# 1 INTRODUCTION

## 1.1 INTRODUCTION

As the name specifies “Complaint Management System” a software developed for managing complaints . For the past few years the number of educational institutions are increasing rapidly. This particular project deals with the problems on managing complaints  and avoids the problems which occur when carried manually. Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system Which is more user friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks of the existing system. An effective complaints management system is integral to providing quality customer service. It helps to measure customer satisfaction and is a useful source of information and feedback for improving services. Often customers are the first to identify when things are not working properly.

The project tries to accomplish all the following features:

1) Smooth flow of data without any hurdles.

2) Adequate validation checks for data entry.

3) Adequate security of data.

4) Facility to update data from time to time.

5) Prompt and specific retrieval of data.

6) Flexibility in the system according to the changing environment.

7) Controlling redundancy in storing the same data multiple times.

8) Accuracy, timeliness and comprehensiveness of the system output.

9) Stability and operability by people of average intelligence.

10) Enhancement in the completion of work within the constraints of time

The different types of modules present in this project are

1. Admin
2. User

**Admin:**

Manage Servicemens : By Using this software admin can easily manage servicemen , i.e. Admin can verify the servicemen is good or fake by seeing its profile information , admin can also able to delete the servicemen account.

- **Manage Users**
- **Manage Complaints**
- **Manage Services**

Admin can also update his profile, change the password and recover the password.

**User**

- **Register Complaint**
- **View Complaint status**

User can also update his/her own profile and change the password

### 1.2 Problem Statement

This research work was undertaken to uncover the various problems with conventional Complaint management system. These includes;

1. Ineffectiveness in dealing with customer concerns or complaints

2. Incomprehensive complaints history

3. Inconsistency in customer interaction

4. Lack of immediate information storage and retrievals: - The information generated by various transactions takes time and efforts to be stored at right place and accessing such information is time consuming.

5. Lack of prompt updating as to when a complaint issues has been resolved.

## 1.3 Scope and Relevance of the Project

The main scope and deliverables of the project would be to :

- Understand and prepare detailed requirement and specifications
- Prepare high level and detailed design specifications of the system
- Prepare test plan and test cases
- Develop the system and coding
- Perform unit testing, integration testing and system testing
- Demonstrate bug free application after suitable modification if needed

Relevance:

By successfully implementing the project a substantial knowledge has been acquired on the implementation of a database system using net technologies. The knowledge will be useful in the future in creating any type of desktop application or online database systems

## 1.4 Objective

# CHAPTER 2

# SYSTEM ANALYSIS

# INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute's detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

## 2.1 EXISTING SYSTEM

All processes in existing system are handled manually. All the work that is done in the existing system is done by the human intervention. As all the work is done manually, there were a lot of workload on complaint officer and it also increases the maximum chances of errors. This is so slow and time consuming. Due to increase in number of user's the process become more difficult. In the system. This big problem is the searching; sorting and updating of the user/staff data and no any notification method available for giving information to student except the notice board.

## 2.2 PROPOSED SYSTEM

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

1. User friendliness is provided in the application with various controls.

2. The system makes the overall project management much easier and flexible.
3. There is no risk of data mismanagement at any level while the project development is under process.
4. It provides high level of security with different level of authentication.

## 2.3 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, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development. The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features: -

### 2.3.1 Economical Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

The cost of project, **Complaint Management System**was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open source software.

### 2.3.2 Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, 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 requires High Resolution Scanning device and utilizes Cryptographic techniques. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using **HTML, CSS, JAVASCRIPT** in front end and **PYTHON DJANGO, SQLITE** in server in back end, the project is technically feasible for development. The System used was also of good performance of Processor Intel i3 core; RAM 4GB and, Hard disk 1TB.

### 2.3.3 Operational Feasibility

Operational feasibility is concerned with the operating of the system after the installation. This does not require much resources as once the site is hosted, it only requires some fees yearly as server rent..

