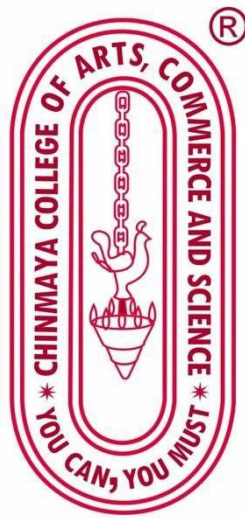


CHINMAYA COLLEGE OF ARTS, COMMERCE AND SCIENCE

Layam Road, Thrippunithura – 682301

(Affiliated to Mahatma Gandhi University, Kottayam)



BACHELOR OF COMPUTER APPLICATIONS

MAIN PROJECT

ON

STORMLIGHT : DISASTER MANAGEMENT SYSTEM

Submitted By,

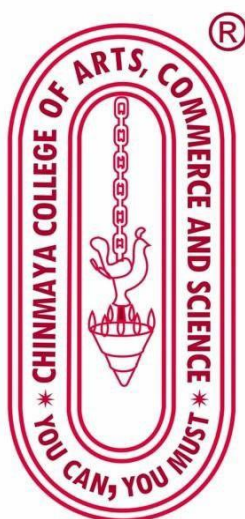
SREERAJ S CHANDRAN

Reg. No.: 210021088687

CHINMAYA COLLEGE OF ARTS, COMMERCE AND SCIENCE

Layam Road, Thrippunithura – 682301

(Affiliated to Mahatma Gandhi University, Kottayam)



CERTIFICATE

This is to certify that the Main Project Report entitled

STORMLIGHT : DISASTER MANAGEMENT SYSTEM

has been submitted by

SREERAJ S CHANDRAN

Reg. No : 210021088687

in partial fulfilment of the requirements for the award of the degree

BACHELOR OF COMPUTER APPLICATIONS

MAHATMA GANDHI UNIVERSITY

During the academic year 2023-2024

Submitted for the University Examination held on

Principal

Head of the Department

External Examiner

Project Guide

NGP/EKM/23-55/308

13 March,2024

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Sreeraj S Chandran** sixth semester BCA student of Chinmaya College of Arts & Science, Thripunithura has successfully completed his internship program in **"Python-Django"**.

He has completed his internship program (**January 01, 2024 to March 12, 2024**) under the guidance of Mr. Anish Mathew Abraham in partial fulfilment of the requirements for the award of the degree of Bachelors in Computer Applications. During the period of his internship program with us, he was found diligent, hardworking and inquisitive.

We wish him every success in his life and career.

Yours Truly,

For,

NextGenPro Innovations & Edupark Pvt Ltd



Shabna Sara Jithin
HR Manager

ACKNOWLEDGEMENT

By blessing and permission of Almighty God, I was able to complete this work successfully. My sincere thanks to our principal, **Dr. Smitha Harikumar** for her overwhelming and moral support extended towards us.

I would like to thank our head of the department **Mrs. Nisha Sanjay** for her constant encouragement and support for the completion of our project.

I would like to express gratitude to **Mr. Vishnu Mohanan (Project guide)**, for his valuable guidance and support.

I would also like to express my sincere thanks to all my teachers, **Mrs. Sharmila Francis, Mrs. Ranimol V G, Mrs. Remilda Rajan and Mrs. Andal V** for their timely assistance and advice offered to us to make this main project a success.

Finally, I thank my parents for their boundless support and for making our lives so easy and for helping to tackle all those difficulties in life.

DECLARATION

I, **SREERAJ S CHANDRAN**, hereby declare that the Main Project entitled **STORMLIGHT : DISASTER MANAGEMENT SYSTEM** submitted to **Mahatma Gandhi University, Kottayam** in partial fulfilment of the requirements for the **Bachelor's degree in Computer Applications** is a record of original work done by me during the period of study at Chinmaya College Of Arts, Commerce And Science, Thrippunithura under the supervision and guidance of **Mr. Vishnu Mohanan (Project Guide)**, Department of Computer Applications and that this project work has not formed the basis for the award of any diploma/associates-ship/fellowship or similar title to any candidate of any university.

Place: Thrippunithura

SREERAJ S CHANDRAN

Date:

Reg. No.: 210021088687

SYNOPSIS

Stormlight is a web-based disaster management system designed to bridge the gap between those in need during emergencies and the volunteers who can help. It provides a central platform for reporting disasters, coordinating volunteers, and improving overall response efficiency.

Anyone can report a disaster, including public incidents (fires, floods) or private emergencies (medical emergencies, building collapses). They can provide details about the incident and its severity. Individuals willing to assist in disaster relief efforts can register on the platform. This allows them to be considered for assignments based on their location and any specific skills they possess (medical training, search and rescue experience etc.). Authorized personnel (disaster management agencies, NGOs) can review submitted disaster reports. They can verify the details, assess the urgency, and assign registered volunteers to suitable tasks based on their proximity and expertise.

Reporting Options: Users can report disasters through a user-friendly interface.

Incident Details: Report forms allow users to provide critical information about the disaster, such as the nature of the incident, time of occurrence, and the number of people potentially affected.

Location Sharing:

- **Preferred Method:** For faster and more accurate response, Stormlight encourages users to share their location via a Google Maps link. This pinpoints the exact location of the incident on a map, streamlining rescue efforts.
- **Alternative Method:** If a Google Maps link is unavailable, Stormlight can automatically detect the user's location using their IP address. This provides an approximate location to guide initial response efforts.

Individuals willing to volunteer can register on the platform. During registration, they can provide their contact information, location availability, and any relevant skills or experience that could be helpful in disaster situations (e.g., first aid certification, structural engineering knowledge).

Once volunteers are assigned to a disaster report, the system updates the dashboard to reflect this. This provides transparency to all involved parties (admins, volunteers, reporters) regarding the ongoing response efforts.

The front-end of the project is HTML/CSS/JavaScript and backend is Python/MySQL. Hypertext Mark-up Language (HTML) is the standard mark-up language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage collected. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the documents.

TABLE OF CONTENTS

1 Introduction	1
1.1 Project Overview	1
2 System Analysis	2
2.1 Problem Analysis	3
2.1.1 Existing System	3
2.1.2 Proposed System	5
2.1.3 Feasibility System	5
2.1.3.1 Economic Feasibility	5
2.1.3.2 Technical Feasibility	6
2.1.3.3 Behavioral Feasibility	7
2.2 Requirement Specification	7
2.2.1 Software Requirement Specification	7
2.3 Hardware and Software Selection and Justification	8
2.3.1 Hardware Selection	8
2.3.2 Software Selection and Justification	9
2.3.3 About Software Tools and Platforms	9
2.4 Use Case Diagram	12
2.5 Data Flow Diagram	13
2.5.1 Zero Level Diagram	15
2.5.2 First Level Diagram	16
2.6 Entity Relationship Diagram	18
3 System Design	18
3.1 Structured Design Methodologies	19
3.2 User Interface Design	19
3.3 Output Design	20
3.4 Database Design	21
3.4.1 Data and Integrity Constraints	22
3.5 Table Design	23

4 Coding	24
4.1 Program Code Preparation	24
5 Implementation of Security	25
5.1 Data Security	25
5.2 User and Access	26
6 System Testing	26
6.1 Unit Testing	26
6.2 Integration Testing	27
6.3 User Acceptance Testing	27
6.4 Test Case Design	28
6.5 Test Report and Debugging	29
7 System Implementation and Testing	30
8 Scope of the Project	30
9 Future Enhancements	31
10 Conclusion	32
11 Bibliography	32
Appendix –	
I- 33	
Appendix-	
II- 40	

1.INTRODUCTION

1.1 PROJECT OVERVIEW

Stormlight is a web-based disaster management system designed to streamline communication and volunteer coordination during emergencies.

Target Users:

- Users: Anyone in need of assistance during a disaster.
- Volunteers: Individuals willing to help during emergencies.
- Admins: Authorized personnel from disaster management agencies or NGOs

Project Goals:

- Reduce Response Times: Facilitate faster emergency response by enabling accurate location sharing and real-time updates.
- Optimize Volunteer Coordination: Match volunteers with disaster reports based on location and expertise.
- Enhance Transparency: Improve communication and build trust through real-time information sharing about ongoing emergencies and volunteer assignments.

Key Functionalities:

- Disaster Reporting: Users can report public or private disasters with details and location (Google Maps link preferred or automatic IP detection).
- Volunteer Management: Volunteers can register and admins can assign them to tasks based on location and expertise.
- Real-Time Updates: A dashboard displays disaster reports and volunteer assignments for improved transparency.

Project Benefits:

- Saves lives and minimizes damage through faster response times.
- Improves efficiency of search and rescue efforts through optimized volunteer matching.
- Fosters better communication and builds trust through real-time updates.

Project Deliverables:

- A fully functional web-based disaster management system.
- User guides for reporters, volunteers, and admins.
- System documentation for future maintenance and development

2. SYSTEM ANALYSIS

Stormlight is a web-based disaster management system designed to streamline communication and volunteer coordination during emergencies. This report analyzes the system's functionalities, user roles, and potential benefits to identify its strengths and weaknesses.

Functionality

- **Disaster Reporting:** Users can report public or private disasters with details and location (Google Maps link preferred or automatic IP detection).
- **Volunteer Management:** Volunteers register and admins assign them to tasks based on location and expertise.
- **Real-Time Updates:** A dashboard displays disaster reports and volunteer assignments for improved transparency.

Strengths

- **Improved Response Times:** Accurate location sharing and real-time updates facilitate faster emergency response.
- **Efficient Volunteer Management:** Matching volunteers based on location and skills optimizes search and rescue efforts.
- **Enhanced Transparency:** Real-time information sharing fosters better communication between all parties.
- **Scalability:** The system can be scaled to accommodate larger regions and user bases.

Weaknesses

- **Technology Dependence:** Relies on internet connectivity for reporting and updates, potentially excluding those without access.
- **Data Security:** User privacy and sensitive information require robust security measures.
- **Volunteer Verification:** A system for verifying volunteer skills and qualifications could enhance trust.
- **Limited Functionality:** The system currently focuses on reporting and initial response. Expanding functionalities (resource management, communication tools) could further improve disaster response.

