense pleasure we take this opportunity to record our sincere thanks to our guide **Mrs. SIGU K PAUL** for their constant and sincere efforts to help bring out the best in me. I am also grateful for their valuable and most helpful guidance all through the course of the project.

I also thank all other teaching and non-teaching staff of the department of Computer Science, Jai Bharath Arts and Science College, Arackappady for helping me in some way or the other. Last but not the least I thank all my classmates in S6 Bachelor of Computer Application for helping me a lot with their valuable suggestions and for their wholehearted support.



CYBER PRISM SOFTWARE SOLUTIONS

CERTIFICATE

This is to certify that the project entitled “**NGO management**” developed in Python-Django submitted by **Ahammed Shajmeer (Reg.No.200021091591)** student of **Jai Bharat Art's and Science College** in partial fulfillment of the Bachelor of Computer Application is a bonafide record of work done at Cyber Prism Software Solutions, Muvattupuzha. He was along with our development team during the period **December 2022 to April 2023** for the project development.

It is seen that the project was successfully completed and that it complies with all the requirements suggested initially.

DATED THIS: 20-04-2023

FOR CYBER PRISM SOFTWARE SOLUTIONS

A handwritten signature in blue ink, appearing to read 'Naveen T M', is written over a set of horizontal lines.

NAVEEN T M
PROJECT MANAGER



CROWN PLAZA, OPP NIRMALA PUBLIC SCHOOL,
MUVATTUPUZZHA.

MOB : 9446782527 | 9446490528
cyberprismlimited@gmail.com

CONTENT

1. INTRODUCTION.....	10
1.1 SYNOPSIS.....	11
1.2 OVERVIEW.....	13
2. REQUIREMENT ANALYSIS.....	14
2.1 PROBLEM DEFINITION.....	15
2.2 SYSTEM ANALYSIS.....	15
2.3 SOFTWARE DEVELOPMENT MODEL.....	16
2.4 REQUIREMENT ANALYSIS.....	18
2.5 REQUIREMENT SPECIFICATION.....	18
2.6 PROJECT PLANNING.....	19
2.7 PROJECT SCHEDULING.....	20
2.8 FEASIBILITY STUDY.....	20
2.8.1 ECONOMIC FEASIBILITY.....	20
2.8.2 TECHNICAL FEASIBILITY.....	21
2.8.3 OPERATIONAL FEASIBILITY.....	21
2.8.4 BEHAVIOURAL FEASIBILITY.....	21
3. SOFTWARE REQUIREMENT SPECIFICATION.....	22
3.1 INTRODUCTION.....	23
3.1.1 PURPOSE.....	23
3.2 OVERALL DESCRIPTION.....	23
3.2.1 PRODUCT PERSPECTIVE.....	23
3.2.2 PRODUCT FUNCTION.....	23
3.2.3 USER CHARACTERISTICS.....	23

3.2.4 CONSTRAINTS.....	23
3.2.5 ASSUMPTIONS AND DEPENDENCIES.....	24
3.3 SPECIFIC REQUIREMENT.....	25
3.3.1 EXTERNAL INTERFACES.....	25
3.3.2 LOGIN INTERFACE.....	25
3.3.3 HARDWARE INTERFACE.....	25
3.3.4 SOFTWARE INTERFACE.....	25
3.4 FUNCTIONAL REQUIREMENT.....	26
3.5 NON-FUNCTIONAL REQUIREMENT.....	26
4. SOFTWARE AND HARDWARE REQUIREMENT.....	28
4.1 SOFTWARE REQUIREMENT.....	29
4.1.1 PYTHON.....	30
4.1.2 VISUAL STUDIO CODE.....	31
4.1.3 HTML.....	32
4.1.4 XAMPP.....	32
4.2 HARDWARE REQUIREMENT.....	32
5. SYSTEM DESIGN.....	33
5.1 INTRODUCTION.....	34
5.2 INPUT DESIGN.....	35
5.3 OUTPUT DESIGN.....	36
5.4 DATABASE DESIGN.....	36
5.5 DATA FLOW DIAGRAM.....	41
6. CODING.....	47
6.1 CODE SNIPPETS.....	48
7. SYSTEM TESTING.....	70

7.1 TESTING.....	71
7.2 OBJECTIVES OF SYSTEM TESTING.....	71
7.3 TYPES OF TESTING.....	72
7.3.1 UNIT TESTING.....	72
7.3.2 INTEGRATION TESTING.....	73
7.3.3 VALIDATION TESTING.....	73
7.3.4 OUTPUT TESTING.....	74
8. SYSTEM IMPLEMENTATION.....	75
9. SECURITY TECHNOLOGIES AND POLICIES.....	77
9.1 SYSTEM SECURITY.....	78
9.1.1 THREATS TO A SYSTEM SECURITY.....	78
9.2 SECURITY POLICY...../.....	79
9.3 SYSTEM SECURITY MEASURES.....	80
9.4 SECURITY.....	81
10. FUTURE SCOPE OF THE PROJECT.....	82
10.1 FUTURE ENHANCEMENT.....	83
10.2 SYSTEM SCOPE.....	84
11. CONCLUSION.....	85
12. APPENDIX.....	87
12.1 SCREENSHOTS.....	88
13. REFERENCES.....	92

1. INTRODUCTION

1.1 SYNOPSIS

NGO MANAGEMENT is a web application which is place for NGO's or Non-Governmental Organization is an organization that operates independently of any organization. NGO's are typically non-profit organizations and many of them are funded by donations from members, private institutions, general public etc.

Maintaining transparency with respect to how the donations are spent greatly increases the confidence of donors and also improves the credibility of an NGO. Thereby, helping to improve their reach in the long term.

Using conventional paper-based methods for managing the funds of an NGO increases the risk for embezzlement and fraud.

NGOs can use this system to view the list of donations received. The organization can then add their expenditures and view the history of all their transactions.

Meanwhile, the user can choose the NGO of their preference and make a donation to them through this system. The user will be able to see the list of transactions to ensure that their money is utilized in a rightful manner.

Modules:

1. Administration

Admin will look after registration process of NGO's and Users and do their work on website. They verify whether the NGO and User is legitimate or not and approve their registration.

2. NGO

NGO will register with proper documents and can update their profile. NGOs can use this system to create their new projects and view the list of donations received. The organization can then add their expenditures and view the history of all their transactions.

3. Users

User can choose the NGO of their preference and make a donation to them through this system. The user will be able to see the list of transactions to ensure that their money is utilized in a rightful manner.

4. Donation

Donation for the NGO is done through online payment. For this an online transaction payment gateway can be used.

5. Manage account

The users can manage their account details such as profile details, login details etc. Users can also update these details if needed.

1.2 OVERVIEW

In NGO MANAGEMENT system, the admin approves the NGO and can monitor their projects, donation and expenditure history. The NGO can keep a track of their expenditures and donations that they received from different users. User can also get insights over where their donations are being used.

2. REQUIREMENT ANALYSIS

2.1 PROBLEM DEFINITION

The group or individual understand and can prioritize the current challenges that they require to improve. A project is a plan of work that ideally gets you to the desired outcome. Before a project takes place, you really must think about and define why you are taking on the project.

2.2 SYSTEM ANALYSIS

System analysis refers to an orderly structured process for identifying and solving problems using computer. It is the most essential part of the project development. It is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements to the system. Training, experience and common sense are required for the collection of the information needed to do the analysis. To analyse a system, one has to study the system work in detail before designing to the appropriate computer-based system that will meet all requirements of the system. In other words, system analysis specifies what the system should do. System analysis includes the following steps:

- Preliminary investigation
 - Request clarification
 - Feasibility study
 - Request approval
- Determination of system requirements
- Design of the system
- Development of software
- System testing
- Implementation and evaluation

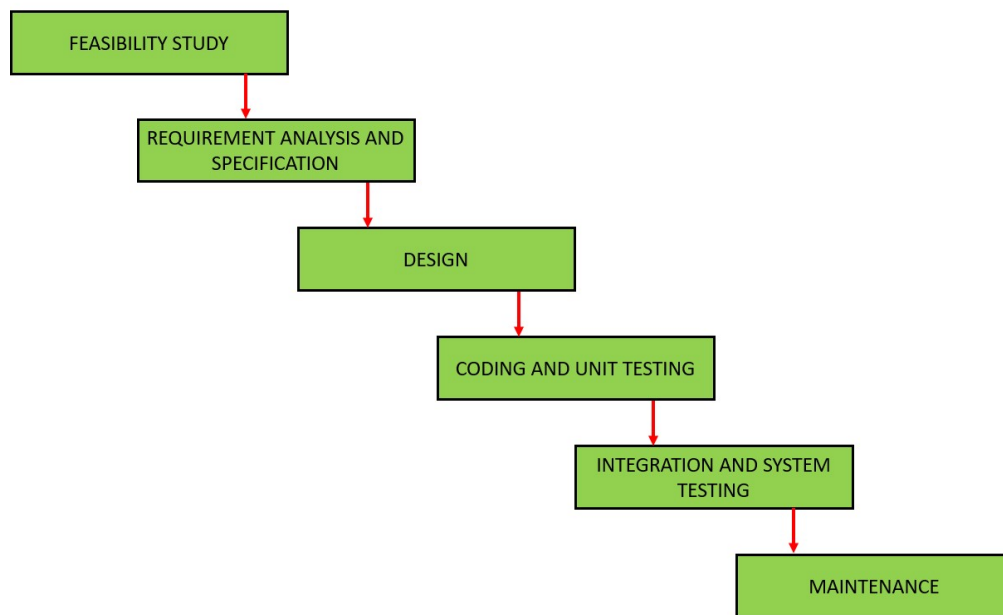
2.3 SOFTWARE DEVELOPMENT MODEL

To solve actual problems in an industry, software developer or a team of developers must incorporate a development strategy that encompasses the process, methods and tools layers and generic phases. This strategy is often referred to as process model or software developing paradigm. A process model for software developing is chosen based on the nature of project and application, the methods and tools to be used, and the controls and deliverable that are required. All software development can be characterized as a problem-solving loop in which four distinct stages are encountered: Status quo, problem definition, technical development and solution integration. Regardless of the process model that chosen for a software project all the stages coexist simultaneously at some level of detail.

Waterfall Model

Waterfall model is the basic software development life cycle model. It is very simple, but idealistic. Earlier this model was very popular but nowadays it is not used. But it is very important because all the other software development life cycle models are based on the waterfall model.

The waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after the completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus, the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of the waterfall model are shown in the below figure:



2.4 REQUIREMENT ANALYSIS

Requirements analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish. Energy should be directed towards ensuring that the final system or product conforms to client needs rather than attempting to mold user expectations to fit the requirements.

2.5 REQUIREMENT SPECIFICATION

These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between the development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.

