CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Everyone enjoys the events that feature a lot of food and other goods, most of which are wasted or remain unused. For various reasons, the majority of us waste food. Most of the time, we tend to squander the same kinds of food for the same two major reasons, however occasionally it's because plans have changed and it's beyond our control. The real power of this project is not just waste management, but it's the love and care for the homeless peoples. Users of the website do not profit from food donations, food receipts, or volunteer work. The Food for Needy Project is a website that accepts requests from donors who want to deal with food waste. Donors can donate to the recipient request comes to them at a specific time, and volunteers pick the order and further deliver that food to a needy place where the majority of people are lacking food in accordance with need status. This website maintains the data in a central location that is simultaneously accessible to all users.

1.2 PROJECT SPECIFICATION

The proposed system is a website in which receiver can book online for food. Also, that the doner can donate food or money for the homeless people according to need status.
The system includes 3 modules. They are:

1. **Approval Management –** In these Approval Management, admin can approve or reject the registered users.

2. **Receiver Management -** Receiver can manage their own profile and they can order the food through the website for their needs and also provide the details like members and quantity they need.

3. **Donation management –** Doners can donate money or food through the donation management module and volunteers provide the information about the donated food.

4. **Order Management -** In Order Management receiver can order the food for their needs and then doners gets the available list of orders from their own district after that doner can proceed with the order either partially or fully, then doners can go with their payment procedure.

5. **Payment Management –** Doners can make the payment after the donation and proceed with donations.

6. **Volunteer Assignment Management –** In this module after the order acceptance by the doner, admin assigns the volunteer to a particular order and then the volunteer delivers the order according to their needs.

7. **Membership management -** This module figures out information about homeless peoples in an area. If there is an increase found in count, the volunteers will update through website.

8. **Delivery Management –** Admin gets the list of orders that are accepted by the doners. After that admin works with the available list of orders and finds the appropriate volunteer and then assign volunteer to the delivery later volunteers pickup and deliver the food for the needy places.

9. **Volunteer Management –** Volunteer can register through the website with their details, after the approval by the admin they can view the task that are assigned by the admin. Volunteer delivers the food fastly for receiver and volunteer can also performs their daily task also and update the membership if any.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The process of gathering and evaluating data, identifying problems, and using the data to recommend system modifications is known as system analysis. During this problem-solving process, there must be considerable communication between the system users and the system developers. A system analysis or study should be the first step in any system development process. The system is carefully inspected and evaluated. The system analyst adopts the position of an interrogator and probes the inner workings of the existing system. The system's input is recognised, and the system is viewed as a whole. The different processes can be connected to the organisational outputs. System analysis involves comprehending the problem, identifying the significant and crucial factors, analysing and synthesising the numerous components, and choosing the best or, at the very least, most acceptable course of action.

The procedure has to be carefully investigated utilising a range of approaches, such as surveys and interviews. The data acquired by various sources has to be thoroughly evaluated in order to draw a conclusion. The conclusion is knowing how the system functions. This system is known as the present one. Now, problem areas have been identified after a detailed examination of the present system. Now the designer takes on the role of a problem-solver and attempts to fix the problems the company is experiencing. The solutions are replaced with proposals. The best option is then selected after an analytical comparison of the proposal and the existing system. The proposal is presented to the user with the option to accept or reject it. On user request, the proposal is assessed and appropriate revisions are implemented. As soon as the user is content with the suggestion, this loop breaks.

Preliminary research is the procedure of gathering and analysing data in order to use it for upcoming system investigations. Initial research requires strong collaboration between system users and developers since it involves problem-solving. It carries out several feasibility studies. These studies give a rough idea of the system activities, and this information may be utilised to choose the methods to employ for effective system research and analysis.

2.1 EXISTING SYSTEM

Existing system is not a fully automated system. Doner can register and they can donate for service. Existing Systems are not only for the food donation but also they provide many services like education loan, awareness etc. Payments are used for the services they provided.

It is necessary to modify the existing system in order to include additional information and make the system more efficient. Using the new system doners can register, view all information about orders that are provided by the receiver and donation can be accepted in payment or food etc.

2.2 DRAWBACKS OF EXISTING SYSTEM

- Difficult to understand.
- It is difficult to maintain important information in books.
- It doesn't bother about the receiver needs

2.3 PROPOSED SYSTEM

The proposed system is defined to meets all the disadvantages of the existing system. It is necessary to have a system that is more user friendly and user attractive for growth of system on such consideration the system is proposed. Users of this proposed system are admin, reciever,doner and volunteer. In our proposed system admin can view all the users and block or unblock the users in anytime. It allows the receiver to make the order based on their needs and the doners to make their donation and do their transactions by using online payment method and by food. Volunteer act as an interface between the receiver and doners. Volunteer will pick up the food and delivers to the needy places This website keeps the data in a centralized way which is available to all the users simultaneously. It is very easy to manage historical data in database. No specific training is required for the user to use this website.

2.4 ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources, and the system will work in almost all configurations. It has got following features:

➢ **Better security: -**

Unauthorized access must be prevented in order for data to stay safe. Data protection means that they are shielded against different types of loss. Security, integrity, privacy, and confidentiality are the four connected problems that make up the system security challenge. Security is maintained by requiring a username and password to sign in. As we use secured databases to maintain the papers, it will also ensure data security.

➢ **Ensure data accuracy: -**

The suggested solution does away with human mistakes made when entering user information during registration..

➢ **Better service: -**

Hard copy storage won't be a burden thanks to the product. For performing the same activity, we can also save time and resources. The data can be kept for a longer time without losing any information..

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

To ascertain if the project will, after completion, achieve the goals of the organisation in relation to the labour, effort, and time put in it, a feasibility study is carried out. The developer can forecast the project's usefulness and possible future thanks to a feasibility study. The premise for a feasibility study is the system proposal's viability, which includes the impact on the organisation, ability to meet user demands, and effective use of resources. As a result, a feasibility evaluation is frequently performed before a new proposal is approved for development.

The paper describes the project's viability and includes a variety of elements, such as its technical, economic, and operational viabilities, that were carefully considered during this project's feasibility assessment. These are its characteristics: -

3.1.1 Economical Feasibility

Analyses of costs and benefits are necessary to support the developing system. to ensure that attention is focused on the project that will produce the best outcomes the quickest. One of the factors is the cost associated with establishing a new system.

The following are a few of the significant fiscal queries raised during the initial inquiry:

- ➢ The expenses carry out an extensive system analysis.
- ➢ The price of the software and hardware.
- ➢ The advantages in terms of lower expenses or less expensive mistakes.

There are no manual costs involved with the suggested system because it was developed as part of a project. Additionally, the availability of all necessary resources suggests that the system might be implemented at a reasonable cost.

System expenses, development costs, and hosting costs made up the project's three cost categories. All estimates show that the project was created at a reasonable cost. Given that only open source software was used throughout its creation.

3.1.2 Technical Feasibility

The system must first undergo a technical evaluation. The assessment of this feasibility must be built around an overview design of the system's needs in terms of input, output, programmers, and processes. After identifying an outline system, the inquiry must next

recommend the type of equipment, essential steps for building the system, and methods of operating the system once it has been created. The investigation's technical difficulties include:

> Does the existing technology sufficient for the suggested one?
> Can the system expand if developed?

The project should be created in such a way that the required performance and functionality are met within the limitations. The project uses cryptographic methods and calls for a high resolution scanning device. The system may still be used even though the technology may become outdated after a while because a newer version of the same software still works with an earlier version. Therefore, this project only has a few limitations. The system was created using PHP for the front end and a MySQL server for the back end; it is technically feasible to complete the project. The system was created using PHP for the front end and a MySQL server for the back end; it is technically feasible to complete the project. The System used was also of good performance of Processor Intel i5 core; RAM 8 GB and, Hard disk 1TB

3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

> Is there sufficient support for the users?
> Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor   -   Intel core i5

RAM   -   8 GB

Hard disk   -   1 TB

3.2.2 Software Specification

Front End   -   HTML, CSS

Backend   -   MYSQL

Client on PC   -   Windows 7 and above.

Technologies used -   JS, HTML, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 PHP

PHP is a server-side scripting language used for general-purpose programming as well as web development. More than 244 million websites and 2.1 million web servers currently use PHP. The PHP group now produces the reference implementation of PHP, which was first developed by Rasmus Ledorf in 1995. PHP, a recursive acronym that once meant for personal Home page, now stands for PHP: Hypertext Preprocessor. A web server's PHP processor module interprets PHP code to produce the final web page. In order to handle data, PHP commands can be directly inserted into an HTML source file rather than calling an external file. The GNU General Public License (GPL) is incompatible with it due to restrictions on the use of the word PHP, and it has grown to incorporate a command-line interface capability. PHP is available for free deployment on the majority of web servers as well as a standalone shell on practically all platforms and operating systems.

3.3.2   MySQL

The most well-known Open Source SQL database management system, MySQL, was developed, distributed, and supported by Oracle Corporation. On the MySQL website, you may get the most latest information on the MySQL programme.

*   **MySQL is a database management system.**
    A database is a planned collection of data. Anything might be it, from a simple grocery list to a photo gallery to the vast quantity of information in a company network. The data stored in a computer database must be added to, retrieved from, and processed using a database management system, such as MySQL Server. Because computers are so good at processing massive amounts of data, database management systems—whether employed as standalone programmes or as a component of other applications—are crucial to computing.

*   **MySQL databases are relational.**
    A relational database stores the data in separate tables rather than consolidating it into one enormous warehouse. The database structures are stored in physical files that are optimised for speed. A versatile programming environment is offered by the logical model, which includes objects like databases, tables, views, rows, and columns. You might create rules to govern the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and "pointers" between other tables. Since a well-designed database upholds these constraints, your application won't ever run across inconsistent, duplicate, orphan, out-of-date, or missing data. The prefix "SQL" in MySQL stands for "Structured Query Language." SQL is the most widely used standard language for database access. You could explicitly input SQL (for instance, to produce reports), embed SQL statements into other languages' code, or utilise a language-specific API that hides the SQL syntax depending on your programming environment. SQL is specified by the ANSI/ISO SQL Standard. The SQL standard has undergone several changes since its creation in 1986. The 1992 standard is referred to in this document as "SQL92," the 1999 standard is referred to as "SQL," and the most recent version of the standard is referred to as "SQL: 2003." "The SQL standard" refers to the SQL Standard as it is at any given time.

- **MySQL software is Open Source.**

  Considering that the programme is open source, anybody may use and alter it. Anyone may use and download the MySQL software for free online. You have the right to go at the source code and make any required changes. The MySQL software adheres to the GPL (GNU General Public License), which outlines what you may and cannot do with the programme under specific conditions. If the GPL makes you uncomfortable or if you need to incorporate MySQL code into a for-profit application, you may buy a commercially licenced version from us. Check out the MySQL Licensing Overview for more information.

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**
  If it is what you're wanting, you should try it. MySQL Server may run easily on a desktop or laptop and needs little to no maintenance in addition to your other programmes, web servers, and other software. If you dedicate a whole workstation to MySQL, you may change the settings to make full use of the RAM, CPU, and I/O capabilities.

