

1. INTRODUCTION

1.1 Overview

The Flood rescue rehabilitation corner system is developed for the flood victims to provide their basic needs. The people can easily access the details of those who are in need for help. The purpose of developing Flood rescue rehabilitation corner system is to computerize the traditional way. Currently, we are facing lack of time and communication problems also some of flood effected people could not getting their basic needs properly. The Flood rescue rehabilitation corner system helps to easily access the people who are in need for help and provides accurate information

1.2 Objective

This project provides a quality browser to the users of different levels.

The various objectives of the project are:

- This is a website for flood victims to help and provide their basic needs systematically.
- Starvation can be minimized by providing foods and also provides other basic needs like clothes etc.
- It will ensure that everything is delivered to the destitute
- Provides exact data collection and immediate action.
- Provide the basic needs for everyone in easy manner.
- The information about Flood Relief Camps (such as nearby relief centres, members of camps, etc...)
- The Helpdesk will be provides important contact of rescue agents, police.
- Awareness about the disaster can make people more cognizant to the situation.

1.3 Scope

The project aim to benefit the people affected by flood, by providing basic needs like food ,cloths, etc...

1.4 Methodology

The methodology used here is “Iterative model”.

1.4.1 SDLC

An SDLC model is a conceptual framework describing all activities in a software development project from planning to maintenance.

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

Some of the SDLC models are:

- Iterative Model
- Spiral Model
- Agile Model

1.4.2 Iterative Model

This model leads the software development process in iterations. It projects the process of development in cyclic manner repeating every step after every cycle of SDLC process.

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

In this model we have to revisit each phase in the subsequent iterations and rework is to be done in these phases. This model is used for getting early output and useful for large projects. Since these are not very necessary for the project we are doing this model was not chose.

1.4.3 Spiral Model

Spiral model is a combination of iterative development process model and sequential linear development model i.e. waterfall model with very high emphasis on risk analysis. It allows for incremental releases of the product, or incremental refinement through each iteration around the spiral.

This model is usually used when requirements are unclear and complex. It is not suitable for smaller projects and costs a lot. Hence this method was not chosen for this project.

1.4.4 Agile Model

By breaking the product into cycles, the [Agile model](#) quickly delivers a working product and is considered a very realistic development approach. The model produces ongoing releases, each with small, incremental changes from the previous release. At each iteration, the product is tested.

This method is the present industry standard but not used for this project as it is usually used by experienced senior developers. Final result keeps on changing as per requirements, but the requirements are fixed in this project.

1.4.5 Prototype Model

Prototype methodology is defined as a Software Development model in which a prototype is built, tested, and then reworked when needed until an acceptable prototype is achieved. It also creates a base to produce the final system. It is an iterative, trial, and error method which take place between the developer and the client. This model allows the user to interact and try out with a working model.

The prototype model is applied when detailed information related to input and output requirements of the system is not available. It is usually used when a system does not exist or when a system is large and complex. However our project already knows the requirements and is fixed.

1.5 Mission of the project

The aim of the proposed system is to address the limitations of the current system. It is an automated web service. All sections are handled by admin. It is only implemented in kerala now. The speed, accuracy and reliability of computers become obvious nowadays. The system is designed in such a way that helps can be done in an easier manner. We can make further extension to the system. The speedy and accurate helps through this system is very effective and less time consuming.

1.6 Company / Organizational Profile

RISS technologies an ISO 9001: 2008 certified company focuses on transforming and running business processes and operations including those are complex and industry specific. Our loom is distinctive :through an unbiased, agile combination of smarter process science , targeted technology and advanced analytics, We help our clients become more competitive by making their enterprises more intelligent: adaptive, innovative, globally effective, and connected to their own clients

2. SYSTEM ANALYSIS

Every software project is initiated with a thorough system analysis. System analysis is a general term that refers to a structured process for identifying and solving problems. Analysis implies the process of breaking something down into parts so that the whole may be understood. The definition of system analysis includes not only the process of analysis; it is the application of the systems approach to problem solving using computers. The ingredients are system elements, processes, and technology. This means that to do systems work, one needs to understand the systems work, one needs to understand the systems concept and how organizations operate as a system, and then design appropriate computer-based systems that will meet an organization's requirements. It is actually a customized approach to the use of computer science for problem solving.

In other words Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system.

System is a way of thinking about organizations and their problems. It also involves a set of techniques that helps in solving problems.

The Analyst who carries out the system analysis must know what information to gather, where to find it, how to collect it, and what to make of it. The proper use of tools for gathering information is the key to successful analysis. The tools used are the traditional

- Interview
- Questionnaire
- On-site observations
- Visiting similar site
- Literature review

Interview

Interview is a face to face interpersonal role situation in which a person called the interviewer asks a person being interviewed questions designed to gather information about a problem area. This is the oldest and most often used device for gathering information in systems work. It can be used for two main purposes

- as an exploratory device to identify relations or verify information and
- to capture information as it exists.

The method chosen for this project is an interview because we need the information from users on what all features can be included in a tutorial website, what all information is unnecessary and what all is lacking in the currently available tutorial websites.

The people interviewed for this project are students who have visited these websites, content developers who have probably worked on similar projects and teachers.

Questionnaire

The method of questionnaire was excluded because the number of people available to us who have used tutorial websites were less and an interview with them was possible (questionnaire is used when the number of people from whom the data is to be collected is large). Also the validity of the questionnaire is less compared to the interview.

On-site observation

We are not including the method of on-site observation as it is difficult to go and observe users using such websites.

Visiting similar sites

We also studied features required for a tutorial website by visiting other similar websites and also by analysing the reviews about a few websites by its users. The websites whose reviews by users are taken into account for this project include:

- www.researchgate.net
- www.ijcsmc.com
- www.github.com
- www.techtarget.com

Literature review

A literature review surveys books, scholarly articles, and any other sources relevant to a particular issue, area of research, or theory, and by doing so, provides a description, summary, and critical evaluation of these works in relation to the research problem being investigated. However there is no literature easily available on tutorial website building.

2.1 Existing System

In existing system all processes are done only manually, but in proposed system it is computerized through the application.

2.1.1 Drawbacks of Existing System

- At present there exist only door to door awareness.
- More time consuming, ineffective and not practical
- Communication risk.

- Not accurate.
- Loss of human life

2.2 Proposed System

The proposed system is designed to overcome the disadvantages of the existing system. In existing system all processes are done only manually but in proposed system we have to computerized through the application.

2.3 module and their description

Admin

Admin module mainly deals with collector and higher authorities. Over all powers comes under admin module such as decision making, approvals

User

User module is one of the main components in this website. User can get into the website easily on registering into the website by their names, locality and details. It provides easy access to nearby available camps using their locality.

Camp officer

Camp officer handles the camp settings such as look after the camp, allocate people and ensure that people who have registered for the camp have reached out. Camp officer and requirements that are needed in the camp.

Volunteer

In this module people who are willing to help can register into the website, volunteers are controlled by camp officer. Camp officer gives instructions to the volunteer.

Donor

In this module people are willing to donate for the flood people. Donors can ensure their donation has reached out to those who need it. They can register on to the donor module and make their donations.

2.4 Feasibility Study

A feasibility study is an assessment of the practicality of a proposed project or system. A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture.

2.4.1 Economic Feasibility

Economic feasibility also known as cost/benefit analysis is a procedure to determine the benefits and savings that are expected from a candidate system and compare them with cost. If benefits outweigh costs, then the decision is made to design and implement the system.

The cost of the resources needed for this project are less and the softwares used for developing this browser is available free. Only cost would be to host the site, maintenance is also less. Thus development of this website is economically feasible.

2.4.2 Technical Feasibility

It involves determining whether or not a system can actually be constructed to solve the problem at hand. If there is an existing system it centres around the existing system and to what extent it can support the proposed system.

The proposed system is technically feasible. Since the necessary technology exists and the required resources for the development and maintenance is easily available as well.

2.4.3 Behavioral Feasibility

People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made on how strong a reaction the user is likely to have to the system.

In this project there is minimum behavioural feasibility as the users are already very well aware of the browser and normally skilled enough to use a computer properly.

2.4.4 Legal Feasibility

Legal feasibility is the study to know if the proposed project conforms to the legal and ethical requirements. Content we distribute should be our own, it should not be copied from any other source, if done we should reference it .

This project abides by the software license of all softwares used for development of the project. The contents used also follows copyright rules if any. This is not a work copied from any other source

2.5 Project Scheduling

Project scheduling is a mechanism to communicate what tasks need to get done and which organizational resources will be allocated to complete those tasks in what timeframe. A project schedule is a document collecting all the work needed to deliver the project on time.

Scheduling can be done efficiently using a Gantt chart. Gantt charts are useful for planning and scheduling projects. They help you assess how long a project should take, determine the resources needed, and plan the order in which you'll complete tasks. An interactive Gantt chart software is very useful. You can add tasks and dates into your Gantt chart to have a visual representation of each task's duration. Better still, as dates change (as they inevitably do) you can simply drag and drop those changes and the whole Gantt chart is updated instantly.