In addition to specifying how the system should behave, the specification also defines at a high-level the main business process that will be supported, what simplifying assumptions have been made and what key performance parameters will need to be met by the system.

Proposed System

- Software programmers and business personals can interact with each other.
- Easy to find developers for small projects and complete work in short time with less cost.
- Scope of reaching maximum number of clients who are located at different parts of the world.

2.6 PROJECT PLANNING

Project planning is at the heart of the project life cycle and tells everyone involved where you're going and how you're going to get there. The planning phase is when the project plans are documented, the project deliverables and requirements are defined, and the project schedule is created. It involves creating a set of plans to help guide your team through the implementation and closure phases of the project. The plans created during this phase will help you manage time, cost, quality, changes, risk, and related issues. They will also help you control staff and external suppliers to ensure that you deliver the project on time, within budget, and within schedule.

Project planning involves:

DEFINING OBJECTIVES

The definition must include what the project is comprised of, its main aim, what it intends to accomplish, and what marks its closure

- EXPLAINING THE SCOPE

The explanation provides details on what the project intends to solve and who will benefit from the project.

- SCHEDULING TASKS

Each task is given a start date, an end date, and provides an estimate of how much time a task would take to complete

2.7 PROJECT SCHEDULING

Scheduling means the allocation of time period to tasks. Determining the sequence of tasks with time period, start and finish time is known as scheduling. When preparing a schedule estimate, consider that transition between activities often takes time. Organizations or resources outside your direct control may not share your sense of schedule urgency, and their work may take longer to complete. Beware of all external dependency relationships. Uncertain resources of talent, equipment, or data will likely result extending the project schedule.

The project schedule includes, at a minimum, a date for when the project begins and a date when the projects is expected to end. The project schedule is considered proposed until the resources needed to complete the project work are ascertained. In addition to the schedule, the project manager should include all the supporting details.

2.8 FEASIBILITY STUDY

A feasibility study, as the name suggests, is designed to reveal whether a project/plan is feasible. It is an assessment of the practicality of a proposed project/plan.

A feasibility study is part of the initial design stage of any project/plan. It is conducted in order to objectively uncover the strengths and weaknesses of a proposed project or an existing business. It can help to identify and assess the opportunities and threats present in the natural environment, the resources required for the project, and the prospects for success.

2.8.1 Economic Feasibility

- It is evaluating the effectiveness of candidate system by using cost/benefit analysis method.
- It demonstrates the net benefit from the candidate system in terms of benefits and costs to the organization.
- The main aim of Economic Feasibility Analysis (EFS) is to estimate the economic requirements of candidate system before investments funds are committed to proposal.

- It prefers the alternative which will maximize the net worth of organization by earliest and highest return of funds along with lowest level of risk involved in developing the candidate system.

2.8.2 Technical Feasibility

- It investigates the technical feasibility of each implementation alternative.
- It analyses and determines whether the solution can be supported by existing technology or not.
- The analyst determines whether current technical resources be upgraded or added it that fulfil the new requirements.

2.8.3 Operational Feasibility

- It determines whether the system is operating effectively once it is developed and implemented.
- It ensures that the management should support the proposed system and its working feasible in the current organizational environment.
- It analyses whether the users will be affected, and they accept the modified or new business methods that affect the possible system benefits.
- It also ensures that the computer resources and network architecture of candidate system are workable.

2.8.4 Behavioral Feasibility

- It evaluates and estimates the user attitude or behaviour towards the development of new system.
- It helps in determining if the system requires special effort to educate, retrain, transfer, and changes in employee's job status on new ways of conducting business

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

3.1.1 Purpose

NGO MANAGEMENT is a platform that manage the activity of an NGO organization. The application will help to manage the NGO members, Donors, and NGO projects. The whole idea behind the application is to manage the Projects,Expenditure and transactions details of the donors or Fundraiser

3.2 OVERALL DESCRIPTION

3.2.1 Product Perspective

It is a multipurpose storage system that is secure. In order to construct a dependable distributed storage system, it offers a threshold proxy re-encryption algorithm and combines it with a decentralized erasure code.

3.2.2 Product Functions

The product function can be represented using Data Flow Diagram. The main purpose of this system is to provide a secure multipurpose storage system.

3.2.3 User Characteristics

Registered users can store their files , share files and also send messages.

3.2.4 Constraints

- System should have installed with XAMPP as back end.
- Correct username and password must be provided for login.

3.2.5 Assumptions and Dependencies

The assumptions are:

- The coding should be error free.
- The system should be user friendly so that the users can easily access data.
- The system should have more storage capacity and provide fast access to database.
- The system should provide system search facilities and provide quick search results.
- The system should save money and time unlike the existing system.
- The user must provide correct username and password to enter to the system.

The dependencies are:

- The specific hardware and software due to which the product will run.
- Based on listing requirements and specification, the system will be developed and run.
- The end-users should have knowledge about the system.
- Updates are to be made correctly and data entered without any mistakes.

3.3 SPECIFIC REQUIREMENTS

3.3.1 External Interfaces

GUI

The software provides good graphical interface for the user and the administrator can operate on the system, performing the required task such as create, update, viewing the details of the users.

- The user interface must be customizable by the administrator.
- All the modules provided with the software must fit into this graphical user interface and accomplish to the standard defined.
- The design should be simple, and all the different interfaces should follow a standard template.

3.3.2 Login Interface

The system is provided with a username and password. If the user gives an incorrect username or password, an error message occurs.

3.3.3 Hardware interfaces

Only the recommended configuration (basic requirements of a computer system), if the camera is not available in the basic computer system external camera device is recommended.

3.3.4 Software Interfaces

Software will depend on the security features provided by the operating system and the language python. The system will have a direct connection with the database MYSQL. The data are stored in the database and retrieved as per requirements.

3.4 FUNCTIONAL REQUIREMENTS

This section describes the functional requirements of the system for those requirements which are expressed in the natural language style.

➤ Insert records:

This action is done to add new records into fields.

➤ Update records:

This event is to modify or update the information on each process.

➤ Delete records:

This action is to remove records from the system whenever they are no longer needed.

➤ Search for records:

Whenever the admin wants to search for a record, this action is performed.

➤ The validation of data entered should be done.

➤ Specific condition must be met.

➤ System shall maintain the sequences of the processes.

3.5 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements which define the system performance are:

1. Accuracy

The level of accuracy in the proposed system will be high. All operations would be done correctly, and it ensures that whatever information that comes from the center is accurate.

2. Reliability:

The reliability of the proposed system will be high. The reason for the increased reliability of the system is that system uses correct formulas to calculate the results.

3. Immediate Response:

The system is highly responsive because it uses well accurate formulas to calculate required results provided the user should enter the valid input data.

4. Easy to Operate:

The system should be easy to operate and should be such that it can be developed within a short period of time and fit in the limited budget of the user.

5. Usability

This section includes all those requirements that effect usability.

- We get the response within seconds.
- The software must have a simple, user friendly interface so customers can save time and confusion.

6. Supportability

The system is designed to be the cross platform supportable.

7. Implementation**8. Interface**

4. SOFTWARE AND HARDWARE REQUIREMENT

4.1 SOFTWARE REQUIREMENTS

A **Software Requirements Specification (SRS)** includes in-depth descriptions of the software that will be developed. A **System Requirements Specification (SyRS)** collects information on the requirements for a system.

The software used for new system are:

Server side:

- Windows 10
- Python
- Visual studio code
- Xampp

Client side:

- Windows 7 or above
- Internet explorer
- Google Chrome
- Mozilla Firefox

4.1.1 Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of their features support functional programming as aspect-oriented programming (including metaprogramming and metaobjects). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It uses dynamic name resolution (late binding), which binds method and variable names during program execution.

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which users enter statements sequentially and receive results immediately.

Python also comes with an Integrated development environment (IDE) called IDLE, which is more beginner-oriented.

Other shells, including IDLE and IPython, add further abilities such as improved auto-completion, session state retention, and syntax highlighting.

As well as standard desktop integrated development environments, there are Web browser-based IDEs, including Sage Math, for developing science- and math-related programs; PythonAnywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial IDE emphasizing scientific computing.

Python is a dynamic, interpreted (bytecode-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs. Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's **cross-platform**, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

4.1.2 Visual Studio Code

Visual Studio 2022 is the best Visual Studio ever. Our first 64-bit IDE makes it easier to work with even bigger projects and more complex workloads. **Visual Studio Code** (famously known as **VS Code**) is a free open source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times. VS Code supports a wide array of programming languages from Java, C++, and Python to CSS, Go, and Dockerfile. Moreover, VS Code allows you to add on and even creating new extensions including code linters, debuggers, and cloud and web development.

4.1.3 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of corner stone technologies for the World Wide Web.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

4.1.4 XAMPP

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

4.2 HARDWARE REQUIREMENTS

The hardware requirement for developing and implementing the proposed system is given below:

Processor : Intel i5 or more

RAM : 4 GB or higher

Hard Disk : 80 GB or more Network : EDGE and Above. Mouse : Logitech

Serial

5. SYSTEM DESIGN

5.1 INTRODUCTION

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.

A systemic approach is required for a coherent and well-running system. Bottom-Up or Top-Down approach is required to take into account all related variables of the system. A designer uses the modelling languages to express the information and knowledge in a structure of system that is defined by a consistent set of rules and definitions. The designs can be defined in graphical or textual modelling languages.

Characteristics of a well-defined system are:

- Acceptability
- Decision making ability
- Economy
- Flexibility
- Reliability
- Simplicity

For the replicated data processing environment using the python scripting in windows XP, developing an efficient system, which is user friendly as well as high in performance, is the main aim. It has been that the system will have the functions and promises of the proposed system. In the design phase various techniques are used to present a simple but efficient environment.

5.2 INPUT DESIGN

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

It is the process of converting the user-oriented inputs in to the computer-based format. The goal of designing input data is to make the automation as easy and free from errors as possible. For providing a good input design for the application easy data input and selection features are adopted. The input design requirements such as friendliness, consistent format and interactive dialogue for giving the right message and help for the user at right time are also considered for the development of the project. In this project, all the textboxes are validated. If any field is not filled, then it will display an error message that the field is blank. List boxes are used to reduce the user inputs.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties –

- It should serve specific purpose effectively such as storing, recording, and retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward.
- It should focus on user's attention, consistency, and simplicity.

Objectives Of Input Design

The objectives of input design are –

- To design data entry and input procedures
- To reduce input volume
- To design source documents for data capture or devise other data capture methods.
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

5.3 OUTPUT DESIGN

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

To develop output design that serves the intended purpose and eliminates the production of unwanted output. To develop the output design that meets the end user's requirements. To deliver the appropriate quantity of output. To form the output in appropriate format and direct it to the right person. To make the output available on time for making good decisions.

5.4 DATABASE DESIGN

Table is a collection of complete details about a particular subject. These data are saved in rows and Columns. The data of each Row are different units. Hence, rows are called RECORDS and Columns of each row are called FIELDS.