- **MySQL Server works in client/server or embedded systems.**
  The MySQL Database Software is a client/server system made up of a multi-threaded SQL server that supports several client applications and libraries, administration tools, and a broad range of application programming interfaces (APIs). Additionally, we provide MySQL Server as an integrated multi-threaded library that you can connect into your programme to create a standalone offering that is smaller, quicker, and simpler to operate.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step in the creation of any engineering system or product. Design is a creative process. A good design is the key to a system that works effectively. "Design" is the process of employing many approaches and concepts to thoroughly outline a process or a system so that it may be physically implemented. The process of employing several approaches and concepts to specify a tool, a procedure, or a system in sufficient detail to enable its physical actuality is one way to put it. Software design serves as the technical foundation of the software engineering process, regardless of the development paradigm used. The system design process develops the architectural detail required to build a system or product. As with any systematic approach, this software underwent the best design phase possible, fine-tuning all efficiency, performance, and accuracy levels. During the design stage, a user-oriented document is transformed into a document for programmers or database employees. Logical design and physical design are the two phases of system design development.

4.2 UML DIAGRAM

The artefacts of a software system are specified, visualised, constructed, and documented using a standard language called UML. A draught of the UML 1.0 specification was provided to the Object Management Group (OMG), which was in charge of creating UML, in January 1997.

UML stands for Unified Modeling Language. UML is different from other well-known programming languages like C++, Java, COBOL, etc. Software designs are made using a visual language called UML. UML is a general-purpose visual modelling language used for the definition, design, development, and documentation of software systems. Although representing software systems is the most popular use of UML, it is not the only one. Additionally, it is employed to simulate non-software-based systems. the process flow in the manufacturing plant, etc. UML is not a programming language, but there are tools that can create code from UML diagrams in a number of other languages. UML is intimately connected to the analysis and design of objects-oriented systems. It has been so well standardised that OMG now recognises UML as a standard. All the components and interactions make to a complete UML diagram that represents a system. The visual effect of the UML diagram is the most important aspect of the entire OMG standardisation process. All the components and interactions make to a complete UML diagram that

represents a system. The visual effect of the UML diagram is the most important factor in the entire process. Utilizing all the other elements completes it. It is completed by using all the other components. The following nine diagrams are part of UML:-

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram
- Statechart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

An illustration of the interactions between system components is a use case diagram. A use case is a method for locating, defining, and organising system needs. The word "system" in this context refers to a thing that is being built or operated, such as a website for mail-order service and product sales. Use case diagrams are used in UML (Unified Modeling Language), a standard language for modelling real-world objects and systems.
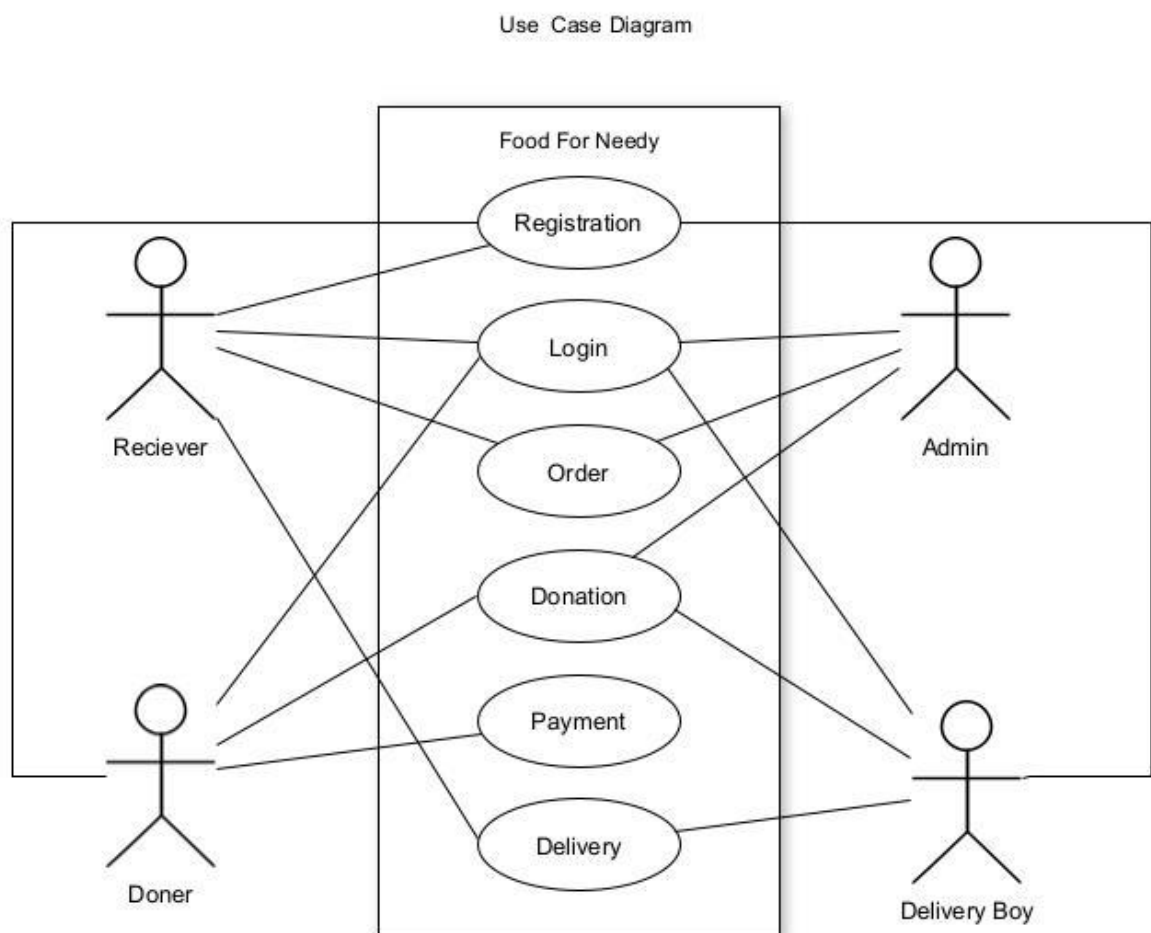
Planning general requirements, validating hardware designs, testing and debugging software products while they are still in development, creating online help resources, and finishing customer support-focused tasks are a few examples of system objectives. Use cases in the context of product sales, for instance, may include item ordering, catalogue updating, payment processing, and customer interactions. In a use case diagram, there are four elements.

- The boundary, which defines the system of interest in relation to the world around it.

- The actors, usually individuals involved with the system defined according to their roles.

- The use cases, which are the specific roles are played by the actors within and around the system.

- The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.

Use Case Diagram

Food For Needy

Registration

Login

Order

Donation

Payment

Delivery

Reciever

Admin

Doner

Delivery Boy

4.2.2 SEQUENCE DIAGRAM

A sequence diagram essentially shows how things interact with one another sequentially, or the order in which these interactions occur. A sequence diagram can also be referred to as event diagrams or event scenarios. Sequence diagrams show the actions taken by the components of a system in chronological order. Businesspeople and software engineers frequently use these diagrams to record and comprehend the requirements for new and current systems..

Sequence Diagram Notations –

i. **Actors –** In a UML diagram, an actor represents a particular kind of role in which it communicates with the system's objects. An actor is always beyond the purview of the system that we want to use the UML diagram to represent. We employ actors to portray a variety of roles, including those of human users and other outside subjects. In a UML diagram, an actor is represented using a stick person notation. In a sequence diagram, there might be several actors..

ii. **Lifelines –** An identifiable component known as a lifeline identifies a specific participant in a sequence diagram. A lifeline, then, essentially acts as a representation for each instance in a sequence diagram. The lifeline elements are located at the top of a sequence diagram.

iii. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:
- Synchronous messages

- Asynchronous Messages

- Create message

- Delete Message

- Self-Message

- Reply Message

- Found Message

- Lost Message

iv. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.
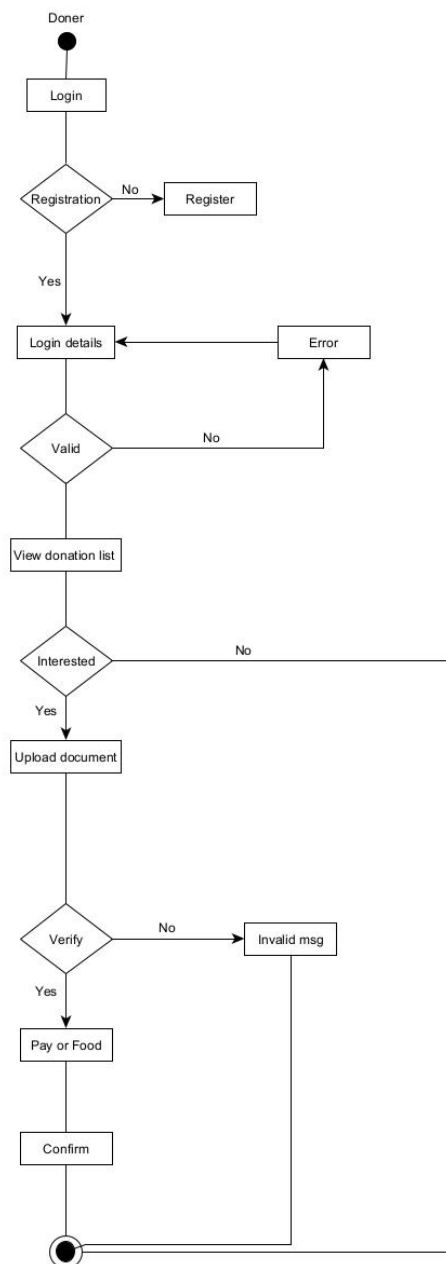
Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.
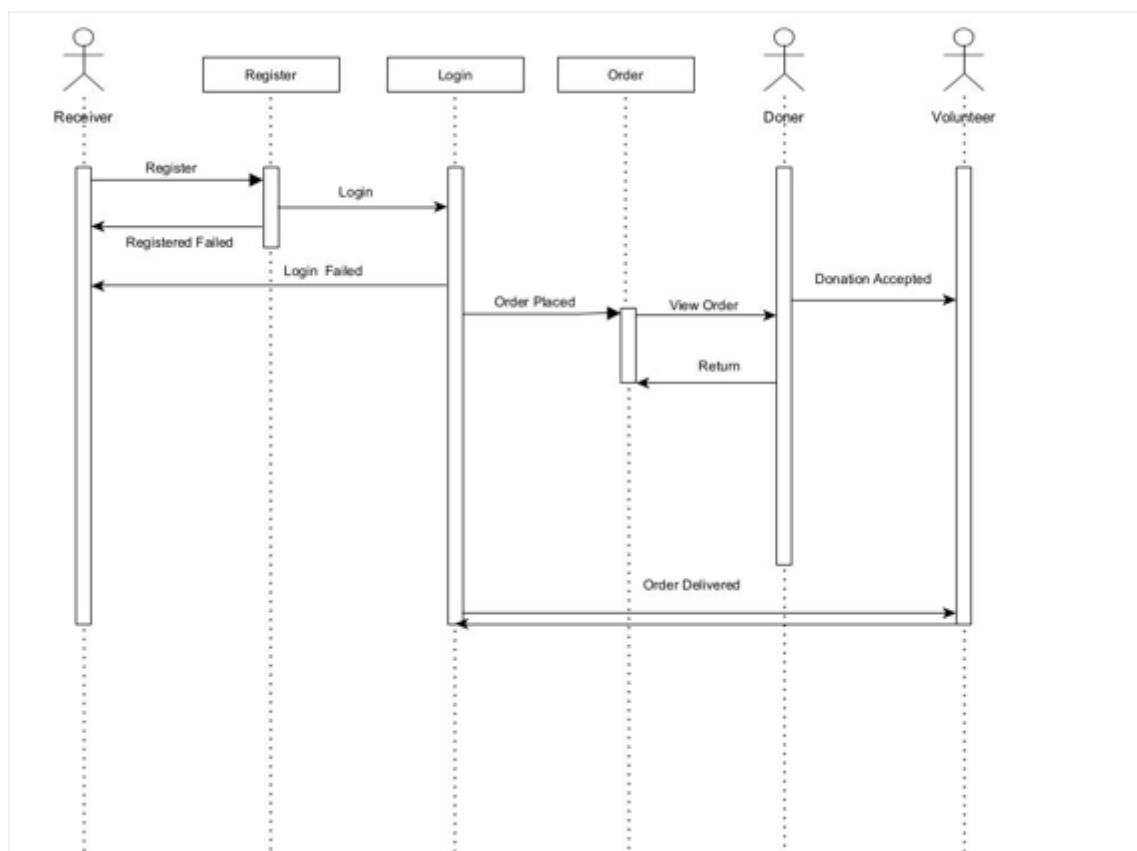
4.2.3 ACTIVITY DIAGRAM