2.6 Cost estimation and scheduling

Cost effective

Scheduling

- Requirement collection : 10 days

- Analysis : 15 days
- Design : 20 days
- Coding : 30 days
- Testing : 10 days
- Implementation : 06 days

Total days : 91 days

2.7 System Specifications

Hardware Required –

- Processor : Intel Pentium IV or above
- Monitor : Min. 14
- RAM :4 Gb
- Hard Disk : 80 GB
- Keyboard : Standard 104 Keys
- Mouse : Serial mouse

Software Requirements –

- IDE : JETBRAINS PYCHARM, ANDROID STUDIO
- OS : Windows 7 or above / Linux New Versions
- Languages : Python , Java
- Other Tools : SQLyog, Adobe Dream Weaver
- Back End: My SQL
- Android Os : Android M or above
- Android Ram Capacity : 2Gb or above
- Front End :HTML,XML

2.8 Software Environment

2.8.1 PYCHARM

PYCHARM enables programmers to write high quality Python code. The editor enables programmers to read code easily through color schemes, insert indents on new lines automatically, pick the appropriate coding style, and avail context-aware code completion suggestions. In our project this editor help us to expand a code block to an expression or logical block, avail code snippets, format the code base, identify errors and misspellings,

detect duplicate code, and auto-generate code. Also, the editor makes it easier for us to analyze the code and identify the errors while writing code.

2.8.2 SQLYOG

SQLYOG is used for creating our web data base .SQLYOG is a fast, easy to use and compact graphical tool for managing our MySQL databases .SQL prefer to work in a visual environment, SQLYOG makes it easy for us to get started and provides us tools to enhance MySQL experience.

2.8.3 Smart Draw

Smart Draw is a diagram tool used to make flowcharts, organization charts, mind maps, project charts, and other business visuals. Smart Draw has two versions: an online edition and a downloadable edition for Windows desktop.

2.8.4 Data Flow Diagram

The DFD of this project was drawn using Smart Draw.

2.8.6 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on [IntelliJ IDEA](#) . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for [Google Cloud Platform](#), making it easy to integrate Google Cloud Messaging and App Engine

3. SYSTEM DESIGN

3.1 Introduction

System design transforms a logical representation of what a given system is required to do into physical specifications. The specifications are converted into physical reality during development. The design forms a blueprint of the system and how the components relate to each other. The design phase proceeds according to an orderly sequence of steps beginning with review and assignment of task and ending with package design. During the first stage, output design, and the analyst determines what data the application produces and how to organize and present that data.

System design is the process of planning a system or to replace or to complement an existing system but before this planning should be done. It must be thoroughly understood about the old system and determine how computers can make its operations more effective. The importance of system design can be understood with a single word and that is quality. design is the phase of where quality is fostered in software development. Design is the only way to transform a customer's requirement into software development and software support steps that follow. Without design there is a risk of building an unstable system that will fail when small changes are made. The most creative and challenging phase of the software life cycle is system design. The term design describes a final system and a process by which it is developed. The design may be defined as the "The process of applying various techniques and principles for the purpose of defining a device, a process or a system with sufficient details to permit its physical realization".

The designer's goal is how the output is to be produced and in what format. Samples of output and input also presented. Second input files and database files have to be designed to meet the requirements of proposed output. The processing phases are handled through the program construction and testing. Finally details related to justification of the system and an estimate of the impact of the candidate system on the user and the organization are documented and evaluated by management as a step towards implementation.

3.2 Input Design

Input design is the process of converting the user originated input into a computer based format. The design for handling input specifies how the data are accepted for computer processing. Input design is a part of overall system design that needs careful attention and includes Specifying the means by which actions are taken. A system user interacting through a workstation must be able to tell the system whether to accept input, produce a report or end processing. The collection of input data is considered to be the most expensive part of the system design. Since the input has to be planned in such a manner as to get relevant information taken, extreme care is taken to be obtained from the information. If data going into the system is then the processing and output will magnify these errors.

Input design is the process of converting user oriented description of the inputs to a computer based business system into a programmer oriented specification. Inaccurate input based system into a programmer oriented specification. Inaccurate input is the most common cause of data processing error. If the input design is poor, particularly where operators must enter data from source permits, data to enter a computer system. The main objective of the

system is to specify how the information is put into a form that is acceptable to the computer. The system also needs to include appropriate messages which ensure that the user can understand the context. The input data is validated to minimize the errors in the data entry. User is never left in a state of confusion as to what is happening.

3.3 Output Design

The output design is an ongoing activity almost from the beginning of the project and follows the principles of forms designing. Computer output is the most important and direct source of information to the user. Efficient, intelligible, and well defined output design improves the relationship of the system and the user. The objective of the output design is to define the format of all printed documents and reports screens that will be produced by the system. Computer output is the most important and direct source of information to the user.

Output design phase of the system is concerned with the convergence of information to the end user friendly manner. The output design should be efficient, intelligible so that system relationship with the end user is improved and thereby enhancing the process of decision making.

3.4 Detailed System Design

Detailed system design is data flow based methodology. The approach begins with the system specification that identifies input and output aspects of the system. The system specification is used as a basis for the graphic representation of the data flow and process. This design phase partitions a program into small independent modules. They are arranged in hierarchy form. It is an attempt to minimize the complexity and make a problem manageable by subdividing into smaller segments.

3.4.1 Use Case Diagram

The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system.

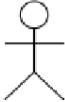

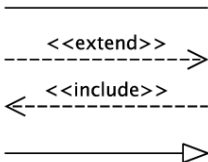
In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

symbols and notation:

- **Use cases:** Horizontally shaped ovals that represent the different uses that a user might have.
- **Actors:** Stick figures that represent the people actually employing the use cases.

- **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.
- **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- **Extend:** It is used when a use case conditionally adds steps to another first class use case.
- **Include:** It is used to extract use case fragments that are duplicated in multiple use cases. The included use case cannot stand alone and the original use case is not complete without the included one.

Symbol	Reference Name
	Actor
	Use case
	Relationship

3.5 DATABASE DESIGN

A database is collection of inter related data stored with minimum redundancy to serve many application. The general objectives considered in database design are controlled redundancy, data dependency, more information at low cost, accuracy and integrity; recover from failure, privacy and security.

In database environment, the Database Management System (DBMS) is the software that provides the interface between the data file on a disk and the management, they differ in the way they structure data. The three types of data structures are hierarchical, network and relational. Structuring in which all and relationship are presented in a flat, two dimensional table called relation. A relation is equivalent to file.

Data structuring is refined through a process called normalization. Data are grouped into simplest way possible, so that later changes can be made with a minimum impact on data structures.

Based on the requirements determined during the definition phase of project life cycle, the data elements describing the entity were determined. They are later submitted to normalization to remove redundancy and to optimize them.

The organization of data in database aims three major objectives

Data integration

In a database information from several files are coordinated, accessed and upon as through it is in a single line. Logically, the information is centralized physically and the data communication facility.

Data integrity

The database all data are stored in one place only and it allows each application to access it. The approach results in more consistent.

Data independence

The objectives seek to allow changes in the content and organization of physical data without reprogramming the application. This approach used designed the system take care of these objectives into consideration so as organization the data.

Detailed System Design

Detailed system design is data flow based methodology. The approach begins with the system specification that identifies input and output aspects of the system. The system specification is used as a basis for the graphic representation of the data flow and process. This design phase partitions a program into small independent modules. They are arranged in hierarchy form. It is an attempt to minimize the complexity and make a problem manageable by subdividing in to smaller segments.

3.5.1 CONCEPTUAL MODEL

The conceptual level represents the major data and relationship between them. Conceptual level describes the essential features of system data. It uses symbols from a modelling called entity relationship analysis. It is an early phase of the design process, in which the broad outlines of function and form of something are articulated. It includes the design of interactions, experiences, processes and strategies. It involves an understanding of people's needs and how to meet them with products, services & processes. Common artefacts of conceptual design are concept sketches and models

3.5.2 NORMALIZATION

The normalization of data refers to the way data item are grouped into record structures. Normalization is used to overcome drawbacks likes repetition of data (redundancy), loss of information and inconsistency. In other words neutralization is a technique of separating redundant fields and breaking up large table into smaller one. In our design all tables have been normalized up to the third normal form. The different normal forms applied during the database design are given below:

3.5.3First Normal Form (1NF)

A relation is said to be in 1NF if it satisfies the constraints :

- Each column contain atomic values.

- In each column value stored should be of same datatype.
- Each column should have unique name.

3.5.4 Second Normal Form (2 NF)

A relation is said to be in 1NF if it satisfies the constraints:

- Should be in 1NF.
- Should not have any partial dependencies.

3.5.5 Third Normal Form (3NF)

A relation is said to be in 1NF if it satisfies the constraints:

- Should be in 2NF.
- Should not have transitive dependency.

3.5.6 Control Redundancy

Redundant occupies space and therefore, is wasteful. If versions of the data are in different phases of updating the system often gives conflicting information. A unique aspect of database design is storing only ones, which controls redundancy and improves system performance

3.6 Data Flow Diagram (DFD)

A data-flow diagram is a way of representing a flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops.

The dataflow diagram is used for classifying system requirements to major transactions that will become programs in system design. This is the starting point of design phase that functionally decompose the required specification down to the lower level of details.