Threats

- **System Outage:** Technical failures could disrupt reporting and volunteer coordination during critical times.
- **Misuse of the System:** False reports or malicious actors could hinder response efforts.
- **Limited Volunteer Availability:** The system's effectiveness depends on a sufficient volunteer base.

2.1 PROBLEM ANALYSIS

Stormlight, while a valuable tool for disaster response, faces some challenges that need to be addressed to ensure its effectiveness. Here's a breakdown of the key problems.

Dependence:

The system relies on internet connectivity for reporting and updates, potentially excluding those without access in remote areas or during outages.

Data Security:

User privacy and sensitive information (location, volunteer skills) require robust security measures to prevent breaches.

Volunteer Verification:

Currently, there's no system to verify volunteer skills and qualifications, which could impact the effectiveness of assigned tasks.

Limited Functionality:

The system currently focuses on reporting and initial response. Expanding functionalities could further improve disaster response.

2.1.1 EXISTING SYSTEM

Stormlight joins a growing field of disaster management systems aiming to improve response and communication during emergencies. Here's a look at some existing systems and how Stormlight can position itself competitively:

Government portals: Many governments have established online platforms for disaster preparedness, alerts, and basic reporting.

Strengths: Official source of information, widely accessible.

Weaknesses: Limited functionality for user-generated reports and volunteer coordination.

Stormlight's Advantage: Focuses on user-driven reporting, volunteer coordination, and real-time updates.

Red Cross and NGO platforms: The Red Cross and other NGOs often have online platforms for disaster relief efforts.

Strengths: Extensive experience in disaster response, established volunteer networks.

Weaknesses: May be limited geographically or require affiliation with the specific organization.

Stormlight's Advantage: Open to a wider range of volunteers and reporters, promotes a community-based approach.

Commercial solutions: Private companies offer disaster management software to businesses and organizations.

Strengths: Often have advanced features for resource management and communication.

Weaknesses: May be cost-prohibitive for individual users or smaller communities, focus on enterprise needs.

Stormlight's Advantage: Free and accessible to the public, designed for both individual and community-based response.

Social media platforms: Social media plays a growing role in disaster response with real-time information sharing.

Strengths: Widespread reach, allows for rapid information dissemination.

Weaknesses: Unverified information, potential for misinformation and confusion, limited focus on volunteer coordination.

Stormlight's Advantage: Provides a structured platform for verified reporting, volunteer assignment, and facilitates focused communication during emergencies

2.1.2 PROPOSED SYSTEM

Stormlight, It outlines the system's functionalities, components, and architecture to provide a clear understanding of how it will operate.

- Disaster Reporting:
 - Users can report public or private disasters.
 - Report forms allow for details like incident type, severity, and number of people affected.
 - Location sharing is encouraged:
 - Preferred method: Users can share their exact location using a Google Maps link.
 - Alternative method: The system can automatically detect user location using IP address (approximate location).
- Volunteer Management:
 - Individuals can register as volunteers through the platform.
 - Volunteer profiles may include contact information, location availability, and relevant skills (e.g., first aid, search and rescue).
 - Authorized admins can review disaster reports and assign volunteers based on:
 - Location proximity to the disaster zone.
 - Reported needs of the incident (medical aid, structural assessment etc.).
 - Volunteer's expertise and skills.
- Real-Time Updates:
 - A designated dashboard displays a real-time feed of submitted disaster reports.
 - The dashboard also reflects volunteer assignments for each report, promoting transparency.

2.1.3 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, time and effort spent on it. Feasibility study lets the developer foresee the future of the project and its usefulness. Finding out whether a new system is required or not. The study is carried out to the best system that meet performance requirement. This entails identification, description and evaluation of candidate system and selection of the best system for the job. It simply identifies whether the propose system is feasible to the organization or not.

There are three aspects in the feasibility study portion of the preliminary investigation

- ✓ Economic feasibility
- ✓ Technical feasibility
- ✓ Behavioral feasibility

2.1.3.1 ECONOMIC FEASIBILITY

Economic analysis is most frequently used method for evaluating the effectiveness of a candidate system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. The Trade-offs involved under economic feasibility are:

Cost of Development: With all the hardware and software provided by the vendors. Selling of the software however does not depend upon the development cost, as the hardware and software bought and installed respectively for development purpose will remain with the developer for further development. The system is a web-based application where user need not develop any cost for hardware and all. The system requires an internet connection to work efficiently.

Only depreciation costs are to be considered for them. This costing varies due to the provision for negation and also as at this initial phase of development this is a rough estimation of cost.

Cost for Maintenance after Implementation: Cost of maintenance is an important issue for the developer developing the system as well as for the organization using the system after development User friendliness is no doubt beneficial; provision of help file is also no doubt beneficial but the main thing to be taken into consideration is that whether the users are capable of using the system.

If the users are new to the world of computers it will be tough for them to understand the workings of the system how much user friendly or well documented it is. Thus, the user training is always necessary and a fixed minimum amount of cost must be kept as provision for them. This cost effects the overall costing of the project depending upon the way the training is given. Maintenance cost thus plays an important role to the developer.

Cost & Profit Ratio: Development of a system is much easier than convincing an organization to use it. Thus cost/profit analysis plays a very important role in determining the economic feasibility of a project.

After a careful analysis of the Economic Feasibility, it could be concluded that treating the project is "Economically Feasible" for development.

2.1.3.2 TECHNICAL FEASIBILITY

The system must be evaluated from technical viewpoint first. The assessment of this feasibility must be based on outline design of the system requirement in the terms of input, output, programs and procedure having identified an outline system, the investigation must go on to suggest the type of equipment, required method of the developing the system method of running the system once it has been designed. The project should be developed such that the necessary functions and performance are achieved within the constraints. The project is developed with latest technology. There are only minimal constraints involved in this project. In the proposed system the data can be easily stored and managed using data base management system software. And for development of project, it uses Python as its front end and MySQL as its backend.

Therefore, the system is technically feasible.

2.1.3.3 BEHAVIOURAL FEASIBILITY

In behavioral feasibility the entire application is checked whether the system will be use if it is developed and implemented. Also, it is checked whether there will be resistance from user that may undermine the possible application benefits. There is no barrier for implementing the system. The system also helps to access the information immediately as need arises. The proposed system has a user-friendly interface and the user can easily get item details, rent amount, get rental instruments. User with minimum internet knowledge can easily access the system, register themselves and can view the various brands items. Thus, the system is found to be operational feasible.

2.2 REQUIREMENT SPECIFICATION

Requirement analysis involves studying the current system to find out how it works and where improvements could be made. A clear idea about the existing system is a must for making improvements where it is required. Proper planning and collection of data serves the purpose.

The requirement engineering process should normally involve writing requirements definition and then expanding this into a requirement specification. The requirement definition is targeted at a managerial level and requirements specification at the technical staff. The software design is based directly on the requirements specification.

System analysis includes two main procedures. They are: -

1. Preliminary Analysis
2. Detailed Analysis

The preliminary analysis stage begins when someone encounters a problem or limitation in an existing system, desires a modification to the existing system. The modification may be either change in existing system or proposing an entire new system.

The detailed analysis expands the preliminary analysis to include a complete analysis of all possible alternative solutions to the problem and a complete explanation of what appears to be the most practical solution.

2.2.1 SOFTWARE REQUIREMENT SPECIFICATION

This documentation aims to define the overall software requirements for “Heal on You” efforts have been made to define the requirements accurately. The final product will be having only features mentioned in this document and assumptions for any additional feature should not be made by any of the parties involved in developing/testing/implementing/using this product. In case it is required to have some additional features, a formal change request will need to be raised and subsequently a new release of this document and product will be produced. This specification document describes the capabilities that will be provided by the software application “Salon book”. It also states the various required constraints by which the system will abide. The

intended audience for this document is the development team, testing team and end user of the product.

Heal on You is a web-based application designed to counter and eliminate problems encountered in the current manual operation of the business. The proposed work “Heal on You” that uses an online platform that makes the task of making an appointment from the doctor easy and reliable for the users

The system provides the user to perform their task in an easy and much less complex way to avoid redundancy. This system ensures that the users accessing the system can ensure maximum efficiency and they can depend on the system for desired results.

Major non-functional specifications:

- ❖ Performance: The system should be able to handle a large volume of data and user requests without experiencing significant downtime or lag. It should also be able to generate accurate stock price predictions within a reasonable timeframe.
- ❖ Reliability: Salon book should be reliable and consistently provide accurate predictions and insights. The system should also be regularly updated and maintained to ensure continued performance and accuracy.
- ❖ Usability: Salon book should be easy to navigate and use, even for users without advanced knowledge of internet. It should also provide clear and concise explanations of its predictions and metrics.
- ❖ Security: The system should be designed with security in mind, including measures to prevent unauthorized access and protect user data. This includes using encryption and secure authentication methods.
- ❖ Scalability: The system should be scalable, meaning it should be able to handle increased demand as the user base grows. This includes ensuring that the system architecture is designed for expansion and that resources can be easily added as needed.
- ❖ Compatibility: Salon book should be compatible with a wide range of devices and web browsers to ensure that users can access the platform from anywhere with an internet connection.

2.3 HARDWARE AND SOFTWARE SELECTION AND JUSTIFICATION

2.3.1 HARDWARE SELECTION

The selection of hardware is very important in the existence and proper working of any software. When selecting hardware, the size and capacity requirements are also important. Below are some of the hardware that is required by the system.

Processor: Intel core i3 and above

RAM: Minimum 2 GB RAM

Hard Disk Space: 100 GB

Input Devices: Mouse, Keyboard

Output Devices: Monitor

2.3.2 SOFTWARE SELECTION AND JUSTIFICATION

We require much different software to make the application which is in making to work efficiently. It is very important to select the appropriate software so that the software works properly. Below are the software requirements.