An activity diagram shows the numerous decision routes that are available throughout the execution of an activity by depicting the control flow from a start point to an end point. Using an activity diagram, we can show both concurrent and sequential processing of activities. They are mostly used to represent the dynamic elements of a system in business and process modelling. A flowchart and an activity diagram are extremely similar.

Reciever

Login

Registration ──No──> Register

Yes

Login details <──── Error

Valid ──No────

View Services

Eligible ──────No──────

Yes

Apply Order

Verify ──No──> Invalid msg

Yes

Confirm

Delivery

Volunteer

Login

Registration — No → Register

Yes

Login details ← Error

Valid — No →

View Task

Interested — No →

Yes

Delivery
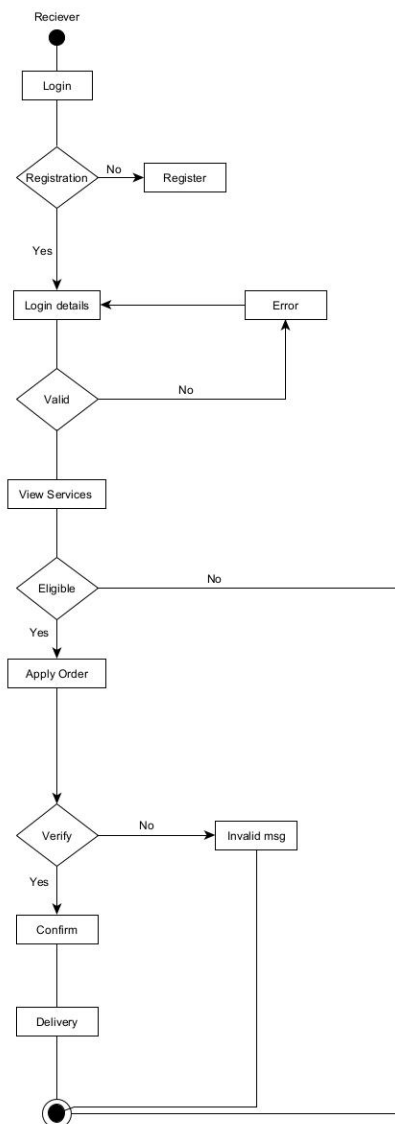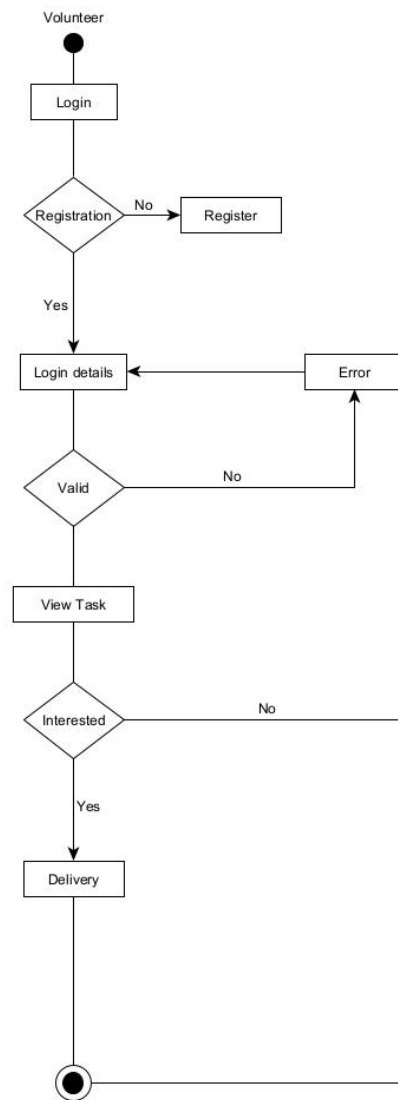
4.2.4 CLASS DIAGRAM

Class diagrams are a type of static diagram. It represents the static view of the application. Class diagrams are useful for visualising, describing, and documenting numerous system components as well as for writing executable code for 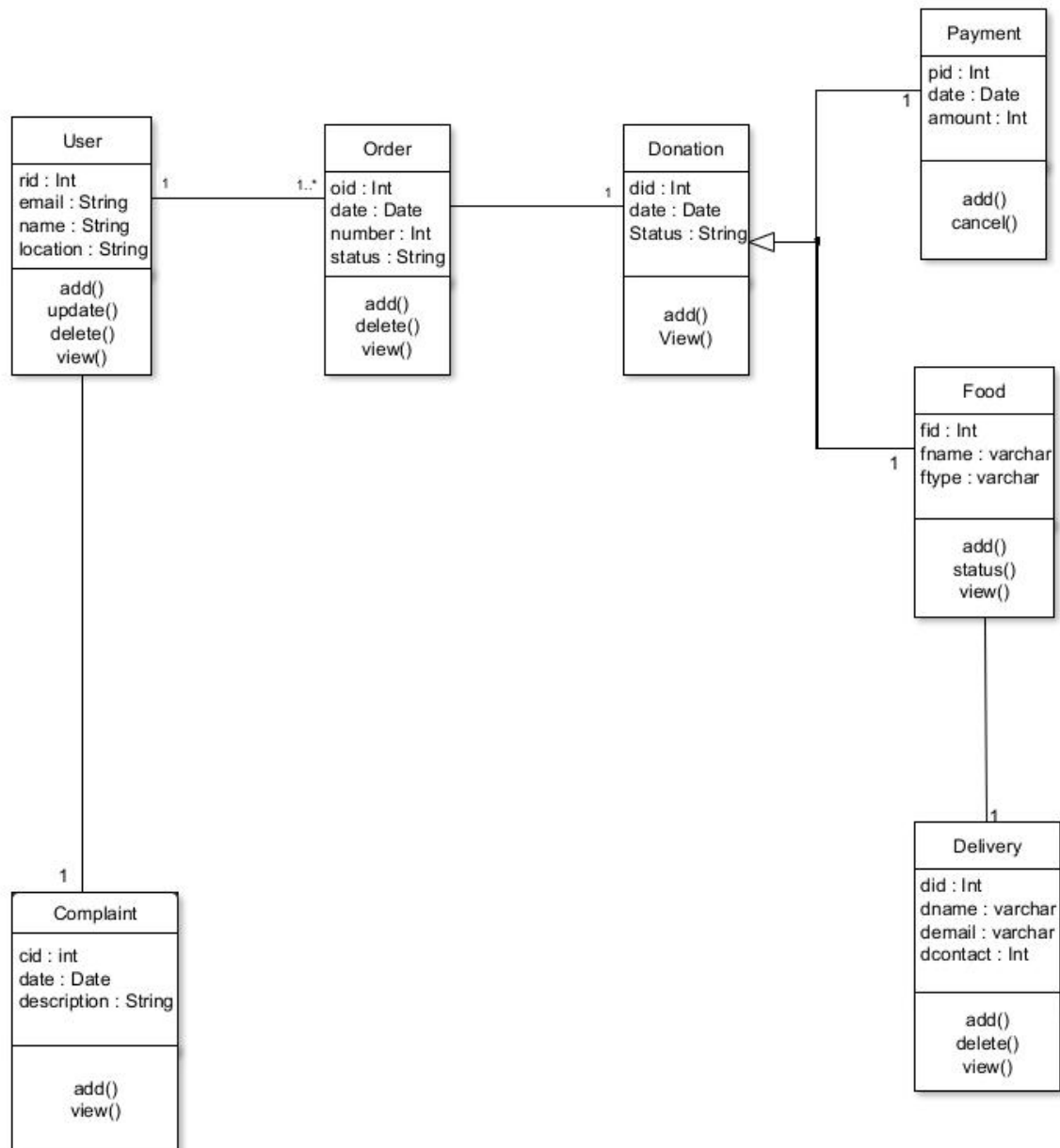software applications. A class diagram describes the constraints imposed on the system together with the properties and operations of a class. Class diagrams are frequently used in the modelling of object-oriented systems since they are the only UML diagrams that can be directly mapped with object-oriented languages. A class diagram shows a collection of classes, interfaces, affiliations, collaborations, and limits. It is also known as a structural diagram.

• This is the only UML that can appropriately depict various aspects of the OOPs concept.

• Proper design and analysis of applications can be faster and efficient.

• It is the base for deployment and component diagram.

• Each class is represented by a rectangle having a subdivision of three compartments name, attributes, and operation.

• There are three types of modifiers that are used to decide the visibility of attributes and operations.

• + is used for public visibility (for everyone)

• # is used for protected visibility (for friend and derived)

• – is used for private visibility (for only me)

User

rid : Int
email : String
name : String
location : String

---

add()
update()
delete()
view()

Order

oid : Int
date : Date
number : Int
status : String

---

add()
delete()
view()

Donation

did : Int
date : Date
Status : String

---

add()
View()

Payment

pid : Int
date : Date
amount : Int

---

add()
cancel()

Food

fid : Int
fname : varchar
ftype : varchar

---

add()
status()
view()

Complaint

cid : int
date : Date
description : String

---

add()
view()

Delivery

did : Int
dname : varchar
demail : varchar
dcontact : Int

---

add()
delete()
view()

4.2.5 COMPONENT DIAGRAM

Component diagrams are a particular kind of UML diagram. Additionally, the purpose is different from the previously given diagrams. It describes the components involved in producing that functionality, but it does not define the functioning of the system as a whole. Component diagrams are used to portray the real physical components of a system from that point of view.

These components include, among others, files, libraries, and packages. Component diagrams may also be thought of as a static implementation view of a system. The components' configuration is shown in a static implementation at a certain point in time. A single component diagram is impossible to depict the entire system, thus a set of diagrams are utilised instead..

The purpose of the component diagram can be summarized as –

☐ Visualize the components of a system.

☐ Construct executable by employing forward and reverse engineering.