Data is stored in tables, which is available in the back-end the items and data, which are entered in the input, form id directly stored in this table using linking of database. We can link more than one table to input forms. We can collect the details from the different tables to display on the output. There are mainly 9 tables in this project.

1. auth_user

Description: This table is used for registration and login

.Primary key: id

Foreign key: username

SL.NO	FIELDNAME	DATA TYPE	CONSTRAINT	DESCRIPTION
1.	id	INT	PRIMARY	TO ENTER THE USER_ID
2.	username	VARCHAR	FOREIGN	TO ENTER THE USERNAME
3.	password	VARCHAR	NOT NULL	TO ENTER THE PASSWORD
4.	first_name	VARCHAR	NOT NULL	TO STORE THE FIRST NAME
5.	last_name	VARCHAR	NOT NULL	TO STORE THE LAST NAME
6.	email	VARCHAR	NOT NULL	TO STORE THE EMAIL

2. ngoapp_ngo_reg

Description: This table is used for registration of NGO

Primary key: id

Foreign key: user_id

SL.NO	FIELDNAME	DATA TYPE	CONSTRAINT	DESCRIPTION
1.	id	INT	PRIMARY	TO ENTER THE USER_ID
2.	oname	VARCHAR	NOT NULL	TO STORE ORGANIZATION_NAME
3.	phone	VARCHAR	NOT NULL	TO ENTER THE FIRST_NAME
4.	user_id	INT	FOREIGN	TO ENTER THE USER_ID
5.	address	VARCHAR	NOT NULL	TO STORE THE ADDRESS
6.	licence	VARCHAR	NOT NULL	TO STORE THE LICENCE

3. ngoapp_user_reg

Description: This table is used for registration of user

Primary key: id

Foreign key: user_id

SL.NO	FIELDNAME	DATA TYPE	CONSTRAINT	DESCRIPTION
1.	id	INT	PRIMARY	TO ENTER THE ID
2.	phone	VARCHAR	NOT NULL	TO STORE THE PHONE NUMBER
3.	address	VARCHAR	NOT NULL	TO STORE THE ADDRESS
4.	idproof	VARCHAR	NOT NULL	TO STORE THE ID_PROOF
5.	user_id	INT	FOREIGN	TO ENTER THE USER_ID
6.	photo	VARCHAR	NOT NULL	TO ENTER THE PHOTO

4.ngoapp_project

Description: This table is used to add project

Primary key: id

Foreign key: ngo_id

SL.NO	FIELDNAME	DATA TYPE	CONSTRAINT	DESCRIPTION
1.	id	INT	PRIMARY	TO ENTER THE ID
2.	pname	VARCHAR	NOT NULL	TO ENTER THE PROJECT_NAME
3.	pdes	VARCHAR	NOT NULL	TO STORE THE PROJECT_DESCRIPTION
4.	amount	VARCHAR	NOT NULL	TO ENTER THE AMOUNT

5.	photo	VARCHAR	NOT NULL	TO STORE THE PHOTO
6.	ngo_id	INT	FOREIGN	TO STORE THE NGO_ID

5. ngoapp_expenditure

Description: Used for show project expenditure

Primary key: id

Foreign key: project_id

SL.NO	FIELDNAME	DATA TYPE	CONSTRAINT	DESCRIPTION
1.	id	INT	PRIMARY	TO ENTER THE ID
2.	etype	VARCHAR	NOT NULL	TO ENTER THE EXPENDITURE_TYPE
3.	amount	VARCHAR	NOT NULL	TO ENTER THE AMOUNT
4.	edes	VARCHAR	NOT NULL	TO STORE THE EXPENDITURE_DESCRIPTION
5.	project_id	INT	FOREIGN	TO STORE THE PROJECT_ID

6. ngoapp_udonation

Description: Used for donation

Primary key: id

Foreign key: ngo_id,project_id,reg_id

SL.NO	FIELDNAME	DATA TYPE	CONSTRAINT	DESCRIPTION
1.	id	INT	PRIMARY	TO ENTER THE ID
2.	amount	INT	NOT NULL	TO ENTER AMOUNT

3.	ngo_id	INT	FOREIGN	TO STORE THE NGO_ID
4.	project_id	INT	FOREIGN	TO STORE THE PROJECT_ID
5.	reg_id	INT	FOREIGN	TO ENTER THE REG_ID
9.	STATUS	VARCHAR	NOT NULL	TO STORE THE STATUS

7. ngoapp_assigndonation

Description: Used for assigning the donation

Primary key: id

Foreign key: donation_id,ngo_id,project_id

SL.NO	FIELDNAME	DATA TYPE	CONSTRAINT	DESCRIPTION
1.	id	INT	PRIMARY	TO ENTER THE ID
2.	amount	INT	NOT NULL	TO ENTER THE AMOUNT
3.	donation_id	VARCHAR	FOREIGN	TO STORE WHO DONATION_ID
4.	ngo_id	INT	FOREIGN	TO ENTER THE NGO_ID
5.	project_id	INT	FOREIGN	TO ENTER THE PROJECT_ID

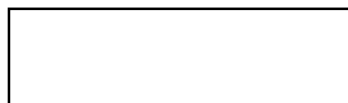
5.5 DATA FLOW DIAGRAM

A DFD, also known as a “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. A DFD consists of a series of bubbles joined by lines. The bubbles represents data transformations and the lines represents data flow in the system.

A data flow diagram may be used to represent a system or software at any level of abstraction. A DFD is a diagram that describes the flow of data and the processes that change or transform data throughout a system. It is a structured analysis and design tool that can be used or flowcharting in place of, or in association with, information oriented and process oriented system flowchart. When analyst prepare the DFD, they specify the user needs at a level of detail that virtually determines the information flow into and out of the system and the required data resources. This network is constructed by using a set of symbols that do not imply a physical implementation. The DFD reviews the current physical system, prepare input and output specification, specifies the implementation plan etc.

Basic data flow diagrams symbols are:

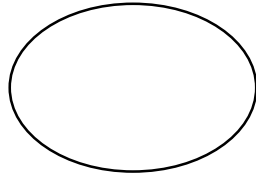
- A “rectangle” defines a source or destination of a system data.



- An “arrow” identifies data flow. It is a pipeline through which information flows.



- A “circle” represents a process that transforms incoming data flow(s) into outgoing data flow(s).