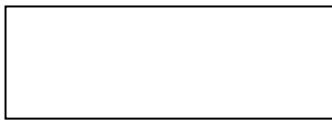
Symbols and notations used:

- **Entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
- **Process:** any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”

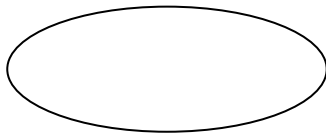
- **Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”
- **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

3.6.1 DFD NOTATIONS

In DFD, there four symbols, they are as follows,



- Source (or) Destination



- Process

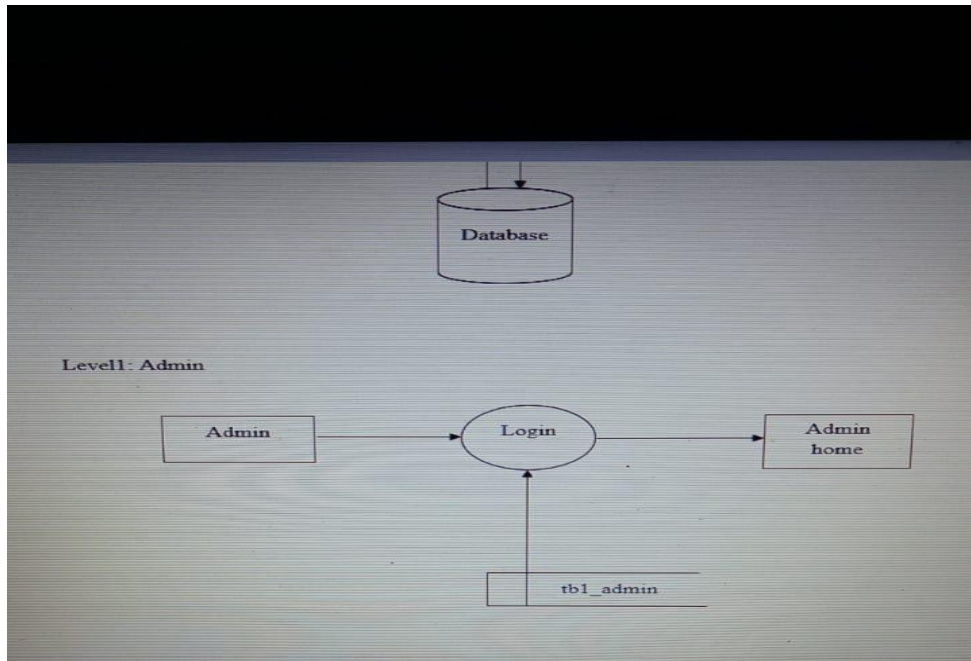


- Data storage

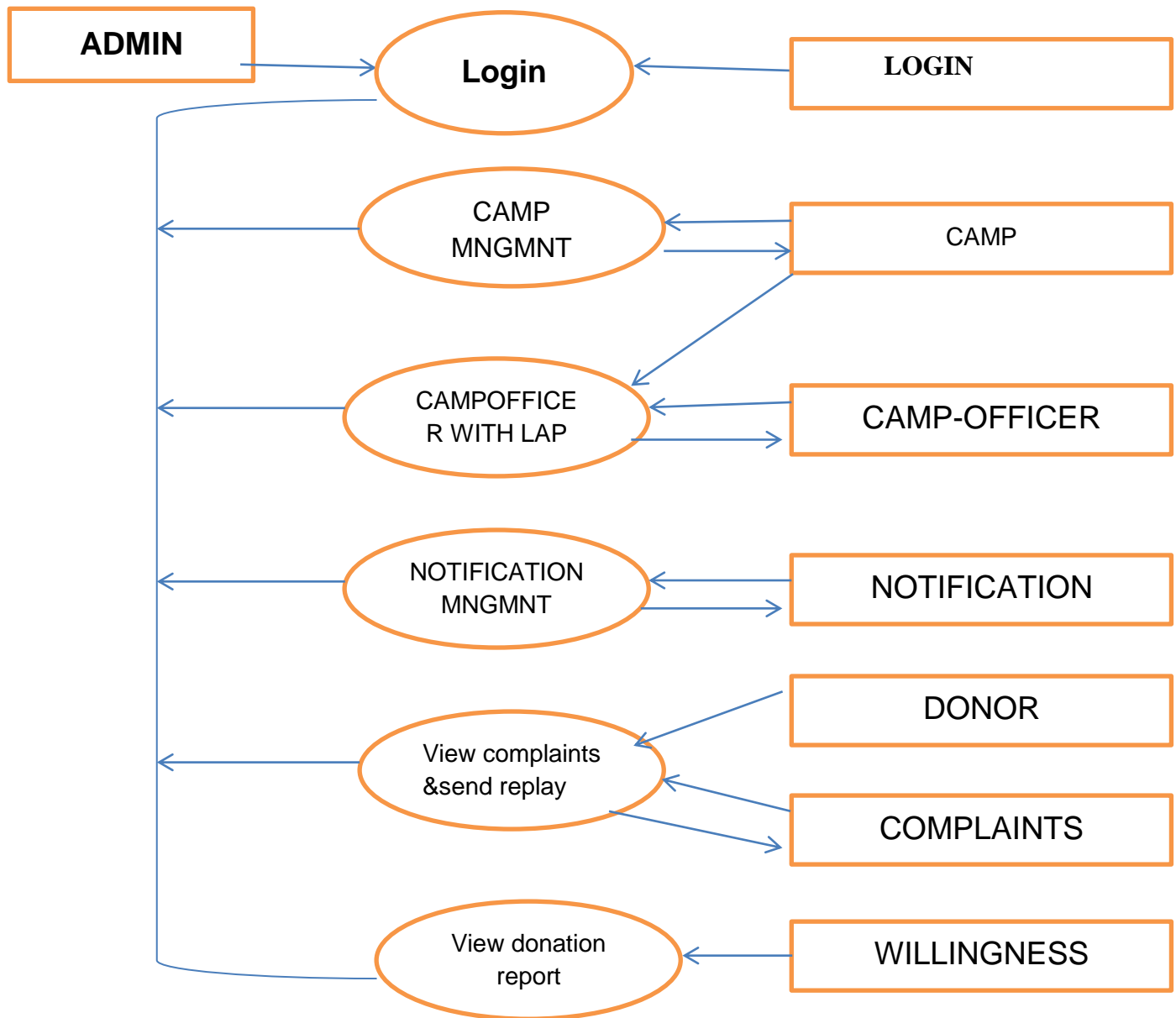


- Arrow

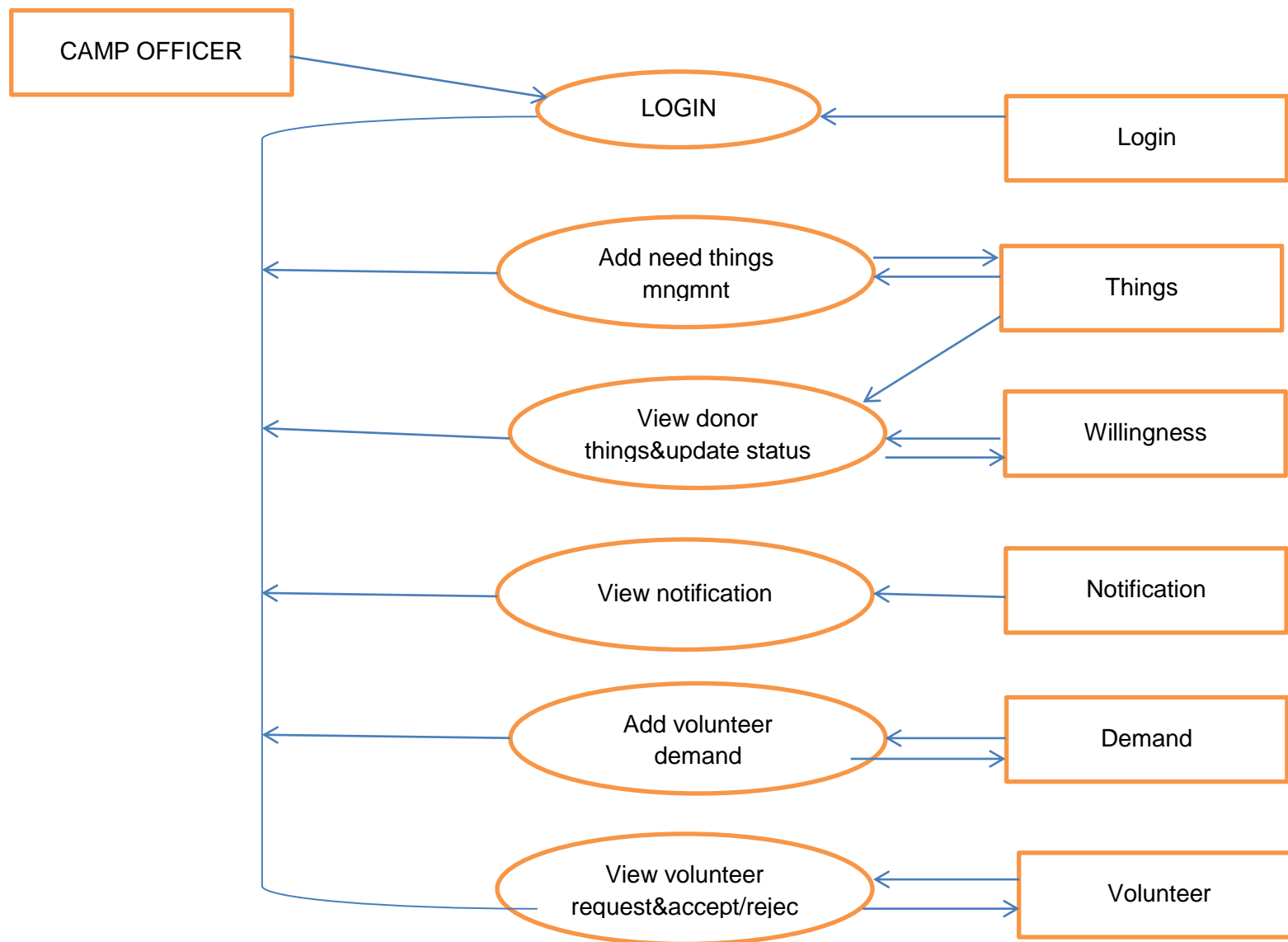
LEVEL 0



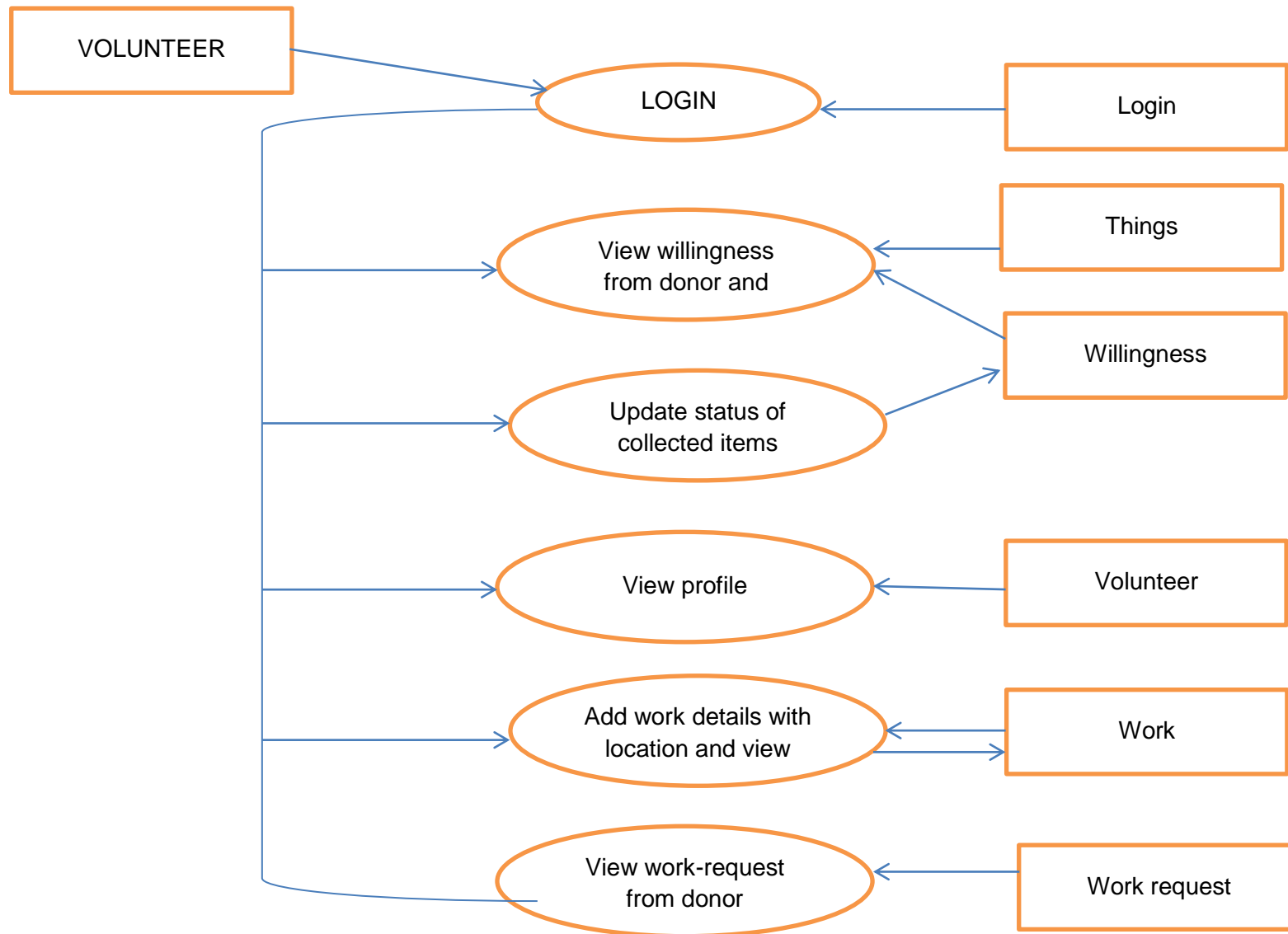
LEVEL 1



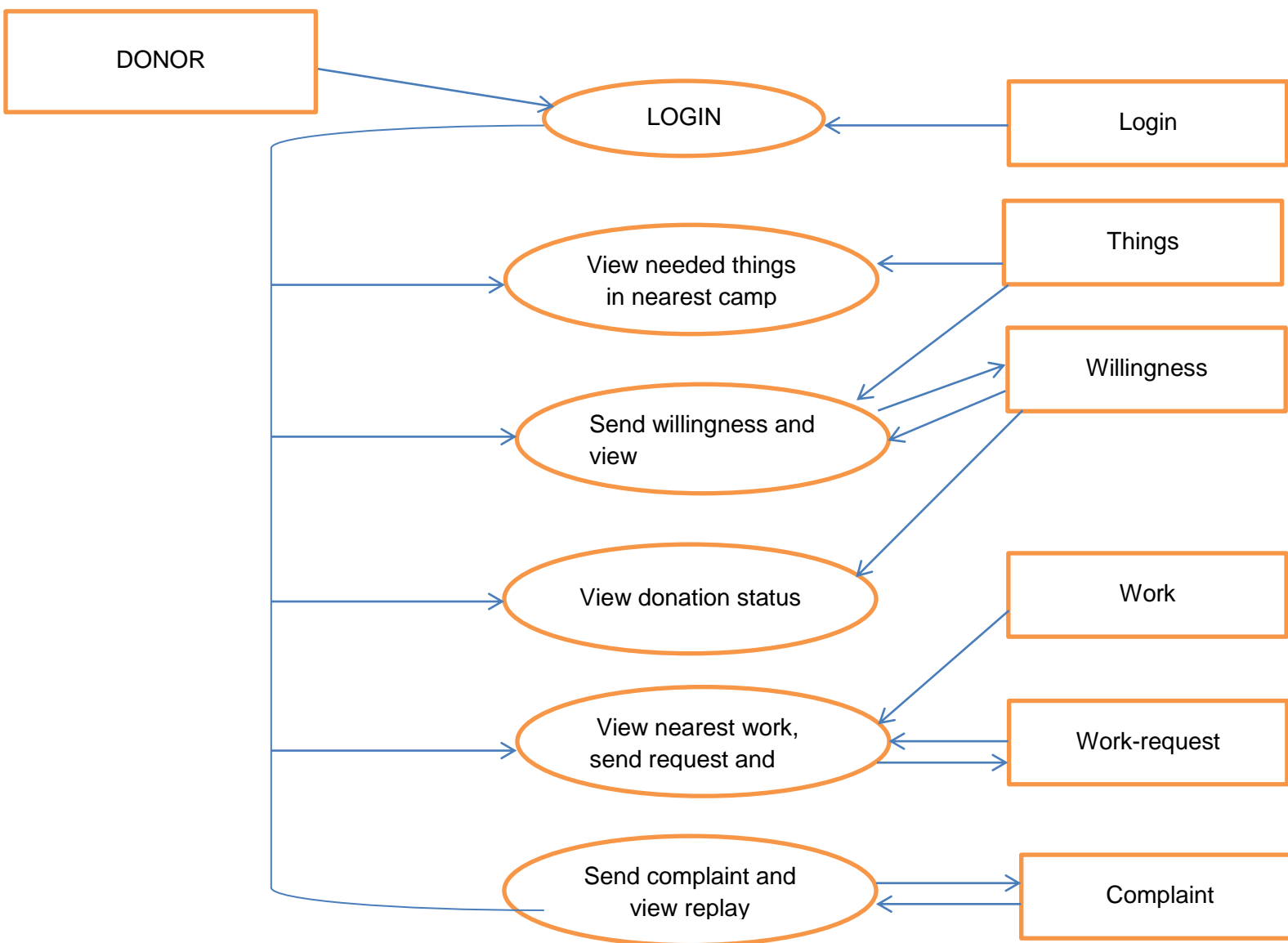
LEVEL 2



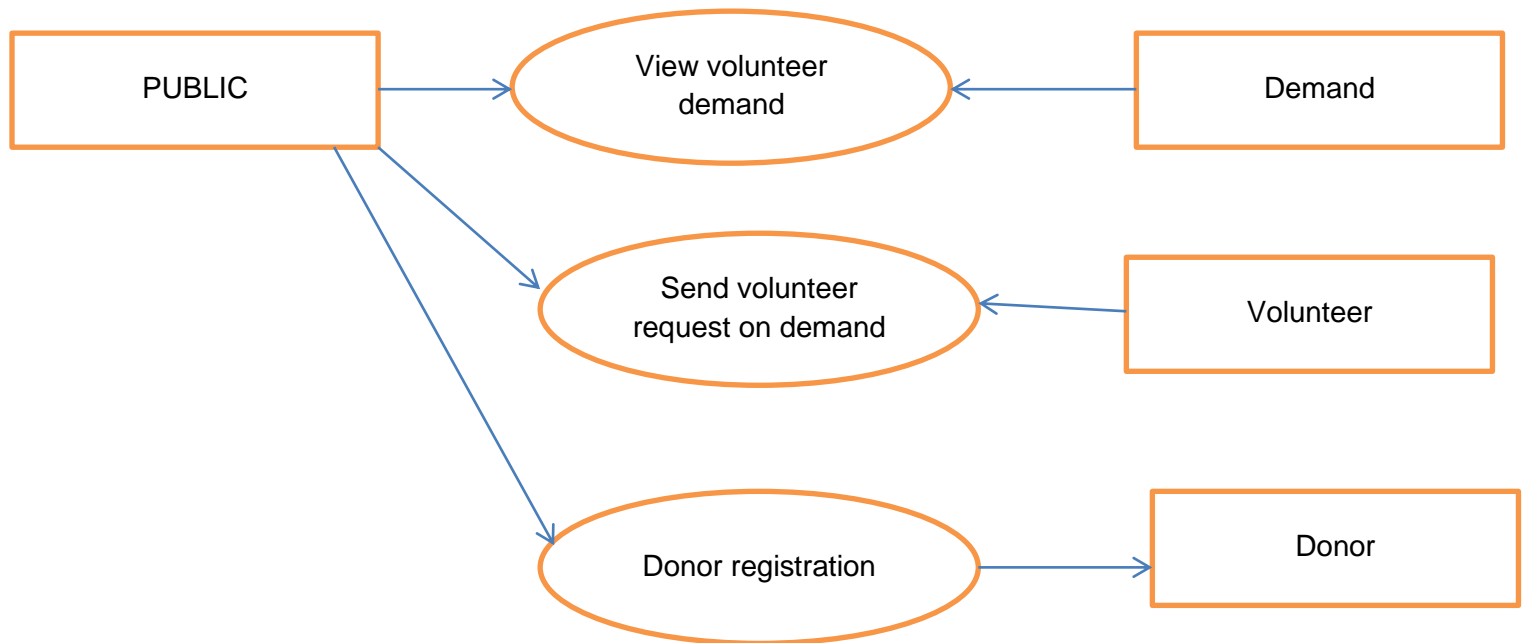
LEVEL 3



LEVEL 4



LEVEL 5

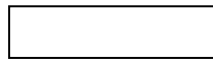


3.7 ER DIAGRAM

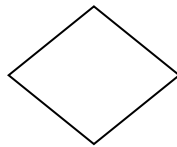