Operating System: Windows 7 or higher versions

Front End: HTML, JavaScript, CSS

Back End: Python (Django Framework), MySQL

Web Browser: Internet Explorer/Google Chrome/Firefox

Web Server: Wamp Server

2.3.3 ABOUT SOFTWARE TOOLS AND PLATFORMS

WINDOWS 10

Windows 10 is a major release of the Windows NT operating system developed by Microsoft. It is the successor to Windows 8.1, which was released nearly two years earlier, and itself was released to manufacturing on July 15, 2015, and broadly released for the general public on July

29, 2015. Windows 10 was made available for download via MSDN and TechNet, as a free upgrade for retail copies of Windows 8 and Windows 8.1 users via the Windows Store, and to Windows 7 users via Windows Update. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users, in addition to additional test builds of Windows 10, which are available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

Microsoft initially aimed to have Windows 10 installed on over one billion devices within three years of its release; that goal was ultimately reached almost five years after release on March 16, 2020. By January 2018, Windows 10 surpassed Windows 7 as the most popular version of Windows worldwide. As of December 2021, it is estimated that 82% of Windows PCs, 61% of all PCs (the rest being older Windows versions and other operating systems such as macOS and Linux), and 27% of all devices (including mobile,

tablet and console) are running Windows 10. On June 24, 2021, Microsoft announced Windows 10's successor, Windows 11, which was released on October 5, 2021.

Python - Django Framework

Python is a high-level, object-oriented programming language that is widely used for web development, scientific computing, data analysis, and artificial intelligence applications. One of the most popular web frameworks for Python is Django, which is an open-source, ModelView-Controller (MVC) architecture web framework that allows for the rapid development of web applications.

Django has several characteristics that make it a popular choice for web development, including:

- DRY (Don't Repeat Yourself) principle: Django follows the DRY principle, which means that developers only need to write code once for a particular task, rather than repeating it in multiple places. This reduces the amount of code needed to develop a web application and makes it easier to maintain.
- Batteries included: Django includes many built-in features and libraries that make it easier to develop complex web applications. This includes features like authentication, admin interface, and automatic database schema generation.
- ORM (Object Relational Mapping): Django uses an ORM to interact with the database, which means that developers can work with database objects as if they were Python objects. This makes it easier to write code that interacts with the database and reduces the likelihood of SQL injection attacks.
- MVC architecture: Django follows the Model-View-Controller (MVC) architecture, which separates the application logic into three separate components: the model (data), the view (presentation), and the controller (logic). This separation of concerns makes it easier to maintain and test web applications.
- URL routing: Django includes a powerful URL routing system that allows developers to map URLs to specific views in the application. This makes it easy to create complex URLs and to handle URL-based logic.

MYSQL

MySQL was created by a Swedish company, MySQL AB, founded by David Axmark, Allan Larsson and Michael "Monty" Widenius. Original development of MySQL by Widenius and Axmark began in 1994. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from MySQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface, while keeping the same API as MySQL. By keeping the API consistent with the MySQL system, many developers were able to use MySQL instead of the (proprietary licensed) MySQL antecedent.

MySQL can be built and installed manually from source code, but it is more commonly installed from a binary package unless special customizations are required. On most Linux distributions, the package management system can download and install MySQL

with minimal effort, though further configuration is often required to adjust security and optimization settings.

Though MySQL began as a low-end alternative to more powerful proprietary databases, it has gradually evolved to support higher-scale needs as well. It is still most commonly used in small to medium scale single-server deployments, either as a component in a LAMP-based web application or as a standalone database server. Much of MySQL's appeal originates in its relative simplicity and ease of use, which is enabled by an ecosystem of open-source tools such as phpMyAdmin. In the medium range, MySQL can be scaled by deploying it on more powerful hardware, such as a multi-processor server with gigabytes of memory.

There are, however, limits to how far performance can scale on a single server ('scaling up'), so on larger scales, multi-server MySQL ('scaling out') deployments are required to provide improved performance and reliability. A typical high-end configuration can include a powerful master database which handles data write operations and is replicated to multiple slaves that handle all read operations. The master server continually pushes bin-log events to connected slaves so in the event of failure a slave can be promoted to become the new master, minimizing downtime. Further improvements in performance can be achieved by caching the results from database queries in memory using Memcached, or breaking down a database into smaller chunks called shards which can be spread across a number of distributed server clusters.

2.4 USE CASE DIAGRAM

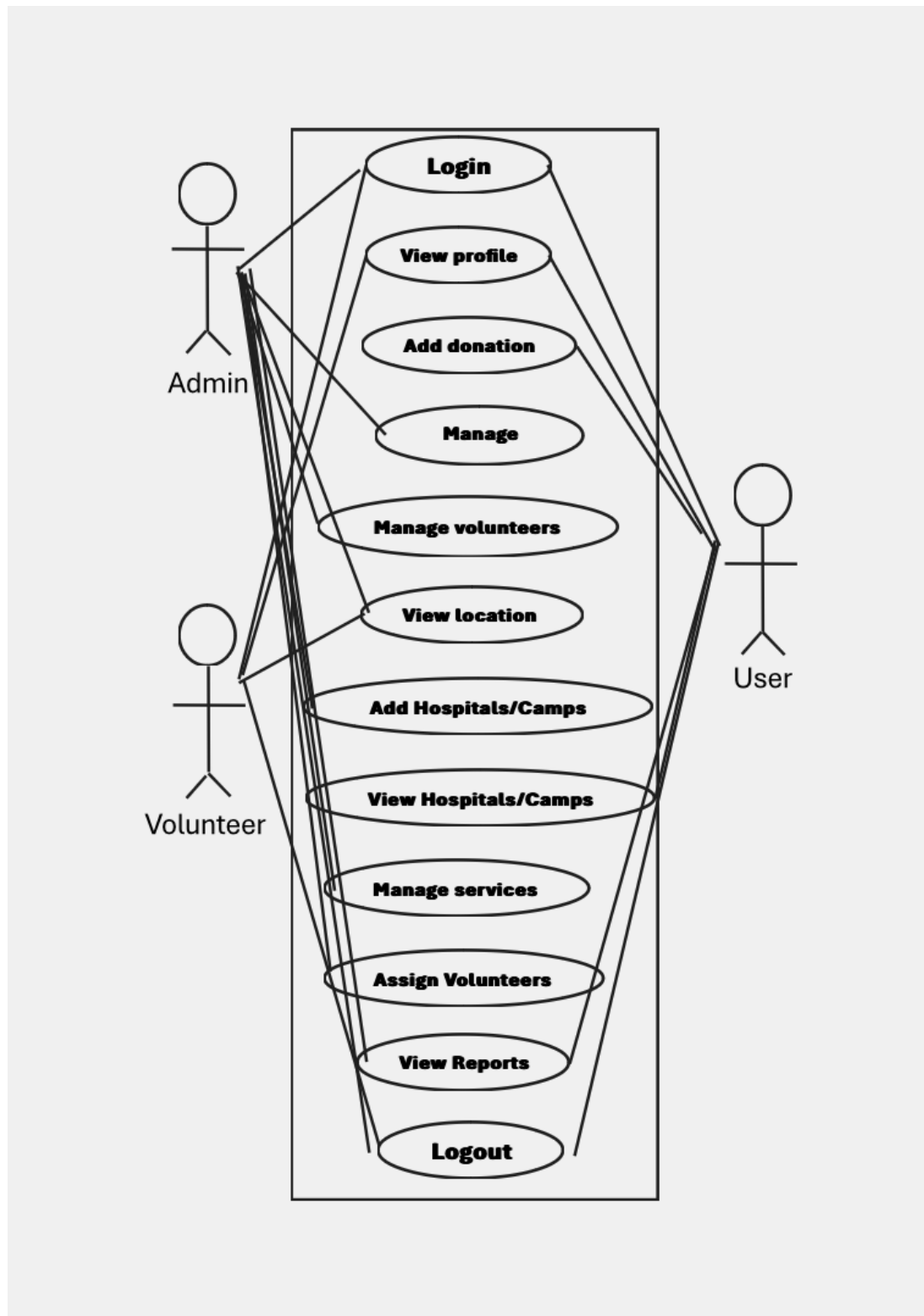


Fig 2.1 Use Case Diagram

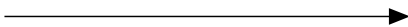
2.5 DATAFLOW DIAGRAM

The DFD also known as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data and the output data generated by the system.

Data flow diagram (DFD) is used to show how data flows through the system and the processes that transform the input data into output. Data flow diagrams are a way of expressing system requirements in a graphical manner. Four simple notations are used to complete a DFD.

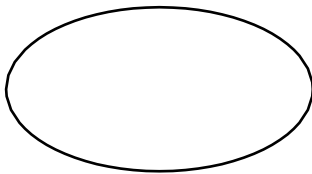
The notations are given below:

DATA FLOW



A directed arc or an arrow is used as a Data Flow Symbol. This represents the data flow occurring between two processes or between an external entity and a process; in direction of the Data Flow Arrow. Data flow Symbols are annotated with corresponding data names.

PROCESS



A function is represented using a circle. This symbol is called a process or bubble. Bubbles are annotated with the names of corresponding functions.

EXTERNAL ENTITY



An external entity such as a user, project manager etc. is represented by a rectangle. The external entities are essentially those physical entities external to the application system, which interact with the system by inputting data to the system or by consuming the data produced by the system. In addition to the human users the external entity symbols can be used to represent external hardware and software such as application software.

DATA STORE



A Data Store represents a logical file; it is represented using two parallel lines. A logical file can represent either Data Store Symbol, which can represent either data structure or a physical file on disk. Each data store is connected to a process by means of a Data Flow Symbol. The direction of the Data Flow Arrow shows whether data is being read from or written into a Data Store. An arrow flowing in or out of a data store implicitly represents the entire area of the Data Store and hence arrows connecting to a data store need not be annotated with the names of the corresponding data items.