## 2.5 Software Engineering Paradigm Applied

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

These can be combined into various categories, though each of them is contained in one another:

<u>Software Development Paradigm</u>

This Paradigm is known as software engineering paradigms where all the engineering concepts pertaining to the development of software are applied. It includes various researches and requirement gathering which helps the software product to build. It consists of –

• Requirement gathering

• Software design

• Programming

<u>Software Design Paradigm</u>

This paradigm is a part of Software Development and includes –

• Design

• Maintenance

• Programming

<u>Programming Paradigm</u>

This paradigm is related closely to programming aspect of software development. This includes

• Coding

• Testing

• Integration

# CHAPTER 3

# SYSTEM DESIGN

## 3.1 Introduction

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

1. **Primary Design Phase:**

In this phase, the system is designed at block level. The blocks are created onthe basis of analysis done in the problem identification phase. Different blocksare created for different functions emphasis is put on minimizing the informationflow between blocks. Thus, all activities which require more interaction are keptin one block.

2. **Secondary Design Phase:**

In the secondary phase the detailed design of every block is performed.

The general tasks involved in the design process are the following:

1. Design various blocks for overall system processes.

2. Design smaller, compact and workable modules in each block.

3. Design various database structures.

4. Specify details of programs to achieve desired functionality.

5. Design the form of inputs, and outputs of the system.

6. Perform documentation of the design.

7. System reviews.

## 3.2 DATABASE DESIGN

In designing a database application, you must set up not only the program 's routines f or maximum performance, but you must pay attention also to the physical layout of the data storage. A good database design does the following:

- Provides minimum search times when locating specific records.
- Stores the data in the most efficient manner possible to keep the database from growing too large.
- Makes data updates as easy as possible.
- It is flexible enough to allow inclusion of new functions required of the program.

**Normalization**

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e., repetition of data in the database, maintain data integrity as well a s handling problems that can arise due to insertion, updating, deletion anomalies.

Insertion anomaly: Inability to add data to the database due to absence of other data.

Deletion anomaly: Unintended loss of data due to deletion of other data.

Update anomaly: Data incenses- ten key resulting from data redundancy and partial update. Decomposing is the process of splitting

relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation. Normal Forms are the rules for structuring relations that eliminate anomalies.

**First Normal Form:**

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

**Second Normal Form:**

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

- Primary key is a not a composite primary key.
- No non key attributes are present.
- Every non key attribute is fully functionally dependent on full set of primary keys.

**Third Normal Form:**

A relation is said to be in third normal form if their exits no transitive dependencies.

Transitive Dependency: self-two non-key attributes depend on each other as well as on the primary key then they a re said to be transitively dependent.

The above normalization principles were applied to decompose the data in multiple ta - bless thereby making the data to be maintained in a consistent state. The database is implemented using a DBMS package. Each particular DBMS has unique characteristics and general technique for database design. The application stores the information relevant for processing to SQL database. This SQL database contains tables where each table corresponding to one particular type of information. Each piece of information in a table is called a field for column. A table also contains records, which is a set of fields. These are primary key field that are uniquely identifying a record in a table. There are also fields that contain primary key form another table called foreign key.

**Candidate key**

In the relational model, a candidate key of a relational variable is a set of attributes of that relation variable such that at all times it holds in the relation assigned to that variable that there are no two distinct tuples with the same values for these attributes and there is not a proper subset of this set of attributes f or which holds.

**Primary key**

In relational database design, a unique key to uniquely identify each row in a table. A uniquely key or primary ley comprises a single column or set of columns. No two distinct rows in a table can have the same value in those columns. Depending on its design, a table may have arbitrarily many unique keys but at most one primary key.A unique key must uniquely identify all possible rows that exist in a table and not only the currently existing rows. Examples are social security numbers.

**Foreign key**

In the context of relational databases, a foreign key is a referential constraint between two tables. The foreign key identifies a column or a set of columns in another table. The columns in the referencing table must be the primary key or other candidate key in the referenced table. The values in one row of the referencing columns must occur in a single row in the referenced table. Thus, a row in the referencing table cannot contain values that don ‘t exists in the referenced table.