E-R model has some means of describing the physical database model; it is basically useful in the design & communication of the logical data model. In this mode, objects of similar structure are collected into an entity set. The relationship between entity sets is requested by a named E-R relationship. The data structure employing the E-R model is usually shown pictorially using entity –relationship (E-R) diagrams.

\

3.7.1 ER NOTATIONS



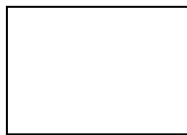
- Entity



- Relationship



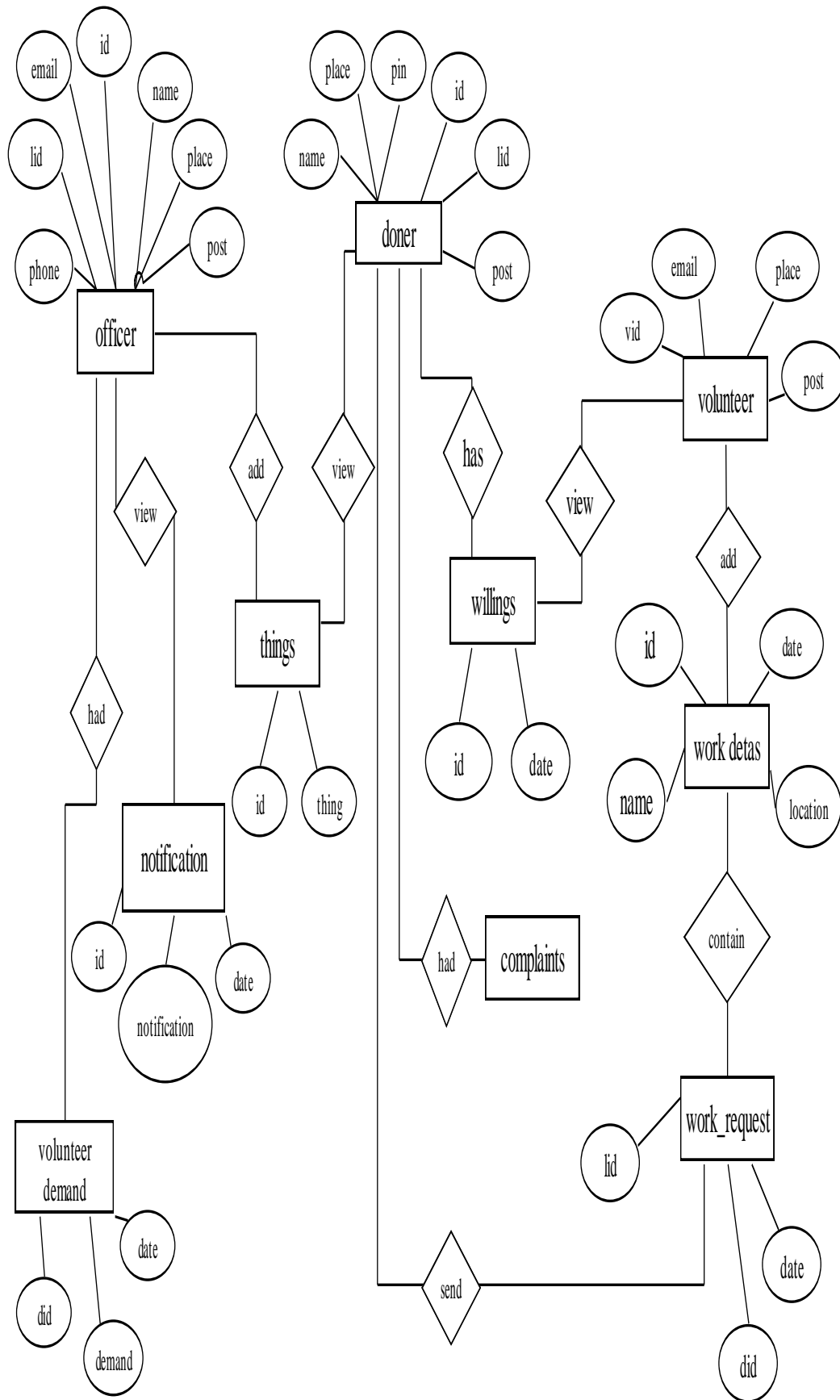
- Attribute



- Weak Relationship



- Link



3.8 DESIGN PROCESS**TABLE DESIGN****Table name : login****use : To Login****Primary key: login-Id**

FIELD NAME	DATATYPE	LENGTH	DESCRIPTION	CONSTRAINTS
Login id	INTEGER	11	Primary key	NULLABLE
Username	VARCHAR	100	Username	NULLABLE
Password	VARCHAR	50	Password	NULLABLE
Type	VARCHAR	50	Type	NULLABLE

Table name : camp

Use : camp

Primary key: camp Id