2.5.1 ZERO LEVEL DIAGRAM

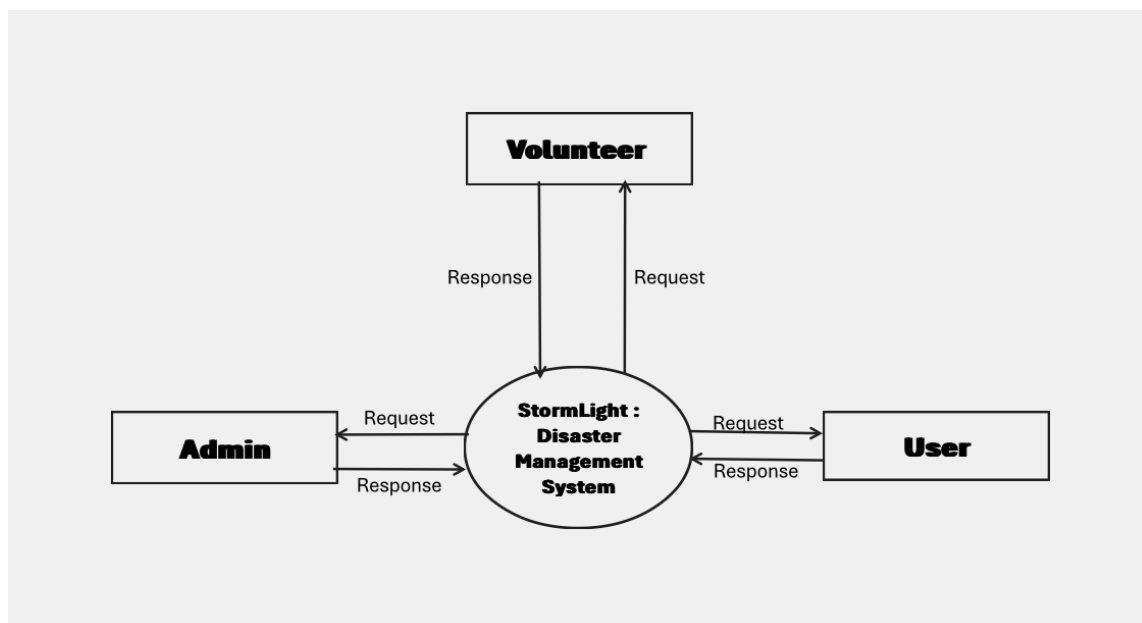


Fig 2.2 Zero Level DFD

2.5.2 FIRST LEVEL DIAGRAM

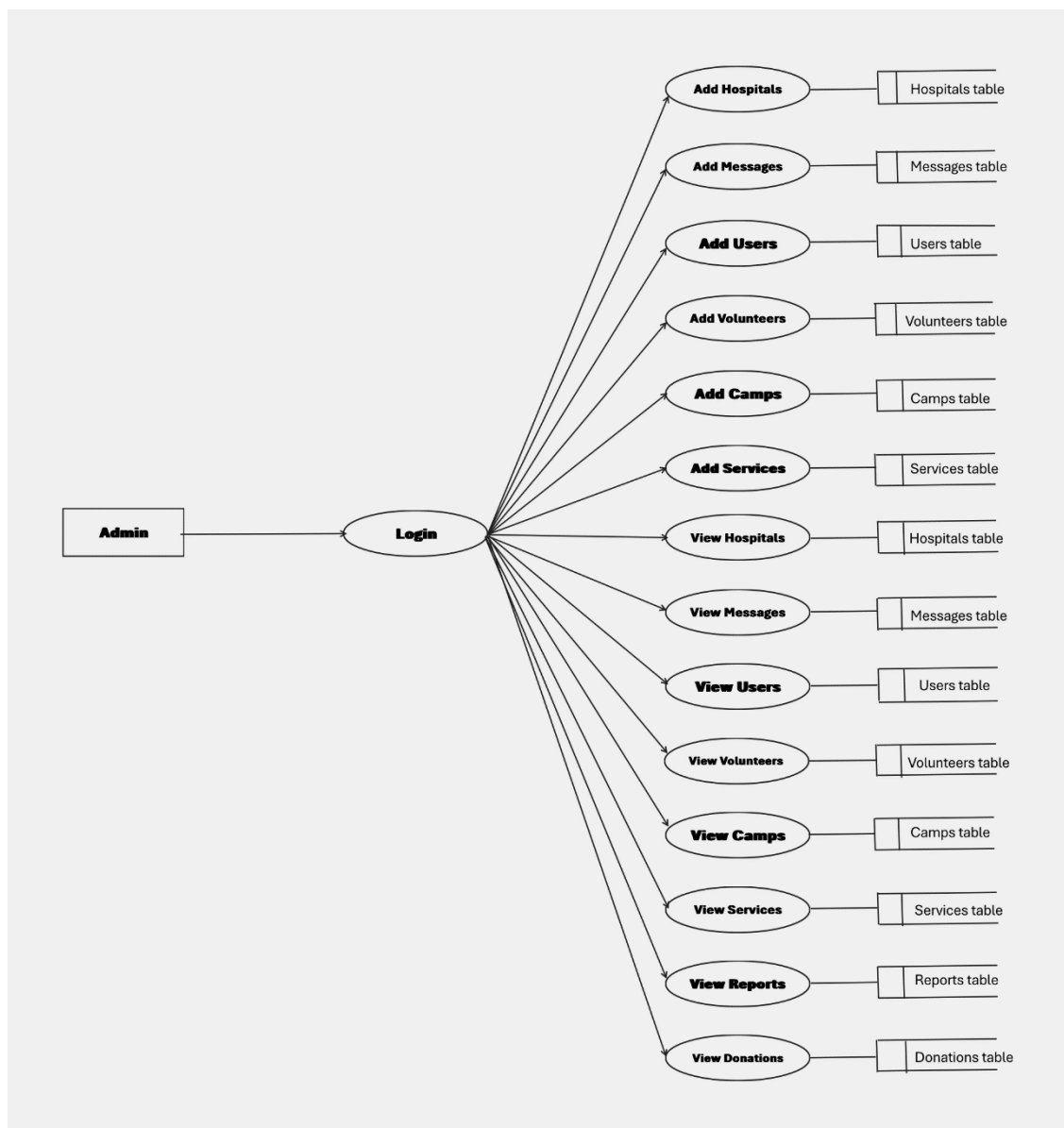


Fig 2.3 First Level DFD (Admin)

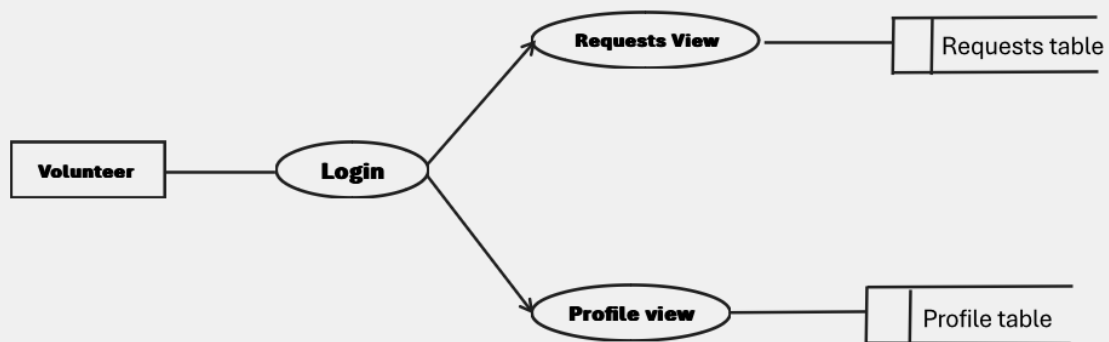


Fig 2.4 First Level DFD (Volunteer)

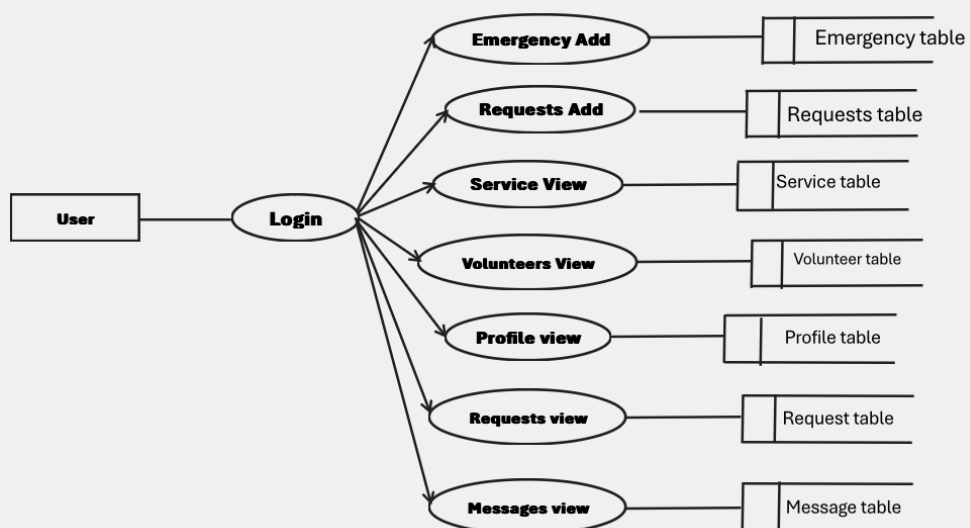


Fig 2.6 First Level DFD (User)

