###### A Project report on

**URGENCE**

###### Submitted in partial fulfillment of the requirement for the award of

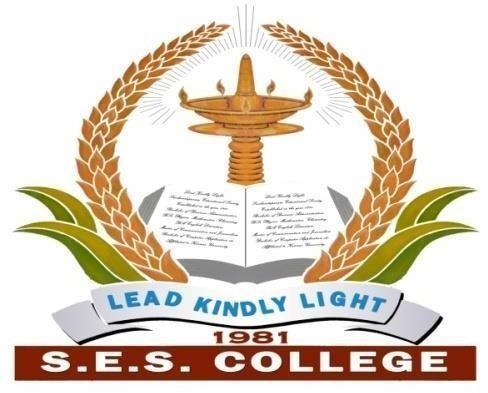
**BACHELOR OF COMPUTER SCIENCE**

**Degree of KANNUR UNIVERSITY**

**By**

**ARYAKRISHNAN R K**

###### Register No: SE20BCAR20



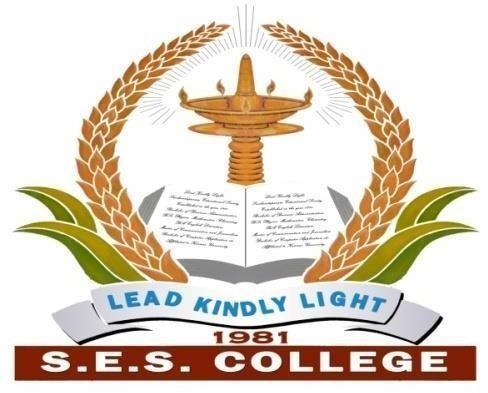
**SES COLLEGE SREEKANDAPURAM**

**KANNUR-670631 KERALA**

**2023**

#### KANNUR UNIVERSITY SES COLLEGE

**SREEKANDAPURAM, KANNUR-670 631**



**CERTIFICATE**

This is to certify that the project entitled “**URGENCE**” submitted in partial fulfillment of the requirement for the award of BCA Degree is a result of bona-fide work of carried out by **Mrs.ARYAKRISHNAN R K** during the year 2023.

###### Internal Guide Head of the institution

Mrs. Manjima Benny SES College

Assistant Professor Sreekandapuram

SES College, Sreekandapuram

**External Examiners 1.(**Name&Signature)**: 2.(**Name&Signature)**:**

###### DECLARATION

I, VI semester BSC COMPUTER SCIENCE student of SES college, Sreekandapuram, under Kannur University do hereby declare that the project entitled “**URGENCE**” is the original work carried out by me under the super vision of Mrs. Manjima Benny., Assistant Professor, SES College, Sreekandapuram towards partial fulfillment of the requirement of BCA Degree, and no part thereof has been presented for the award of any other degree.

Date: ARYAKRISHNAN R K

Reg No: SE20BCAR20

Dept of Computer Science & Application

SES College Sreekandapuram, Kannur -670631

###### ACKNOWLEDGEMENT

I am thankful to GOD Almighty for his blessing to me. I express my sincere gratitude to Mrs.Deepthi K K, Head of the Department of Computer Science and Application, SES College, Sreekandapuram for his valuable advice and help throughout this project.

I express my deep sense of gratitude to Riss Technologies, was my External Guide for valuable support and suggestions during the project period.

My sincere thanks to Mrs. Manjima Benny, Asst. Professor Department of Computer Science and Application, SES College Sreekandapuram, who was my Internal Guide, for his valuable support and help during the project period.

I convey my thanks to all the staff members of the Department of Computer Science and Application, SES College Sreekandapuram, and my friends who helped me during the project.

**ABSTRACT**

The aim of the project is to develop an android application that lets its users to send notification in case of as terrorist attack, medical, natural disaster, women protection. The users can send multiple messages, WhatsApp message, phone calls and emails on the press of a single button and there are four other sessions names hospital, drugs, health, workouts.

1. Medical session has lot of other session like hospital session, drug session, health session, doctor session, workout session
   1. Natural Disaster session provide news fields, live weather podcast, Natural Disaster surviving tutorial and provide emergence number.
   2. Terrorist Aid session provide self-defense and emergence button make the user can sent automatic SMS and call to the emergence authorities.
   3. Women protection session provide self-defense tutorial, User can able to buy self-protection tool, emergence button makes the user can sent automatic SMS and call to the emergence authorities.

Module

1. Admin
2. Doctor
3. User

The phone numbers, email ids and the contents of the call, text, WhatsApp and email messages can be set from within the application. The text, WhatsApp messages and emails sent along with the content also have the last known location of the user. This is very helpful in tracking the whereabouts of the person. The user can also call 100,108 directly from within the application, if the nature of the situation demands it.

Additionally, the user of the application may allow the app to track their location. If this option is selected, the application fetches the device’s location at about every 10 minutes and stores it in a database. This information is very useful and can be used in a variety of ways. One such use of the location data is from within the Android app where the user can view a map that shows their location history over a period of time for a particular day.

| Chapters | Content | Page NO |
| --- | --- | --- |
| **1** | **INTRODUCTION** | 5 |
| 1.1 | Software Development Life Cycle Model | 7 |
| 1.1.1 | Waterfall Model | 8 |
| **2** | **SYSTEM ANALYSIS** | 9 |
| 2.1 | Existing System | 10 |
| 2.2 | Proposed System | 11 |
| 2.3 | Feasibility Study | 11 |
| 2.3.1 | Technical Feasibility | 12 |
| 2.3.2 | Economical Feasibility | 13 |
| 2.3.3 | Behavioral Feasibility | 14 |
| 2.4 | System Requirements | 14 |
| 2.4.1 | Hardware Requirement | 14 |
| 2.4.2 | Software Requirement | 15 |
| 2.5 | System Requirement Specification | 16 |
| 2.5.1 | Actor Identification | 17 |
| 2.5.2 | Use case Identification | 17 |
| 2.5.2.1 | Use cases | 23 |
| 2.5.2.2 | Use case Diagram | 25 |

| **3** | **SYSTEM DESIGN** | 26 |
| --- | --- | --- |
| 3.1 | Database Design | 27 |
| 3.1.1 | E R Diagram | 29 |
| 3.1.2 | Table Design | 32 |
| 3.2 | Architectural Design | 42 |
| 3.2.1 | Data flow Diagram | 45 |
| 3.3 | Interface Design | 51 |
| 3.3.1 | Input Design | 51 |
| 3.3.2 | Output Design | 53 |
| **4** | **CODING** | 55 |
| 4.1 | Description of Software | 56 |
| 4.2 | Coding Principle | 62 |
| **5** | **TESTING** | 65 |
| 5.1 | Unit Testing | 67 |
| 5.2 | Integration Testing | 70 |
| 5.3 | Validation Testing | 71 |
| 5.4 | System Testing | 71 |
| **6** | **IMPLEMENTATION** | 72 |
| **7** | **Conclusion and Future Enhancement** | 74 |
| 7.1 | Conclusion | 75 |
| 7.2 | Future Enhancement | 76 |
| **8** | **REFERENCES** | 77 |
| **9** | **APPENDIX** | 79 |

# INTRODUCTION

###### 1.Introduction

Urgence is an innovative and user-friendly emergency and SOS mobile application that is designed to provide quick and efficient access to medical services and assistance in times of emergency. The application is built using advanced technologies such as HTML, CSS, and Bootstrap in the frontend and Python Flask in the backend.

With its three distinct modules, the Urgence application provides comprehensive functionality to its users, including administrators, doctors, and regular users. The administrator module is responsible for managing all activities within the application, including creating and updating user profiles, providing access to medical services, and controlling the overall functioning of the application.

The user module enables regular users to access a range of functionalities, including purchasing products and services, viewing medical details, and triggering the SOS functionality in case of an emergency. Users can also access a range of additional features such as diet plans, natural disaster management, self-defense tips, women protection functionality, news feed, and other essential services.

The doctor module is designed to provide medical professionals with a dedicated platform to offer their services to users of the Urgence application. Doctors can provide medical services, access medical records of their patients, and communicate with other medical professionals to provide the best possible care to patients.

The Urgence application is packed with a range of features that make it an essential tool for emergency response and medical care. With the SOS functionality, users can trigger a request for emergency medical assistance with a single tap of a button. The application also provides users with access to contact details for the nearest hospitals, medical practitioners, and other essential emergency services

In conclusion, the Urgence mobile application is an excellent example of the power of advanced technology in providing critical emergency and medical services to users. Its comprehensive functionality, user-friendly interface, and advanced features make it an indispensable tool for anyone looking to access emergency medical assistance quickly and efficiently.

#### 1.1. Software Development Life Cycle Model (SDLC)

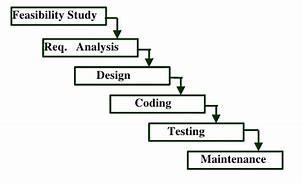
A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. Thus, no element which life cycle model is followed; the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

**1.1.1. WATERFALL MODEL**

The waterfall model was selected as the SDLC model due to the following reasons:

* Requirements were very well documented, clear and fixed.
* Technology was adequately understood.
* Simple and easy to understand and use.
* There were no ambiguous requirements.
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Clearly defined stages.
* Well understood milestones. Easy to arrange tasks.



**SYSTEM ANALYSIS**

1. **SYSTEM ANALYSIS**

Requirement analysis result in the specification of software’s operational characteristics indicates software’s interface with other system elements, and establishes constraints that software must meet. The requirements are analyzed in order to identify inconsistencies, defects, omissions etc. are describe requirements in terms of relationships and resolve conflicts, if any. Analysis report contains a detailed study of existing system like how it is working, what are the drawbacks of the system, etc. From that study we can understand the need of new system. We tried to find any risk associated with the system and study the feasibility of the system. The gathering can be done by following ways.

1. Interview.
2. Questionnaire.
3. Site visit.
4. Website.

For requirement analysis we choose the method questionnaire. For that we prepare some questions for the sellers. The reason behind to choose this method is, we want to collect information about existing system

**Existing system**

In our real-life emergency situations, we handle very simply and depending on other people. In that time there is also the possibility that there are no people to help and time wasting it causes people to die and the situation gets worse. For example,

A woman walking alone on her way home some people are attacking her. That time she can only call for others and scream. Within that time, she may become a victim or someone may arrive there and help her.

In medical emergency situations we directly go to hospital, enquire about the hospital, and take the medicines ourselves. Sometime they may not accurate

In a place there may occur a terrorist attack, in that time the terrorist makes conditions with the officials. After the officials realizing the terrorist attack is happening, they take the necessary action.

When a natural disaster occurs, the people can only handle the situation by the knowledge they know. The way they handle the situation with their knowledge

may be wrong or there is a possibility that they do not have the right knowledge to handle the situation. It may cost their life.

**Proposed System**

Here we propose a system to overcome the disadvantages of existing system. In this system admin can able to add e-commerce details, modify the existing user. App can able to overcome some real-life issue, its maybe an emergency or not. So, by using this app user overcome it.