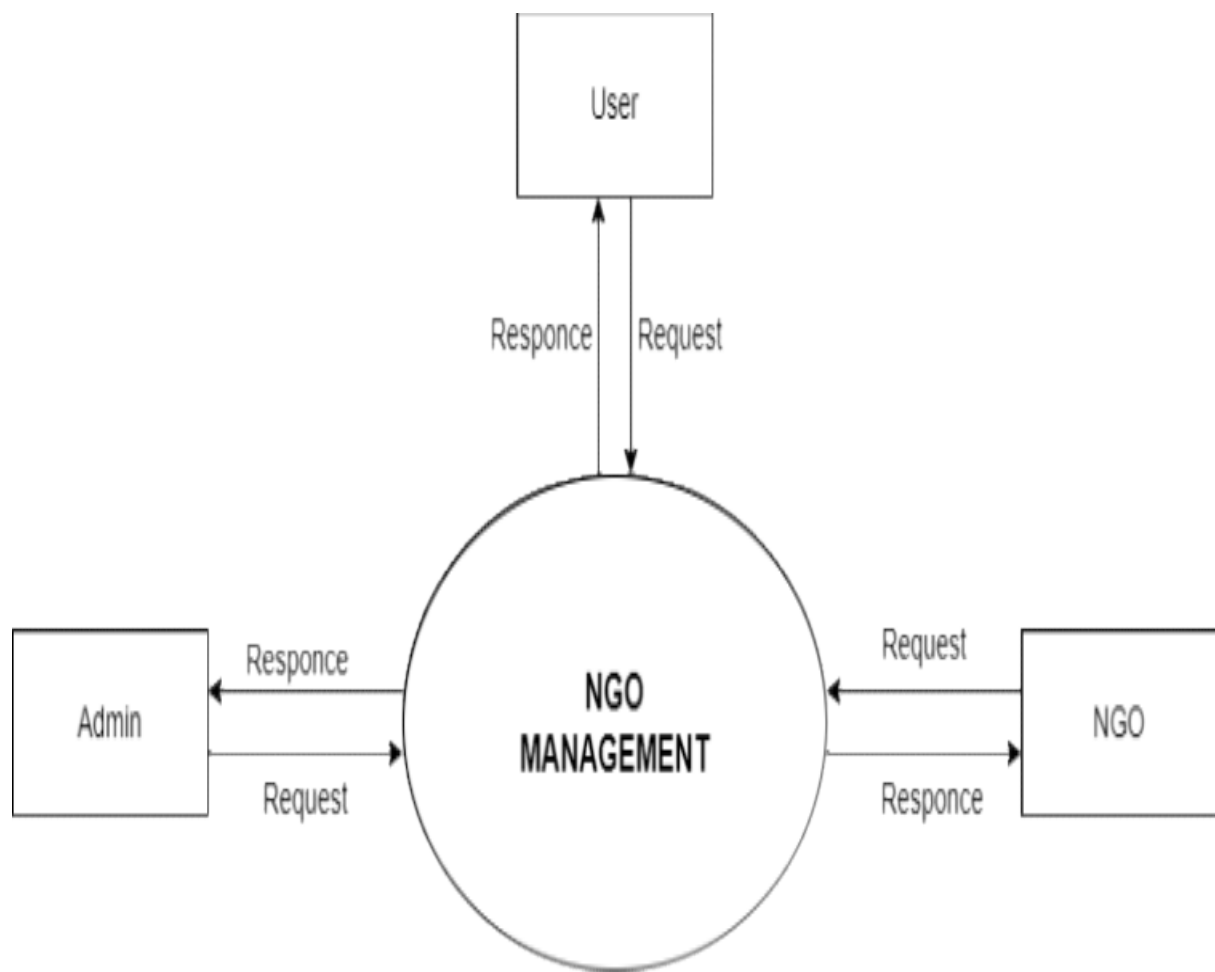
- An “open rectangle” is a data store

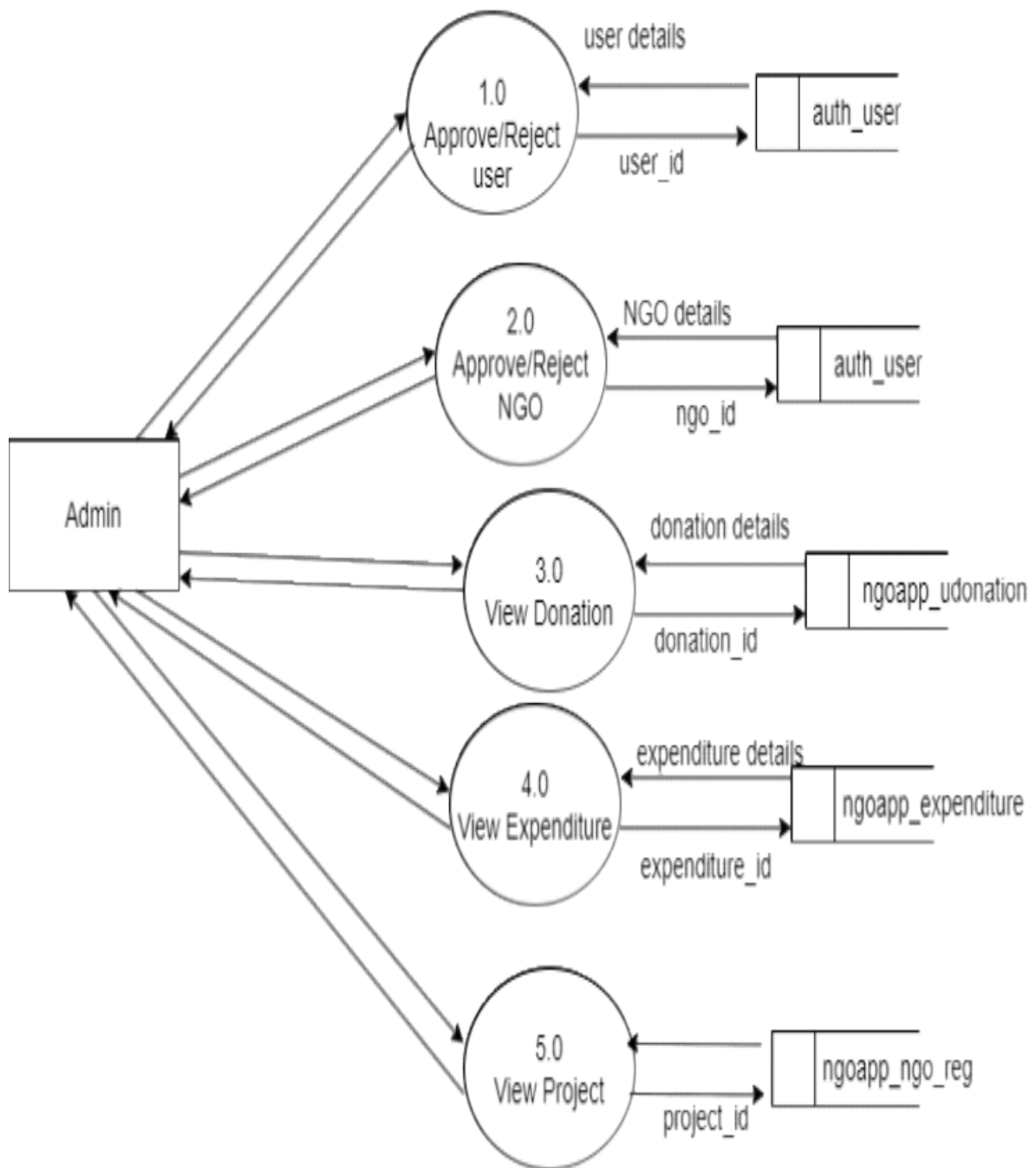


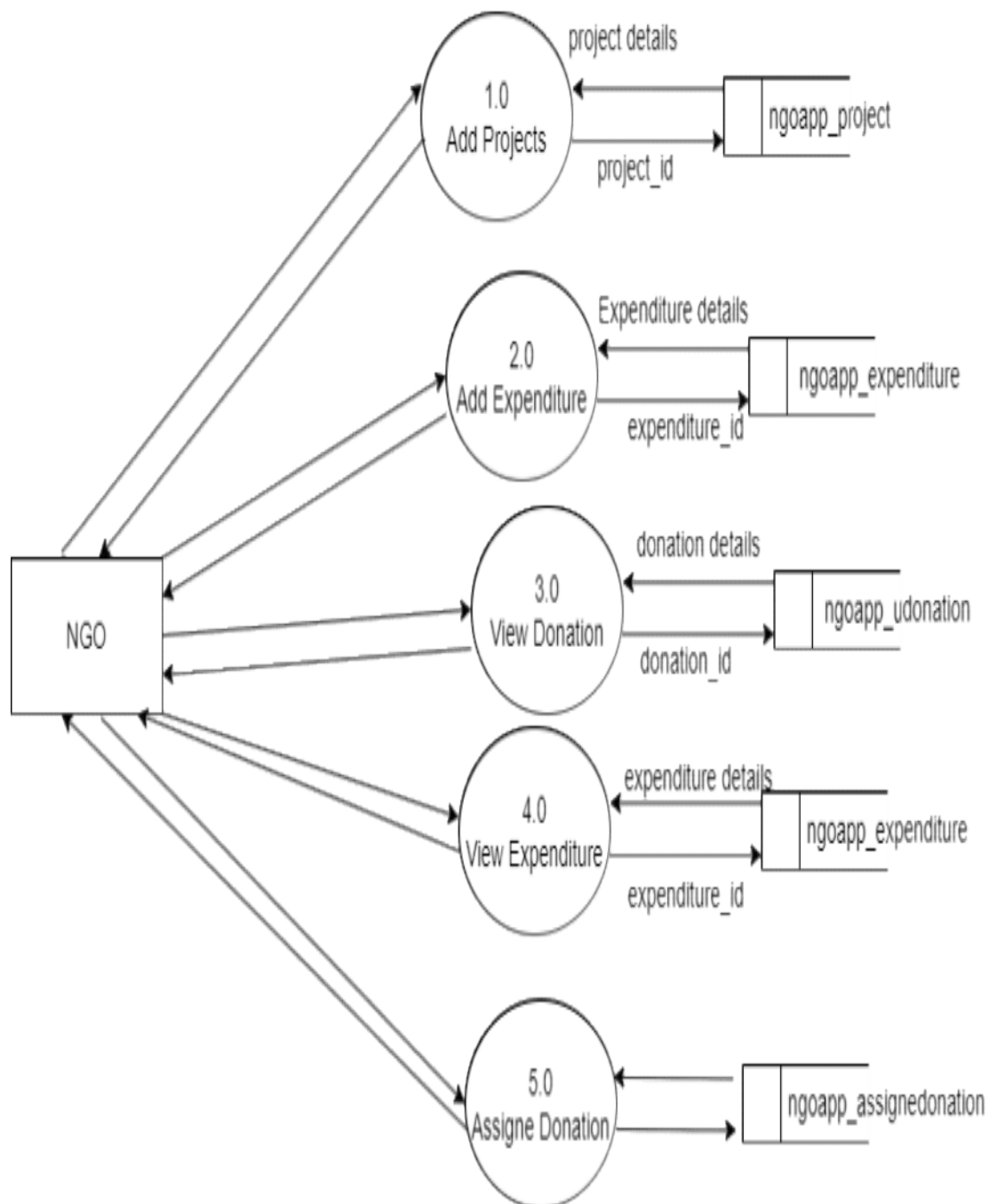
A context diagram is a level-0 DFD and represents the entire system elements as a single bubble with input and output data indicated by incoming and outgoing arrows respectively. The user gives data or commands as input and the user will get the details as output. Steps to construct data flow diagram:

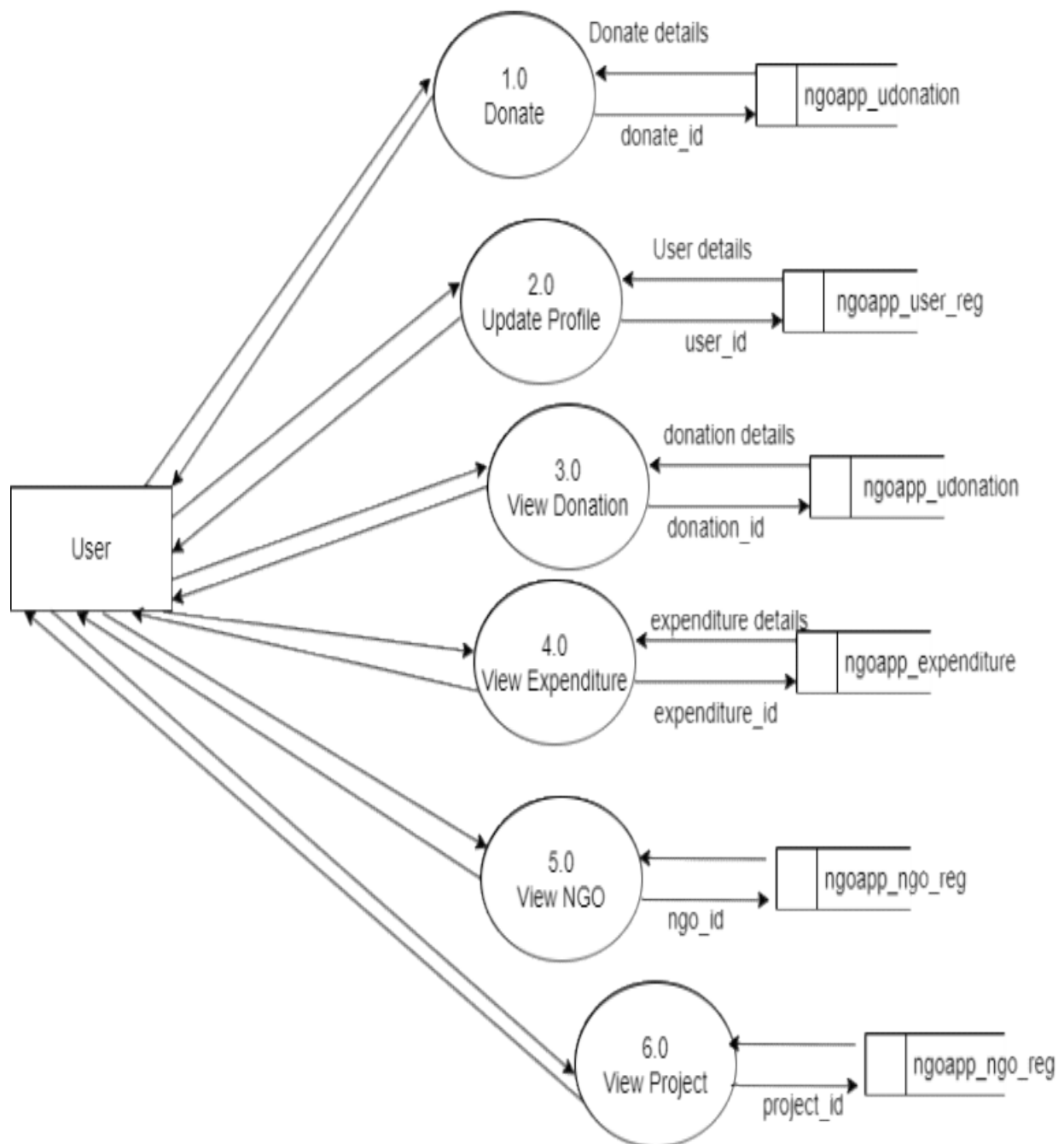
Four steps are commonly used to construct a DFD:

- Process should be named and numbered for easy reference.
- Each name should be representative of the process.
- The direction of flow is from top to bottom and from left to right.
- Then a process is exploded in to lower level details they are numbered.