2.6 ER DIAGRAM

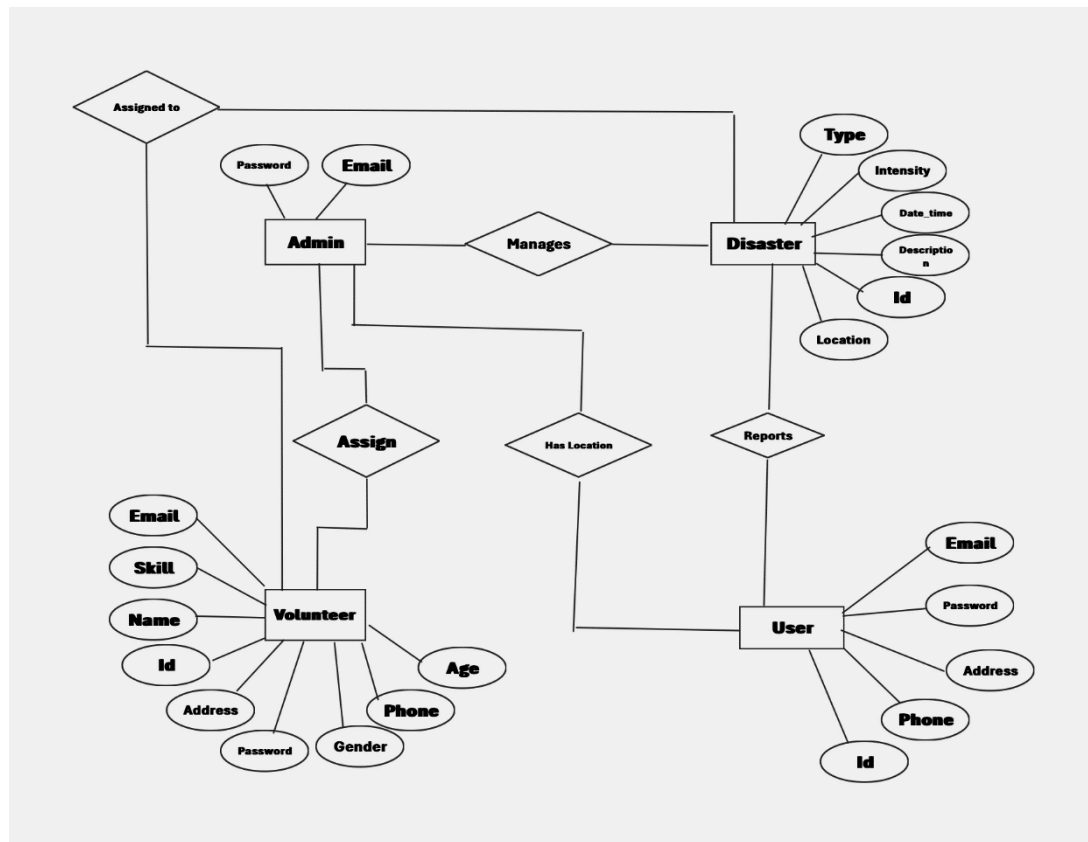


Fig 2.7 ER Diagram

3.SYSTEM DESIGN

The most creative and challenging phase of the system development is system design, is a solution to how to approach to the creation of the proposed system. It refers to the technical specification that will be applied. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Design goes through the logical and physical stages of development. At an early stage in designing a new system, the system analyst must have a clear understanding of the objectives, which the design is aiming to fulfil. The first step is to determine how the output is to be produced and in what format. Second, input data and master files (database) have to be designed to meet the requirements of the proposed output. The operational (processing) phases are handled through program construction and testing.

It is to be developed in a way that two users can access Admin and Customer with different permissions. It is a solution to a “how to” approach compared to system analysis which is a “what is” orientation. It translates the system requirements into ways of making them operational. The design phase focuses on the detailed implementation of the system recommended in the feasibility study.

3.1 STRUCTURED DESIGN METHODOLOGIES

Design methodology refers to the development of a system or method for a unique situation. Design methodology stresses the use of brainstorming to encourage innovative ideas and collaborative thinking to work through each proposed idea and arrive at the best solution. Meeting the needs and requirements of the end user is the most critical concern. To employ design methodology, various analyses and testing have been done so as to meet the desired user needs. Every input that the user input is being tested in this software. That means the validity of each data is being checked and if found invalid necessary warning and prompting messages are displayed. The output forms are also tested in detail to see whether the desired output is met or not. Also, the output forms are made clearer and more meaningful for the user to understand.

3.2 USER INTERFACE DESIGN

User interface design is one of the major functions in developing a system. It is a good understanding of the user needs very clearly. Because the user is the person who has to interact with system being developed. So that it should seek the needs of the user before developing it.

The system is designed in a very user-friendly manner that makes the user with little knowledge of computer and of the organization can work very easily with the system.

The input forms that are used to enter the data are made clearer and easier to understand. Every time the user enters data the system is designed to check the validity of the data and if found as invalid meaningful prompting and warning messages are displayed. This

makes the user comfortable to interact with the system. Also, when a user login to the system it checks the username and password entered to see whether it is valid user or not. It ensures security of the system and database. The data storage and data processing are made more efficient so that accurate results are being displayed on the output forms. And also, the retrieval of specific records as demanded by the user is made very faster that saves the user time.

3.3 OUTPUT DESIGN

The output is the most important and direct source of information to the user. The output should be provided in a most efficient formatted way. The output design has been done so that the results of process should be communicated to the user. Effective output design will improve the clarity and performance of outputs. Output is the main reason for developing the system and the basis on which they will evaluate the usefulness of the application. Output design phase of the system is concerned with the convergence of the 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. The various types of outputs required by most systems are:

☐ External Output:

Whose destination is outside the organization and which require special attention.

☐ Internal Output:

Whose destination is within the organization and which require careful design because they are user's main interface with the computer.

☐ Operational Output:

Whose use is purely within the computer department,

☐ Interactive Output:

Which involves the user in communicating with the computer. Specific output design is shown in APPENDIX II.

3.4 DATABASE DESIGN

Database design is one of the most important parts of the system design phase. In a database environment common data are available and are used by several users. Instead of each program managing its own data, authorized users share data across application with the database software managing the data as an entity. The primary objective of a database design is fast response time to enquiries, more information at low cost, control of redundancy, clarity and ease of use, date and program independence, accuracy and integrity of the system, fast recovery and availability of powerful end-user languages. The theme behind a database is to handle information as an integrated whole thus the main objective is to make information as access easy, quick, inexpensive and flexible for the users.

Data directory specifies the major element in the system, and care should be taken while designing, in order to avoid unnecessary duplication of data. The entire package depends on how the data are maintained in the system. Several tables are maintained in the system

to store data that are required for the processing of various data as well as storing intermediate or final processed results.

Database design mainly aims at handling large volumes of information, involving the definitions for the structure of storage and provisions for the manipulation of information, providing safety of information despite of system crashes due to unauthorized access. Some conditions are satisfied in database design stage.

3.4.1 DATA AND INTEGRITY CONSTRAINTS

The primary of a database design is fast response time to inquiries, more information at low cost, control of redundancy, clarity and ease of use, accuracy and integrity of the system, fast recovery and availability of powerful end-user languages. The theme behind a database is to handle information as an integrated whole thus the main objective is to make information as access easy, quick, inexpensive and flexible for the users. In this project, we mainly concentrated into relational databases.

Relational database stores data in tables, which is turn, are composed of rows also known as records, columns also known as fields. The fields in the relational model are: -

Primary Key:

The key which is uniquely identify records. They also notify the not null constraints.

Foreign Key:

The key which references the primary key, is the data inserted in the primary key column of the table.

Normalization:

After the conceptual level, the next level of process of database design to organize the database structure into a good shape called Normalization. The different normal forms obtained during the database design are given below:

In the database design, we create a database with different tables that is used to store the data. We normalize the data in the table. Database normalization is the process of organizing the fields and tables in a relational database to minimize redundancy and dependency. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

In the project I have made used of the 3rd normal form, Third Normal Form (3NF) is a property of database tables. A relation is in third normal form if it is in Second Normal Form and there are no functional (transitive) dependencies between two (or more) non primary key attributes. The overall objective in the development of database technology has been to treat data as an organizational resource and as an integrated whole. Database Management System allows data to be protected and organized separately from other resources. Database is an integrated collection of data. This is the difference between logical and physical data.

3.5 TABLE DESIGN

userreg

FIELD NAME	DATATYPE	NULL	DEFAULT
id	tinyint(4)	YES	NULL
name	varchar(300)	YES	NULL
email	varchar(300)	YES	NULL
address	varchar(300)	YES	NULL
phone	varchar(300)	YES	NULL
password	varchar(300)	YES	NULL

emc_servicerequest

FIELD NAME	DATATYPE	NULL	DEFAULT
uid	tinyint(4)	NO	NULL
e_uid	varchar(12)	No	NULL
District	varchar(12)	No	NULL
city	varchar(12)	No	NULL
maplink	varchar(12)	YES	NULL
intensity	varchar(12)	NO	NULL
image	varchar(12)	YES	NULL
message	varchar(12)	YES	NULL
service	varchar(12)	NO	NULL
status	varchar(12)	NO	NULL
catogory	varchar(12)	NO	NULL
Ip_contry	varchar(12)	YES	NULL
Ip_regionName	varchar(12)	YES	NULL
Ip_city	varchar(12)	YES	NULL
Ip_zip	varchar(12)	YES	NULL
Ip_lat	varchar(12)	YES	NULL
Ip_zip	varchar(12)	YES	NULL
Ip_ipaddr	varchar(12)	YES	NULL

donations

FIELD NAME	DATATYPE	NULL	DEFAULT
id	tinyint(4)	YES	NULL
name	bigint(20)	YES	NULL
type	varchar(17)	YES	NULL
quantity	varchar(15)	YES	NULL
phone	tinyint(4)	YES	NULL
place	tinyint(4)	YES	NULL
u_id	tinyint(4)	YES	NULL

camps

FIELD NAME	DATATYPE	NULL	DEFAULT
id	tinyint(4)	YES	NULL
name	varchar(18)	YES	NULL
date	bigint(20)	YES	NULL
phone	varchar(14)	YES	NULL
place	varchar(71)	YES	NULL
map	mediumint(9)	YES	NULL

chats

FIELD NAME	DATATYPE	NULL	DEFAULT
id	tinyint(4)	YESS	NULL
cus_id	varchar(0)	YESS	NULL
admin	varchar(34)	YESS	NULL
cus_msg	tinyint(4)	YESS	NULL
admin_msg	varchar(0)	YESS	NULL

volregg

FIELD NAME	DATATYPE	NULL	DEFAULT
id	tinyint(4)	YESS	NULL
name	varchar(400)	YESS	NULL
email	varchar(300)	YESS	NULL
address	varchar (20)	YESS	NULL
state	varchar(23)	YESS	NULL
phone	varchar(16)	YESS	NULL
age	varchar(60)	YESS	NULL
gender	varchar (400)	YESS	NULL
service	varchar (400)	YESS	NULL
password	varchar (400)	YESS	NULL

4. CODING

4.1 PROGRAM CODE PREPARATION

When considered as a step-in software engineering, coding is viewed as a natural consequence of design. However, programming language characteristics and coding style can profoundly affect software quality and maintainability. The coding step translates a detail design representation into a programming language realization. The translation process continues when a compiler accepts source code as input and produces machine-independent object code as output. The initial translation step in detail design to programming language is a primary concern in the software engineering context. Improper interpretation of a detail design specification can lead to erroneous source code. Style is an important attribute of source code and can determine the intelligibility of a program. The elements of a style include internal documentation, methods for data declaration, procedures for statement construction, and I/O coding and declaration. In all cases, simplicity and clarity are key characteristics. An offshoot of coding style is the execution time and/or memory efficiency that is achieved. Coding is the phase in which we actually write programs using a programming language. In the coding phase, design must be translated into a machine-readable form. If design is performed in a detailed manner, coding can be accomplished mechanistically. It was the only recognized development phase in early or unsystematic development processes, but it is just one of several phases in a waterfall process. The output of this phase is an implemented and tested collection of modules. In developing Salon book, I have utilized an array of cutting-edge technologies to create a robust and user-friendly web application. As the primary programming language, I have leveraged Python to establish the application's backend functionality. To expedite the development process and ensure a seamless user experience, I have incorporated the Django web framework to design a sleek and intuitive user interface. Finally, MySQL serves as the primary database for Salon book, ensuring secure and dependable data storage for user data and other pertinent information.