**Feasibility Study**

During system analysis a feasibility study of the proposed system “” is carried out to see whether it is beneficial to society. It is a test of proposed system regarding, its workability, impact on organization, and the ability to meet the user requirements. It is performed to choose the system that meets the performance requirements at lower cost. Primarily, whether a project that was initiated by one must consider for further development or not will be decided through preliminary investigation. The study begins by classifying the problem definition. Feasibility is to determine if it is worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analysed carefully. The study was done in four areas:

* Technical feasibility
* Economic feasibility
* Behavioural feasibility

**Technical Feasibility**

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on outline design of system requirements in terms of input, processes, output, fields, programs, and procedures. This can be qualified in terms of volume of data, trends, frequency of updating in order to introduce the technical system. The application is the fact that it has been developed on Windows 7/8 and above operating system, primary memory 4GB RAM and above and Intel Pentium Core i3 and above processor. This is technically feasible. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

**Front-end selection:**

* Extremely Light Weight.
* Full Featured database classes with support for several platforms.
* Active Record Database Support.
* Data encryption

According to the above stated featured we selected python3 as the front-end for developing our project.

**Back-end Selection:**

* Cross platform support
* Updatable views
* Provide inherent features for security.
* Embeddable database library
* Manipulate Data base is Sql

According to above stated features We selected MySQL as the back- end. The technical feasibility is frequently the most difficult area encountered at this stage.

It is essential that the process of analysis and definition be conducted in parallel with an assessment to technical feasibility. It centres on the existing system (hardware, software etc.) and to what extent it can support the proposed system.

**Economic Studies**

Establishing the cost-effectiveness of the proposed system i.e., if the benefits do not outweigh the costs, then it is not worth going ahead. In the fast-paced world today there is a great need of online social networking facilities. Thus, the benefits of this project in the current scenario make it economically feasible. The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

**Behavioural-Feasibility**

The proposed system is very much user friendly and operations on it can be done very easily. It has descriptive and user-friendly interfaces, which enables the user to operate on it very easily. The proposed system is very helpful to students to know their extra activities. In addition, it can be maintained easily. In the existing system, it requires more human effort to maintain records. In the proposed system, and both human effort and time are managed efficiently. The proposed system is beneficial to students and teachers. Hence, the system is behaviourally feasible.

**ORGANISATION OF THE REPORT**

INTRODUCTION

This section includes the overall view of the project i.e. the basic problem definition and the general overview of the problem which describes the problem in layman terms. It also specifies the software used and the proposed solution strategy.

**SOFTWARE REQUIREMENTS SPECIFICATION**

This section includes the Software and hardware requirements for the smooth running of the application.

**DESIGN & PLANNING**

This section consists of the Software Development Life Cycle model. It also contains technical diagrams like the Data Flow Diagram and the Entity Relationship diagram.

**IMPLEMENTATION DETAILS**

This section describes the different technologies used for the entire development process of the Front-end as well as the Back-end development of the application.

**RESULTS AND DISCUSSION**

This section has screenshots of all the implementation i.e., user interface and their description.

**SUMMARY AND CONCLUSION**

This section has screenshots of all the implementation i.e., user interface and their description

**SYSTEM REQUIREMENTS**

Hardware and software requirements for the installation and smooth functioning of this project could be configured based on the requirements needed by the component of the operating environment that works as front-end system here we suggest minimum configuration for the both

hardware and software components.

Working off with this software is requirements concrete on system environments. It includes two phases

 Hardware requirements

 Software requirement

**HARDWARE REQUIREMENTS**

* + INPUT DEVICE : MOUSE, KEYBOARD
  + OUTPUT DEVICE : MONITOR
  + MEMORY : 4GB RAM or above
  + PROCESSOR : Intel core i3 or above

**SOFTWARE REQUIREMENTS**

* + OPERATING SYSTEM : WINDOWS 8 or above
  + FRONT END : Android Java
  + BACK END : Python and MySQL

###### SOFTWARES USED : Android SDK, JetBrains PyCharm

###### 2.5. SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement specification is a document that completely describes all of the functions of a proposed system and the constraints under which operate. System requirements are expressed in a system requirement document. The requirement document indicates the requirements definitions and the requirement specifications. The software requirement document is not a design document. It should set out what the system should do without specifying how it should be done. The requirement set out in this document is complete and consistent. The Software Requirement Specification document as it typically contains the following.

* A complete description of the software’s purpose and functionality.
* Details as to how the software will perform in terms of speed, response time, availability, portability, maintainability and more.
* Use cases of how users will use the software.
* It specifies the system behaviors.
* It specifies constraints on the implantation.

###### 2.5.1. ACTOR IDENTIFICATION

An actor is someone or something that interacts with the system. An actor is he/she what uses the system. An actor exchanges information with the system. Asking certain questions as detailed below can identify the actors of the system.

| 1 | Who will use the main functionality of the system? | Admin ,Doctor ,Users |
| --- | --- | --- |
| 2 | Who will lead support from the system and  do their daily tasks? | Admin , Doctor ,Users |
| 3 | Who will maintain and administrator of the  system? | Admin |
| 4 | Which hardware devices does the system  need to handle? | Monitor ,Mobile  ,Keybord |
| 5 | With which other system does this system  need to interact? | Database |
| 6 | Who are the benefiters of result produced by the system? | Users |

The answers to this question bring out the actors of the system as:

* + Admin
  + Doctor
  + User

###### 2.5.2. USE CASE IDENTIFICATION

A use cases represents the functionality of an actor. It is defined as a set of actions performed by a system, which yields an observable result. An ellipse containing its name inside the ellipse or below it represents it. It is placed

Inside the system boundary and connected to an actor with an association.

#### Use case for Admin

| **1** | Who is the super user of the system? | Admin has the privilege of super user |
| --- | --- | --- |
| **2** | What function does the admin require for the system? What does the Admin need to do? | Admin must have username and password. View complaints , hospital management, Medical management, Tip management, Doctor management, News management, NDST management, Women management, send reply , Self-defense tool management, View purchase, Notification management, View Self-defense tool |
| **3** | Does the Admin need to read, create, destroy, modify or store some kind of information in the system? | Yes, Admin need to create, view and edit the data if required. |
| **4** | Could the Admin’s daily work be simplified or made more efficient by adding new functions to the system | Yes, the admin daily work could be made more efficient. |

The above questions give the following use cases for the actor admin:

1. Login

2. Hospital management

3.View complaints

4.hospital management

5.Medical management

6.Tip management

7.Doctor management

8.News management

9.NDST management

10.Women management

11send reply

12.Self-defense tool management

13.View purchase

14.Notification management

16.View Self-defense tool

**Use case for Doctor**

| **1** | What does the Doctor require from the system? What does Doctor need to do | Doctor requires username and password. View profile, Schedule Management, View appointment and upload details |
| --- | --- | --- |
| **2** | Does the Doctor need to read, create, destroy, modify or store some kind of information in the system? | Yes, Doctor need to create, view and edit the data if require |
| **3** | Could the Doctor daily work be simplified or made more efficient by adding new Schedule to the system? | Yes, the Doctor work could be more efficient and reduced. |

The above questions give the following use cases for the actor Doctor

1. View profile,
2. Schedule management
3. View appointment and upload details

**Use case for User**

| 1 | What function does the Users require from the system? What does user need to do? | User must have a username and password, must have the permission to Change password, Registration, Login,  Send complaint and view reply, view hospital, view medicine, view tip, view doctor and book, view booking history and prescription, view diet profile, view news, view NDST, view Self-defense, view booking history of tools. |
| --- | --- | --- |
| 2 | Does the Users need to read, create, destroy, modify or store some kind of information in the system? | Yes, Users need to create, view and edit the data if require. |
| 3 | Does the Users need to read ,create ,destroy ,modify or store some kind of information in the system | Yes, Users need to create, view and edit the data if required. |

The above questions give the following use cases for the actor Customers.

1. Login,

2.Send complaint

3.view reply,

4.view hospital,

5.view medicine,

6.view tip,

7.view doctor and book,

8.view booking history and prescription,

9.view diet profile,

10.view news,

11.view NDST,

12.view Self-defense,

13.view booking history of tools

**2.5.2.1. Use Cases**

The use case represents the functionality of an actor. It is defined as a set of actions performed by the system, which yields an observable result. In the analysis section three actors such as admin, authority and user. There are four basic elements in use case diagram:

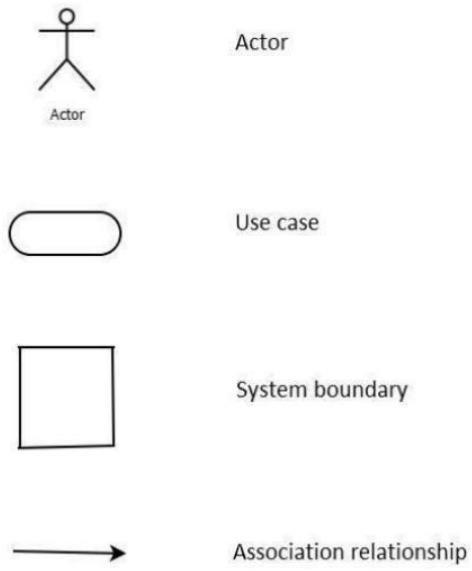
* Actor is a person or system that derives benefits from and is external to the system, is placed outside the system boundary.
* Use case represents a major piece of system functionality, is placed inside