LEVEL 0

LEVEL 1(ADMIN)

LEVEL 1(NGO)

LEVEL 1(USER)

6. CODING

6.1 CODE SNIPPETS

Views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import auth, User
from django.contrib import messages
from django.views.generic import TemplateView
from django.contrib.auth import login
from django.core.mail import EmailMessage
from django.conf import settings
from django.core.files.storage import FileSystemStorage

from ngoapp.models import UserType, ngo_reg, user_reg

# from restaurantapp.models import UserType, customer_reg, restaurant_reg
# from college.models import UserType, shop_reg, user_reg

class IndexView(TemplateView):
    template_name = 'index.html'

class UserReg(TemplateView):
    template_name = 'user_reg.html'

class NgoReg(TemplateView):
    template_name = 'ngo_reg.html'

    def post(self, request, *args, **kwargs):

        name = request.POST['nname']
        oname = request.POST['oname']

        ph = request.POST['phone']
        em = request.POST['email']
        department = request.POST['address']
        image = request.FILES['licence']
```



```
username = request.POST['username']
password = request.POST['password']
im = FileSystemStorage()
images = im.save(image.name, image)

try:
    user =
    User.objects.create_user(first_name=name,email=em,password=password,username=username,last_name='0')
    user.save()
    se=ngo_reg()
    se.user=user
    se.ename=ename
    se.licence=images

    se.phone=ph
    se.address=department
    se.save()

    usertype = UserType()
    usertype.user = user
    usertype.type = 'ngo'
    usertype.save()
    return render(request, 'index.html', {'message': "successfully Registered"})
except:
    messages = "Enter Another Username, user already exist"
    return render(request,'index.html',{'message':messages})

class UserReg(TemplateView):
    template_name = 'user_reg.html'

    def post(self, request, *args, **kwargs):

        name = request.POST['name']

        ph = request.POST['phone']
        em = request.POST['email']
```

```
department = request.POST['address']
image = request.FILES['ID']
photo = request.FILES['photo']

username = request.POST['username']
password = request.POST['password']
im = FileSystemStorage()
images = im.save(image.name, image)
pm = FileSystemStorage()
photo = pm.save(photo.name, photo)

try:
    user =
    User.objects.create_user(first_name=name,email=em,password=password,username=username,last_name='0')
    user.save()
    se=user_reg()
    se.user=user
    se.idproof=images
    se.photo=photo
    se.phone=ph
    se.address=department
    se.save()

    usertype = UserType()
    usertype.user = user
    usertype.type = 'user'
    usertype.save()
    return render(request, 'index.html', {'message': "successfully Registered"})
except:
    messages = "Enter Another Username, user already exist"
    return render(request,'index.html',{'message':messages})

class login_view(TemplateView):
    template_name="login.html"
    def post(self,request,*args,**kwargs):
        unname=request.POST['username']
        password =request.POST['password']
```

```
user=auth.authenticate(username=username,password=password)
if user is not None:
    login(request,user)
    if user.last_name=='1':
        if user.is_superuser:
            return redirect('/admin')
        elif UserType.objects.get(user_id=user.id).type=="ngo":
            return redirect('/ngo')
        elif UserType.objects.get(user_id=user.id).type == "user":
            return redirect('/user')

    else:
        return render(request,'login.html',{'message':" User Account Not
Authenticated"})

    else:
        return render(request,'login.html',{'message':" User Account Not
Authenticated"})
    else:
        return render(request,'index.html',{'message':"Invalid Username or
Password"})
```

models.py

```
from django.contrib.auth.models import User

# Create your models here.
class UserType(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE)
    type = models.CharField(max_length=50)

class ngo_reg(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE)
    oname= models.CharField(max_length=50)
    phone= models.CharField(max_length=50)
```

```
address= models.CharField(max_length=50)
licence= models.ImageField(null=True)

class user_reg(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE)
    phone= models.CharField(max_length=50)
    address= models.CharField(max_length=50)
    idproof= models.ImageField(null=True)
    photo= models.ImageField(null=True)

class project(models.Model):
    ngo= models.ForeignKey(ngo_reg,on_delete=models.CASCADE)
    pname=models.CharField(max_length=150)
    pdes= models.CharField(max_length=1000)
    amount= models.CharField(max_length=150)
    photo= models.ImageField(null=True)
```

admin_view.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import auth, User
from django.contrib import messages
from django.views.generic import TemplateView, View
from django.contrib.auth import login
from django.core.mail import EmailMessage
from django.conf import settings
from django.core.files.storage import FileSystemStorage

from ngoapp.models import assigneddonation, expenditure, ngo_reg, project,
user_reg

class AdminIndexView(TemplateView):
    template_name = 'admin/index.html'
```

```
class ngo_verify(TemplateView):
    template_name = 'admin/approve_ngo.html'

    def get_context_data(self, **kwargs):
        context = super(ngo_verify,self).get_context_data(**kwargs)

        app_rj =
        ngo_reg.objects.filter(user__last_name='0',user__is_staff='0',user__is_active='1')

        context['app_n'] = app_rj
        return context

class user_verify(TemplateView):
    template_name = 'admin/approve_user.html'

    def get_context_data(self, **kwargs):
        context = super(user_verify,self).get_context_data(**kwargs)

        app_rj =
        user_reg.objects.filter(user__last_name='0',user__is_staff='0',user__is_active='1')

        context['app_u'] = app_rj
        return context

class ApproveView(View):
    def dispatch(self, request, *args, **kwargs):
        id = request.GET['id']
        user = User.objects.get(pk=id)
        user.last_name='1'
        user.save()
        return render(request,'admin/index.html',{'message':" Account Approved"})

class RejectView(View):
    def dispatch(self, request, *args, **kwargs):
        id = request.GET['id']
        user = User.objects.get(pk=id)
        user.last_name='1'
```

```
user.is_active='0'
user.save()
return render(request,'admin/index.html',{'message':"Account Rejected"})

class ViewNgo(TemplateView):
    template_name = 'admin/viewngo.html'

    def get_context_data(self, **kwargs):
        context = super(ViewNgo,self).get_context_data(**kwargs)

        app_user =
ngo_reg.objects.filter(user__last_name='1',user__is_staff='0',user__is_active='1')

        context['ngo'] = app_user
        return context

class View_projects(TemplateView):
    template_name = 'admin/view_project.html'

    def get_context_data(self, **kwargs):

        id =self.request.GET['id']

        context = super(View_projects,self).get_context_data(**kwargs)
        # user = User.objects.get(pk=self.request.user.id)
        # ngo=ngo_reg.objects.get(user_id=user.id)
        c = project.objects.filter(ngo_id=id)
        context['p'] = c
        return context

class Viewspecificexpenditure(TemplateView):
    template_name = 'admin/view_expenditure.html'
```

```
def get_context_data(self, **kwargs):

    id=self.request.GET['id']
    context = super(Viewspecificexpenditure,self).get_context_data(**kwargs)
    # user = User.objects.get(pk=self.request.user.id)
    # ngo=ngo_reg.objects.get(user_id=user.id)

    c = expenditure.objects.filter(project_id=id)
    p=project.objects.get(id=id)

    price = 0
    for i in c:
        price = price + int(i.amount)

    context['a'] = price
    context['p'] = p
    context['ex'] = c

    return context
```

```
class Viewproject(TemplateView):
    template_name = 'admin/view_projects.html'

    def get_context_data(self, **kwargs):

        context = super(Viewproject,self).get_context_data(**kwargs)
        # user = User.objects.get(pk=self.request.user.id)
        # ngo=ngo_reg.objects.get(user_id=user.id)
        c = project.objects.all()
        context['p'] = c
        return context
```

```
class Viewspecificddonation(TemplateView):
    template_name = 'admin/view_specificdonations.html'

    def get_context_data(self, **kwargs):
```

```
id =self.request.GET['id']
context = super(Viewspecificddonation,self).get_context_data(**kwargs)
# user = User.objects.get(pk=self.request.user.id)
# ngo=ngo_reg.objects.get(user_id=user.id)

c = assigneddonation.objects.filter(project_id=id)
p=project.objects.get(id=id)

price = 0
for i in c:
    price = price + int(i.amount)

context['asz'] = price
context['fe'] = c
context['p'] = p

return context
```

user_view.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import auth, User
from django.contrib import messages
from django.views.generic import TemplateView, View
from django.contrib.auth import login
from django.core.mail import EmailMessage
from django.conf import settings
from django.core.files.storage import FileSystemStorage

from ngoapp.models import assigneddonation, expenditure, ngo_reg, project,
udonation, user_reg
```



```
class UserindexView(TemplateView):
    template_name = 'user/index.html'

class ProfileView(TemplateView):
    template_name = 'user/view_profile.html'

    def get_context_data(self, **kwargs):

        context = super(ProfileView,self).get_context_data(**kwargs)
        user = User.objects.get(pk=self.request.user.id)

        c = user_reg.objects.get(user_id=user.id)
        context['fe'] = c
        return context

class EditProfile(TemplateView):
    template_name = 'user/edit_profile.html'

    def get_context_data(self, **kwargs):

        context = super(EditProfile,self).get_context_data(**kwargs)
        user = User.objects.get(pk=self.request.user.id)

        c = user_reg.objects.filter(user_id=user.id)
        context['fe'] = c
        return context

    def post(self, request, *args, **kwargs):

        name = request.POST['name']
        id = request.POST['uid']
        id2 = request.POST['rid']
        ph = request.POST['phone']
        em = request.POST['email']
        department = request.POST['address']
        image = request.FILES['ID']
        username = request.POST['username']
```