FIELD NAME	DATA TYPE	LENGTH	DESCRIPTION	CONSTRAINTS
Camp Id	INTEGER	11	Primary key	NULLABLE
Camp-place	VARCHAR	100	Place	NULLABLE
Camp-pin	INTEGER	11	Pin	NULLABLE
Camp-post	VARCHAR	60	Post	NULLABLE
Camp-phone	INTEGER	11	Phone	NULLABLE
Camp-lat	VARCHAR	100	Latitude	NULLABLE
Camp-long	VARCHAR	100	Longitude	NULLABLE

FLOOD RESCUE REHABILITATION CORNER

Table name: camp-officer

Use: camp officer

Primary key : id

Field name	Data type	Description	Constraint
Officer-id	INTEGER(11)	officer id	Primary key
Officer-name	VARCHAR (100)	Officer id	NULLABLE
Officer-gender	VARCHAR (10)	Gender	NULLABLE
Officer-place	VARCHAR (100)	Gender	NULLABLE
Officer-post	VARCHAR (100)	Pin	NULLABLE
Officer-phone	VARCHAR (10)	Post	NULLABLE
Officer-image	INTEGER(11)	Image	NULLABLE
Officer-email	INTEGER(11)	Email	NULLABLE
Officer _qualification	VARCHAR(100)	Qualification	NULLABLE
Officer _ logid	INTEGER(11)	Login-id	NULLABLE
Camp _id	INTEGER(11)	Camp-id	NULLABLE