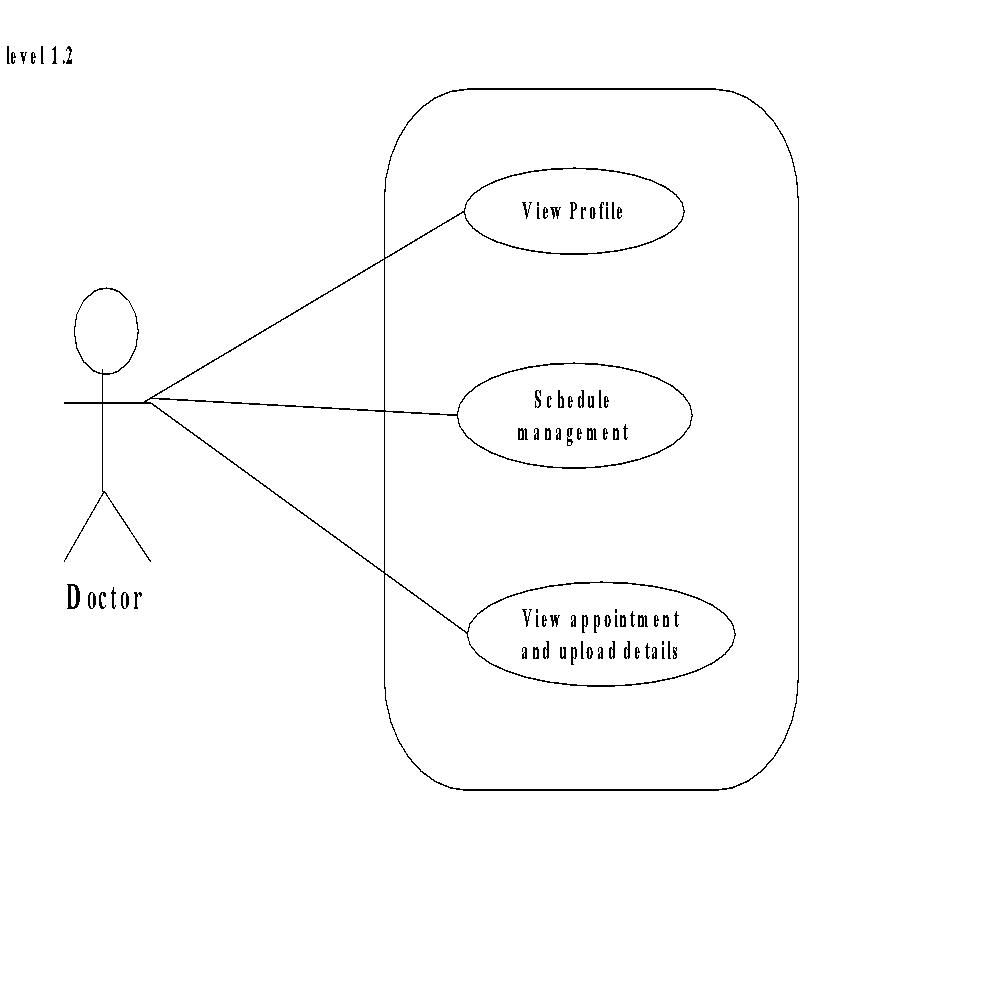
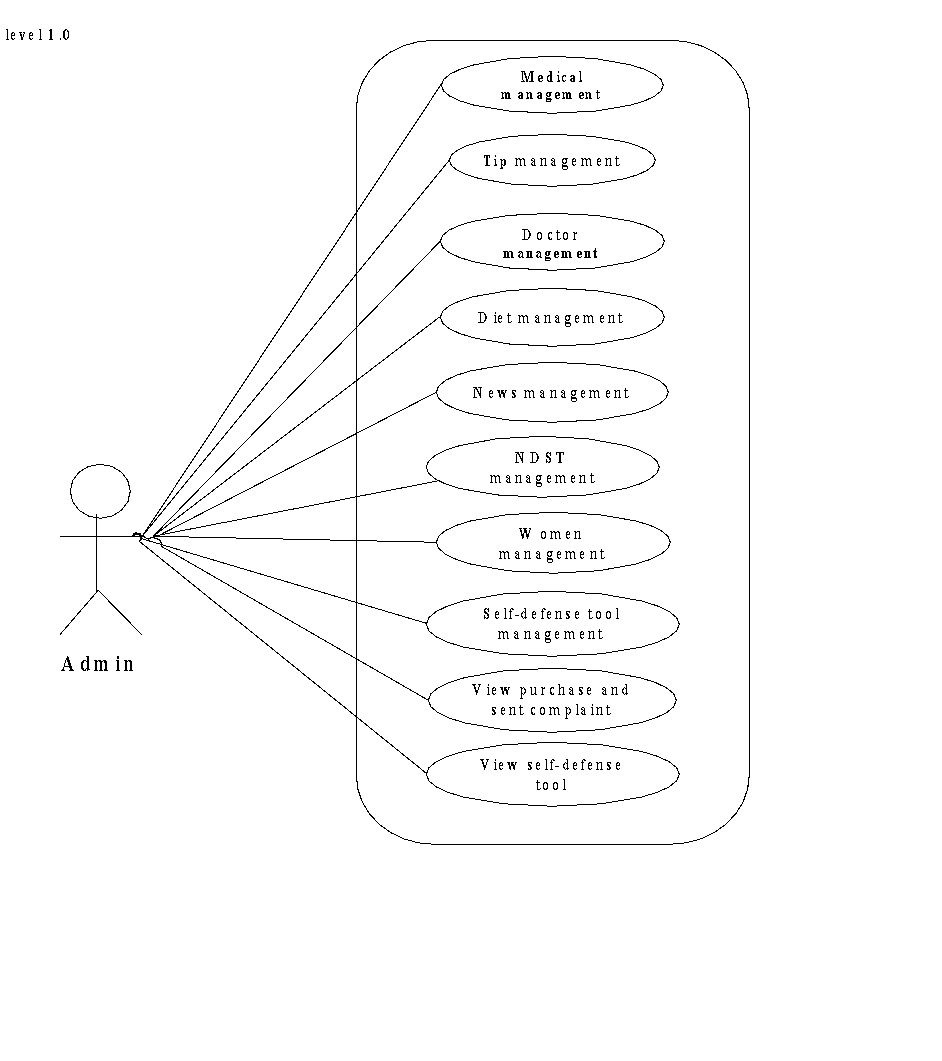
the system boundary.

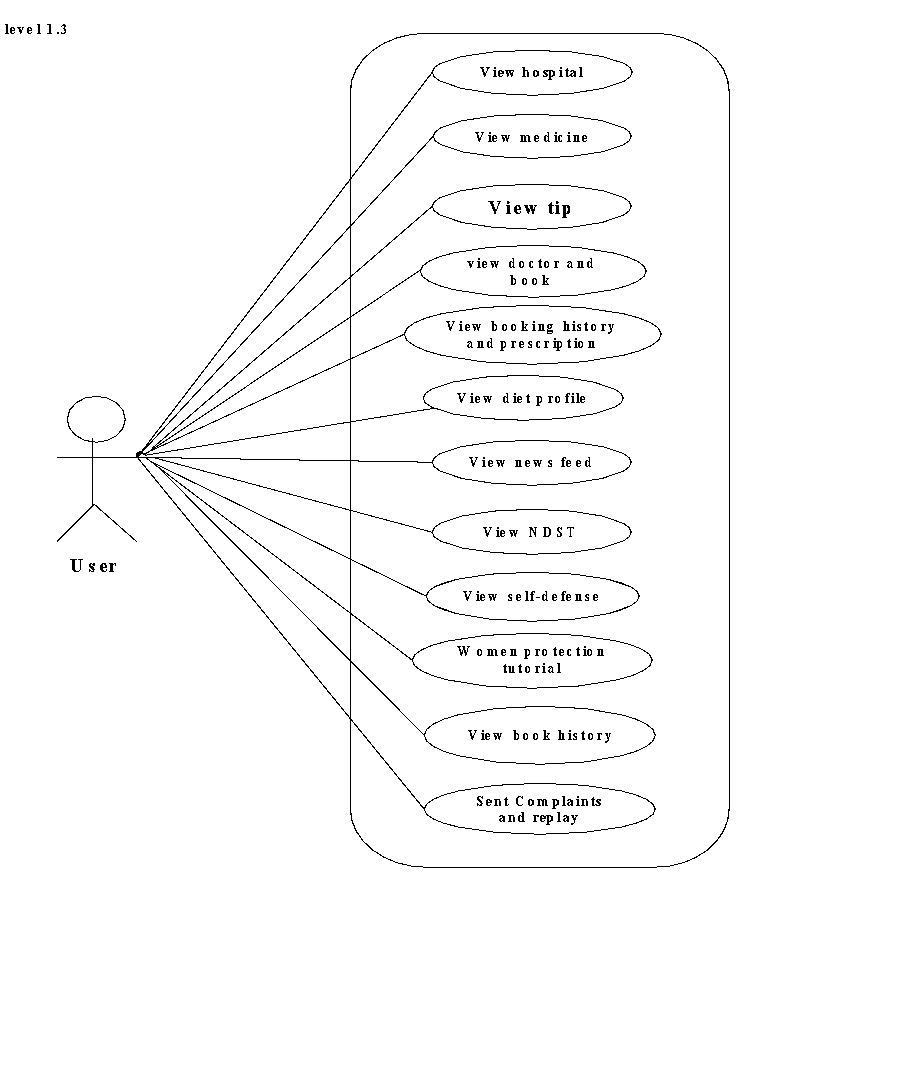
* System boundary includes the name of the system inside or on top, represents the scope of the system.
* Association relationship links an actor with the use case with which it

interacts.

#### Use Case Notations







**SYSTEM DESIGN**

1. **SYSTEM DESIGN**

System design is the second phase of the system life cycle. The detailed design of the system selected in the study phase is accomplished in the design phase. The principal activity performed during this phase includes allocation of function between computer programs equipment and manual tasks and databases design and test requirement definition. In the course of design phase, the performance specification is expanded into the design specification. A design phase report is prepared after the completion of design phase activities and the review is held with the user organization in order to determine whether or not the computer based business information system project is ready to the development phase. The goal of the design phase is to plan a solution of the problem specified by requirements document. Major activities during the design phase are:

* Data Base Design
* Architectural Design
* Interface Design
* Modular Design

###### 3.1 DATABASE DESIGN

A database is a collection of inter related data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy, quick, inexpensive and flexible for the user. In a database environment, common data are available in which several users can use. The concept behind a database is an integrated collection of data and provides a centralized access to the data from the program. When too many attributes are grouped together to form entities, some attributes are found to be entities themselves. Further normalization of these entities into attributes linked by common data elements to form relationships improves the effectiveness of the DBMS.

Guidelines for designing a database:

* Design a relational schema so that it is easy to explain its meaning. Do not combine attributed from multiple entity and relationship types into a single relation.
* Design the database schema so that no insertion, deletion or modification

anomalies are present in the relation.

* As per as possible, avoid placing attributes in a base relation whose values

may frequently be null.

* Design relation schema so that they can be joined with equality conditions on attributes that are either primary keys or foreign keys in a way that n spurious tuples are generated.

Advantages

* + Ease of use
  + Data independence
  + Accuracy and integrity
  + Avoiding inordinate delays
  + Recovery from failures
  + Private and security

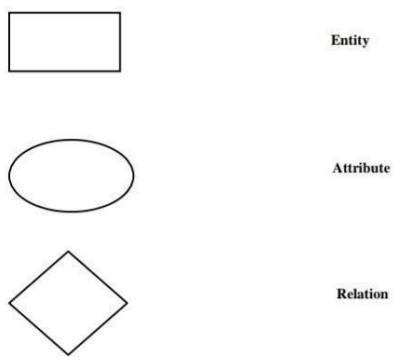
###### 3.1.1 Entity Relationship (ER) Diagram

It is a detailed logical representation of the data for an organization and uses three main constructs data entities, relationship and their associated attributes. An entity is a fundamental thing of an organization about which data may be maintained. A relationship is a reason for associating two entity types degree of relationship is the number of entity type that participates in that relationship. There are three basic elements in ER models:

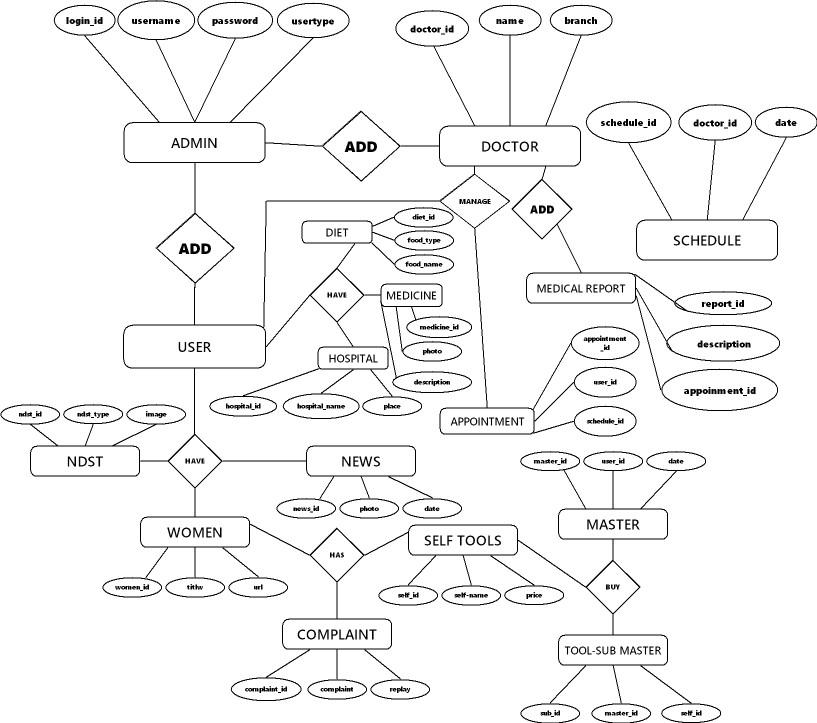
* + Entities are the “things” about which we seek information.
  + Attributes are the data we collect about entities.
  + Relationships provided the structure needed to draw information

from multiple entities.