```
im = FileSystemStorage()
images = im.save(image.name, image)
photo = request.FILES['photo']
pm = FileSystemStorage()
photos = pm.save(photo.name, photo)

try:
    i = User.objects.get(pk=id)
    u = user_reg.objects.get(pk=id2)
    i.first_name=name
    i.email=em
    i.username=username
    i.save()

    u.idproof=images
    u.phone=ph
    u.photo=photos
    u.address=department
    u.save()

    return render(request, 'user/index.html', {'message': "Updated
Successfully"})
except:
    messages = "Can't update"
    return render(request,'user/index.html',{'message':messages})

class Donation(TemplateView):
    template_name = 'user/donation.html'

    def get_context_data(self, **kwargs):

        context = super(Donation,self).get_context_data(**kwargs)
        # user = User.objects.get(pk=self.request.user.id)

        c = ngo_reg.objects.all()
```

```
context['fe'] = c
return context

def post(self, request, *args, **kwargs):

    user = User.objects.get(pk=self.request.user.id)
    reg=user_reg.objects.get(user_id=user.id)

    nid=request.POST['id']
    ph = request.POST['amount']
    em = request.POST['amt']

    try:

        se=udonation()
        se.reg_id=reg.id
        se.amt=em
        se.amount=ph
        se.ngo_id=nid
        se.status="not used yet"
        se.save()

        return render(request, 'user/index.html', {'message': "Successfully
Donated"})
    except:
        messages = "Cant donate"
        return render(request,'user/index.html',{'message':messages})

class ViewDonation(TemplateView):
    template_name = 'user/view_donation.html'

    def get_context_data(self, **kwargs):

        context = super(ViewDonation,self).get_context_data(**kwargs)
        user = User.objects.get(pk=self.request.user.id)
        reg=user_reg.objects.get(user_id=user.id)

        c = udonation.objects.filter(reg_id=reg.id)
```

```
        context['fe'] = c
        return context

class ViewNgo(TemplateView):
    template_name = 'user/view_ngo.html'

    def get_context_data(self, **kwargs):
        context = super(ViewNgo,self).get_context_data(**kwargs)

        app_user =
        ngo_reg.objects.filter(user__last_name='l',user__is_staff='0',user__is_active='1')

        context['ngo'] = app_user
        return context

class View_projects(TemplateView):
    template_name = 'user/view_project.html'

    def get_context_data(self, **kwargs):

        id =self.request.GET['id']

        context = super(View_projects,self).get_context_data(**kwargs)
        # user = User.objects.get(pk=self.request.user.id)
        # ngo=ngo_reg.objects.get(user_id=user.id)
        c = project.objects.filter(ngo_id=id)
        context['p'] = c
        return context

class Viewproject(TemplateView):
    template_name = 'user/view_projects.html'

    def get_context_data(self, **kwargs):

        context = super(Viewproject,self).get_context_data(**kwargs)
        # user = User.objects.get(pk=self.request.user.id)
        # ngo=ngo_reg.objects.get(user_id=user.id)
        c = project.objects.all()
```

```
context['p'] = c
return context

class Viewspecificddonation(TemplateView):
    template_name = 'user/view_specificdonations.html'

    def get_context_data(self, **kwargs):

        id = self.request.GET['id']
        context = super(Viewspecificddonation, self).get_context_data(**kwargs)
        # user = User.objects.get(pk=self.request.user.id)
        # ngo = ngo_reg.objects.get(user_id=user.id)

        c = assigneddonation.objects.filter(project_id=id)
        p = project.objects.get(id=id)

        price = 0
        for i in c:
            price = price + int(i.amount)

        context['asz'] = price
        context['fe'] = c
        context['p'] = p

        return context

class Viewspecificexpenditure(TemplateView):
    template_name = 'user/view_expenditure.html'

    def get_context_data(self, **kwargs):

        id = self.request.GET['id']
        context = super(Viewspecificexpenditure, self).get_context_data(**kwargs)
        # user = User.objects.get(pk=self.request.user.id)
        # ngo = ngo_reg.objects.get(user_id=user.id)

        c = expenditure.objects.filter(project_id=id)
```

```
p=project.objects.get(id=id)

price = 0
for i in c:
    price = price + int(i.amount)

context['a'] = price
context['p'] = p
context['ex'] = c

return context
```

settings.py

```
"""
```

Django settings for NGO project.

Generated by 'django-admin startproject' using Django 4.1.1.

For more information on this file, see

<https://docs.djangoproject.com/en/4.1/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/4.1/ref/settings/>

```
"""
```

```
import os
```

```
from pathlib import Path
```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.
```

```
BASE_DIR = Path(__file__).resolve().parent.parent
```

```
# Quick-start development settings - unsuitable for production
```

```
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
```

```
SECRET_KEY = 'django-insecure-
```

```
189(n5b9#kv!$)n3o=uiub3o_o@npfl+mtat4m!10l)&9h&rcq'
```

```
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'ngoapp'  
]
```

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]
```

```
ROOT_URLCONF = 'NGO.urls'
```

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [os.path.join(BASE_DIR, 'templates')],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  
                'django.template.context_processors.debug',
```

```
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
]

WSGI_APPLICATION = 'NGO.wsgi.application'

# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'ngo',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'localhost',
    }
}

# Password validation
# https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
]
```



```
{
    'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]
```

```
# Internationalization
```

```
# https://docs.djangoproject.com/en/4.1/topics/i18n/
```

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/4.1/howto/static-files/
```

```
STATIC_URL = 'static/'
```

```
STATICFILES_DIRS=[os.path.join(BASE_DIR,'static')]
```

```
STATIC_ROOT=os.path.join(BASE_DIR,"")
```

```
MEDIA_URL='/media/'
```

```
MEDIA_ROOT=os.path.join(BASE_DIR,'media')
```

```
# Default primary key field type
```

```
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Login.html

```
{% extends 'base.html' %}
```

```
{% load static %}
```

```

{% block content %}

{% if message %}
<script>
alert('{{ message }}')
</script>
{% endif %}


<script>
.cont
{
    top=50%;
    left=50%;
    position=absolute;
}
</script>


<div class="site-section bg-image overlay-primary" style="background-image: url('{{ static
'images/img_1.jpg' }}');">
<div class="container">
<div class="row align-items-stretch">
    {% comment %} <div class="col-md-6">
        
    </div> {% endcomment %}
    <div class="col-md-6">
        <div class="bg-white h-100 p-4 shadow">

            <div class="col-md-12">
                {% comment %} <div class="bg-gray h-100 p-4 shadow"> {% endcomment %}
                <h3 class="mb-4 text-cursive">Login</h3>
                <form action="" method="post" >
                    {% csrf_token %}
                    <div class="form-group">
                        <input type="text" class="form-control" placeholder="Username" name="username"
required>
                    </div>
                    <div class="form-group">
                        <input type="password" class="form-control" placeholder="Password"
name="password" required>
                    </div>

                    <div class="form-group">
                        <input type="submit" value="LOGIN" class="btn btn-primary">
                    </div>
                </form>
                {% comment %} </div> {% endcomment %}
            </div>
        </div>
    </div>
</div>

```

```

        </div>
    </div>
</div>
</div>
</div>

<br>
<br><br><br><br>
<div class="container" >

</div>

{% endblock %}

```

ngo_reg.html

```

{% extends 'base.html' %}

{% load static %}

{% block content %}

{% if message %}
<script>
alert("{{ message }}")
</script>
{% endif %}

<br><br>

<div class="site-section bg-image overlay-primary" style="background-image: url('{{ static
'images/img_1.jpg' %}}');">
    <div class="container">
        <div class="row align-items-stretch">
            {% comment %} <div class="col-md-6">
                
            </div> {% endcomment %}
            <div class="col-md-6">
                <div class="bg-white h-100 p-4 shadow">
                    <h3 class="mb-4 text-cursive">NGO Registration</h3>
                    <form action="" method="post" enctype="multipart/form-data" >
                        {% csrf_token %}
                        <div class="form-group">
                            <input type="text" name='nname' class="form-control" placeholder="NGO
Name"required >
                        </div>
                        <div class="form-group">
                            <input type="text" name='oname' class="form-control" placeholder="Founder Name"
required >

```

```

    </div>
    <div class="form-group">
        <input type="email" name="email" class="form-control" placeholder="Email" required >
    </div>
    <div class="form-group">
        <input type="number" name="phone" class="form-control" placeholder="Contact"
required >
    </div>
    <div class="form-group">
        <textarea name="address" class="form-control w-100" cols="62" rows="6" placeholder
= 'Address' style="border-0"></textarea>
    </div>
    <div class="form-group">
        <label>Upload Licence</label>
        <input type="file" name="licence" class="form-control" placeholder="" required >
    </div>
    <div class="form-group">
        <input type="text" name='username' class="form-control"
placeholder="Username"required >
    </div>
    <div class="form-group">
        <input type="password" name='password' class="form-control" placeholder="Password"
required >
    </div>
    <div class="form-group">
        <input type="submit" value="Register Now" class="btn btn-primary">
    </div>
</form>
</div>
</div>
</div>
</div>
</div>

{% endblock %}

```

user_reg.html

```

{% extends 'base.html' %}

{% load static %}

{% block content %}

{% if message %}
<script>
alert("{} {{ message }}")
</script>
{% endif %}

<br><br>

```

```

<div class="site-section bg-image overlay-primary" style="background-image: url('{% static
'images/img_1.jpg' %}');">
  <div class="container">
    <div class="row align-items-stretch">
      {% comment %} <div class="col-md-6">
        
      </div> {% endcomment %}
      <div class="col-md-6">
        <div class="bg-white h-100 p-4 shadow">
          <h3 class="mb-4 text-cursive">User Registration</h3>
          <form action="" method="post" enctype="multipart/form-data" >
            {% csrf_token %}
            <div class="form-group">
              <input type="text" name='name' class="form-control" placeholder="Name"required >
            </div>

            <div class="form-group">
              <input type="email" name="email" class="form-control" placeholder="Email" required >
            </div>
            <div class="form-group">
              <input type="number" name="phone" class="form-control" placeholder="Contact"
required >
            </div>
            <div class="form-group">
              <textarea name="address" class="form-control w-100" cols="62" rows="6" placeholder
= 'Address' style="border:0"></textarea>
            </div>
            <div class="form-group">
              <label>Upload ID proof</label>
              <input type="file" name="ID" class="form-control" placeholder="" required >
            </div>
            <div class="form-group">
              <label>Upload Photo</label>
              <input type="file" name="photo" class="form-control" placeholder="" required >
            </div>
            <div class="form-group">
              <input type="text" name='username' class="form-control"
placeholder="Username"required >
            </div>
            <div class="form-group">
              <input type="password" name='password' class="form-control" placeholder="Password"
required >
            </div>
            <div class="form-group">
              <input type="submit" value="Register Now" class="btn btn-primary">
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
  {% endblock %}

```

7. SYSTEM TESTING

7.1 TESTING

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be covering the end-to-end system's actual routine.

This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system. This testing checks complete end-to-end scenarios as per the customer's point of view.

Functional and Non-Functional tests also done by System testing. All things are done to maintain trust within the development that the system is defect-free and bug-free. System testing is also intended to test hardware/software requirements specifications. System testing is more of a limited type of testing; it seeks to detect within the "inter-assemblages".