5. IMPLEMENTATION OF SECURITY

The software quality assurance is comprised of a variety of tasks associated with five major activities.

- Application of technical methods
- Conduct of formal technical reviews.
- Software testing
- Enforcement of standard
- Record keeping and recording

The quality begins with a set of technical methods and tools that help the analyst to achieve high quality specification and the designer to develop high quality design. The next activity involves assessment for quality for the design that is created which is the formal technical review. Software testing combines a multi-step strategy with a series of test case design methods that help to ensure effective error detection. To enforce data integrity, you can constrain or restrict the data values that users can insert, delete, or update in the database. Recordkeeping refers to the entire range of functions involved in creating and managing records throughout their life cycle. It includes: creating / capturing adequate records. Maintaining them in trustworthy recordkeeping systems for defined retention periods.

5.1 DATA SECURITY

The software maintains a well-organized database for storing the details that are provided by the user. This helps us to eliminate the entering of invalid data. Data is not accessible to unauthorized users. The system analyst will provide the test data, specially designed to show that the system will operate successfully in all its aspects and produce expected results under expected conditions. Preparation of test data and the checking of results should be carried out in conjunction with the appropriate users and operational departments. Also, the extent to which the system should be tested must be planned.

5.2 USER AND ACCESS RIGHTS

Admin: Authorized personnel from disaster management agencies or NGOs will have admin privileges

Volunteers: Individuals willing to help during emergencies can register as volunteers.

Users: Anyone can register as a reporter. They can Report a Disaster or can seek for help.

6.SYSTEM TESTING

System testing is the stage of implementation highly aimed at ensuring that the system works accurately and efficiently before the live operation commences. Testing is vital to the success of the system. The primary objective of testing is to derive a set of tests that has the highest likelihood for uncovering defects in then software. The system test in implementation should conform that all is correct and an opportunity to show the users that the system works as expected. It accounts the largest percentage of technical effort in the software development process. Testing phase in the development cycle validates the code against the functional specification.

There are mainly two approaches of testing namely, functional testing and structural testing. Functional testing is based on the functionality of the program and not the structure of the program. The test cases are solely on the basis of specification or requirements of the program. This type of testing is also called as black box testing. Structural testing is also called as white box testing or glass box testing. Here the internal structure of the program is tested. Test cases are designed by examining the logic of the program. “Salon book” focuses on the functionality of the system and hence it mainly does the functional/black box testing. The test cases of this system are completely based on the specifications of the system.

The application was tested and found to be working as expected. There was no abnormal behaviour reported during the testing of the program. Testing is a method by which we try reducing the testing efforts and bringing out the maximum output. Testing helps us in knowing whether the logical assumptions that we have taken for the system are correct, and if they are correct, we have obtained our goal. We test the system to know the errors, to check the validity of the information, to also group the modules with the aim that we meet the system requirements according to the system needs.

Testing is vital to the success of the system. System testing makes logical assumption that if all the parts of the system are correct, we have achieved the mission successfully. System testing is the stage of implementation that is aimed at assuring that the system works accurately and efficiently before the live operation commences.

Testing includes several levels of testing. They are:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

Client-Side Validation

Client-side validation is something that will happen on users' browser. The validation will occur before the data gets posted back to server. It is a good idea to have client-side validation as the user gets to know what needs to be changed immediately, i.e., no trips to servers are made.

JavaScript is most widely used to perform client-side validation.

Server-Side Validation

Server-side validation occurs at server. The benefit of having server-side validation is that if the user somehow bypasses the client-side validation (accidentally or deliberately), then we can catch the problem on the server side. So, having server-side validation provides more security and ensures that no invalid data gets processed by the application. Server-side validation is done by writing our custom logic for validating all the input.

The different types of testing are as follows:

6.1 UNIT TESTING

Unit testing is the first level of testing. In this process the code produced during the coding phase is verified. The goal is to test the internal logic of the modules. Each unit was found to be working satisfactorily. This testing is carried out during the programming stage itself. In this testing step, each module is found to be working satisfactorily as regards to the expected output from the module. Using a method called white box testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose and where each module or component of the software is tested individually. In the unit test case, we will be testing the separate modules of the Software. We will test the components by passing data through it and we will be monitoring data to find the errors. We will be looking for entry and exit conditions of the data. We will make sure that all the components work without any troubles.

Function Tested	Test Condition	Expected Results	Actual Results	Status
Username and Password	Invalid Username and Password	Access Denied	Access Denied	Pass
Logout Function	Invalid Infiltration with wrong credentials	Access Denied	Access Denied	Pass

6.2 INTEGRATION TESTING

After splitting the program into units, the units were tested together to see the defects between each module and function. It is testing two or more modules or functions together with the intent of finding interface defects between the modules or functions. Testing completed at as part of unit or functional testing, and sometimes, becomes its own standalone test phase. On a larger level, integration testing can involve putting together groups of modules and functions with the goal of completing and verifying that the system meets the system requirements.

6.3 USER ACCEPTANCE TESTING

Heal on You was tested by a small client community to see if the program met the requirements defined the analysis stage. It was found to be satisfactory. In this phase, the system is fully tested by the client community against the requirements defined in the analysis and design stages, corrections are made as required, and the production system is built. User acceptance of the system is key factor for success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with the users at the time of developing and making changes whenever required.

This was done regard to the following point:

- Input Screen Design
- Output Screen Design

6.4 TEST CASE DESIGN

Test case design refers to how you set-up your test-cases. It is important that your tests are designed well, or you could fail to identify bugs and defects in your software during testing. Designing good test cases ensure that every aspect of your software gets tested so that you can find and fix any issues.

Function Tested	Test Condition	Expected Result	Actual Result	Status
Name	Entered noncharacters	Not Allowed	Not Allowed	Pass
Phone Number	Entered more than 10 digits	Not Allowed	Not Allowed	Pass

Login	Invalid email id or password	Not Allowed	Not Allowed	Pass
E-mail	Entered invalid email	Not Allowed	Not Allowed	Pass

6.5 TEST REPORT AND DEBUGGING

Testing means verifying correct behaviour. Testing can be done at all stages of module development: requirements analysis, interface design, algorithm design, implementation, and integration with other modules. In the following, attention will be directed at implementation testing. Implementation testing is not restricted to execution testing. An implementation can also be tested using correctness proofs, code tracing, and peer reviews, as described below.

Debugging is a cyclic activity involving execution testing and code correction. The testing that is done during debugging has a different aim than final module testing. Final module testing aims to demonstrate correctness, whereas testing during debugging is primarily aimed at locating errors. This difference has a significant effect on the choice of testing strategies.

- Report error conditions immediately - Much debugging time is spent zeroing in on the cause of errors. The earlier an error is detected, the easier it is to find the cause. If it is not detected until the symptoms appear in the client interface then may be difficult to narrow down the list of possible causes.
- Maximize useful information and ease of interpretation - It is obvious that maximizing useful information is desirable, and that it should be easy to interpret. Ease of interpretation is important in data structures. Some module errors cannot easily be detected by adding code checks because they depend on the entire structure. Thus, it is important to be able to display the structure in a form that can be easily scanned for correctness.
- Minimize useless and distracting information - Too much information can be as much of a handicap as too little. If you have to work with a printout that shows entry and exit from every procedure in a module then you will find it very difficult to find the first place where something went wrong. Ideally, module execution state reports should be issued only when an error has occurred. As a general rule, debugging information that says "the problem is here" should be preferred in favour of reports that say "the problem is not here".
- Avoid complex one-use testing code - One reason why it is counterproductive to add module correctness checks for errors that involve the entire structure is that the code to do so can be quite complex. It is very discouraging to spend several hours debugging a problem, only to find that the error was in the debugging code, not the module under test. Complex testing code is only practical if the difficult parts of the code are reusable.

7.SYSTEM IMPLEMENTATION AND MAINTENANCE

Implementation is an activity that is contained throughout the development phase. It is the process of bringing a developed system into operational use and turning it over to the user. The new system and its components are to be tested in a structured and planned manner. A successful system should be delivered and users should have the confidence that the system would work efficiently and effectively. The more complex the system being implemented the more involved will be the system analysis and design effort required for implementation. Implementation is the stage of the system when the theoretical design is turned into working system. The implementation involves careful planning investigation of the current system and its constraints on implementing, design of methods to achieve the changeover, training of user over procedure and evaluation change over method.

There are three types of implementations:

1. Implementation of a computer system to replace a manual system. The problems involved are converting files, training users, creating accurate files, and verifying printouts for legacy.
2. Implementation of a new computer system to replace an existing one. This is usually a difficult conversion. If not properly planned, there can be many problems. Some larger systems have taken as long as a year to convert.
3. Implementation of a modified application to replace an existing one using the same computer. This type of conversion is relatively easy to handle, provided there are no major changes in files.

8.SCOPE OF PROJECT

Salon book aims to provide a user-friendly platform that allows customers to find best salons. The main objective of System is to design and develop an online web-based application to counter and eliminate problems encountered in the current manual operation of the business.

The proposed work “Salon book” that uses an online platform that makes the task of finding salon easier and can also book their service

Salon book also prioritizes data security by implementing user and access rights and using MySQL as a database to store user details. This ensures that customers' sensitive financial information is protected and only accessible by authorized users.

By prioritizing data security and using advanced technologies, Salon book system is a timesaving and efficient project. The System can use it to efficiently store all the data in a secure database. It is less prone to errors as the program checks the data entered before saving it to database. If it finds any data to be unsatisfactory it shows a warning to the user to correct the error.