###### ER Diagram Symbols:



**ER DIAGRAM**



**3.1.2. TABLE DESIGN**

In the database all the information are stored in the form of tables. A table is simply a way storing data in rows and columns. In the system data is stored in many tables.

Table 1: Login

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| Login\_id | int(15) | Primary key | Id of lodin |
| username | Varchar(25) | Not null | Name of user |
| password | Varchar(100) | Not null | Password of user |
| usertype | Varchar(25) | Not null | Type of user |

Table 2: user

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| User\_id | int(15) | Primary key | Id of user |
| name | Varchar(25) | Not null | Name of user |
| age | int(10) | Not null | age of user |
| gender | char(10) | Not null | gender of user |
| height | int(15) | Not null | height of user |
| weight | int(15) | Not null | weight of user |
| number | bigint(15) | Not null | number of user |
| email | Varchar(25) | Not null | email of user |
| image | Varchar(255) | Not null | image of user |
| place | Varchar(200) | Not null | place of user |
| post | Varchar(200) | Not null | post of user |
| pin | int(11) | Not null | pin of user |

Table 3: Hospital

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| hospital\_id | int(15) | Primary key | Id of hospital |
| Hospital\_name | Varchar(50) | Not null | Name of hospital |
| place | Varchar(50) | Not null | place of hospital |
| post | Varchar(50) | Not null | post of hospital |
| pin | int(50) | Not null | pinof hospital |
| district | Varchar(500) | Not null | district of hospital |
| Phone\_no | bigint(255) | Not null | Phone number of hospital |
| Email | Varchar(100) | Not null | email of hospital |
| description | Varchar(500) | Not null | description of hospital |

Table 4: Doctor

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| doctor\_id | int(15) | Primary key | Id of doctor |
| photo | Varchar(500) | Not null | photo of doctor |
| name | Varchar(25) | Not null | name of doctor |
| branch | Varchar(25) | Not null | Specialization of doctor |
| number | bigint(15) | Not null | Number of doctor |
| place | Varchar(25) | Not null | Place of doctor |
| pin | int(15) | Not null | Pin number of doctor |
| post | Varchar(25) | Not null | post of doctor |
| Hospital \_name | Varchar(25) | Not null | Hospital\_name of doctor |
| district | Varchar(25) | Not null | district of doctor |
| email | Varchar(25) | Not null | email of doctor |
| phone | bigint(20) | Not null | Phone number of doctor |
| fees | bigint(20) | Not null | fees of doctor |

Table 5: Medicine

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| Medicine\_id | int(15) | Primary key | Id of medicine |
| photo | Varchar(100) | Not null | Photo of medicine |
| Medicine\_type | Varchar(25) | Not null | type of medicine |
| Medicine\_name | Varchar(25) | Not null | name of medicine |
| description | Varchar(500) | Not null | description of medicine |
| Manufacturing\_company | Varchar(25) | Not null | Manufacturing company of medicine |
| price | int(15) | Not null | price of medicine |

Table 6: Medical report

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| Report\_id | int(15) | Primary key | Id of medical report |
| description | Varchar(500) | Not null | Description of medical report |
| appoinment \_id | int(15) | Not null | Id\_Appointment |

Table 7: Specialization

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| spid | int(11) | Primary key | Id of speciality |
| speciality | varchar(100) | Not null | Name of speciality |
| d1 | varchar(100) | Not null | Description of speciality |
| d2 | varchar(100) | Not null | Description of speciality |
| icon | varchar(100) | Not null | icon of speciality |

Table 8: Schedule

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| Schedule\_id | int(15) | Primary key | Id of schedule |
| date | varchar(255) | Not null | date of schedule |
| Doctor\_id | int(15) | Not null | Id of doctor |
| start | varchar(255) | Not null | Start number of schedule |
| end | varchar(255) | Not null | End number of schedule |
| tokencount | int(11) | Not null | tokencount of schedule |

Table 9: Appointment

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| appoinment\_id | int(15) | Primary key | Id of appoinment |
| User\_id | int(15) | Not null | Id of user |
| Schedule\_id | int(15) | Not null | Id of schedule |
| tid | int(11) | Not null | Id of token |
| status | Varchar(10) | Not null | Status of the appoinment |

Table 10: Diet

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| diet\_id | int(15) | Primary key | Id of appoinment |
| Food\_type | Varchar(25) | Not null | Id of user |
| Food\_name | Varchar(25) |  |  |
| ingredient | Varchar(25) | Not null | Id of schedule |
| Recipe\_description | Varchar(500) | Not null | Id of token |
| d\_photo | Varchar(200) | Not null | Status of the appoinment |

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| Tip\_id | int(15) | Primary key | Id of tip description |
| Disease\_type | Varchar(25) | Not null | Type of disease |
| description | Varchar(500) | Not null | Description of disease and tip |

Table 11: Tip

Table 12: Complaint

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| complaint\_id | int(15) | Primary key | Id of complaint |
| complaint | Varchar(200) | Not null | Details of complaint |
| reply | Varchar(200) | Not null | Reply for complaint |
| Reply\_date | Varchar(50) | Not null | Date of reply |
| Sub\_id | int(15) | Not null | Id of |

Table 13: Emergency contact

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| eid | int(11) | Primary key | Id of emergency contact |
| uid | int(11) | Not null | id of user |
| eph | bigint(20) | Not null | Phone number of contact users |

Table 14: News

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| news\_id | int(15) | Primary key | Id of news |
| photo | Varchar(100) | Not null | photo of news |
| Disaster\_type | Varchar(25) | Not null | News type |
| place | Varchar(25) | Not null | Place of news happening |
| date | date | Not null | Date of news |
| description | Varchar(500) | Not null | Details of news |

Table 15: Natural disaster

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| ndst | int(15) | Primary key | Id of natural disaster |
| ndst\_type | varchar(25) | Not null | type of natural disaster |
| image | varchar(100) | Not null | photo of natural disaster |
| ndst\_url | mediumtext | Not null | External link of natural disaster |

Table 16: Tool delivery

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| master\_id | int(15) | Primary key | Id of master |
| User\_id | int(15) | Not null | Id of user |
| date | date | Not null | Date of delivery |
| status | varchar(25) | Not null | Status of process |
| Deliver\_status | varchar(25) | Not null | Status of delivery |

Table 17: Self-defense tool

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| Self\_id | int(15) | Primary key | Id of tool |
| Self\_name | varchar(25) | Not null | Name of tool |
| image | varchar(100) | Not null | Image of tool |
| price | int(100) | Not null | Price of tool |
| description | varchar(500) | Not null | Description of tool |

Table 18: Tool

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| Sub\_id | int(15) | Primary key | Id of tool |
| Master\_id | int(15) | Not null | Name of tool |
| Self\_id | int(15) | Not null | Image of tool |
| counts | int(15) | Not null | Price of tool |

Table 19: Women safety

| **Field Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| women\_id | int(15) | Primary key | Id of women safety visuals |
| title | varchar(25) | Not null | title of women safety visuals |
| url | mediumtext | Not null | url of women safety visuals |

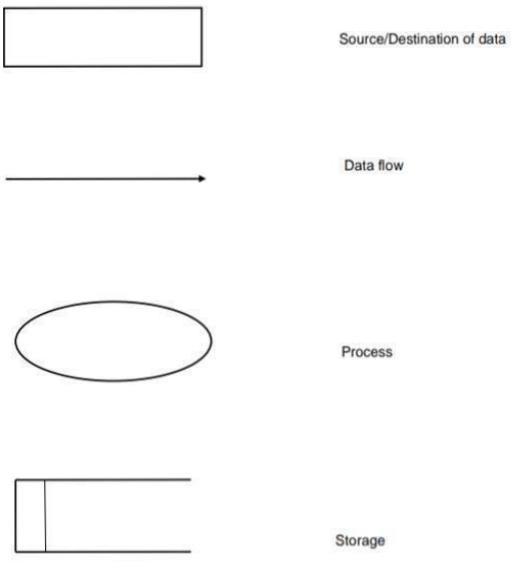
###### 3.2 ARCHITECTURAL DESIGN

The architectural design develops a modular program structure and represents the control relationships between modules. It also defines interfaces that enable data to flow through the program.

###### 3.2.1DATAFLOW DIAGRAMS

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system. DFD can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kind of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored.