☐ Describe how the elements are arranged and connected.
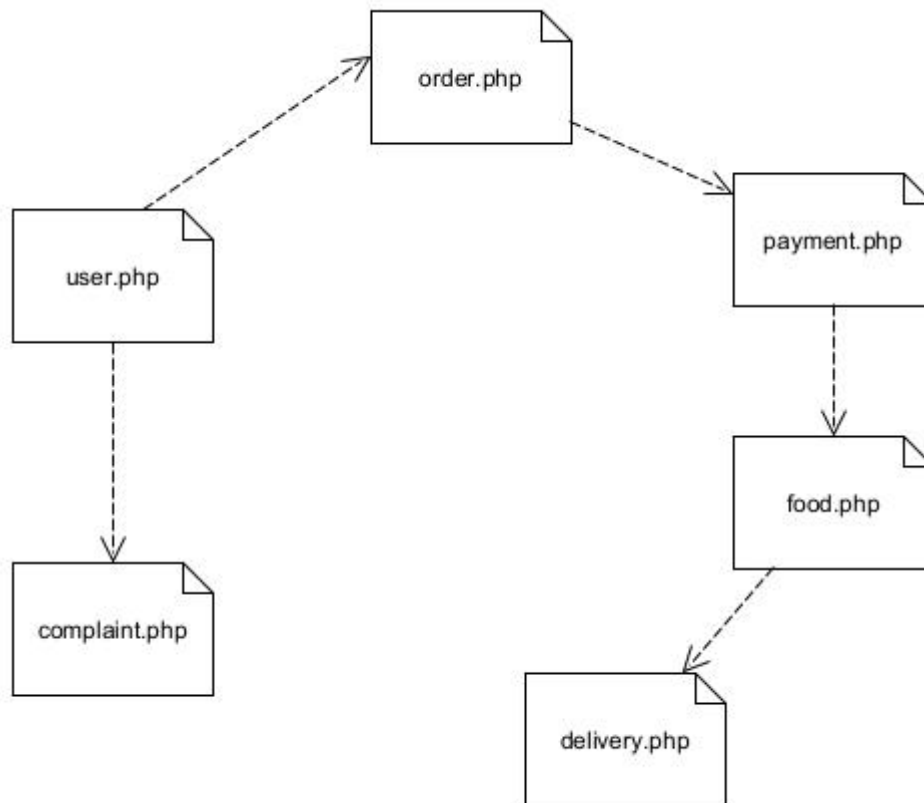
4.2.6 OBJECT DIAGRAM

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Object diagrams are used to render a set of objects and their relationships as an instance.

Purpose of Object Diagram:

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

The purpose of a diagram should be understood clearly to implement it practically. The purposes of object diagrams are similar to class diagrams. The difference is that a class diagram represents an abstract model consisting of classes and their relationships. However, an object diagram represents an instance at a particular moment, which is concrete in nature.

It means the object diagram is closer to the actual system behaviour. The purpose is to capture

the static view of a system at a particular moment.

The purpose of the object diagram can be summarized as –

• Forward and reverse engineering.

• Object relationships of a system

• Static view of an interaction.

• Understand object behavior and their relationship from practical perspective.
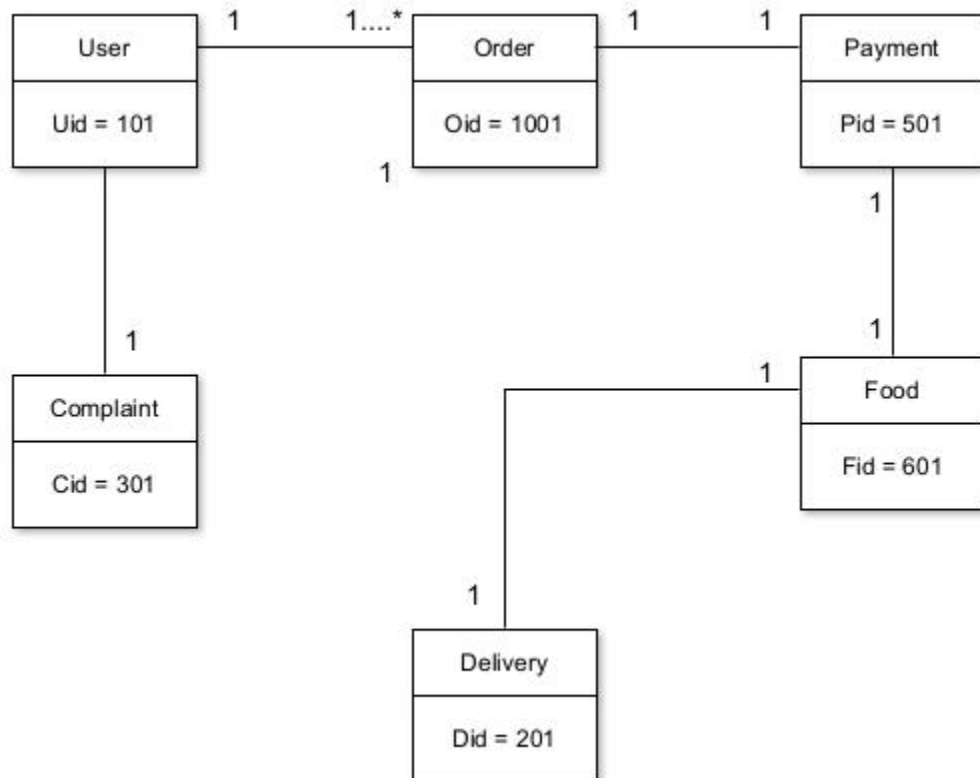
4.2.7 DEPLOYMENT DIAGRAMS

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships. Purpose of Deployment Diagrams The term Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components, where software components are deployed. Component diagrams and deployment diagrams are closely related. Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware**.**

UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components. Most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on the hardware topology of a system. Deployment diagrams are used by the system engineers.

The term Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components, where software components are deployed. Component diagrams and deployment diagrams are closely related. Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware. UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components.

Most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on the hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as –

• Visualize the hardware topology of a system.

• Describe the hardware components used to deploy software components.

• Describe the runtime processing nodes.
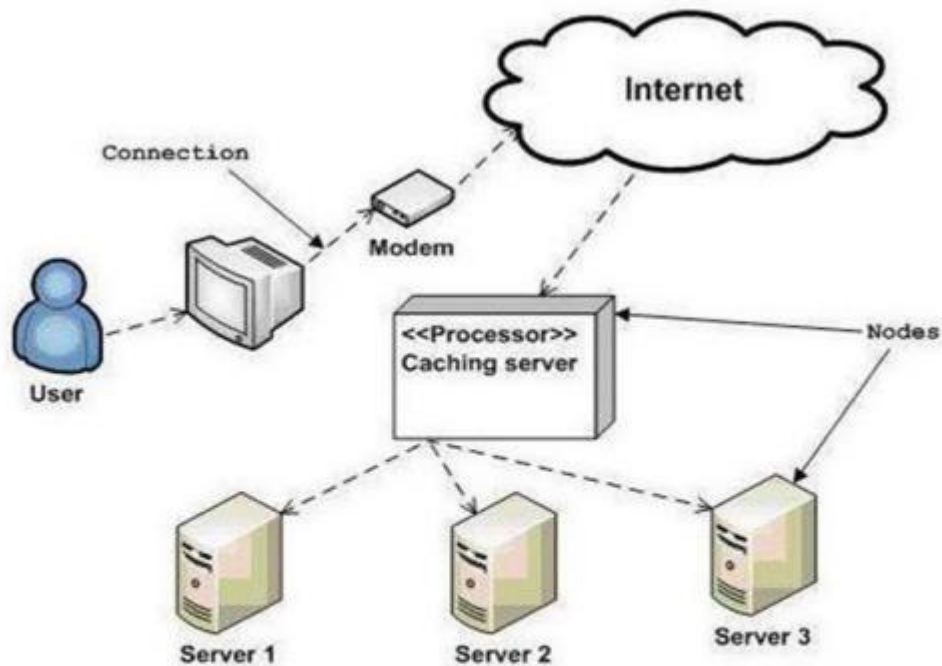
4.2.8 STATE DIAGRAM

It describes various system components' statuses. The states are unique to a particular system item or component. A state machine is described in a Statechart diagram. State machine can be described as a device that distinguishes between an object's various states, and these events either internal or external control states. They specify several object states over its lifespan, and events alter these states. Statecharts are helpful for simulating the responsive systems. A system that reacts to internal or external events is known as a reactive system. The transition from one state to another is depicted in a statechart. According to definitions, states are conditions in which objects exist and undergo changes.

The main goal of a statechart diagram is to model an object's lifespan from creation to destruction. For both forward and backward engineering of a system, statechart diagrams are employed. But modelling the reactive system is the fundamental objective. Following are the main purposes of using Statechart diagrams –

- To simulate a system's dynamic component.

- To simulate a reactive system's lifetime.

- To describe different states of an object during its life time.

- Create a state machine to represent an object's states

4.3 USER INTERFACE DESIGN

4.3.1-INPUT DESIGN

Form Name   : User Registration

Form Name   : User Login

Form Name   : Doner Home
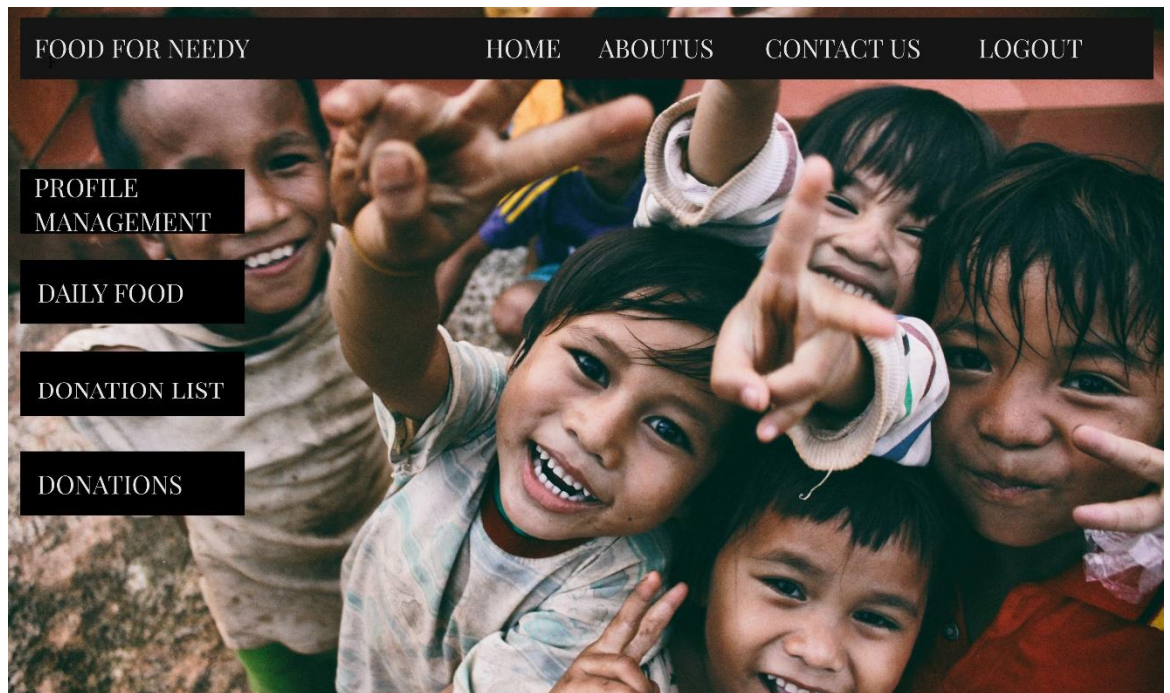
Form Name   : Home Page

4.3.2 OUTPUT DESIGN

User Login

User Registration

4.6 DATABASE DESIGN

A database is a structured system with the capacity to store information and allows users to access stored information quickly and effectively. Any database's primary goal is its data, which need protection.