FLOOD RESCUE REHABILITATION CORNER

Table name: notification

Use: notification

Primary key: id

Field name	Data type	Description	Constraint
Notification-id	INTEGER(11)	Notification id	Primary key
Notification	VARCHAR(50)	Notification	NULLABLE
Dates	VARCHAR(50)	Dates	NULLABLE

Table name complaint

Use: store complaint details

Primary key :complaint id

Field name	Data type	Description	Constraint
Complaint-id	INTEGER(11)	Complaint id	Primary key
d-id	INTEGER(11)	d- id	NULLABLE
Complaint	VARCHAR (200)	Complaint	NULLABLE
Reply	VARCHAR (200)	Reply	NULLABLE
Date	VARCHAR (10)	Date	NULLABLE

FLOOD RESCUE REHABILITATION CORNER

Table name: donor

Use: to view donation

Primary key: donor id

Field name	Data type	Description	Constraint
d-id	INTEGER(11)	Donor id	Primary key
d- name	VARCHAR (50)	Donor name	NULLABLE
d-gender	VARCHAR (10)	Donor gender	NULLABLE
d-dob	VARCHAR (10)	Donor dob	NULLABLE
d- place	VARCHAR (100)	Donor place	NULLABLE
d-pin	VARCHAR (10)	Donor pin	NULLABLE
d-phone	VARCHAR 10)	Donor phone	NULLABLE
d- image	VARCHAR (100)	Donor image	NULLABLE
d- email	VARCHAR (50)	Donor email	NULLABLE
d- lid	INTEGER(100)	Lid	NULLABLE

Table name: willingness

Use : willingness

Primary key: willingness id

Field name	Data type	Description	Constraint
Wil _id	INTEGER(11)	Willingness id	Primary key
donor_ lid	INTEGER(11)	Donor lid	NULLABLE
things _id	INTEGER(11)	Things id	NULLABLE
Date	VARCHAR (10)	Date	NULLABLE
Status	VARCHAR (100)	Status	NULLABLE
Volunteer _id	VARCHAR (100)	Volunteer id	NULLABLE

Table name: things

Use: things

Primary key: things id

Field name	Data type	Description	Constraint
Things _id	INTEGER(11)	Things id	Primary key
Things name	VARCHAR (100)	Thing name	NULLABLE
Quantity	VARCHAR (100)	Quantity	NULLABLE
Camp _id	INTEGER(11)	Camp id	NULLABLE

Table name: volunteer

Primary key: volunteer id

Field name	Data type	Description	Constraint
Volunteer id	INTEGER(11)	Volunteer id	Primary key
Volunteer name	VARCHAR (100)	Name	NULLABLE
Volunteer phone	VARCHAR (10)	Phone	NULLABLE
Volunteer gender	VARCHAR (10)	Gender	NULLABLE
Volunteer dob	VARCHAR (10)	Date of birth	NULLABLE
Volunteer place	VARCHAR (100)	Place	NULLABLE
Volunteer pin	VARCHAR (10)	Pin	NULLABLE
Volunteer post	VARCHAR (100)	Post	NULLABLE
Volunteer email	VARCHAR (100)	Email	NULLABLE
Volunteer image	VARCHAR (100)	Image	NULLABLE
Volunteer lid	INTEGER(11)	Volunteer Lid	NULLABLE
Camp id	INTEGER(11)	Camp id	NULLABLE

Table name: volunteer demand

Primary key :null

Field name	Data type	Description	Constraint
Demand _id	INTEGER(11)	Demand id	Primary key
Description	VARCHAR (500)	Description	NULLABLE
Camp _id	INTEGER(11)	Camp id	NULLABLE
Date _need	VARCHAR (10)	Date	NULLABLE

Table name: work

Primary key: work id

Field name	Data type	Description	Constraint
Work _id	INTEGER(11)	Work id	Primary key
Volun_ lid	INTEGER(11)	Volunteer lid	NULLABLE
Work	VARCHAR (100)	Work	NULLABLE
Place	VARCHAR (100)	Place	NULLABLE
Pin	VARCHAR (10)	Pin	NULLABLE
Post	VARCHAR (100)	Post	NULLABLE
Latitude	VARCHAR (100)	Latitude	NULLABLE
Longitude	VARCHAR (100)	Longitude	NULLABLE

Table name: work request

Primary key: work id

Field name	Data type	Description	Constraint
Work _id	INTEGER(11)	Work request id	Primary key
Donor _lid	INTEGER(11)	Donor lid	NULLABLE
Work _id	INTEGER(11)	Work id	NULLABLE
Date	VARCHAR (10)	Date	NULLABLE
Time _from	VARCHAR (10)	Time from	NULLABLE
Time _to	VARCHAR (10)	Time to	NULLABLE

FLOOD RESCUE REHABILITATION CORNER

Date_ from	VARCHAR (10)	Date from	NULLABLE
Date_ to	VARCHAR (10)	Date to	NULLABLE
No_of_members	VARCHAR (100)	No of members	NULLABLE
Status	VARCHAR (200)	Status	NULLABLE

4. SYSTEM CODING

Coding is the process of designing and building an executable computer program to accomplish a specific computing result.

4.1 Introduction

The goal of coding is to translate the design of the system protected during the design phase into code in a programming language, which can be executed by a computer. When considered as a step in the software engineering process. The coding translates a detailed design representation of software into a programming language realization which can be executed by the computer. The quality of source code can be improved by the use of structured coding techniques, good coding style and readable, consistent source code.

The goal of the coding phase is to produce simple and clear programs. It should be constructed in a way that is easy to read and understand. Maintenance phase in any software development takes a lot of time.

4.2 Coding Design

The design must be translated into a machine readable form. The coding steps perform the task. If the design is performed in a detailed manner, coding can be accomplished mechanically. The goal of the coding phase is to translate the design of the system code into a given programming language.

4.3 Programming Paradigm

All design contains hierarchies as creating a hierarchy is a natural way to manage complexity. Most design methodology for software also produces hierarchy. The question at coding time is given to the hierarchy of modules being built starting from the top level or starting from the bottom level.

The programming paradigm used here is OOP.

Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOP paradigm was used in this project since it provides many features like code reusability, access specifiers, polymorphism etc which are extremely useful in coding. Other uses of OOP include:

- OOPs makes development and maintenance easier whereas in Procedure-oriented programming languages it is not easy to manage if code grows as project size grows.
- OOPs provide data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.
- OOPs provide the ability to simulate real-world events much more effectively. We can provide the solution of real word problems if we are using the Object-Oriented Programming language.

5. TESTING AND IMPLEMENTATION

5.1 Introduction

Testing is a set of activities that can be planned in advance and conducted. Systematically, this is aimed at ensuring that the systems work accurately and efficiently before live operation commences. Testing is the process of correcting a program with the intent of finding an error. A good test case is one that has a high probability of finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error.

5.2 Testing Objectives

There are several rules that can serve as testing objectives

Testing is a process of executing a program with the intent of finding an error.

A good test case is one that has a high probability of finding an undiscovered error. Testing is vital to the success of the system to the success of the system. System testing makes a logical assumption that all parts of the system are subject to a variety of tests: online response, volume, stress, recovery and security and usability tests. A series of tests are performed before the system is ready for user acceptance testing.

5.3 Testing and Strategies

White Box Testing

Black Box Testing

5.3.1 White Box Testing

White box testing is also known as code testing. The code checking strategy checks for the correctness of every statement in the program. To follow this strategy, there should be cases that result in execution of every instruction in the program or module, which is every path in the program, is tested. The test cases should be guaranteed that independent paths within modules are executed once.

Exercise all logical decisions on their true or false sides.

Execute all loops at their boundaries and within their operational bounds.

This testing strategy, on the face of it, sounds exhaustive. If every statement in the program is checked for its validity, there does not seem to be much scope of error.

5.3.2 Black Box Testing

Black box testing is also known as specification testing. To perform black box testing, the analyst examines the result, the analyst can examine specifications taking what the program or module should do and how it should perform on the various conditions and submitted for processing . By examining the result, the analyst can examine whether the program performs according to the specified requirements.

5.4 Types Of Testing

Different types of testing are:

- Unit Testing
- Integration Testing
- Validation Testing
- Output Testing
- User Acceptance Testing

5.4.1 Unit Testing

In this testing we test each module individually and integrate the overall system. Unit testing focuses verification efforts on the smaller unit of software design in the module. This is also known as “module” testing. The module of the system is tested separately. The testing is carried out during the programming stage itself. In this testing step each module is found to be working satisfactory as regard to the expected output from the module. There are some validation checks for verifying the data input given by the user which both the formal and validity of the entered. It is very easy to find errors when debugging the system.

5.4.2 Integration Testing

Data can be lost across an interface; one module can have an adverse effect on the other sub functions, when combined by May not produce the desired major functions. Integrated Testing is the systematic testing for constructing the uncovered errors within the interface. This testing was done with sample data. The developed system has run successful for this sample data. The need for an integrated test is to find the overall system performance.