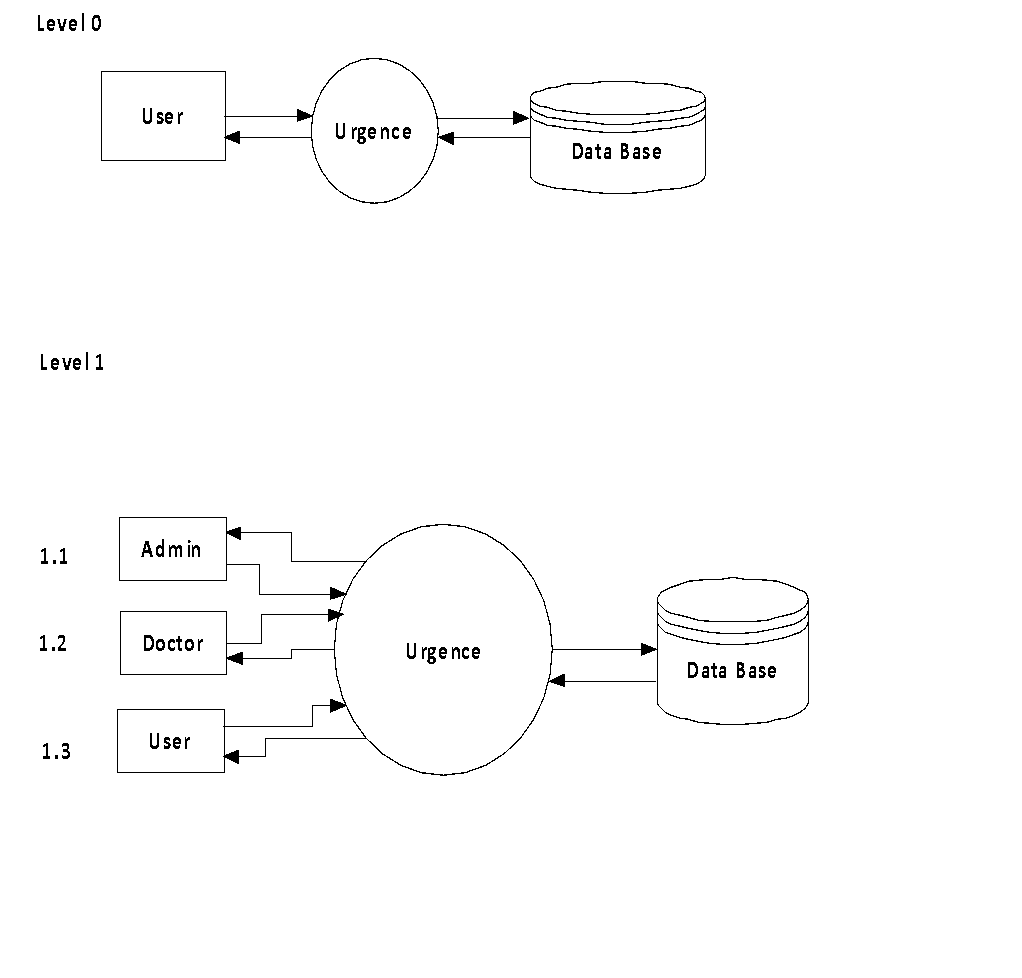
**Data flow diagram symbol**



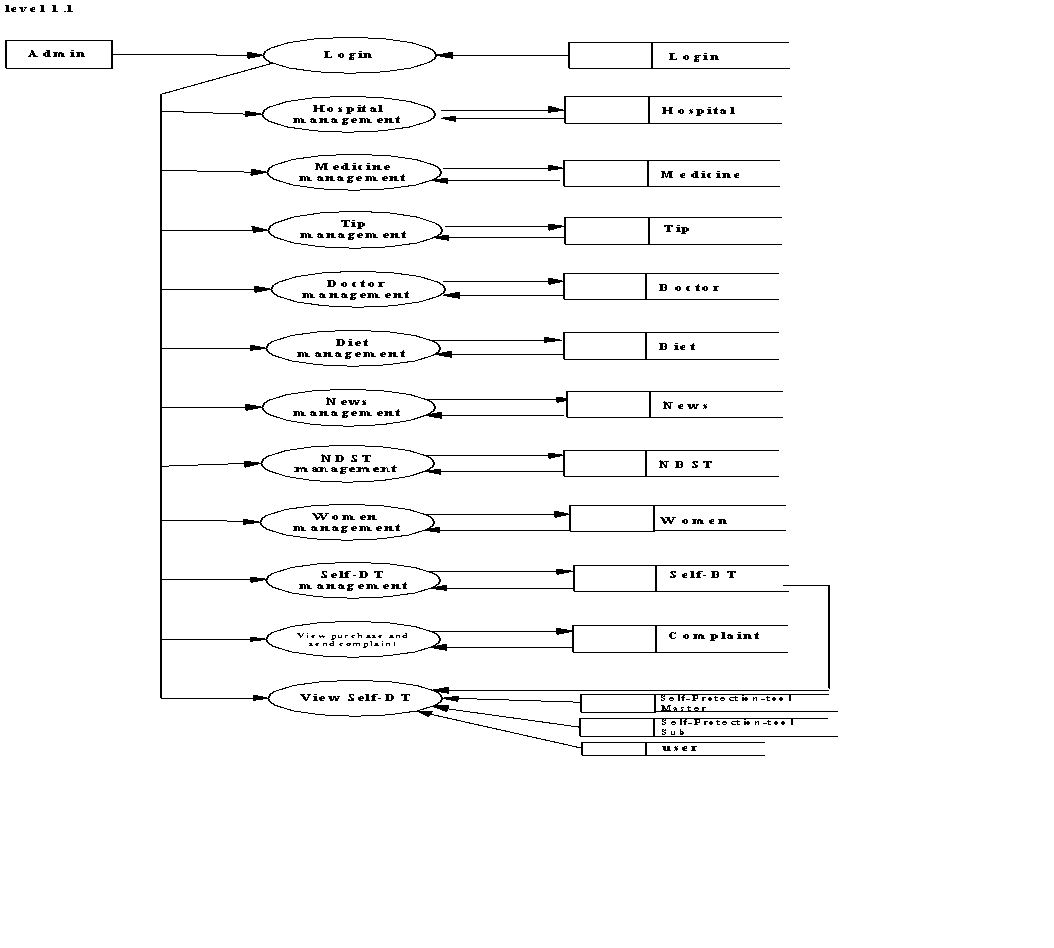
The rules used in constructing data flow diagram are as follows:

* + Process should be named and numbered.
  + The direction of flows is from top to bottom & from left to right.
  + After exploding, lower-level details of process are to be numbered.
  + The name of data stores, sources, destinations are written in upper cases processes & data flow name have first letter of each word capitalized.

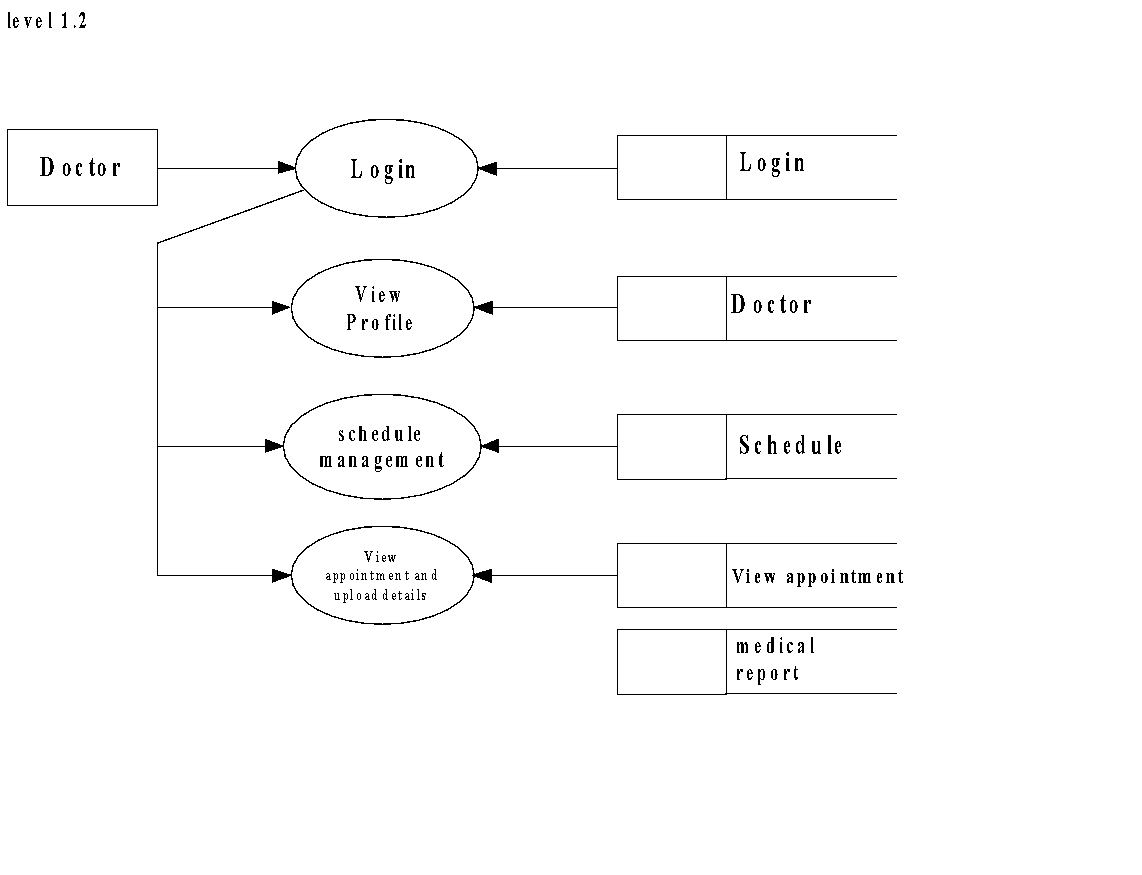
###### LEVEL 0



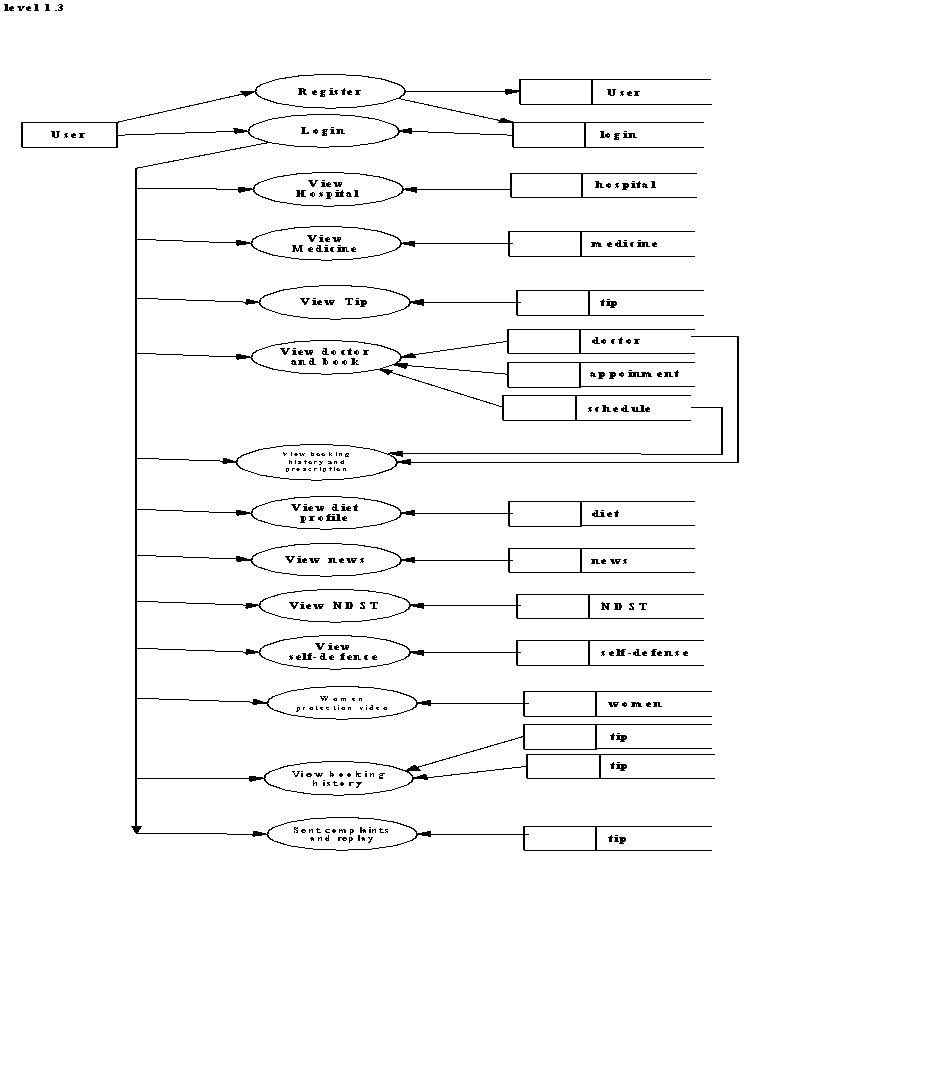
**ADMIN**



**DOCTOR**



**User**



**3.3. INTERFACE DESIGN**

An interface design element for the software tells how information flows into and out of the system and how it is communicated among the components as part of the architecture.