There are two stages to the database design process. The user needs are obtained in the first phase, and a database is created to as clearly as possible satisfy these criteria. This process, known as information level design, is carried out independently of all DBMSs.

The design for the specific DBMS that will be used to construct the system in issue is converted from an information level design to a design in the second stage. Physical Level Design is the stage where the characteristics of the particular DBMS that will be utilised are discussed. Parallel to the system design is a database design. The database's data arrangement aims to accomplish the following two main goals.

• Data Integrity

• Data independence

4.4.1 Relational Database Management System (RDBMS)

The database is represented as a set of relationships in a relational paradigm. Each relation appears as a table of values or a file of records. A row is referred to as a tuple, a column heading is referred to as an attribute, and the table is referred to as a relation in the formal language of the relational model. In a relational database, the tables are divided into categories and given unique names. A row in a tale corresponds to a collection of linked values.

Relations, Domains & Attributes

A relation is a table. Tuples are the units of a table's rows. An ordered group of n items is a tuple. Attributes are referred to as columns. Every table in the database has relationships already established between them. This guarantees the integrity of both referential and entity relationships. A group of atomic values make up a domain D. Choosing a data type from which the domain's data values are derived is a typical way to define a domain. To make it easier to understand the values of the domain, it is also helpful to give it a name.

Every value in a relationship is atomic, meaning it cannot be broken down.

Relationships

• Key is used to create table connections. Primary Key and Foreign Key are the two principal keys that are most crucial. Referential integrity as well as entity integrity These are the basics of establishing relationships.

• Entity Integrity enforces that no Primary Key can have null values.

• No Primary Key may contain null values, according to Referential Integrity.

• Referential Integrity: A Primary Key value in the same domain must correspond to each unique Foreign Key value. Super Key and Candidate Keys are additional keys.

4.4.2 Normalization

The simplest possible grouping of data is used to put them together so that future modifications may be made with little influence on the data structures. The formal process of normalising data structures in a way that reduces duplication and fosters integrity. Using the normalisation approach, superfluous fields are removed and a huge table is divided into several smaller ones. Anomalies in insertion, deletion, and updating are also prevented by using it. Keys and relationships are two notions used in the standard style of data modelling. A row in a table is uniquely identified by a key. Primary keys and foreign keys are the two different kinds of keys. A primary key is an element, or set of components, in a database that serves as a means of distinguishing between records from the same table. A column in a database known as a foreign key is used to uniquely identify records from other tables. Up to the third normal form, all tables have been normalised.

It means placing things in their natural form, as the name suggests. By using normalisation, the application developer aims to establish a coherent arrangement of the data into appropriate tables and columns, where names may be quickly connected to the data by the user. By removing recurring groups from data, normalisation prevents data redundancy, which places a heavy demand on the computer's resources. These include:

    4.4.2.1   Normalize the data.

    4.4.2.2   Choose proper names for the tables and columns.

    4.4.2.3   Choose the proper name for the data.

First Normal Form

According to the First Normal Form, each attribute's domain must only include atomic values, and each attribute's value in a tuple must be a single value from that domain. In other words, 1NF forbids using relationships as attribute values within tuples or relations within relations. Single atomic or indivisible values are the only attribute values that are permitted under 1NF. The data must first be entered into First Normal Form. This may be accomplished by separating data into tables of a similar type in each table. Depending on the needs of the project, a Primary Key or Foreign Key is assigned to each table. For each nested relation or non-atomic property, additional relations are formed in this process. This got rid of data groupings that were repeated. If a relation solely meets the constraints that include the main key, it is said to be in first normal form.

Second Normal Form

No non-key attribute should be functionally dependent on a portion of the main key for relations when the primary key has several attributes, according to Second Normal Form. This involves breaking down each partial key into its dependent characteristics and setting up a new relation for each one. Keep the original primary key and any properties that are entirely dependent on it in your database. This procedure aids in removing data that depends only on a small portion of the key. If and only if a connection meets all the requirements for first normal form for the main key and every one of its non-primary key qualities completely depends on its primary key alone, then that relationship is said to be in second normal form.

Third Normal Form

Relation should not have a non-key attribute that is functionally determined by another non-key property or by a collection of non-key attributes, according to the Third Normal Form. The main key should not be transitively dependent, in other words. The non-key qualities that functionally determine other non-key attributes are decomposed in this way put up in relation. This procedure is used to eliminate everything not wholly dependent on the Primary Key. A relationship is only considered to be in third normal form if it is in second normal form, and it should also not depend on any other relationships' non-key properties.

TABLE DESIGN

Table No   : 01
Table Name   : registration
Primary Key   : rid
Table Description   : To store user Registration information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	rid	Int	Primary Key	Registration Id
2	fname	Varchar	Not Null	First Name
3	lname	Varchar	Not Null	Last Name
4	email	Varchar	Not Null	Email
5	phoneno	Varchar	Not Null	Phone Number

Table No   : 02
Table Name   : login
Primary Key   : lid
Foreign Key   : rid
Table Description   : To store user login information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	lid	Int	Primary Key	Login Id
2	rid	Int	Foreign Key	Registration Id
3	email	Varchar	Not Null	User Name
4	password	Varchar	Not Null	Password
5	Role	Varchar	Not Null	Role

Table No   : 03
Table Name   : receiver
Primary Key   : reid
Foreign Key   : lid
Table Description   : To store receiver information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	reid	Int	Primary Key	Receiver Id
2	lid	Int	Foreign Key	Login Id
3	fname	Varchar	Not Null	First Name

4	lname	Varchar	Not Null	Second Name
5	rtype	Varchar	Not Null	Reciever Type
6	houseno	Varchar	Not Null	House Name
7	street	Varchar	Not Null	Street
8	pincode	Int	Not Null	Pincode
9	district	Varchar	Not Null	District
10	id proof no	Varchar	Not Null	Id Proof Number
11	id proof image	Varchar	Not Null	Id Proof Image
12	phneno	Varchar	Not Null	Phone Number

Table No   : 04
Table Name   : doner
Primary Key   : did
Foreign Key   : lid
Table Description : To store doner information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	did	Int	Primary Key	doner Id
2	lid	Int	Foreign Key	Login Id
3	dfname	Varchar	Not Null	First Name
4	dlname	Varchar	Not Null	Second Name
5	dtype	Varchar	Not Null	Doner Type
6	houseno	Varchar	Not Null	House Name
7	Street	Varchar	Not Null	Street
8	pincode	Int	Not Null	Pincode
9	district	Varchar	Not Null	District
10	Email	Varchar	Not Null	Email
11	phone no	Varchar	Not Null	Phone Number

Table No   : 05
Table Name   : volunteer
Primary Key   : vid
Foreign Key   : lid
Table Description   : To store volunteer information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	vid	Int	Primary Key	Volunteer Id
2	lid	Int	Foreign Key	Login Id
3	fname	Varchar	Not Null	First Name
4	lname	Varchar	Not Null	Second Name
5	idnumber	Varchar	Not Null	Id Proof No
6	idimage	Varchar	Not Null	Id Proof Image
7	Area	Varchar	Not Null	Area
8	location	Int	Not Null	Location
9	pretiming	Varchar	Not Null	Pretiming
10	Email	Varchar	Not Null	Email
11	Phone no	Varchar	Not Null	Phone Number

Table No   : 06
Table Name   : order
Primary Key   : oid
Foreign Key   : reid
Table Description   : To store order information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	oid	Int	Primary Key	Order Id
2	reid	Int	Foreign Key	Receiver Id
3	o_des	Varchar	Not Null	Order description
4	o_quan	Varchar	Not Null	Order Quantity
5	o_type	Varchar	Not Null	Order Type
6	o_date	Varchar	Not Null	Order Date
7	o_time	Varchar	Not Null	Order Time
8	status	Int	Not Null	Status

Table No   : 07
Table Name   : payment
Primary Key   : pid
Foreign Key   : did
Table Description : To store payment information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	pid	Int	Primary Key	Payment Id
2	did	Int	Foreign Key	Doner Id
3	oid	Int	Foreign Key	Order Id
4	amtpaid	Varchar	Not Null	Amount Paid
5	pay_type	Varchar	Not Null	Payment Type
6	pay_date	Date	Not Null	Payment Date

Table No   : 08
Table Name   : delivery
Primary Key   : delid
Foreign Key   : reid,vid
Table Description : To store user doner information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	delid	Int	Primary Key	Delivery Id
2	reid	Int	Foreign Key	Receiver Id
3	vid	Int	Foreign Key	Volunteer Id
4	status	Varchar	Not Null	Status

Table No   : 09
Table Name   : complaints
Primary Key   : cid
Foreign Key   : reid
Table Description : To store user doner information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	cid	Int	Primary Key	Delivery Id
2	vid	Int	Foreign Key	Receiver Id
3	fname	Varchar	Not Null	complainee

| 4 | com | Varchar | Not Null | Complaints |

Table No   : 10
Table Name   : food
Primary Key   : fid
Foreign Key   : did
Table Description   : To store user food information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	fid	Int	Primary Key	Order Id
2	did	Int	Foreign Key	Receiver Id
3	dname	Varchar	Not Null	Order description
4	dtype	Varchar	Not Null	Order Quantity
5	fname	Varchar	Not Null	Order Type
6	ftype	Varchar	Not Null	Order Date
7	fquan	Int	Not Null	Order Time

Table No   : 11
Table Name   : assign
Primary Key   : aid
Foreign Key   : vid,doid
Table Description   : To store volunteer information

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	aid	Int	Primary Key	Assign Id
2	vid	Int	Foreign Key	Volunteer Id
3	doid	Int	Foreign Key	Donation Id
4	status	Varchar	Not Null	Status

Table No   : 12
Table Name   : mealsplan
Primary Key   : mid
Table Description   : To store mealplans

SLNO	FIED NAME	TYPE	CONSTRAINS	DESCRIPTION
1	mid	Int	Primary Key	Assign Id
2	meals	Int	Foreign Key	Volunteer Id

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is the practise of carefully controlling the execution of software in order to determine if it behaves as intended. The words verification and validation are frequently used in conjunction with software testing. Validation is the process of examining or evaluating a product, including software, to determine if it complies with all relevant specifications. One type of verification, software testing, employs methods including reviews, analyses, inspections, and walkthroughs as well. Verifying that what has been specified matches what the user truly want is the process of validation.

The processes of static analysis and dynamic analysis are additional ones that are frequently related to software testing. Static analysis examines the software's source code, searching for issues and obtaining statistics without actually running the code. Dynamic analysis examines how software behaves while it is running in order to offer data like execution traces, timing profiles, and test coverage details.