5.4.3 Validation Testing

At the culmination of black box testing software is completely assembled as a package, interface errors have been uncovered and corrected and the final series of software test, validation test begins. Validation testing can be defined many ways but a simple definition is that validation succeeds when the software functions in a manner that can be reasonably accepted by the customer. After the validation test has been conducted one of the two possible conditions exists. The function of performance characteristics confirm to specification and are accepted. A deviation from specification is uncovered and a deficiency list is created.

5.4.4 Output Testing

After performing the validation testing the next test is output testing of the proposed system since no system could be useful if it does not produce the required data in the specific format. The output displayed or generated by the system under consideration is tested by, asking the user about the format displayed. The output format on the screen is found to be correct as the format was designed in the system according to the user needs. Hence the output is testing doesn't any correction in the system.

5.4.5 User Acceptance Testing

User acceptance of the system is the key factor for the success of a system. User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the

software system before moving the software application to the production environment. The main purpose of UAT is to validate the end to end business flow.

5.5 TEST CASE

Admin

Login

Test case	Input	Result	Remark
1.Admin login	Enter correct username and password	Login successful	Same output as expected
2.Admin login	Incorrect username and correct password	Incorrect username	Same output as expected
3.Admin login	Correct username and incorrect password	Incorrect password	Same output as expected

User

Login

Test case	Input	Result	Remark
1.User login	Enter correct username and password	Login successful	Same output as expected
2.User login	Incorrect username and correct password	Incorrect username	Same output as expected
3.User login	Correct username and incorrect password	Incorrect password	Same output as expected

5.6 Implementation

Implementation is the stage of the project when theoretical design is turned into a working system. Most crucial stage is achieving a successful system and confidence that the new system will work effectively. It involves careful planning, investigation of the manual system and new system. Implementation means converting a new or revised system design into an operational one. The implementation includes all the activity that takes place to convert from the old system to new one. There are several activities involved while implementing a project:

- Careful planning
- Investigating the current system and its constraints on implementation
- Design of methods to achieve the changeover.

- Training of the staff in the changeover procedure and evaluation of change over method.

However in this project the above points including training of staff are not necessary as it targets a younger generation who is skilled at using a pc and internet. Implementation of this project is focused on only hosting of this website.

6. MAINTENANCE

6.1 Introduction

Software maintenance is the various modification activities that occur following the product release. The project is coded in an efficient manner that facilitates easy understanding and there by easy maintenance. Modification is made to enhance, adapt, and correct errors in software products. Maintenance includes all the activities after the installation of the software that is performed to keep the system operational. Two forms of maintenance are being used. They are adaptive maintenance and corrective maintenance.

6.2 Corrective Maintenance

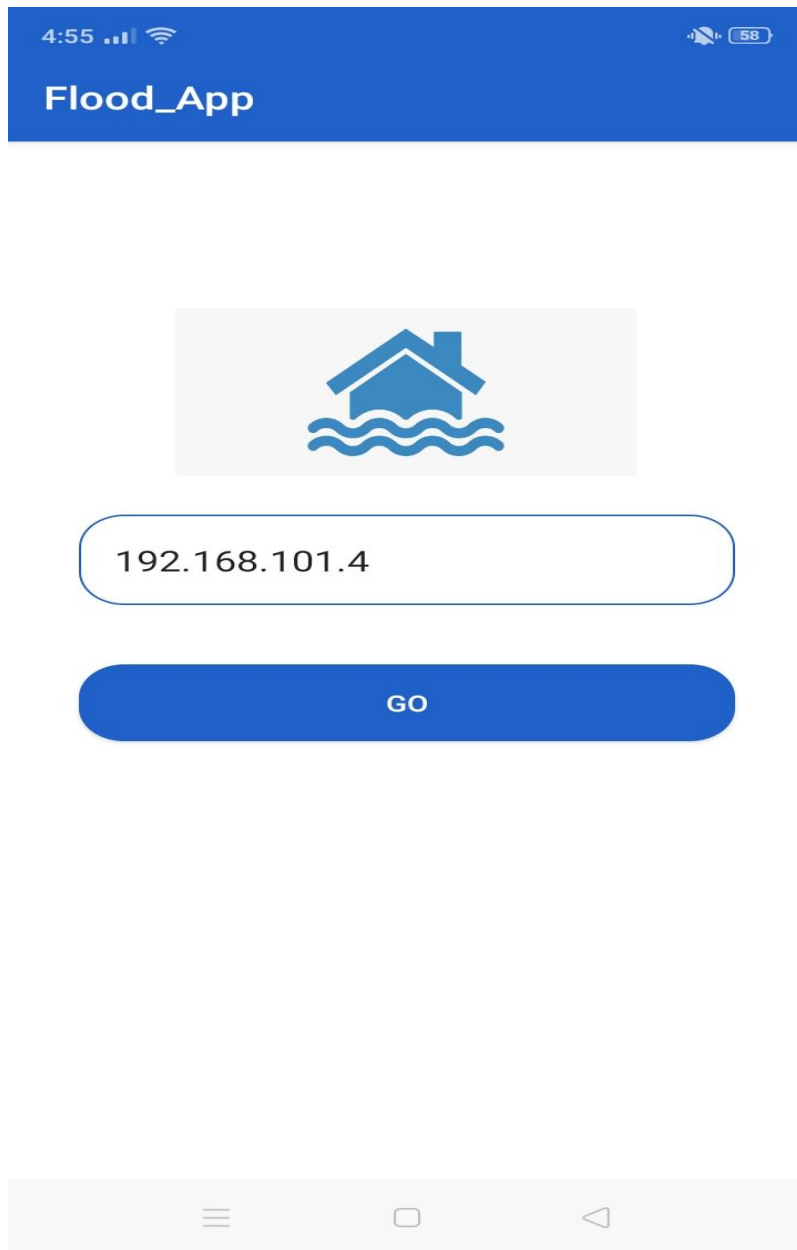
It is generally agreed that for large systems, removing all the faults before delivery is extremely difficult. Maintenance activities related to the fixing of errors falls under the corrective maintenance.

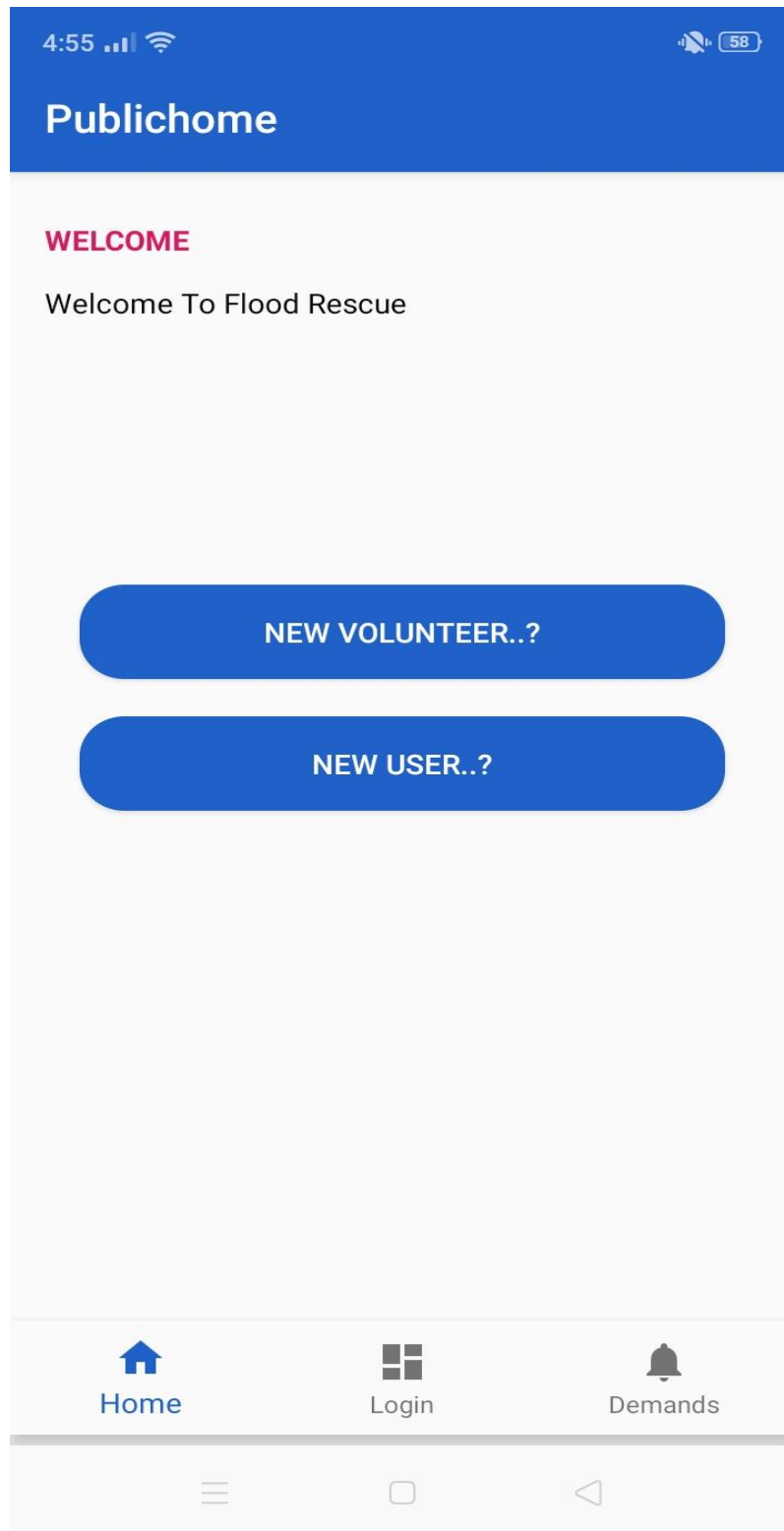
6.3 Adaptive Maintenance






Removing errors is one of the activities of maintenance. Adaptive maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system. Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment. Adaptive maintenance consists of adapting software to changes in the environment such as the hardware or the operating system.