9. FUTURE ENHANCEMENTS

Stormlight has a strong foundation for improving disaster response through user-driven reporting and volunteer coordination. Here are some potential future enhancements to consider:

- Expanding Functionalities
 - Resource Management: Integrate functionalities for managing critical supplies (food, water, medicine) during emergencies. This could involve:
 - An inventory system to track available resources.
 - A functionality for requesting and allocating resources based on disaster zone needs.
 - Communication Tools: Develop features to facilitate real-time communication during emergencies:
 - In-app chat or messaging system for volunteers and admins to collaborate.
 - Push notifications to alert volunteers about new assignments or updates.
 - Post-Disaster Recovery: Consider incorporating functionalities to support recovery efforts:
 - Damage assessment tools to gather data on infrastructure damage.
 - Resource allocation for rebuilding efforts.
 - Volunteer opportunities for long-term recovery projects.
- Advanced Features
 - AI-powered Matching: Utilize Artificial Intelligence (AI) to optimize volunteer assignment. AI algorithms could consider factors like:
 - Volunteer skills and past performance data.
 - Real-time traffic conditions and travel time estimations.
 - Evolving needs of the disaster zone.
 - Multilingual Support: Expand accessibility by offering Stormlight in multiple languages to cater to a wider user base.
 - Offline Functionality (Mobile App): Develop a mobile app that allows for basic disaster reporting (with text-based location descriptions) and volunteer task management even in areas with limited internet connectivity.
- Scalability and Sustainability
 - Integration with Emergency Services: Integrate Stormlight with existing emergency response systems to expedite dispatch and resource allocation.
 - Partnerships: Partner with disaster management agencies, NGOs, and volunteer organizations to promote Stormlight and expand its reach.
 - Monetization Strategies (Optional): Explore sustainable funding models (e.g., grants, corporate sponsorships) to ensure long-term system maintenance and development, balancing accessibility with financial viability (if applicable).
- Addressing Emerging Challenges
 - Cybersecurity: Regularly update the system to address evolving cybersecurity threats and ensure data privacy.
 - Misinformation Management: Develop mechanisms to identify and flag potentially false reports to prevent misuse of the system.
 - Mental Health Support: Consider integrating resources for mental health support for volunteers and disaster-affected individuals.

10. CONCLUSION

In conclusion, Stormlight offers a promising solution for improving disaster response by empowering communities to report emergencies, connect with volunteers, and coordinate relief efforts. Here's a summary of the key points:

- **Strong Foundation:** Stormlight addresses the critical need for faster response times through accurate location sharing and real-time updates.
- **Efficient Volunteer Coordination:** Matching volunteers based on location and skills optimizes search and rescue efforts and resource allocation.
- **Enhanced Transparency:** Real-time information sharing fosters better communication and trust between those affected by the disaster, volunteers, and disaster management personnel.

By addressing potential challenges like cybersecurity threats, misinformation, and mental health support, Stormlight can become a truly comprehensive disaster management system.

11. BIBILIOGRAPHY

www.stackoverflow.com

www.w3schools.com

www.tutorialspoint.com

BOOK OF STUDY

Software Engineering (3rd ed.) By K.K. Aggarwal & Yogesh Singh

APPENDIX-I

```
views.py

1 from django.http import HttpResponse,HttpResponseRedirect
2 from django.shortcuts import render
3 from django.shortcuts import redirect
4 from django.urls import reverse
5 from django.core.files.storage import FileSystemStorage
6 import datetime
7 import pycurl
8 from urllib.parse import urlencode
9 from .models import *
10 from django.db.models import Q
11 import requests
12 import json
13
14
15 # Create your views here.
16
17 def first(request):
18     return render(request,'index.html')
19
20 def index(request):
21     return render(request,'index.html')
22
23 def register(request):
24     return render(request,'register.html')
25
26 def adduser(request):
27     if request.method=="POST":
28         name=request.POST.get('name')
29         email=request.POST.get('email')
30         address=request.POST.get('address')
31         phone=request.POST.get('phone')
32         password=request.POST.get('password')
33
34
35         cus=userregg(name=name,email=email,address=address,phone=phone,p
assword=password)
36         cus.save()
37         return render(request,'index.html', {'message1':'successfully
Registered'})
38
39 def service(request):
40     return render(request,'addservice.html')
```

views.py

```
1
2 def login(request):
3     return render(request, 'login.html')
4
5 def addlogin(request):
6     email = request.POST.get('email')
7     password = request.POST.get('password')
8     if email == 'admin@gmail.com' and password == 'admin':
9         request.session['logintdetail'] = email
10        request.session['admin'] = 'admin'
11        return redirect(index)
12
13    elif
14        userregg.objects.filter(email=email,password=password).exists():
15
16        userdetails=userregg.objects.get(email=request.POST['email'],
17        password=password)
18        if userdetails.password == request.POST['password']:
19            request.session['uid'] = userdetails.id
20            request.session['uname'] = userdetails.name
21            #request.session['number'] = userdetails.number
22            request.session['uemail'] = email
23
24            return redirect(index)
25
26    elif
27        volregg.objects.filter(email=email,password=password).exists():
28
29        userdetails=volregg.objects.get(email=request.POST['email'],
30        password=password)
31        if userdetails.password == request.POST['password']:
32            request.session['vid'] = userdetails.id
33            request.session['vname'] = userdetails.name
34            request.session['vemail'] = email
35
36            return redirect(index)
37
38    elif
39        emergency.objects.filter(name=email,password=password).exists():
40        userdetails=emergency.objects.get(name=email,
41        password=password)
42        if userdetails.password == request.POST['password']:
43            request.session['eid'] = userdetails.id
44            request.session['ename'] = userdetails.name
45
46            return redirect(index)
47
48    else:
49        return render(request, 'login.html', {'message4': 'Invalid
50        Email or Password'})
```

views.py

```
1 def addservice(request):
2     if request.method=="POST":
3         name=request.POST.get('service')
4
5         cus=services(name=name)
6         cus.save()
7         return render(request,'addservice.html',
8             {'message1':'successfully Added'})
9
10 def volregister(request):
11     sel=services.objects.all()
12     return render(request,'volregister.html',{'result':sel})
13
14 def addvol(request):
15     if request.method=="POST":
16         name=request.POST.get('name')
17         email=request.POST.get('email')
18         address=request.POST.get('address')
19         state=request.POST.get('state')
20         phone=request.POST.get('phone')
21         age=request.POST.get('age')
22         gender=request.POST.get('gender')
23         service=request.POST.get('service')
24         password=request.POST.get('password')
25
26         cus=volregg(state=state,name=name,email=email,address=address,ph
27             one=phone,age=age,gender=gender,service=service,password=password
28             )
29         cus.save()
30         return render(request,'index.html', {'message2':'successfully
31             Registered'})
32
33 def emrgnyreg(request):
34     return render(request,'emergencyreg.html')
35
36 def addemrgnyreg(request):
37     if request.method=="POST":
38         name=request.POST.get('name')
39         password=request.POST.get('password')
40
41         cus=emergency(name=name,password=password)
42         cus.save()
43         return render(request,'emergencyreg.html',
44             {'message2':'successfully Registered'})
45
```

views.py

```
1 def logout(request):
2     session_keys = list(request.session.keys())
3     for key in session_keys:
4         del request.session[key]
5     return redirect(first)
6
7 def view_emergencyservice(request):
8     sel=services.objects.all()
9     return render(request,'view_emrgcysr.html',{'result':sel})
10
11 def serrequest(request,id):
12     sel=services.objects.get(id=id)
13     return render(request,'servicereq.html',{'result':sel})
14
15 def addserreq(request):
16     if request.method=="POST":
17
18         e_uid=request.session['ename']
19         service=request.POST.get('service')
20         date=request.POST.get('date')
21         location=request.POST.get('location')
22         state=request.POST.get('state')
23         phone=request.POST.get('phone')
24         message=request.POST.get('message')
25
26         cus=servicerequest(state=state,uid=0,e_uid=e_uid,service=service
27 ,date=date,location=location,phone=phone,message=message,status='
28 pending')
29         cus.save()
30     return redirect(view_emergencyservice)
31
32 def addserreq_user(request):
33     if request.method=="POST":
34
35         uid=request.session['uname']
36         service=request.POST.get('service')
37         date=request.POST.get('date')
38         location=request.POST.get('location')
39         state=request.POST.get('state')
40         phone=request.POST.get('phone')
41         message=request.POST.get('message')
42
43         cus=servicerequest(state=state,uid=uid,e_uid=0,service=service,d
44 ate=date,location=location,phone=phone,message=message,status='pe
45 nding')
46         cus.save()
47     return redirect(view_emergencyservice)
```

views.py

```

1 def profile(request):
2     user=request.session['uid']
3     sel=userregg.objects.filter(id=user)
4     return render(request,'profile.html',{'result':sel})
5
6 def userproupdt(request,id):
7     xy=request.session['uid']
8     xyz=userregg.objects.get(id=xy)
9     return render(request,'proupdt.html',{'result':xyz})
10
11 def pupdate(request,id):
12     if request.method=='POST':
13         name=request.POST.get('name')
14         email=request.POST.get('email')
15         phone=request.POST.get('phone')
16         address=request.POST.get('address')
17         password=request.POST.get('password')
18
19         sp=userregg(name=name,email=email,phone=phone,address=address,pa
20         ssword=password,id=id)
21         sp.save()
22         return redirect(profile)
23
24 def emcservices(request):
25     return render(request,'emergencyser.html')
26
27 def addemcservices(request):
28     if request.method=="POST":
29         ip = requests.get('https://api.ipify.org?format=json')
30         ip_data = json.loads(ip.text)
31         res = requests.get('http://ip-
32         api.com/json/'+ip_data["ip"])
33         location_data_test = res.text
34         location_data = json.loads(location_data_test)
35
36         uid=request.session['uname']
37         #number=request.session['number']
38         District=request.POST.get('district')
39         city=request.POST.get('city')
40         maplink=request.POST.get('maplink')
41         category=request.POST.get('category')
42         intensity=request.POST.get('intensity')
43         service=request.POST.get('service')
44         message=request.POST.get('message')
45         if 'image' in request.FILES:
46             myfile = request.FILES['image']
47             fs = FileSystemStorage()
48             filename = fs.save(myfile.name, myfile)
49         else:
50             filename = None
51
52         cus=emc_servicerequest(ip_contry=location_data['country'],ip_reg
53         ionName=location_data['regionName'],ip_city=location_data['city']
54         ,ip_zip=location_data['zip'],ip_lat=location_data['lat'],ip_lon=l
55         ocation_data['lon'],ip_ipaddr=location_data['query'],maplink=mapl
56         ink,District=District,city=city,category=category,uid=uid,e_uid=0
57         ,intensity=intensity,image=filename,message=message,service=servi
58         ce,status='pending')
59         cus.save()
60         return render(request,'index.html', {'message4':'successfully
61         Registered'})