Testing is a collection of activities that may be planned ahead of time and carried out in a methodical manner. Testing starts with individual modules and progresses to the integration of the full computer-based system. Nothing is complete without testing, as it is essential to the system's testing goals. Several guidelines can be used as testing goals. They are:

Executing a programme during testing is a procedure used to look for errors.

- A excellent test case is one that has a strong chance of revealing a mistake that hasn't been found yet.
- A test that finds a mistake that hasn't been noticed is successful.

If a test is successfully carried out in accordance with the aforementioned aims, it will reveal software bugs. Additionally, testing shows that the software functions seem to be functioning in accordance with the specifications and that the performance requirements seem to have been satisfied.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Testing for correctness is used to ensure that a programme performs precisely as it was intended to. This is considerably harder than it would initially seem, especially for big projects.

5.2 TEST PLAN

A test plan suggests a number of required steps that need be taken in order to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer programme, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfils the purpose for which it was intended. In order to solve the inherent issues with allowing the builder evaluate what they have developed, there is an independent test group (ITG). Testing's precise goals should be laid forth in quantifiable language. Therefore, the test plan should include information on the mean time to failure, the cost to locate and correct the flaws, the residual defect density or frequency of occurrence, and the number of test labour hours required for each regression test.

The levels of testing include:

- ❖ Unit testing
- ❖ Integration Testing
- ❖ Data validation Testing
- ❖ Output Testing

5.2.1 Unit Testing

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a reference for testing crucial control pathways to find faults inside the module's perimeter. the level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test appropriately, the modular interface is checked. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. To confirm that each statement in a module has been performed at least once, boundary conditions are evaluated. All error handling pathways are then evaluated.

Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling pathways must be put up to cleanly redirect or halt work when an error does arise. The final phase of unit testing is boundary testing. Software frequently experiences boundary issues

In the Sell-Soft System, unit testing was carried out by considering each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had certain issues, which were fixed. Each module is tested and run separately after development. To guarantee that every module functions properly and produces the desired outcome, all extraneous code was deleted.

5.2.2 Integration Testing

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a programme structure that has been determined by design using unit tested components. The software as a whole is tested. Correction is challenging since the size of the overall programme makes it hard to isolate the reasons. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All of the modules were merged once unit testing was completed in the system to check for any interface inconsistencies. Additionally, variations in programme architectures were eliminated, and a singular programme structure emerged.

5.2.3 Validation Testing or System Testing

The testing process comes to an end here. This involved testing the complete system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include "Black Box" testing and "System tests."

The functional requirements of the programme are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer may create sets of input circumstances that will thoroughly test every programme requirement.

Problems in data structures or external data access, erroneous or missing functions, interface faults, performance issues, initialization issues, and termination issues are all types of errors that black box testing looks for.

5.2.4 Output Testing or User Acceptance Testing

User approval of the system under consideration is tested; in this case, it must meet the needs of the company. When development, the programme should stay in touch with the user and perspective system to make any necessary modifications. This done with respect to the following points:

☐ Input Screen Designs,

☐ Output Screen Designs,

The aforementioned testing is carried out using a variety of test data. The preparation of test data is essential to the system testing process. The system under investigation is then put to the test using the prepared test data. When testing the system, test data issues are found again and fixed using the testing procedures described above. The fixes are also logged for use in the future.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The project's implementation phase is where the conceptual design is transformed into a functional system. It can be regarded as the most important stage in creating a successful new system since it gives users assurance that the system will operate as intended and be reliable and accurate. User documentation and training are its main concerns. Usually, conversion happens either during or after the user's training. Implementation is the process of turning a newly updated system design into an operational one, and it simply refers to placing a new system design into operation.

The user department now bears the most of the workload, faces the most disruption, and has the biggest influence on the current system. The implementation might lead to confusion and turmoil if it is not adequately planned or managed.

Implementation encompasses all of the steps used to switch from the old system to the new one. The new system might be entirely different, take the place of an existing manual or automated system, or it could be modified to work better. A dependable system that satisfies organisational needs must be implemented properly. System implementation refers to the process of actually using the built system. This comprises all the processes involved in switching from the old to the new system. Only after extensive testing and if it is determined that the system is operating in accordance with the standards can it be put into use. The system employees assess the system's viability. The system analysis and design work needed to implement the three key components of education and training, system testing, and changeover will increase in complexity as a system is implemented.

The implementation state involves the following tasks:

- ☐ Careful planning.
- ☐ Investigation of system and constraints.
- ☐ Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

Software implementation refers to the complete installation of the package in its intended environment, as well as to the system's functionality and fulfilment of its intended applications. The software development project is frequently commissioned by someone who will not be using it. People have early reservations about the programme, but it's important to watch out for the following to prevent resistance from growing:

- The active user has to understand the advantages of utilising the new system.

- Their faith in the software is increased.

- The user is given the appropriate instruction so that he feels comfortable using the programme.

Before examining the system, the user must be aware that the server software has to be running on the server in order to access the results. The real procedure won't happen if the server object is not active and functioning on the server.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer based system, it is essential for the people who will be involved to be confident of theirrole in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

6.2.2 Training on the Application Software

The user will need to receive the essential basic training on computer awareness after which the new application software will need to be taught to them. This will explain the fundamental principles of how to use the new system, including how the screens work, what kind of help is displayed on them, what kinds of errors are made while entering data, how each entry is validated, and how to change the date that was entered. Then, while imparting the program's training on the application, it should cover the knowledge required by the particular user or group to operate the system or a certain component of the system. It's possible that this training will vary depending on the user group and the degree of hierarchy.

6.2.3 System Maintenance

The mystery of system development is maintenance. When a software product is in the maintenance stage of its lifecycle, it is actively working. A system should be properly maintained after it has been effectively installed. An essential part of the software development life cycle is system maintenance. In order for a system to be flexible to changes in the system environment, maintenance is required. Of course, software maintenance involves much more than just "Finding Mistakes".

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The current system working technology is old fashioned and there is no usage of commonly used technologies like internet, digital money. The proposed system introduces facility for customer to book service online and view all information.

7.2 FUTURE SCOPE

- Users can able to do advanced search options

- Users can able to add feedbacks .

- Data security can be enhanced.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, "*System Analysis and Design*", 2009.

- Roger S Pressman, "*Software Engineering*", 1994.

- PankajJalote, "S*oftware engineering*: a precise approach", 2006.

- James lee and Brent ware Addison, "Open source web development with LAMP", 2003

- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- www.w3schools.com

- www.jquery.com

- http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf

- www.agilemodeling.com/artifacts/useCaseDiagram.html

CHAPTER 9

APPENDIX

9.1 Sample Code

Login.html

```html
<!doctype html>
<html lang="en">
 <head>
   <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
   <link href="https://fonts.googleapis.com/css?family=Roboto:300,400&display=swap" rel="stylesheet">

   <link rel="stylesheet" href="fonts/icomoon/style.css">

   <link rel="stylesheet" href="css/owl.carousel.min.css">

   <!-- Bootstrap CSS -->
   <link rel="stylesheet" href="css/bootstrap.min.css">

   <!-- Style -->
   <link rel="stylesheet" href="css/style.css">

   <title>Login #2</title>
 </head>
 <body>


 <div class="d-lg-flex half">
   <div class="bg order-1 order-md-2" style="background-image: url('images/f1.jpg');"></div>
   <div class="contents order-2 order-md-1">

     <div class="container">
       <div class="row align-items-center justify-content-center">
         <div class="col-md-7">
           <h3> <strong><center>Food For Needy</center></strong></h3>
           <p>
                                            Sign In now, Let's start your Grocery Shopping.
Don't have an account?

                                            <a href="../thecharity/selection.php">
                                                    Sign Up Now</a>
                                    </p>
           <p class="mb-4"></p>
           <form action="login1.php" method="post">
             <div class="form-group first">
               <label for="username">Username</label>
               <input type="text" name= "mail" class="form-control" placeholder="your-email@gmail.com" id="email" required onchange="return Validata();">
```

```
            </div>
              <span id="msg3" style="color:red;"></span>

              <script>
              function Validata()
{
   var val = document.getElementById('email').value;

   if (!val.match(/([A-z0-9_\-\.]){1,}\@([A-z0-9_\-\.]){1,}\.([A-Za-z]){2,4}$/))
   {
      document.getElementById('msg3').innerHTML="Enter a Valid Email";

              document.getElementById('email').value = "";
      return false;
   }
document.getElementById('msg3').innerHTML=" ";
   return true;
}
              </script>

         <div class="form-group last mb-3">
          <label for="password">Password</label>
          <input type="password"  name = "pass" class="form-control" placeholder="Your
Password" id="pwd" required onchange="return Validp();">
         </div>
          <span id="msg10" style="color:red;"></span>
                   <script>
function Validp()
{
   var val = document.getElementById('pwd').value;

   if (!val.match(/^[A-Za-z0-9!-*]{6,15}$/))
   {
      document.getElementById('msg10').innerHTML="Password should contain atleast 6
characters";

              document.getElementById('pwd').value = "";
      return false;
   }
document.getElementById('msg10').innerHTML=" ";
   return true;
}

</script>
         <!-- <div class="d-flex mb-5 align-items-center">
          <label class="control control--checkbox mb-0"><span class="caption">Remember
me</span>
             <input type="checkbox" checked="checked"/>
             <div class="control__indicator"></div>
             </label>
             <span class="ml-auto"><a href="#" class="forgot-pass">Forgot
```

Password
               </div>-->

```html
            <input type="submit" value="Login" class="btn btn-block btn-primary">

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


 </div>


   <script src="js/jquery-3.3.1.min.js"></script>
   <script src="js/popper.min.js"></script>
   <script src="js/bootstrap.min.js"></script>
   <script src="js/main.js"></script>
 </body>
</html>
```

Login.php

```php
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
</head>

<body>
  <?php
  include('pconnection.php');

  $u=$_POST['mail'];
  $p=$_POST['pass'];

  $a=mysqli_query($conn,"SELECT * FROM login where email='$u' and password ='$p'");
  $row=mysqli_fetch_array($a);
   $status= $row['status'];
   if(!empty($row))
   {
          if($row['role']==1)
          {
                  session_start();
                  $_SESSION['mail']=$row['email'];
                  $_SESSION['pass']=$row['password'];

        if( $status == 1)
         {
          header("Location:../thecharity/doner_home.php");
```

```
                }
             elseif( $status == 0)
             {
              header("Location:login.php");
             }

                }
              else if($row['role']==2)
              {
                        session_start();
                        $_SESSION['mail']=$row['email'];
                        $_SESSION['pass']=$row['password'];

          if( $status == 1)
           {
            header("Location:../thecharity/reciever_home.php");
           }
          elseif( $status == 0)
          {
           header("Location:login1.php");
          }

                }
       else if($row['role']==3)
            {
                        session_start();
                        $_SESSION['mail']=$row['email'];
                        $_SESSION['pass']=$row['password'];

          if( $status == 1)
           {
            header("Location:../thecharity/vol_home.php");
           }
          elseif( $status == 0)
          {
           header("Location:login1.php");
          }

              }

              else
              {
                        if($row['role']==0)
              {
                        session_start();
            header("Location:../web/index.html");
           }


                }
              }

      else
              echo "<script>
  alert('Login failed');
```

```
        window.location.href='index.html';
        </script>";



        ?>
        </body>
        </html>
```