7.2 OBJECTIVES OF SYSTEM TESTING

The primary objectives of System testing are as below:

- One of the primary objectives of System testing is to reduce risk. Even after individual testing of components, risk of how they will all come together to form a complete System still exists. System testing eliminates this risk by ensuring that it will function as per customer requirements.
- System testing must verify whether the design of the functional and non-functional behaviors of the system is as per the customer's specifications.
- Validate that the system is complete and will work as expected.
- System testing aims to build confidence in the quality of the system.
- System testing also aims to find defects and to prevent defects from escaping to

higher test levels or production. Additionally, it is the only phase that occurs on the full System just before the User Acceptance testing. So, it's critical to find all the possible defects at this stage, and they don't leak to production.

- System Testing results are used by stakeholders to make release decisions. The Entry criteria for User Acceptance testing is the basis completion of System Testing. System testing may also adhere to legal or regulatory requirements or standards.

7.3 TYPES OF TESTING

- Unit Testing
- Integration Testing
- Validation Testing
- Output testing

7.3.1 Unit Testing

Unit Testing is a software testing technique by means of which individual units of software i.e., group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer.

In my system I want to check the information like whether the inputs are saved to back and correctly. So, every form includes testing because I wanted to maintain my database, because information like document to be saved, the personal information, security features are so sensitive and should check it perfectly by each module from the beginning. These are checked step itself.

7.3.2 Integration Testing

Data can be test across an interface; one module can have adverse effect on another, sub function when combined may not produce the desired function. Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated within the interface.

The objective is to take unit tested modules and built a program structure that has been dictated by design. All modules are combined in this testing step. The entire program is tested. Correction is difficult at this stage because the isolation of causes is complicated by the vast expense of the program. Thus, in the integration testing step all the errors uncover are corrected for the next testing step. Primarily I had met with several errors like data saving and table linking. These were corrected well. There were some simple mistakes like setting the start-up page and all. Finally, I fixed login page as start-up page and run again. So, the integration testing was done well.

7.3.3 Validation Testing

Interfacing errors have been uncovered and corrected and a final series of software test validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the user. Software validation is achieved through a series of tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- The functions or performance characteristics confirm to specifications and are accepted.
- Validation from specification is uncovered and a deficiency created.

Deviation or error discovered at this step in this project is corrected prior to completion of the project with the help of the user. Thus, the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

7.3.4 Output Testing

The next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specific format. The output generated or displayed by the system under consideration is tested asking the users about the format required by them. Here, the output is considered in two ways: one is on the screen and the other is printed format.

The output format on the screen is found to be correct as the format designed according to the user needs. For the hard copy also, the output comes out as specified by the user. Hence output testing doesn't result in any connection in the system.

8. SYSTEM IMPLEMENTATIONS

The purpose of System Implementation can be summarized as follows:

making the new system available to a prepared set of users (the deployment) and positioning on-going support and maintenance of the system within the Performing Organization (the transition).

At a finer level of detail, deploying the system consists of executing all steps necessary to educate the Consumers on the use of the new system, placing the newly developed system into production, confirming that all data required at the start of operations is available and accurate, and validating that business functions that interact with the system are functioning properly. Transitioning the system support responsibilities involves changing from a system development to a system support and maintenance mode of operation, with ownership of the new system moving from the Project Team to the Performing Organization.

A key difference between System Implementation and all other phases of the life cycle is that all project activities up to this point have been performed in safe, protected, and secure environments, where project issues that arise have little or no impact on day-to-day business operations. Once the system goes live, however, this is no longer the case. Any miscues at this point will almost certainly translate into direct operational and/or financial impacts on the Performing Organization. It is through the careful planning, execution, and management of System Implementation activities that the Project Team can minimize the likelihood of these occurrences and determine appropriate contingency plans in the event of a problem.

9. SECURITY TECHNOLOGIES AND POLICIES

9.1 SYSTEM SECURITY

9.1.1 Threats To A System Security

Any act or object that poses a danger to computer assets is known as threats. Management must be aware of the various kinds of threats facing the organization. By examining each threat category, the management can effectively protect the assets through policy, training and technology.

A threat can be either "intentional" (i.e., hacking: an individual cracker or a criminal organization) or "accidental" (e.g. the possibility of a computer malfunctioning, or the possibility of a natural disaster such as an earthquake, a fire, or a tornado) or otherwise a circumstance, capability, action, or event.

There are several types of threats.

- Attempt to access websites and modify or destroy its contents.
- Send malicious programs such as viruses, worms and Trojans to a web server by a browser.

The potential network security threats can be classified into two:

- ❖ Passive threats
- ❖ Active threats.

1. Passive Threats: The monitoring and recording of data while the data are being transmitted over a communication facility, by an unauthorized user is a passive threat. The goal of the attacker is to obtain information that is being transmitted. Two types of passive threats are there- (a) Release of message content and (b) Traffic analysis.

In release of message content threat, an e-mail message or file transferred over the internet is read and recorded by the attacker. Traffic analysis attack enables the attacker to understand the location and identity of communicating hosts. Passive threats are difficult to detect since they do not involve any alteration of data.

2. Active Threats: Active threat involves the alternation of digital data or generation of spurious data by an attacker. These involve some modification of data stream or the creation of the false stream. Active threats may fall in three categories, (a) message stream modification, (b) denial of message service and (c) masquerade.

Message stream modification means that some portion of the original message is being altered, replaced or the message is delayed.

The denial of service prevents the normal use or management of communication facilities. This attack may have a specific target.

9.2 SECURITY POLICY

An organization concerned about protecting its data, network or e-commerce assets should develop a security policy. A security policy is a written statement describing which assets to protect and why they are being protected, who is responsible for that protection, and which behaviours are acceptable, and which are not. The policy primarily addresses physical security, network security, access authorization, virus protection and disaster recovery. Security policy is usually developed over a period of time, and it is reviewed and updated at regular intervals.

Most organizations follow a five-step process when creating a security policy.

- Determine which assets must be protected from which threats.
- Determine who should have access to various parts of the system or specific information.
- Identify resources available or needs to protect the assets.
- Develop a security policy based on the information gathered in the first three steps.
- Following the written policy, develop or buy software, hardware and physical barriers to implement the security policy.

Comprehensive plan for security should protect a system's privacy, integrity, and availability and authenticate users.

9.3 SYSTEM SECURITY MEASURES

Security for e-commerce means complying with several key requirements, while making sure that online security measures are adopted. Having the right security measures in place is essential to maintain customer trust and confidence, and to ensure customer retention and loyalty.

To protect the confidential data from intruders, there are several security measures available today.

- ❖ Passwords
- ❖ Virus protection
- ❖ Firewalls
- ❖ Encryption

1. Passwords

The most inexpensive and perhaps the easiest method of security is achieved through password protection. A password is a secret word or string of characters that is used for user authentication to prove identity, or for access approval to gain access to a resource. Now passwords are commonly used by people to login to a computer system, access ATMs, etc. A password is also known as a passkey. It serves as a security measure against unauthorized access to data. Password security adds no cost to the network and is potentially a useful security measure.

2. Virus Protection

A computer virus is a type of malicious codes or program written to alter the way a computer operates and is designed to spread from one computer to another. A virus operates by inserting or attaching itself to a legitimate program or document that supports macros in order to execute its code. In the process, a virus has the potential to cause unexpected or damaging effects, such as harming the system software by corrupting or destroying data.

Antivirus software or anti-virus software, also known as anti-malware, is a computer

program used to prevent, detect, and remove malware. Antivirus software was originally developed to detect and remove computer viruses, hence the name. However, with the proliferation of other kinds of malware, antivirus software started to provide protection from other computer threats.

3. Firewall

One widely used strategy to improve system security is to use a firewall. A firewall consists of software and hardware setup between an internal computer network and the internet. A computer network manager sets up the rules for the firewall to filter out unwanted instructions. These rules are set up in such a way that unauthorized access is much more difficult. A system administrator can decide, for example, that only users within the firewall can access files, or that those outside the firewall have limited capabilities to modify the files.

4. Encryption:

One way to keep files and data safe is to use encryption. This is often used when data is transferred over the internet, where it could potentially be seen by others. Encryption is the process of encoding messages so that it can only be viewed by authorized individuals. An encryption key is used to make the message unreadable, and a secret decryption key is used to decipher the message. Encryption is widely used in systems like e-commerce and internet banking, where the databases contain very sensitive information.

9.4 SECURITY

Proper security checking is done so that no external intruders enter into the overall system and make changes in it. To implement this, we should supply a login screen to enter the password. The password can be letters, digits, special characters or combination of them. The user cannot see the password while entering it. If the entered password is correct then he can proceed with the program, else an error message is provided to him, and the control goes back to the login window. As a result of this the number of users using the software can be controlled. This makes our program more compact. Here, in this system admin and customer are the users who are provided with the login.

10. FUTURE SCOPE OF THE PROJECT

10.1 FUTURE ENHANCEMENT

Almost every project is subjected to change on depending on the client requirements. Since this system is subjected to change for each and every client, there is always a scope for further enhancement. The system and the architecture of the assessment system is a compatible one, so addition of new modules can be done without much difficulty.

The software is developed in visual basic which makes the system more reliable and compatible with other environments. The application proves better extensibility and flexibility for future enhancements. Any further requirement application is possible with the same feature guaranteed. It is a user –friendly system, which is very easy and convenient to use.

The system is complete in the sense that it is operational and it is tested by entering data and getting reports in proper order. During the development of this project coding standards are followed for easy maintainability and extensibility. Though the new system provides a base for improving the efficiency of operations, there are lots of further enhancement that can be added to this project. Keeping this in view, provision has been made in the system to facilitate easy modification updating in future. Any modification will not affect the normal working of the system.

The developed system is very interactive, coded in such a way to ensure maximum user friendliness and also allows flexibility for future. The system developed automates most needed activities in an organization. The new system can be combined with an existing system as well. More and better advanced separation system can be build on top of the proposed system as and when the need arises. This is one of the main special feature of the proposed system.

10.2 SOFTWARE SCOPE

- **Extensibility:** This software is extendable in ways that its original developers may not expect. The following principles enhance extensibility like hide data structure, avoid traversing multiple links or methods, avoid case statements on object type and distinguish public and private operations.
- **Reusability:** Reusability is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort over several designs. Reducing the amount of code also simplify understanding, which increases the likelihood that the code is correct. We follow up both types of reusability: Sharing of newly written code within a project and reuse of previously written code on new projects.
- **Understandability:** A method is understandable if someone other than the creator of the method can understand the code (as well as the creator after a time lapse). We use the method, which small and coherent helps to accomplish this.
- **Cost-effectiveness:** Its cost is under the budget and make within given time period. It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy the entire requirement.