###### 3.3.1. INPUT DESIGN

The input design is the process of converting the user-oriented inputs into the computer-based format. It also includes determining the record media, method of input speed of capture, and entity into the system. Inaccurate input data are the most common cause of errors in data processing. Errors entered by the data entry operator can be controlled by input design. The goal of designing input data is to make the processing easy and free from errors. Input design is a part of overall design, which requires careful attention. The major objective of the input design is to make the data entry easier, logical and error free. With this objective the screen for the system is developed. Input design is the link between the information system and users and those steps that are necessary to put transaction data into a usable or for processing data entry.

Instructing the computer to read data from a written printed document can active the activity of putting data into the computer for processing or it can occur by keying data directly into the system. The design of input focusing on controlling the errors, avoid delay, and keeping the process simple.

System analyst decides the following input design details.

* + What data to input?
  + What medium to use?
  + How the data is arranged and code?

The following are the input forms present in this project:

* + - Login Form
    - Registration Form
    - Add Hospital
    - Add Doctor
    - Add medicine
    - Add medical report
    - Send Reply
    - Add Complaints
    - Add notification
    - Add natural disaster
    - Add news
    - Update tool details
    - Update tool delivery
    - Update women’s self-defense visuals
    - Update Property
    - Send Request
    - Payment management
    - Schedule Date appointment
    - Update token booking
    - Add tips nots
    - Add diet chart

E.g.: **Input design for User registration.**

###### 3.3.2 OUTPUT DESIGN

A quality output is the one, which meets the requirements of the end user and present the information clearly in any system results of processing given to the users through outputs. In the output design it is determined how the information is to be displayed for immediate need. Output design should improve the relationship of the system with the use and help in decision making. The objective of the output design is to define the format of all printed documents and of the system screens that will be produced by the system. The output has been designed as per the need of the institution. The suggestions of the users are also taken into consideration while designing the layouts and the field that are to be included in the project. When designing the output, system analyst must accomplish the following:

* + Determine the information present.
  + Decide whether to print, display the information and select output

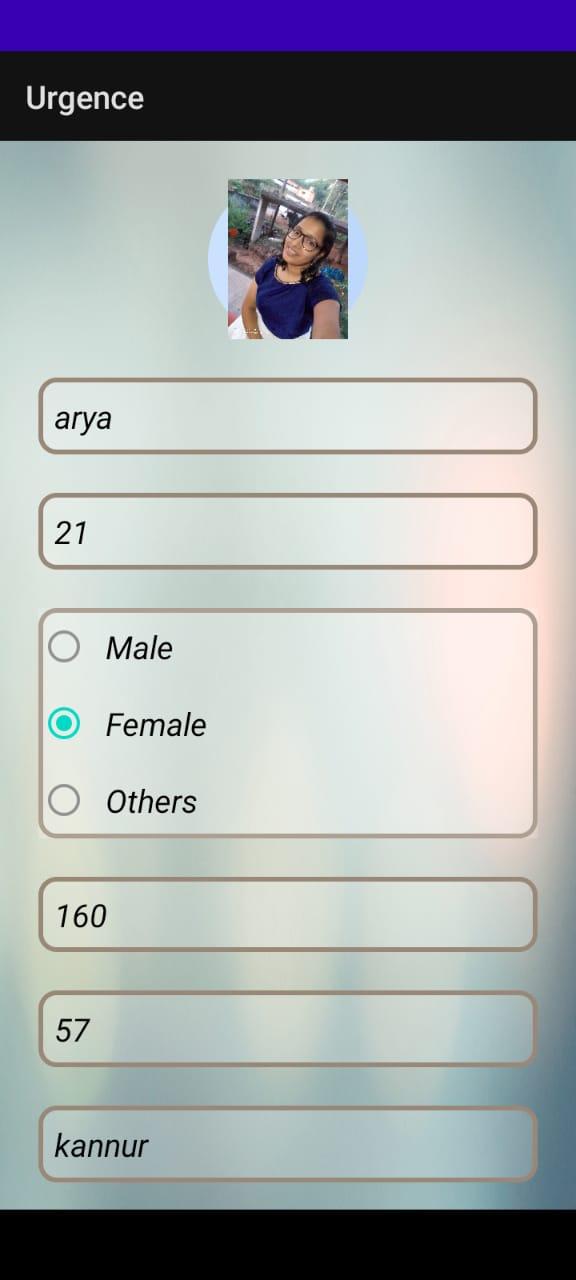
medium.

* + Arrange information in acceptable format

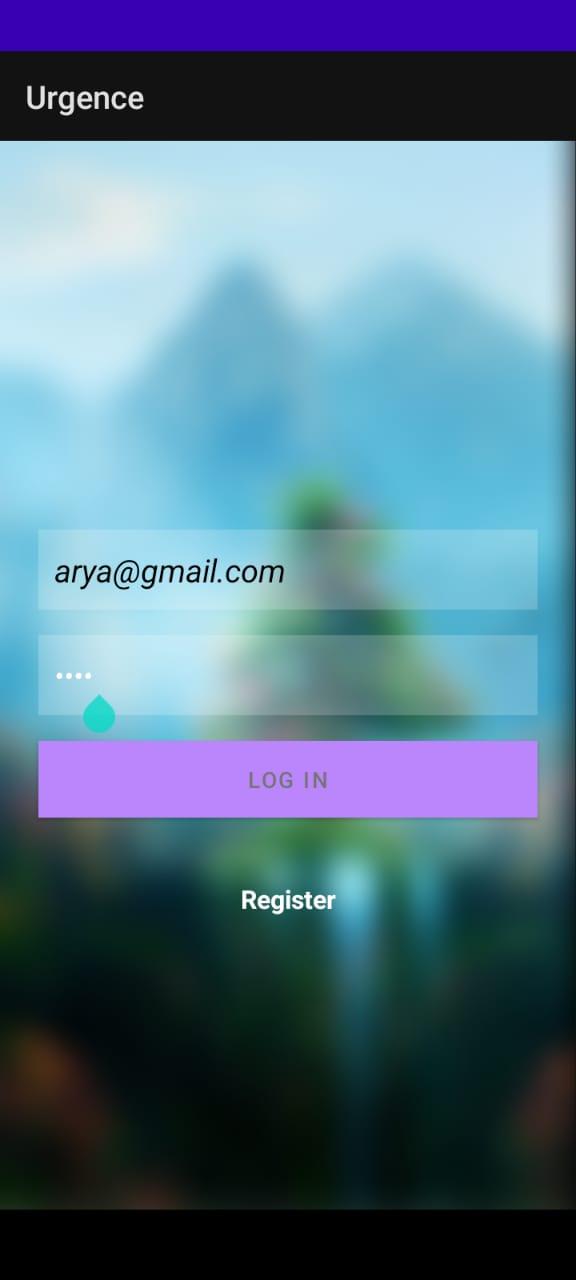
The following are the output forms present in this project:

* + View hospital
  + View doctor
  + View User
  + View medicine and medical report
  + View Notification
  + View Complaints
  + View diet chart
  + View tips nots
  + View self-defense tool
  + View contact list
  + View booking
  + View nature disaster protection methods
  + View news
  + View women safety methods

#### E.g.: Output design for User Profile



**E g. Login page**



**CODING**

###### Coding

###### 4.1. SOFTWARE DESCRIPTION

#### FRONT END

**HTML **

Hypertext Markup Language (HTML) is the standard markup 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. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

##### 

##### **JavaScript**

JavaScript s a high-level, interpreted scripting language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object orientation, and first-class functions. Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

The terms Vanilla JavaScript and Vanilla JS refer to JavaScript not extended by any frameworks or additional libraries. Scripts written in Vanilla JS are plain JavaScript code. Google’s Chrome extensions, Opera's extensions, Apple's Safari 5 extensions, Apple's Dashboard Widgets, Microsoft's Gadgets, Yahoo! Widgets, Google Desktop Gadgets, and Serence Klipfolio are implemented using JavaScript.

##### **Css**

##### 

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

CSS information can be provided from various sources. These sources can be the web browser, the user and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium. The style sheet with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style. The process is called cascading.

One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

**BACK END**

**Python **

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source[32] reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

**Flask (Framework)**

**Flask** is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

### History of Flask

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brandl created a bulletin board system written in Python in 2004, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.



**MySQL**

MySQL is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL).Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL is pretty easy to master in comparison with other database software like Oracle Database, or Microsoft SQL Server. MySQL can run on various platforms UNIX, Linux, Windows, etc. You can install it on a server or even in a desktop. Besides, MySQL is reliable, scalable, and fast. The official way to pronounce MySQL is My Ess Que Ell, not My Sequel. However, you can pronounce it whatever you like, who cares?

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation).In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress. MySQL is also used by many popular websites, including Facebook, YouTube, Twitter and so on.

### 4.2 CODING PRINCIPLE

The input to the coding phase is the design document. During coding phase, modules identified in the design document are coded according to the module specification. Objectives of coding phase are, to transform design into code and unit test the code.

###### Coding Guidelines

* Code should be easy to understand.
* Don’t take pride in cryptic code.
* Code should be well documented.
* Comments should be presents.
* Functions should be small.
* Do not use goto statement

###### LOGIN FORM

{% extends 'login home.html' %}

{% block body %}

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>Untitled Document</title>

</head>

<body>

<form id="form1" name="form1" method="post" action="">

<table width="200" border="4" class="table-active" style="color: #1d2124" cellpadding="20" cellspacing="20">

<tr>

<th scope="row">user name</th>

<td><label for="textfield"></label>

<input type="text" name="textfield" id="textfield" class="form control" /></td>

</tr>

<tr>

<th scope="row">password</th>

<td><label for="textfield2"></label>

<input type="password" name="textfield2" id="textfield2" class="form control" /></td>

</tr>

<tr>

<th colspan="2" scope="row"><input type="submit" name="button" id="button" value="login" class="btn btn-dark" /></th>

</tr>

</table>

</form>

</body>

</html>

{% endblock %}

**TESTING**

1. **TESTING**

Testing is the penultimate step of software development. An elaborate testing of data is prepared and the system is using test data while doing testing, errors are noted and correction is made. The users are trained to operate the developed system. Both hardware and software securities are made to run the developed system successfully. System testing is aimed at ensuring the system accurately before live operation commences. Testing is vital to the system. A series of testing are performed for the proposed system before the system is ready for user acceptance testing success of the system. The various steps in testing the system can be listed as below:

1. Running the program to identify any errors that might have occurred while feeding the program into the system.
2. Applying the screen formats to regulate users to extend, so that the screens are comprehensible to the system.
3. Presenting the formats to administration for the purpose of obtaining approval and checking if any modification has to be done. Obtaining feedbacks from customer and analyzing the scope for improvement.
4. Checking the data accessibility from the data server and whether any improvement is needed or not.