Recieverreg.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>Sign Up Form by Colorlib</title>

   <!-- Font Icon -->
   <link rel="stylesheet" href="fonts/material-icon/css/material-design-iconic-font.min.css">

   <!-- Main css -->
   <link rel="stylesheet" href="css/style.css">
</head>
<body>

   <div class="main">
     <div class="container">
       <div class="signup-content">
         <div class="signup-img">
           <img src="images/v1.jpg"  height="1500" alt="">
         </div>
         <div class="signup-form">

           <?php if (isset($_SESSION['success_message']) && !empty($_SESSION['success_message']))
{ ?>
             <div class="success-message" style="margin-bottom: 20px;font-size: 20px;color:
green;"><?php echo $_SESSION['success_message']; ?></div>
             <?php
             unset($_SESSION['success_message']);
           }
           ?>
           <form method="POST" class="register-form" id="register-form" action="regirec.php">
             <h2>Receiver Registration</h2>

             <div class="form-row">
               <div class="form-group">
                 <label for="name"> Organization Name :</label>
                 <input type="text" name="firstname" id="name"  placeholder=" [Eg:Karuniya]" required
onchange="Validate();" required/>
               </div>
```

```
                    <span id="msg1" style="color:red;"></span>
<script>

function Validate()
{
   var val = document.getElementById('name').value;

   if (!val.match(/^[A-Z][A-Za-z]{3,}$/))
   {
      document.getElementById('msg1').innerHTML="Start with a Capital letter & Only alphabets without
space are allowed!!";
                    document.getElementById('name').value = "";
      return false;
   }
document.getElementById('msg1').innerHTML=" ";
   return true;
}
                    </script>

                    <div class="form-group">
                       <label for="father_name">Category :</label>
                       <div class="form-select">
                          <select name="lastname" id="state">
                             <option value=""></option>
                             <option value="Orphanage">Orphanage</option>
                             <option value="Old Age Home">Old Age Home</option>


                          </select>

                       <span class="select-icon"><i class="zmdi zmdi-chevron-down"></i></span>

                       </div>
                    </div>
                     <span id="msg2" style="color:red;"></span>
                    <script>
                       function Validate1()
{
   var val = document.getElementById('lname').value;

   if (!val.match(/^[A-Z][a-z]{0,}$/))
   {
      document.getElementById('msg2').innerHTML="Start with a Capital letter & Only alphabets without
space are allowed!!";
                          document.getElementById('lname').value = "";
      return false;
   }
document.getElementById('msg2').innerHTML=" ";
   return true;
}
</script>

                    </div>
```

```
            <div class="form-group">
            <label for="email">Email ID :</label>
            <input type="email" name="email" id="email" placeholder="E-mail [Eg:
xyz@gmail.com]" required onchange="return Validata();" />
            </div>

            <span id="msg3" style="color:red;"></span>
            <script>
function Validata()
{
   var val = document.getElementById('email').value;

   if (!val.match(/([A-z0-9_\-\.]){1,}\@([A-z0-9_\-\.]){1,}\.([A-Za-z]){2,4}$/))
   {
      document.getElementById('msg3').innerHTML="Enter a Valid Email";

                    document.getElementById('email').value = "";
      return false;
   }
document.getElementById('msg3').innerHTML=" ";
   return true;
}

            </script>

             <div class="form-group">
             <label for="address">Phone Number :</label>
             <input type="number" name="phne" id="phone" placeholder="Phone Number" required
onchange="Validat();"required/>
            </div>
             <span id="msg4" style="color:red;"></span>
              <script>
function Validat()
{
   var val = document.getElementById('phone').value;

   if (!val.match(/^[7-9][0-9]{1,9}$/))
   {
      document.getElementById('msg4').innerHTML="Only Numbers are allowed and must contain 10
number";

                       document.getElementById('phone').value = "";
      return false;
   }
document.getElementById('msg4').innerHTML=" ";
   return true;
}

</script>

            <div class="form-group">
              <label for="address">Address :</label>
              <input type="text" name="addre" id="add"
                 placeholder="Address" required onchange="Validadd();" required/>
```

```
            </div>
            <span id="msg5" style="color:red;"></span>
              <script>
function Validadd()
{
   var val = document.getElementById('add').value;

   if (!val.match(/^[A-Z][a-z" "]{3,}$/))
   {
      document.getElementById('msg5').innerHTML="Start with a Capital letter & Only alphabets are
allowed";
                        document.getElementById('add').value = "";
      return false;
   }
document.getElementById('msg5').innerHTML=" ";
   return true;
}

            </script>
            <div class="form-row">
              <div class="form-group">
                <label for="state">State :</label>
                <div class="form-select">
                 <select name="state" id="state">
                     <option value=""></option>
                      <option value="kerala">Kerala</option>
                     </select>


                 <span class="select-icon"><i class="zmdi zmdi-chevron-down"></i></span>
                 </div>
             </div>
             <div class="form-group">
                <label for="city">District :</label>
                <div class="form-select">
                   <select name="dis" id="state">
                     <option value=""></option>
                     <option value="Thiruvananthapuram">Thiruvananthapuram</option>
                       <option value="Kollam">Kollam</option>
                      <option value="Pathanamthitta">Pathanamthitta</option>
                      <option value="Alappuzha">Alappuzha</option>
                      <option value="Kottayam">Kottayam</option>
                      <option value="Idukki">Idukki</option>
                      <option value="Ernakulam">Ernakulam</option>
                      <option value="Thrissur ">Thrissur </option>
                      <option value="Palakkad">Palakkad</option>
                      <option value="Malappuram">Malappuram</option>
                      <option value="Kozhikode">Kozhikode</option>
                      <option value="Wayanad">Wayanad</option>
                      <option value="Kannur">Kannur</option>
                      <option value="Kasaragod">Kasaragod</option>

                   </select>

                 <span class="select-icon"><i class="zmdi zmdi-chevron-down"></i></span>
```

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

        <div class="form-group">
          <label for="pincode">Pincode :</label>
          <input type="number" name="pincode" id="pin" placeholder="Pincode" required
onchange="Validpin();" required/>
        </div>
        <span id="msg9" style="color:red;"></span>
        <script>
function Validpin()
{
   var val = document.getElementById('pin').value;

   if (!val.match(/^[1-9][0-9]{5}$/))
   {
      document.getElementById('msg9').innerHTML="Starting digit in the pin code ranging from 1 to 9,next
5 digits in the pin code ranging from 0 to 9,Exactly contains 6 Digits";
                     document.getElementById('pin').value = "";
      return false;
   }
document.getElementById('msg9').innerHTML=" ";
   return true;
}
</script>
        <div class="form-group">
          <label for="course">Id Proof :</label>
            <div class="form-select">
              <select name="idp" id="state" required>
                <option value=""></option>
                <option value="Society Liensce">Society liensce</option>
                <option value="Organization Liensce"> Organization Liensce</option>
              </select>

            <span class="select-icon"><i class="zmdi zmdi-chevron-down"></i></span>

        </div>
        <div class="form-group">
          <label for="course">Id Number :</label>

            <input type="text" name="idnum" id="pincode" required/>


        </div>

        <div class="form-group">
          <label for="course">Supporting Document :</label>

        <input type="file" name="idimg" id="ss5" required/>


        </div>
          <div class="form-group">
          <label for="email">Password :</label>
```

```
                        <input type="password" name="password"id="pwd" required onchange="return Validp();"
/>
              </div>
                 <span id="msg10" style="color:red;"></span>
                 <script>
function Validp()
{
   var val = document.getElementById('pwd').value;

   if (!val.match(/^[A-Za-z0-9!-*]{6,15}$/))
   {
      document.getElementById('msg10').innerHTML="Password should contain atleast 6 characters";

                     document.getElementById('pwd').value = "";
      return false;
   }
document.getElementById('msg10').innerHTML=" ";
   return true;
}

</script>


                 <div class="form-group">
                 <label for="email">Confirm Password :</label>
                 <input type="password" name="confirm" id="confirm"  required onchange="return
check();"/>
              </div>
                 <span id="msg11" style="color:red;"></span>
                                     <script>
        function check()
{
var pas1=document.getElementById("pwd");
                                             var pas2=document.getElementById("confirm");

                                             if(pas1.value=="")
      {
              document.getElementById('msg11').innerHTML="Password can't be null!!";
              pas1.focus();
              return false;
      }
      if(pas2.value=="")
      {
              document.getElementById('msg11').innerHTML="Please confirm password!!";
              pass2.focus();
              return false;
      }
      if(pas1.value!=pas2.value)
      {
              document.getElementById('msg11').innerHTML="Passwords does not match!!";
              pas1.focus();
              return false;
      }
   document.getElementById('msg11').innerHTML=" ";
        return true;
```

```
}
        </script>


                <div class="form-submit">
                    </div>
                    <center><input type="submit" value="Register" class="submit" name="submit"
id="submit" />
                    </center>

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

</div>

  <!-- JS -->
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/main.js"></script>
</body><!-- This templates was made by Colorlib (https://colorlib.com) -->
</html>
```

Recreg.php

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
</head>

<body>
        <?php
        include('pconnection.php');
   $sql="";
        $u=$_POST['firstname'];
        $p=$_POST['lastname'];
   $e=$_POST['email'];
   $ph=$_POST['phne'];
   $g=$_POST['password'];
   $ad=$_POST['addre'];
   $st=$_POST['state'];
   $di=$_POST['dis'];
   $pi=$_POST['pincode'];
   $idr=$_POST['idp'];
   $idn=$_POST['idnum'];
    $idi=$_POST['idimg'];
   $folder='../Profile';
$path= $folder.$idi;
   $sqlu="insert into registration(fname,lname,email,phnenum,pswd) values ('$u','$p','$e',$ph,'$g')";
   $a="SELECT * FROM registration";
   $sql=mysqli_query($conn,$sqlu);
```

```php
$x=mysqli_insert_id($conn);
$sql1="insert into login(rid,email,password,role,status) values ('$x','$e','$g',2,0)";
mysqli_query($conn,$sql1);
$sql2="insert into reciever(rid,rname,rtype,email,addr, state,dist,pincode,idproof,idno,idimg)values
('$x','$u','$p','$e','$ad','$st','$di',$pi,'$idr','$idn','$idi')";
mysqli_query($conn,$sql2);
if($sql2==TRUE)
{
 $_SESSION['success_message'] = "Registered successfully.";
 header("location:../login/index.html");
     exit();
}
else
{
echo "Registration Falied";
}

?>
</body>
</html>
```

Order.php

```html
<html>
  <head>
    <style>

        /* Style inputs, select elements and textareas */
    input[type=text], select, textarea{
      width: 100%;
      padding: 12px;
      border: 1px solid #ccc;
      border-radius: 4px;
      box-sizing: border-box;
      resize: vertical;
    }

    /* Style the label to display next to the inputs */
    label {
      padding: 12px 12px 12px 0;
      display: inline-block;
    }

    /* Style the submit button */
    input[type=submit] {
      background-color: #ff4800;

      color: white;
      padding: 12px 20px;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      text-align: center;
    margin: auto;
```

```
        }

    /* Style the container */
    .container {
        align-content: center
      border-radius: 5px;
      background-color:none;
      padding: 20px;
    }