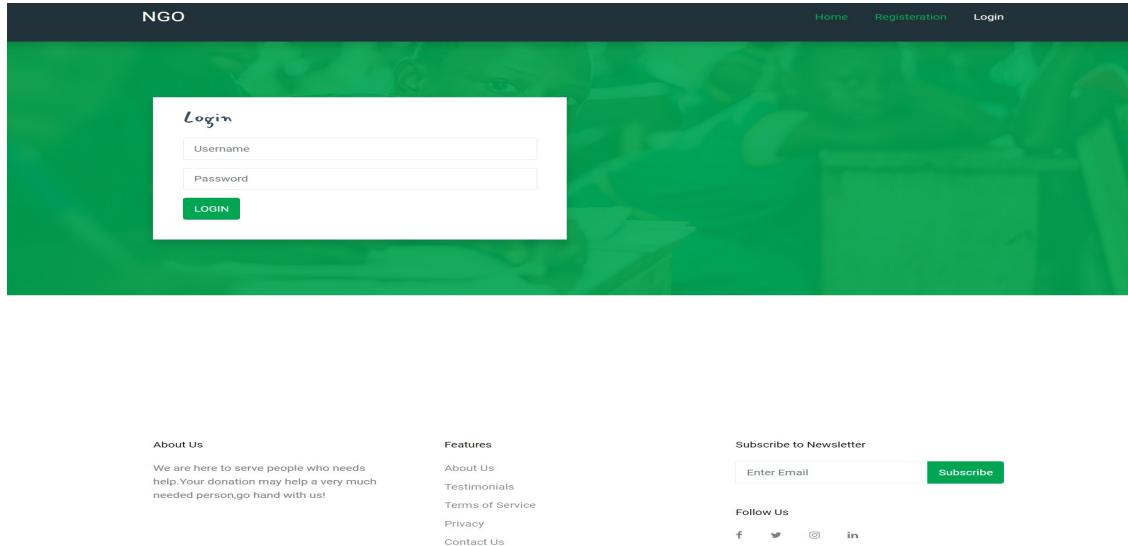
11. CONCLUSION

NGO Management made easy to manage the fund raising process than normal way. Most of the NGOs are usually structured around specific issues like health, human rights or environment. An NGO provides expertise and analysis and thus assists in monitoring international agreements. NGOs are very important since they offer an organization for local communication, action and also distributing resources when there are no existing local organizations. In fact, an NGO provides a mechanism that could possibly work where the government has failed. As a result, it supports grass roots initiatives as well as recognizing and responding to the realities of the local people. Next, cheaper to implement. Since NGOs are actual non-profit organizations, various projects can be achieved without having to use the government's money. This is because there are many private donors who support the NGOs and this means that there will never be a lack of resources. Additionally, NGOs provide a good alternative to creating mass access structures. These mass access structures are extremely cumbersome, unreliable and costly. Another major advantage of NGOs is that they have the capability of communicating at all levels. This means that they can easily interact with the local people and relay their messages to top levels of the government. They are also capable of recruiting highly motivated staff and experts with lesser restrictions than employees working for the government.

12. APPENDIX

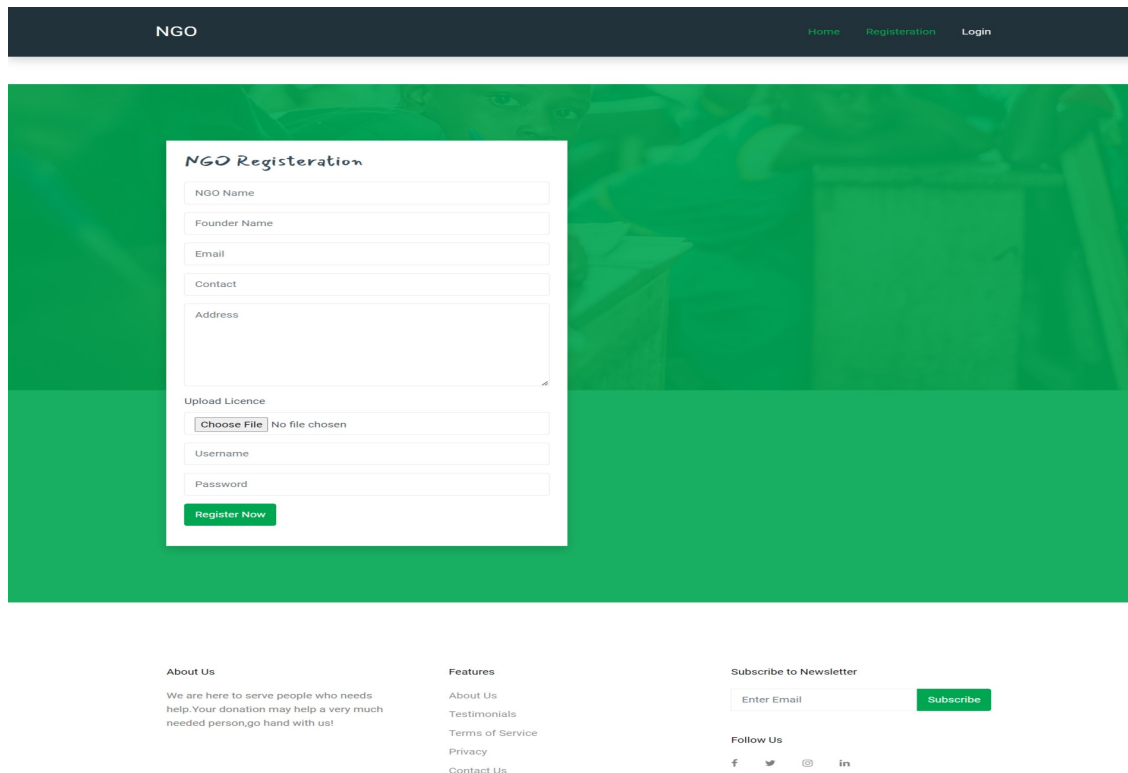
12.1 SCREENSHOTS

Login Page



The screenshot shows the login page of an NGO management system. The page has a dark blue header with the text "NGO" on the left and navigation links "Home", "Registration", and "Login" on the right. The main content area has a green background with a faint image of people. In the center, there is a white login form titled "Login". The form contains two input fields: "Username" and "Password", followed by a green "LOGIN" button. Below the login form, there is a footer section with three columns. The first column is titled "About Us" and contains the text: "We are here to serve people who needs help.Your donation may help a very much needed person,go hand with us!". The second column is titled "Features" and contains links: "About Us", "Testimonials", "Terms of Service", "Privacy", and "Contact Us". The third column is titled "Subscribe to Newsletter" and contains an "Enter Email" input field and a green "Subscribe" button. Below this, there is a "Follow Us" section with social media icons for Facebook, Twitter, Instagram, and LinkedIn.

NGO Registration



The screenshot shows the registration page of an NGO management system. The page has a dark blue header with the text "NGO" on the left and navigation links "Home", "Registration", and "Login" on the right. The main content area has a green background with a faint image of people. In the center, there is a white registration form titled "NGO Registration". The form contains several input fields: "NGO Name", "Founder Name", "Email", "Contact", "Address", "Upload Licence" (with a "Choose File" button and "No file chosen" text), "Username", and "Password". At the bottom of the form is a green "Register Now" button. Below the registration form, there is a footer section with three columns. The first column is titled "About Us" and contains the text: "We are here to serve people who needs help.Your donation may help a very much needed person,go hand with us!". The second column is titled "Features" and contains links: "About Us", "Testimonials", "Terms of Service", "Privacy", and "Contact Us". The third column is titled "Subscribe to Newsletter" and contains an "Enter Email" input field and a green "Subscribe" button. Below this, there is a "Follow Us" section with social media icons for Facebook, Twitter, Instagram, and LinkedIn.

NGO

[Home](#)[Registration](#)[Login](#)

User Registration

Name

Email

Contact

Address

Upload ID proof

Choose File

No file chosen

Upload Photo

Choose File

No file chosen

Username

Password

Register Now

About Us

We are here to serve people who needs help.Your donation may help a very much needed person.go hand with us!

Features

About Us

Testimonials

Terms of Service

Privacy

Contact Us

Subscribe to Newsletter

Enter Email

Subscribe

Follow Us

f

t

@

in

The screenshot shows a website with a dark blue header. On the left, the text 'NGO' is displayed. On the right, there are links for 'Home' and 'Logout'. The main content area has a heading 'Ngo List' in a handwritten-style font. Below this, there is a dark blue rectangular box containing the text 'Kothamangalam NGO' and a green button labeled 'View Projects'. At the bottom of the page, there are three columns of content. The first column is titled 'About Us' and contains the text: 'We are here to serve people who needs help.Your donation may help a very much needed person,go hand with us!'. The second column is titled 'Features' and lists: 'About Us', 'Testimonials', 'Terms of Service', 'Privacy', and 'Contact Us'. The third column is titled 'Subscribe to Newsletter' and features a white input field with the placeholder text 'Enter Email' and a green 'Subscribe' button. Below this, there is a section titled 'Follow Us' with icons for Facebook, Twitter, Instagram, and LinkedIn.

Edit Profile

NGO
Home
Logout

Profile

Name :Ahammed Shajmeer
Phone :7034854686
Email :shajmeer@gmail.com
Address :
Photo:
ID Proof:
Edit Profile

About Us

We are here to serve people who needs help.Your donation may help a very much needed person,go hand with us!

Features

About Us
Testimonials
Terms of Service
Privacy
Contact Us

Subscribe to Newsletter

Enter Email

Follow Us

f t @ in

User View Donation

NGO
Home
Logout

My Donation

Ngo Name	Amount in words	Amount	Status	Project Name
Kothamangalam NGO	twentythousand only/-	20000	used	home
Kothamangalam NGO	twentythousand only/-	500	not used yet	

About Us

We are here to serve people who needs help.Your donation may help a very much needed person,go hand with us!

Features

About Us
Testimonials
Terms of Service
Privacy
Contact Us

Subscribe to Newsletter

Enter Email

Follow Us

f t @ in

Project Form

NGO

[Home](#)
[Add](#)
[Donations](#)
[View Project](#)
[Logout](#)

Project Form

No file chosen

About Us

We are here to serve people who needs help. Your donation may help a very much needed person, go hand with us!

Features

[About Us](#)
[Testimonials](#)
[Terms of Service](#)
[Privacy](#)
[Contact Us](#)

Subscribe to Newsletter

Follow Us

[f](#)
[t](#)
[@](#)
[in](#)

Project Expenditure Form

NGO

[Home](#)
[Add](#)
[Donations](#)
[View Project](#)
[Logout](#)

Project Form

No file chosen

About Us

We are here to serve people who needs help. Your donation may help a very much needed person, go hand with us!

Features

[About Us](#)
[Testimonials](#)
[Terms of Service](#)
[Privacy](#)
[Contact Us](#)

Subscribe to Newsletter

Follow Us

[f](#)
[t](#)
[@](#)
[in](#)

13. REFERENCES

- <https://docs.microsoft.com/en-us/dotnet/>
- <https://stackoverflow.com/>
- <https://www.tutorialspoint.com/index.html>
- <https://medium.com/>