```

views.py

```

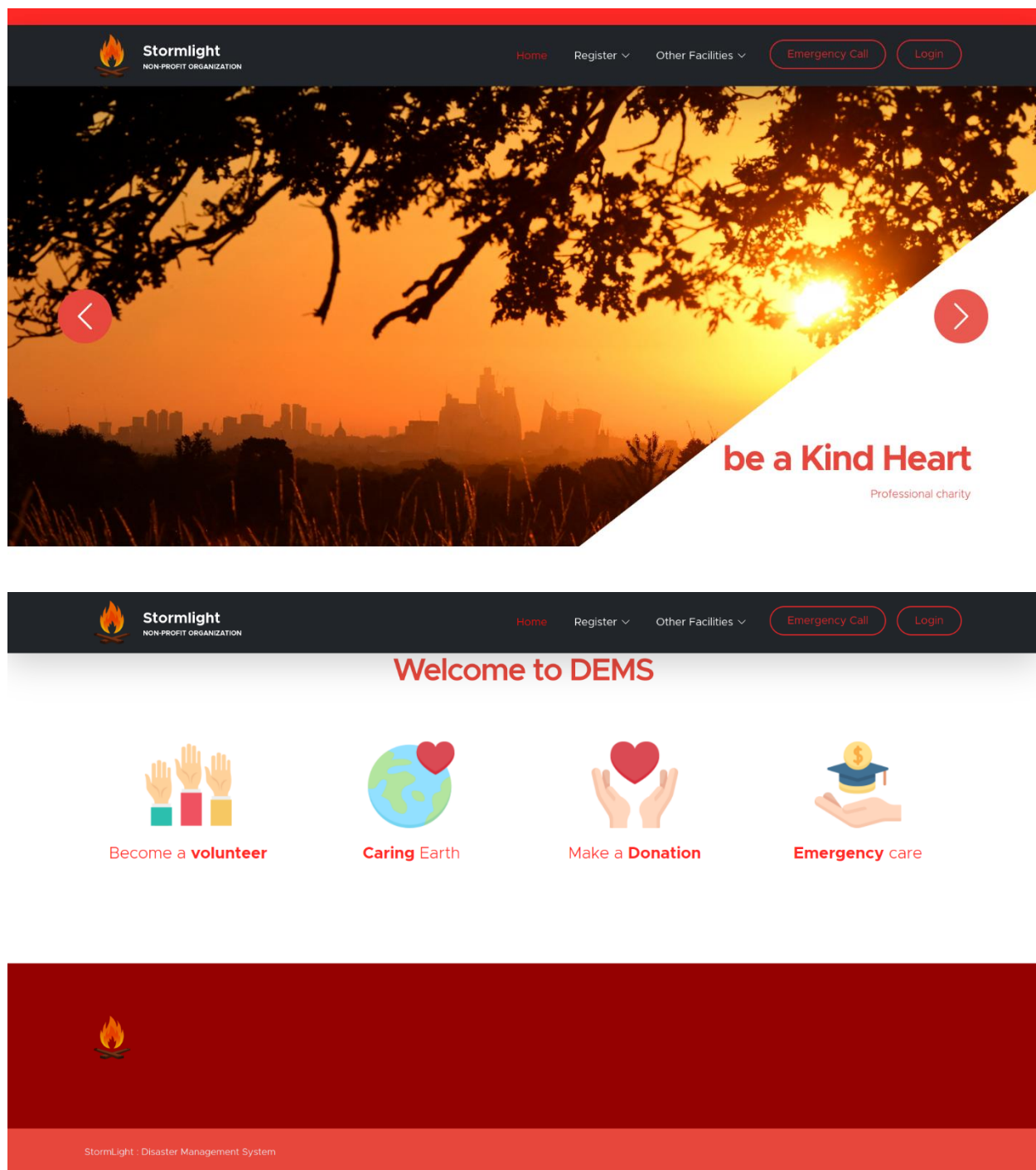
1 def addemcservices2(request):
2     if request.method=="POST":
3         ip = requests.get('https://api.ipify.org?format=json')
4         ip_data = json.loads(ip.text)
5         res = requests.get('http://ip-
api.com/json/'+ip_data["ip"])
6         location_data_test = res.text
7         location_data = json.loads(location_data_test)
8
9         e_uid=request.POST.get('name')
10        number=request.POST.get('number')
11        District=request.POST.get('district')
12        city=request.POST.get('city')
13        maplink=request.POST.get('maplink')
14        category=request.POST.get('category')
15        intensity=request.POST.get('intensity')
16        service=request.POST.get('service')
17        message=request.POST.get('message')
18        if 'image' in request.FILES:
19            myfile = request.FILES['image']
20            fs = FileSystemStorage()
21            filename = fs.save(myfile.name, myfile)
22        else:
23            filename = None
24
25
26
27        cus=emc_servicerequest(ip_contry=location_data['country'],ip_reg
ionName=location_data['regionName'],ip_city=location_data['city']
,ip_zip=location_data['zip'],ip_lat=location_data['lat'],ip_lon=l
ocation_data['lon'],ip_ipaddr=location_data['query'],maplink=mapl
ink,District=District,city=city,category=category,uid=0,e_uid=e_u
id,intensity=intensity,image=filename,message=message,service=ser
vice,status='pending')
28        cus.save()
29        return render(request,'index.html', {'message4':'successfully
Registered'})
30
31 def viewvol(request):
32     sel=volregg.objects.all()
33     return render(request,'viewvolunteer.html',{'res':sel})
34
35 def search(request):
36     if request.method == "POST":
37         title = request.POST.get('title')
38         print(title)
39
40         crp = volregg.objects.filter(Q(address__icontains=title))
41         return render(request,'viewvolunteer.html', {'res': crp})
42     else:
43         crp = volregg.objects.all()
44         return render(request,'viewvolunteer.html', {'res': crp})


```

views.py

```
1 def add_hospital(request):
2     if request.method=="POST":
3         name=request.POST.get('name')
4         phone=request.POST.get('phone')
5         place=request.POST.get('place')
6         map=request.POST.get('map')
7
8         cus=hospitals(map=map,name=name,place=place,phone=phone)
9         cus.save()
10        return render(request,'index.html',
11        {'message9':'successfully Added'})
12    else:
13        return render(request,'add_hospital.html')
14
15 def v_hos(request):
16     sel=hospitals.objects.all()
17     return render(request,'v_hos.html',{'result':sel})
18
19 def remove_hos(request,id):
20     sel=hospitals.objects.get(id=id)
21     sel.delete()
22     return redirect(v_hos)
23
24 def customer_chat(request):
25     cus_id = request.session['uid']
26     cht = chats.objects.filter(cus_id=cus_id)
27     return render(request,'customer_chat.html', {'chat':
28     cht,'team':'Admin'})
29
30 def addchat_customer(request):
31     if request.method == 'POST':
32         cus_id = request.session['uid']
33         message = request.POST.get('message')
34
35         ins = chats(cus_id=cus_id, cus_msg=message,admin='admin')
36         ins.save()
37
38     return redirect(customer_chat)
39
40 def view_chats(request):
41     member=userregg.objects.all()
42     return render(request,'viewchat.html',{'result':member})
43 \
```


APPENDIX - II




Stormlight
NON-PROFIT ORGANIZATION

[Home](#)
[Register](#)
[Other Facilities](#)
[Emergency Call](#)
[Login](#)


Donation Registration

Become a Donor today

Submit


About DEMS

The District Emergency Management System involves a coordinated and integrated approach to minimize the impact of disasters on people, property, and the environment.


Stormlight
NON-PROFIT ORGANIZATION

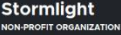
[Home](#)
[Register](#)
[Other Facilities](#)
[Emergency Call](#)
[Login](#)

Emergency Request



About STORMLIGHT


The District Emergency Management System involves a coordinated and integrated approach to minimize the impact of disasters on people, property, and the environment.

**Stormlight**
NON-PROFIT ORGANIZATION

[Home](#) [Register](#) [Other Facilities](#) [Emergency Call](#) [Login](#)


User Registration

Become a Member today



About STORMLIGHT

The District Emergency Management System involves a coordinated and integrated approach to minimize the impact of disasters on people, property, and the environment.

**Stormlight**
NON-PROFIT ORGANIZATION


[Home](#) [Register](#) [Other Facilities](#) [Emergency Call](#) [Login](#)

Volunteer Registration

Become a Volunteer today

Select Gender

Select Service




About STORMLIGHT

The District Emergency Management System involves a coordinated and integrated approach to minimize the impact of disasters on people, property, and the environment.

CHINMAYA COLLEGE OF ARTS, COMMERCE AND SCIENCE




DEPARTMENT OF COMPUTER APPLIATIONS



Stormlight
NON-PROFIT ORGANIZATION

[Home](#)
[Requests](#)
[Messages](#)
[ADD](#)
[View](#)
[Emergency Requests](#)
[Logout](#)

Emergency Requests


Emergency Users Requests

Emergency User Name	Requested Service	Intensity	Message/Description	Location	ip based details	Map Link	Category	Image	Report
sreeraj s	Water Emergency	very_high	kachkjd	vaikom, Eranakulam, Kerala	ip : 103.148.20.155 Region : Kerala City : Kochi Zip : 682018 Lat : 9.9185 Lon : 76.2558	jfwdnjn	Public		Report
worker	Water Emergency	high	vvfdv	sdv, Eranakulam, Kerala	ip : 103.148.20.155 Region : Kerala City : Kochi Zip : 682018 Lat : 9.9185 Lon : 76.2558	dvvfdv	Public		Report
worker	Water Emergency	high	vvfdv	sdv, Eranakulam, Kerala	ip : 103.148.20.155 Region : Kerala City : Kochi Zip : 682018 Lat : 9.9185 Lon : 76.2558	dvvfdv	Public		Report



Stormlight
NON-PROFIT ORGANIZATION

[Home](#)
[Emergency](#)
[Services](#)
[Volunteers](#)
[My Requests](#)
[Chat](#)
[Profile](#)
[Disasters](#)
[Logout](#)


Our Services




NCC




NSS



Red Cross



Mahila Samajam



127.0.0.1:8000/view_emergencyservice