    /* Floating column for labels: 25% width */
    .col-25 {
      float: left;
      width: 25%;
      margin-top: 6px;
    }

    /* Floating column for inputs: 75% width */
    .col-75 {
      float: left;
      width: 75%;
      margin-top: 6px;
    }

    /* Clear floats after the columns */
    .row:after {
      content: "";
      display: table;
      clear: both;
    }

    /* Responsive layout - when the screen is less than 600px wide, make the two columns stack on top of
    each other instead of next to each other */
    @media screen and (max-width: 600px) {
      .col-25, .col-75, {
        width: 100%;
        margin-top: 0;
        align-content: center;
      }
    }
        </style>
      </head>
      <body>
        <?php
          include('header.php');
        include('../pconnection.php');

    $b=$_SESSION['mail'];

        ?>
        <div class="container">
          <?php if (isset($_SESSION['success_message']) && !empty($_SESSION['success_message']))
    { ?>
                  <div class="success-message" style="margin-bottom: 20px;font-size: 20px;color:
    green;"><?php echo $_SESSION['success_message']; ?></div>
```

```php
                    <?php
                    unset($_SESSION['success_message']);
                }
                ?>
```

```html
<form action="order_action.php" method="post">
    <div class="row">
    <div class="col-25">
     <label for="meals">Meal Plan</label>
    </div>
    <div class="col-75">

     <select id="meals" name="meal" placeholder="select">
        <option value=0>Select</option>
```
```php
     <?php
       $q="select * from mealplan";
       $result=mysqli_query($conn,$q);
while($r=mysqli_fetch_array($result))
{
?>
```
```html
     <option value="<?php echo $r['meals'];?>"> <?php echo $r['meals'];?>
</option>
```
```php
        <?php

}?>
```
```html
     </select>
   </div>
  </div>
  <div class="row">
   <div class="col-25">
    <label for="fname">Description</label>
   </div>
   <div class="col-75">
    <input type="text" id="des" name="desc" placeholder="description" required >
   </div>
  </div>

   <div>
    <div class="row">
   <div class="col-25">
    <label for="lname">Veg</label>
   </div>
   <div class="col-75">
    <input type="number" id="quan" name="veg" placeholder="vegetarian" required>
   </div>
  </div>
   <div class="row">
   <div class="col-25">
    <label for="lname">Non-Veg</label>
   </div>
   <div class="col-75">
    <input type="number" id="quan" name="nveg" placeholder="Non-vegetarian" required>
   </div>
  </div>
```

```
      </div>

   <div class="row">
     <div class="col-25">
       <label for="lname"> Order Date</label>
     </div>
     <div class="col-75">
       <input type="Date" id="quan" name="odate" placeholder="" min="2022-03-03" max="2025-12-
31">

     </div>
   </div>
    <div class="row">
     <div class="col-25">
       <label for="lname"> Delivery Time</label>
     </div>
     <div class="col-75">
       <input type="time" id="quan" name="otime" placeholder="" min='today'>
     </div>
   </div>
   <div>
       <br>
       <br>
       <center><input type="submit" onclick="myfun()" value="Submit"
                  ></center>
   </div>
    <script>
        function myfun() {
  alert("You are ordered successfully");
     </script>
  </form>
</div>
      <?php
      include('footer.php')
      ?>
    </body>
</html>
```

Orderaction.php

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
</head>

<body>
  <?php
  include('../pconnection.php');
  session_start();
  $u=$_SESSION['mail'];
  $sql="";
  $m=$_POST['meal'];
  $d=$_POST['desc'];
```

```
      $v=$_POST['veg'];
      $n=$_POST['nveg'];
      $q=$v+$n;
      $od=$_POST['odate'];
      $ot=$_POST['otime'];

    $sql = mysqli_query($conn, "SELECT * FROM reciever where email='$u'");
    while ($row = mysqli_fetch_array($sql))
    {
       $x=$row['rid'];
       $y=$row['dist'];
    }

    $sql1="INSERT INTO `rorder`(`rid`, `o_type`, `o_des`, `o_quan`,`cquan`, `o_veg`,`cveg`,
  `o_non`,`cnon`, `o_date`, `o_time`, `o_status`, `dist`) VALUES
  ('$x','$m','$d','$q','$q','$v','$v','$n','$n','$od','$ot','Pending','$y')";
    mysqli_query($conn,$sql1);
    if($sql1==TRUE)
       {
       $_SESSION['success_message'] = "Ordered Successfully.";
       header("location:order.php");
       exit();
       }
    else
       {
        echo "Server problem, Try after sometime.";
       }

  ?>
  </body>
  </html>
```

Profile.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    .content-table {
      text-align: center;
      border-collapse: collapse;
      margin: 25px 0;
      font-size: 20px;
      min-width: 400px;
      border-radius: 5px 5px 5px 5px;
      overflow: hidden;
      box-shadow: 0 0 20px rgba(0,0,0,0,15);
    }
```

```
      .content-table th
      {
         background-color: #ff4800;
         color: white;
         text-align: center;
         font-weight: bold;


      }

      .content-table td
      {
         padding: 12px 15px;
      }
      .content-table        {
         border-bottom: 1px solid #dddddd;
         border-bottom: 2px solid #ff4800;
}
   </style>
</head>
<body>
   <?php

include '../pconnection.php';
session_start();
 $u=$_SESSION['mail'];
  include('header.php');

$sql = mysqli_query($conn, "SELECT * FROM reciever where email='$u'");

?>

   <center>
      <h1> PROFILE</h1>
      <table align="center" class="content-table" >
         <tr>
            <th>Organization Name</th>
            <th>Category</th>
            <th>Address</th>
            <th>State</th>
            <th>Pincode</th>
            <th>Idproof</th>
             <th>Idnumber</th>
            <th>Idimage</th>
            <th>Action</th>

         </tr>
         <?php while ($row = mysqli_fetch_array($sql)) {
         ?>
            <tr>
               <td><?php echo $row['rname']; ?></td>
               <td><?php echo $row['rtype']; ?></td>
               <td><?php echo $row['addr']; ?></td>
               <td><?php echo $row['state']; ?></td>
               <td><?php echo $row['pincode']; ?></td>
```
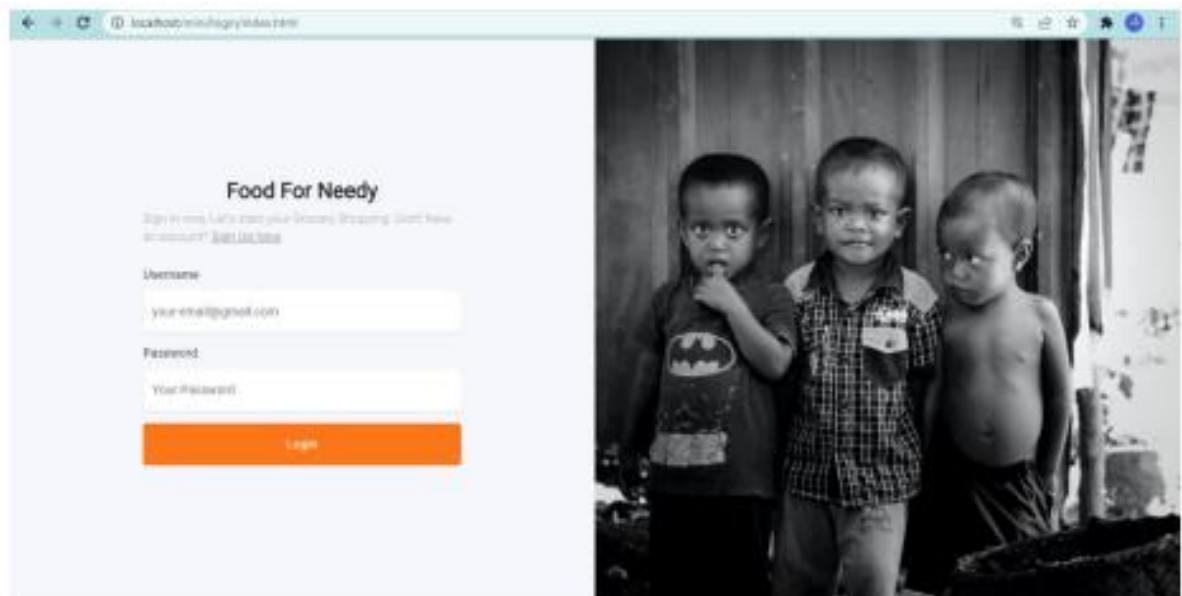
```
        <td><?php echo $row['idproof']; ?></td>
        <td><?php echo $row['idno']; ?></td>

        <td><img src="../Profile/<?php echo $row['idimg'];?>" height="100" width="100"></td>
        <td><a href="profile_action.php?rid=<?php echo $row['rid']; ?>">Edit Profile</a>

        </td>
      </tr>

    <?php } ?>
   </table>

  </center>
  <br>
  <br>
  <br>
  <br>
  <?php
  include('footer.php')
  ?>
</body>

</html>
```
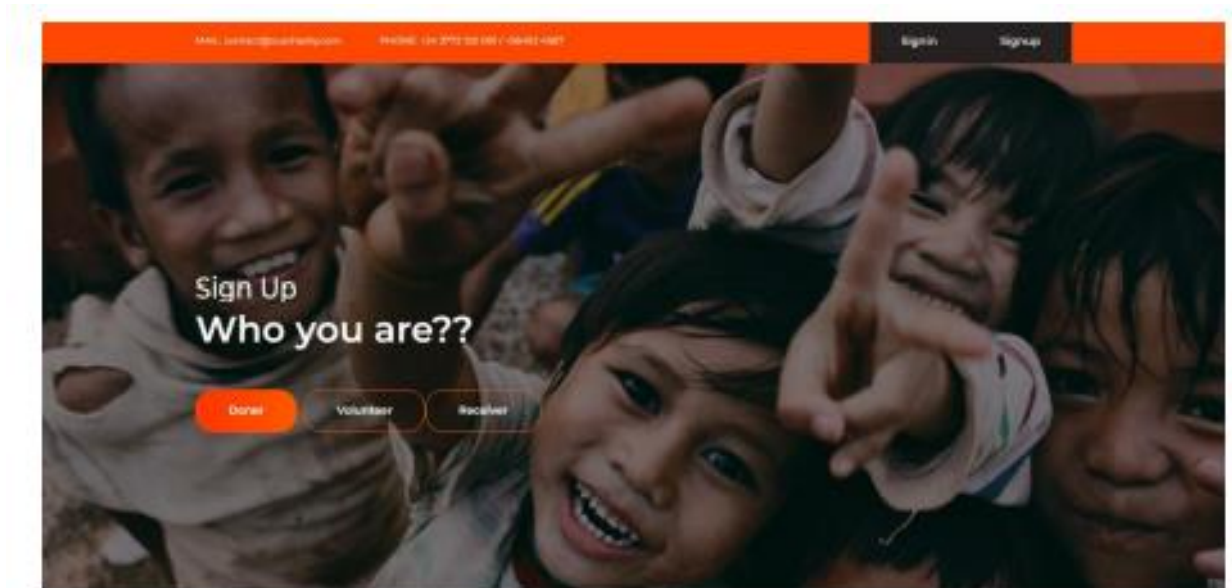
9.2 Screen Shots

Doner Home

Donations

Login

Signup

Registration