The entire testing process can be divided in to four parts

* + Unit testing
  + Integration testing
  + System testing
  + Validation testing

###### UNIT TESTING

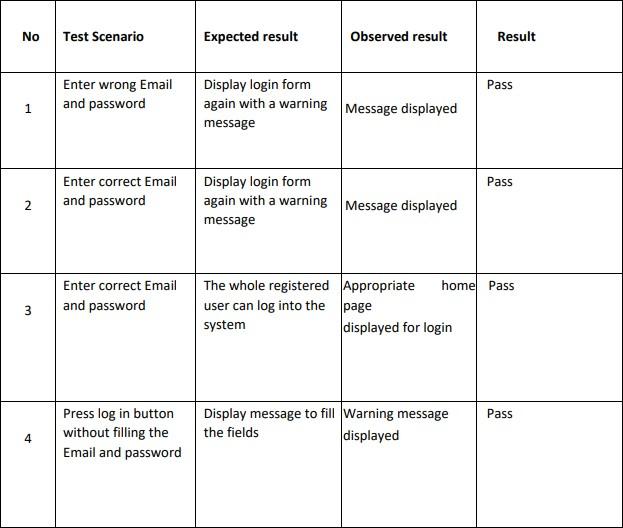
Unit testing focuses verification effort on the smallest unit of software designs the module. To check whether each module in the software works properly so that it gives input. All validations and conditions are tested in the module level in the unit test. Control path are tested to ensure the information property flows into an out of the program unit an out of the program unit under test. Boundary condition is tested to ensure that the modules operate at boundaries. All independent paths through the control structures ensure that all statements in a module have been executed at least once. In unit testing,

* + Module interface is tested to ensure that information properly flows

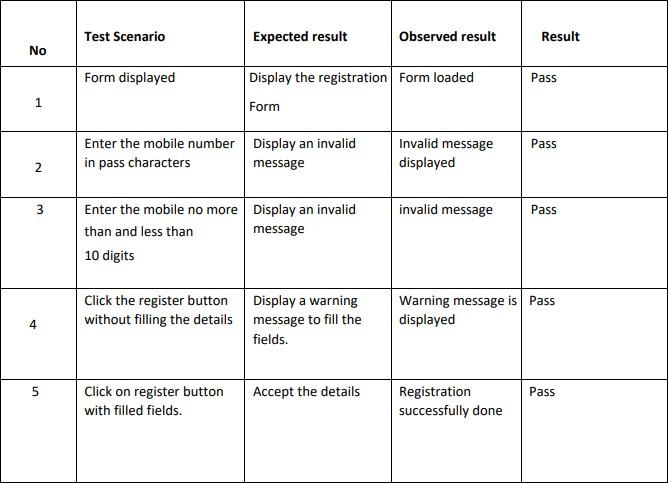
into and out of the program under test.

* + Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in algorithm execution.
  + Boundary Condition is tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
  + All independent paths through the control structures are executed to ensure that all statements in the module have been executed at least once.
  + Error handling paths are also tested.

###### Login Form



**User Registration Form**



1. **BLACK BOX TESTING**

This testing method focuses on the functional requirements of the software. It attempts to find out the errors of the following categories such as incorrect and missing functions, interface error, error in data structures, performance error an initialization.

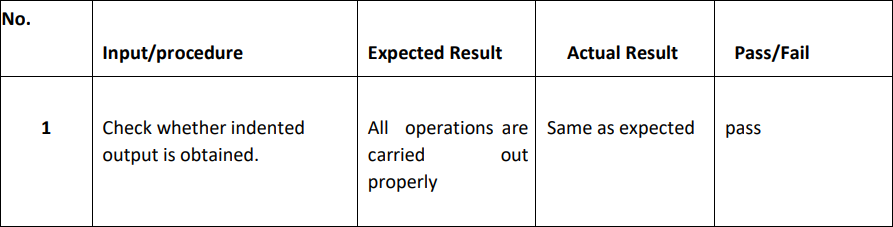
###### WHITE BOX TESTING

This testing method is called path testing. It is a test case design method that uses the control structures to the procedural design to drive test case. In this system, unit testing has been successfully handled. The test data was given to each and every module in all respects and got the desired output. Each module has been tested found working properly.

###### INTEGRATION TESTING

The major concerns of integration testing are developed an incremental strategy that will limit the complexity of the entire actions among components as they are added to the system. Developing components as they are added to the system, developing an implementation and integration schedules that will make the modules available when needed and designing test cases that will demonstrate the validity of the evolving system though each program works individually, they should work together. This also referred to as interfacing.

Each database or table manipulation operation was written as single program was tested again with numerous test data to check for its functionality. The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These design items or group of units, are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystem is exercised through their input interface.



* 1. **DATA VALIDATION TESTING**

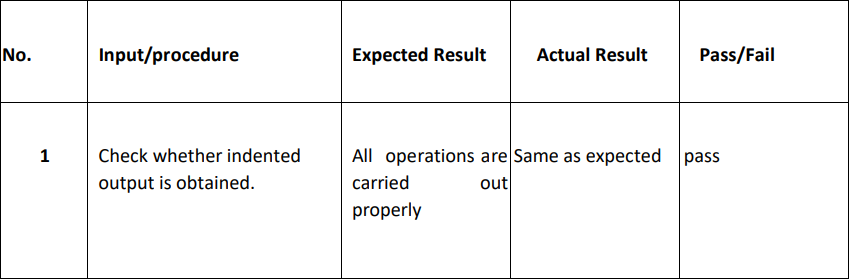
Data validation is one to see whether the corresponding entries made in the tables are correct. Proper validations are one in case of insertions and updating of tables if any case arise the proper error messages or warning, if any, has to be displayed.

###### SYSTEM TESTING

When a system is developed it is hope that it perform properly. The main objectives of system testing are

* To ensure during operation the system will perform as per specification.
* To make sure that the system meets user’s requirements user’s operations.
* To verify that the controls incorporated in the system function as intended.
* To see when correct input are fed to the system the output are correct.

To make sure that operation, incorrect input and output will be detected. System testing is used test the entire system (Integration of the all modules).It also tests to find the discrepancies between the system and the original objective, current specification and system documentation. The entire system is checked to correct deviation to achieve correctness.



# SYSTEM IMPEMENTATION

#### SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the uses that it will work efficiently and effectively. It involves careful planning, investigation of current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation, of change over methods. Apart from planning major task of preparing the implementation are education and training of users. The more complex system being implemented, the more involved will be the system analysis and the design effort required just for implementation, on implementation coordinating committee based on policies of individual organization has been appointed. The implementation process begins with preparing the plan for the implementation for the system. According to this plan, the activities are to be carried out, discussion made regarding the equipment and resources and the additional equipment as to be acquired to implement the new system. The implementation is the final and important phase. The most critical stage in achieving successful new system and in giving the user confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it found to working according to the specification. This method also offers the greatest security. Since the old system can take over if the errors are found or inability to handle certain type of transaction while using new system. **URGENCE** is developed as a Web and Android application. The language selected to program this software is Python. The reason Python is selected is that it is simple and powerful language that is used specially to develop both web and android applications.

**CONCLUSION AND FUTURE ENHANCEMENT**

###### CONCLUSION AND FUTURE ENHANCEMENT

###### CONCLUSION

The Urgence mobile application is an innovative and user-friendly solution for emergency and medical assistance that is designed to provide quick and efficient access to medical services and assistance in times of emergency. The application is built using advanced technologies such as HTML, CSS, and Bootstrap in the frontend and Python Flask in the backend.

With its three distinct modules, the Urgence application provides comprehensive functionality to its users, including administrators, doctors, and regular users. The administrator module is responsible for managing all activities within the application, including creating and updating user profiles, providing access to medical services, and controlling the overall functioning of the application.

The user module enables regular users to access a range of functionalities, including purchasing products and services, viewing medical details, and triggering the SOS functionality in case of an emergency. Users can also access a range of additional features such as diet plans, natural disaster management, self-defense tips, women protection functionality, news feed, and other essential services.

The doctor module is designed to provide medical professionals with a dedicated platform to offer their services to users of the Urgence application. Doctors can provide medical services, access medical records of their patients, and communicate with other medical professionals to provide the best possible care to patients.

In conclusion, the Urgence mobile application is a powerful tool for providing critical emergency and medical services to users. Its comprehensive functionality, user-friendly interface, and advanced features make it an indispensable tool for anyone looking to access emergency medical assistance quickly and efficiently. The application's advanced technology and efficient design provide a great user experience, making it easier for users to navigate the application and access the features they need.

###### FUTURE ENHANCEMENT

Another potential enhancement is to expand the geographical coverage of the application to make it accessible to more users worldwide. This could involve partnering with local healthcare providers, hospitals, and emergency services to provide a more comprehensive network of medical services and assistance.

Integrating the application with wearable devices such as smartwatches or fitness trackers could also enable the application to monitor users' health and provide immediate medical assistance in case of an emergency. This would make the application more proactive and responsive to users' needs, potentially saving lives in critical situations.

Furthermore, adding more features such as real-time updates on natural disasters, pandemic information, and health tips could make the application more useful for its users. This would allow users to stay informed and prepared in times of crisis, reducing the risk of injury or harm.

In conclusion, the Urgence mobile application has great potential for future enhancements and opportunities. Incorporating more advanced technologies, expanding the geographical coverage, integrating with wearable devices, and adding more features could make the application even more powerful and useful for its users. As technology continues to evolve, the Urgence application will have the opportunity to incorporate new advancements and features to provide even more comprehensive emergency and medical services to users.