During the modification of the software, the effects of change have to be clearly understood by the maintainer since introducing undesired side effects to the system during the modification is easy. To test whether those aspects of the system that are not supported to be modified will operate as they were before modification, regression testing is done. This test includes old test cases to test that no new errors have been introduced.

7. SCREENSHOT





2:19    ...  26 

Flood_App

27-08-2000

☐ Male ☒ Female

vythiri

673576

Sherinshahna626@gmail.com

9207994165

REGISTER

2:26 [signal icons] [battery icon] 28 [battery level]

Flood_App

WELCOME

[Login to Continue](#)

admin

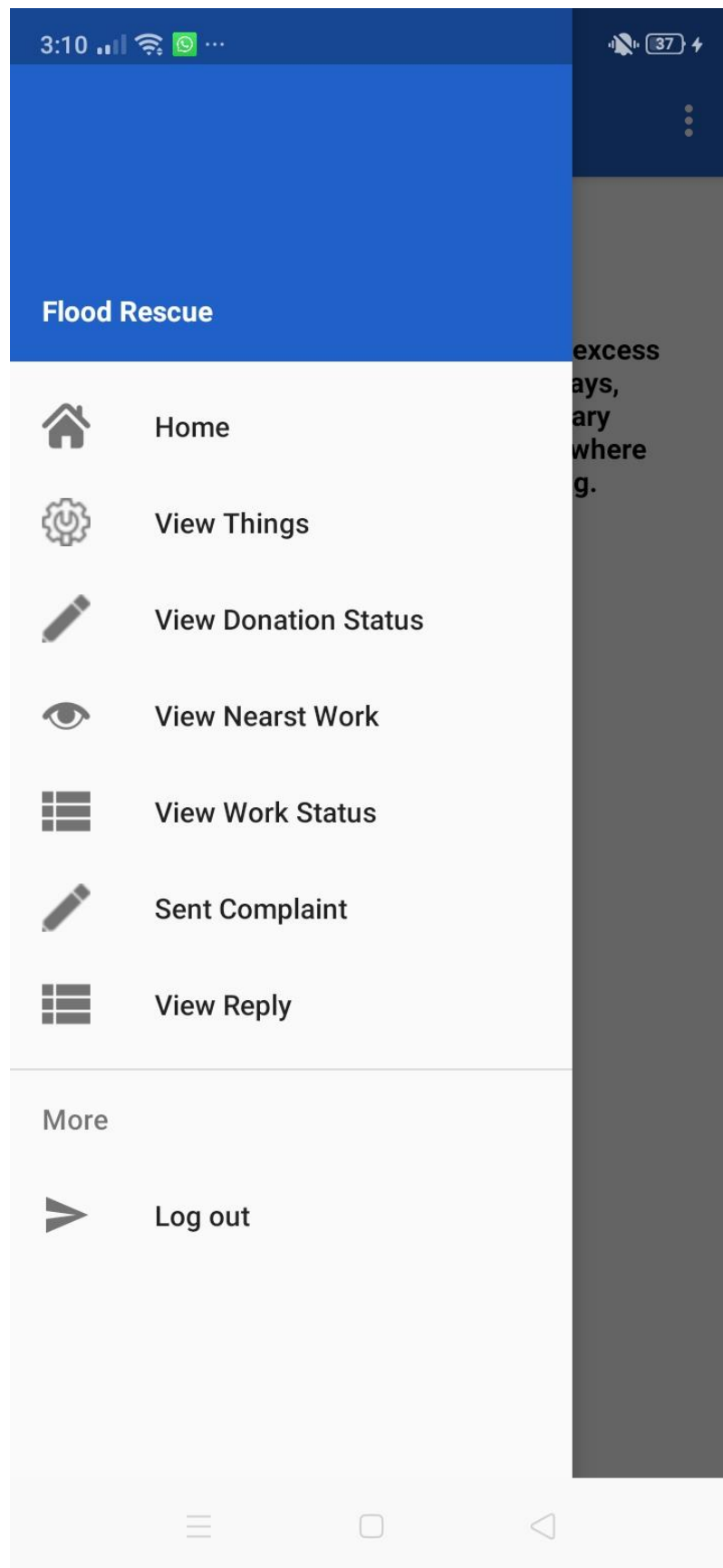
.....

LOG IN

[New User...? Register Now](#)

[New Volunteer..? Register Now](#)

[Navigation Bar: Menu, Home, Back icons]



8. CONCLUSION

FLOOD RESCUE REHABILITATION CORNER

The “Kerala flood rescue and rehabilitation corner” is a website for the people those who are affected by flood and people in flood activities. We aimed to develop human sustainability in those crucial days, by providing certain facilities such as awareness help desk. People can access it from anywhere of the country.

In this project people who are affected by the flood can be minimized and provides sufficient resources to the needy.

9. BIBLIOGRAPHY

- : www.researchgate.net
- www.ijcsmc.com
- www.github.com
- www.techtarget.com

10. CODING

FLOOD RESCUE REHABILITATION CORNER

```
package com.example.flood_app;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;

public class Login extends AppCompatActivity implements
View.OnClickListener {
    EditText editText, editText2;
    Button button, button4;
    ImageView imageView3;
    TextView tt, ttk;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        editText = (EditText) findViewById(R.id.e1);
        editText2 = (EditText) findViewById(R.id.e2);
        button = (Button) findViewById(R.id.button);
        tt = (TextView) findViewById(R.id.textView5);
        ttk = (TextView) findViewById(R.id.textView5);

        button.setOnClickListener(this);
        ttk.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {

        if (v == button) {
```



```

final String uname = editText.getText().toString();
final String password = editText2.getText().toString();

if (uname.length() == 0) {
    editText.setError("Please enter the Username");
} else if (password.length() == 0) {
    editText2.setError("Please enter the password");
} else {

    SharedPreferences sh =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
    String hu = sh.getString("ip", "");
    String url = "http://" + hu + ":5000/android_login_post";

    RequestQueue requestQueue =
Volley.newRequestQueue(getApplicationContext());
    StringRequest postRequest = new
StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                // Toast.makeText(getApplicationContext(),
response, Toast.LENGTH_LONG).show();

                // response
                try {
                    JSONObject jsonObj = new
JSONObject(response);

                    if
(jsonObj.getString("status").equalsIgnoreCase("ok")) {
                        String type =
jsonObj.getString("type");

                        SharedPreferences sh =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
                        SharedPreferences.Editor ed =
sh.edit();

                        ed.putString("lid",
jsonObj.getString("id"));

                        ed.commit();

                        if (type.equalsIgnoreCase("donor"))
{

                            Intent iri=new
Intent(getApplicationContext(), LocationService.class);
                            startService(iri);

```

```

Intent ii=new
Intent(getApplicationContext(),Donorhome.class);
startActivity(ii);

//

}

if
(type.equalsIgnoreCase("volenteer")) {

Intent ii=new
Intent(getApplicationContext(),Volunterhome.class);
startActivity(ii);

//
Intent i = new
Intent(getApplicationContext(), Agenthomepage.class);
startActivity(i);
//

} else {
Toast.makeText(Login.this, "Invalid
username or password", Toast.LENGTH_SHORT).show();
}

// }

} catch (Exception e) {
Toast.makeText(getApplicationContext(),
"Error" + e.getMessage().toString(), Toast.LENGTH_SHORT).show();
}
},
new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error)
{
// error
Toast.makeText(getApplicationContext(),
"eeeeee" + error.toString(), Toast.LENGTH_SHORT).show();
}
}
) {
@Override
protected Map<String, String> getParams() {
SharedPreferences sh =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
Map<String, String> params = new HashMap<String,

```

```

String>();

        params.put("name", uname);
        params.put("pass", password);

        return params;
    }
};

int MY_SOCKET_TIMEOUT_MS = 100000;

postRequest.setRetryPolicy(new DefaultRetryPolicy(
    MY_SOCKET_TIMEOUT_MS,
    DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
    DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
requestQueue.add(postRequest);

    }
}

if(v==tt){

    Intent ii=new
Intent(getApplicationContext(),Userregistration.class);
    startActivity(ii);

}

if(v==ttk){

    Intent ii=new
Intent(getApplicationContext(),Volregistration.class);
    startActivity(ii);

}

}

}

```