## REFERENCES

#### REFERENCE

###### WEBSITES

* + [http://www.mysql.com](http://www.w3schools.com/)
  + [http://www.w3schools.com](http://www.w3schools.com/)
  + [http://www.google.com](http://www.google.com/)
  + [http://www.answers.com](http://www.answers.com/)
  + http://[www.tutorialspoint.com](http://www.tutorialspoint.com/)
  + [http://www.codeproject.com](http://www.codeproject.com/)

###### REFERENCE BOOKS

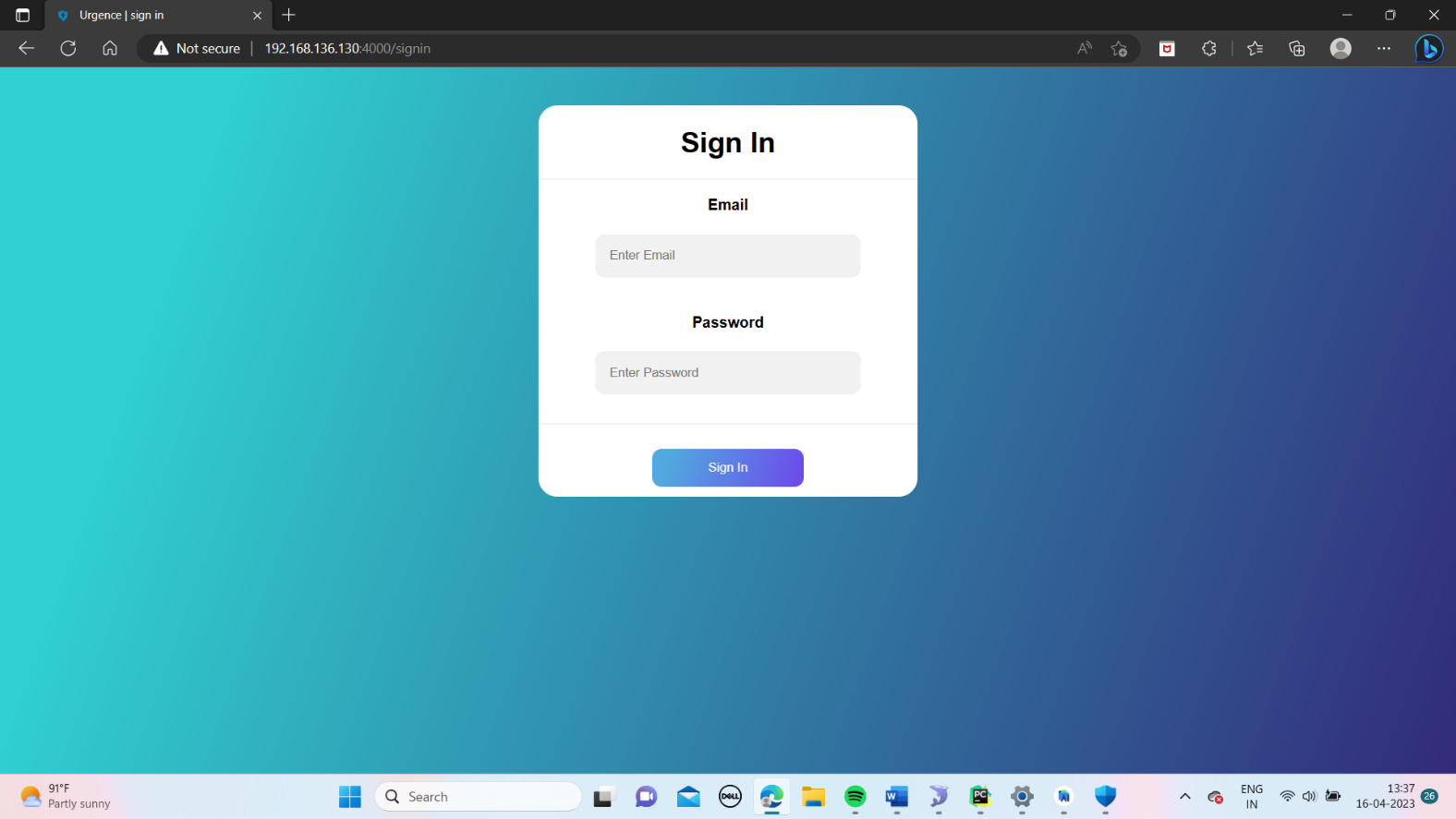
Æ Rogers Pressman, “Software Engineering”, 5th edition by McGraw-Hill Publications 1987

Æ Silberschatz Korth, Sudarshan, “Database System Concepts”, 4th Edition McGraw-Hill.

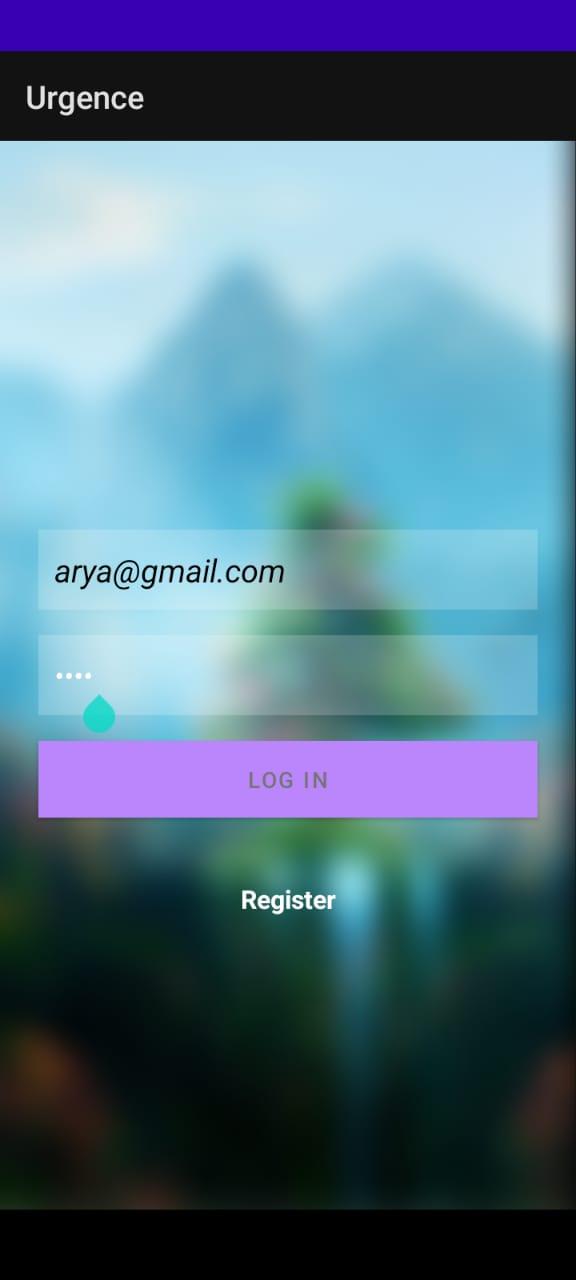
## APPENDIX

## APPENDIX

#### Form for Login



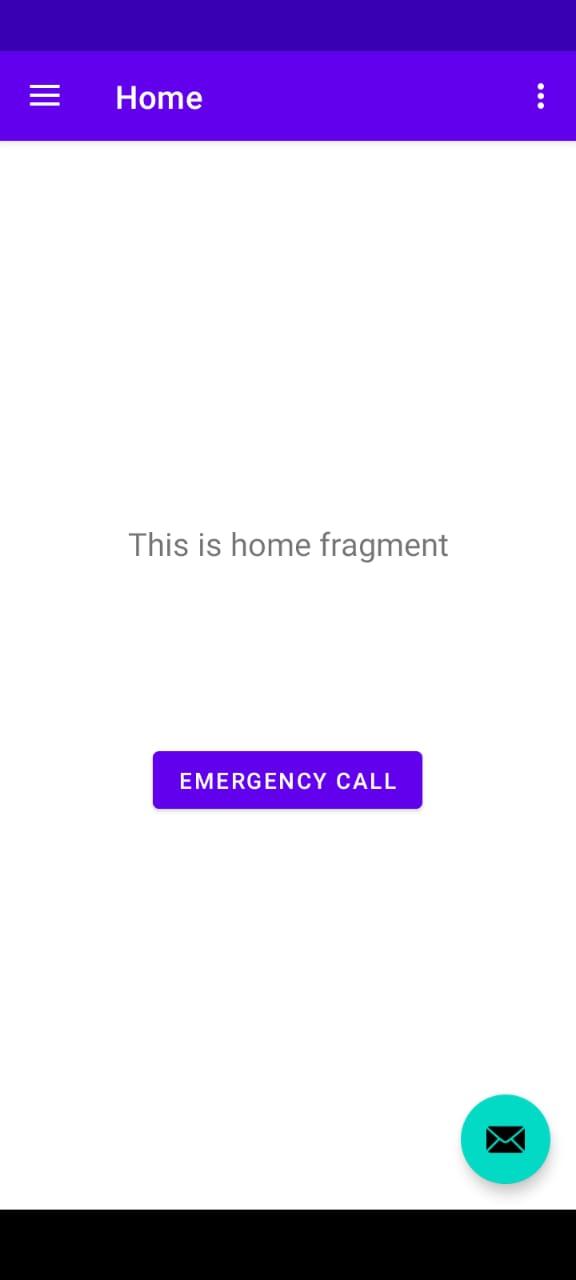
**Form for User Login**



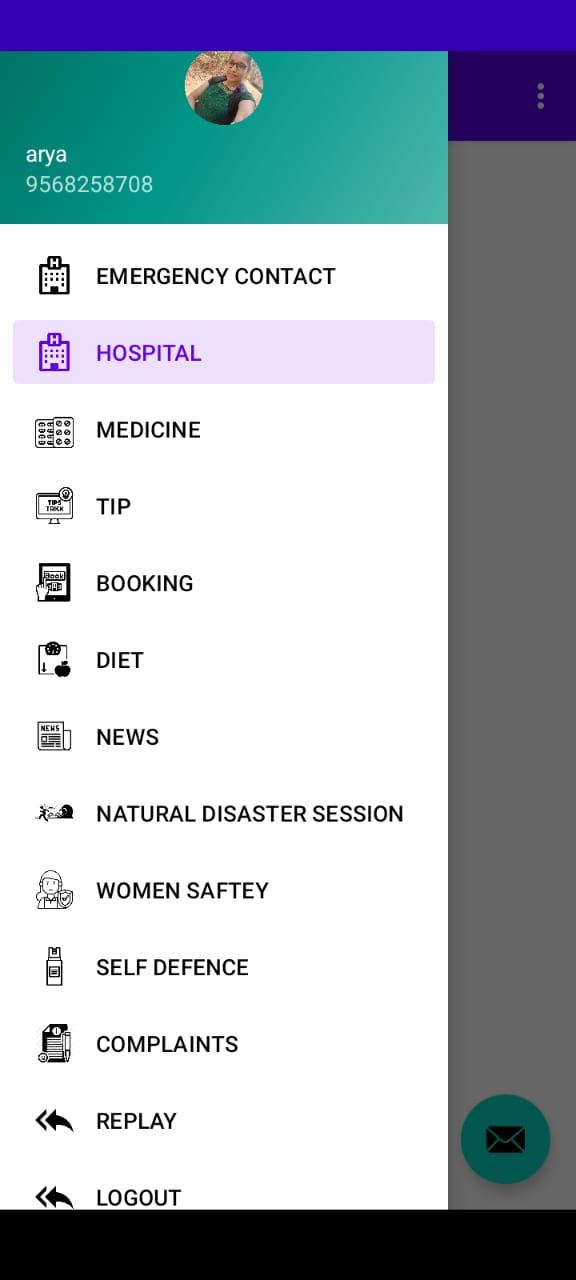
**Form for User Registration**

**Front page of the app**



**Menu bar of the app**



Featurs of the app