**3.2.1 TABLE DESIGN**

Table No : 1
Table name : complaint
Primary Key :

| Field name | Data type | size | constraints | Description |
|---|---|---|---|---|
| id | integer | | Primary Key | Complaint id |
| category | varchar(100) | 100 | None | Complaint category |
| subcategory | varchar(100) | 100 | None | Complaint subcategory |
| Complaint_type | varchar(100) | 100 | None | Complaint type |
| state | varchar(100) | 100 | None | State |
| Complaint_details | varchar(300) | 300 | None | Complaint details |
| Complaint_file | varchar(100) | 100 | None | Complaint related file |
| regdate | date | | None | Complaint registration date |
| status | varchar(50) | 50 | None | Complaint status |
| Lastupdation_date | date | | None | Last updation date |
| userid_id | integer | | Foreign Key | Customer id |
| noc | varchar(200) | 100 | None | Nature of complaint |
| remark | varchar(200) | 200 | None | remarks |
| Remark_date | date | N/A | None | Remark published date |

Table no : 2
Table name : auth_user
Primary key : id

| Field name | Data type | size | constraints | Description |
|---|---|---|---|---|
| id | integer | | Primary Key | User id |
| password | varchar(128) | 128 | None | User password |
| last_login | datetime | | None | Last login date and time |
| is_superuser | bool | | None | User permission control |
| username | varchar(150) | 150 | None | User name |
| last_name | varchar(150) | 150 | None | Last name |
| email | varchar(254) | 254 | None | Email id |
| is_staff | bool | | None | Admin site access permission |
| is_active | bool | | None | User active state |
| date_joined | datetime | | None | Date joined |
| first_name | varchar(150) | 150 | None | First name |

Table no:3
Table name :tbl_user
Primary Key: id

| Field name | Data type | size | constraints | Description |
|---|---|---|---|---|
| id | integer | | Primary Key | Registration id |
| address | varchar | 300 | None | User Address |
| state | varchar | 50 | None | State |
| country | varchar | 50 | None | Country |
| pincode | varchar | 20 | None | Pincode |
| userimage | varchar | 100 | None | User image |
| regdate | date | | None | Registration date |
| user_id | integer | | Foreign key | User id |

Table no:3
Table name :state
Primary Key: id

| Field name | Data type | size | constraints | Description |
|---|---|---|---|---|
| id | integer | | Primary Key | Id |
| statename | varchar(100) | 100 | None | State Name |
| creationdate | datetime | | None | Creaation date |

Table no:4
Table name :category
Primary Key: id

| Field name | Data type | size | constraints | Description |
|---|---|---|---|---|
| id | integer | | Primary Key | Category id |
| catname | varchar | 100 | None | Category Name |
| catdes | varchar | 300 | None | Category Details |
| creationdate | datetime | | None | Creation date |

Table no:4
Table name :sub_category
Primary Key: id

| Field name | Data type | size | constraints | Description |
|---|---|---|---|---|
| id | integer | | Primary Key | Sub category id |
| subcategoryname | varchar | 100 | None | Sub category name |
| creationdate | datetime | | None | Creation date |
| categoryid_id | integer | | None | Category d |

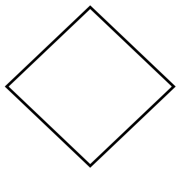
**3.2.2 ENTITY RELATIONSHIP MODEL**

E-R Model is a popular high level conceptual data model. This model and its variations are frequently used for the conceptual design of database application and many database design tools employ its concept.A database that confirms to an E-R diagram can be represented by a collecton of tables in the relational system. The mapping of E-R diagram to the entities are:

•Attributes

•Relations

- Many-to-many
- Many-to-one
- One-to-many
- One-to-one



•Weak entities

•Sub-type and super-type

The entities and their relationshops between them are shown using the following conventions.

•An entity is shown in rectangle.

•A diamond represent the relationship among number of entities.

•The attributes shown as ovals are connected to the entities or relationship by lines.

•Diamond,oval and relationships are labeled.

• Model is an abstraction process that hides super details while highlighting details relation to application at end.

•A data model is a mechanism that provides this abstraction for database application.

•Data modeling is used for representing entities and their relationship in the database.

•Entities are the basic units used in modeling database entities can have concrete existence or constitute ideas or concepts.

•Entity type or entity set is a group of similar objects concern to an organization for which it maintain data,

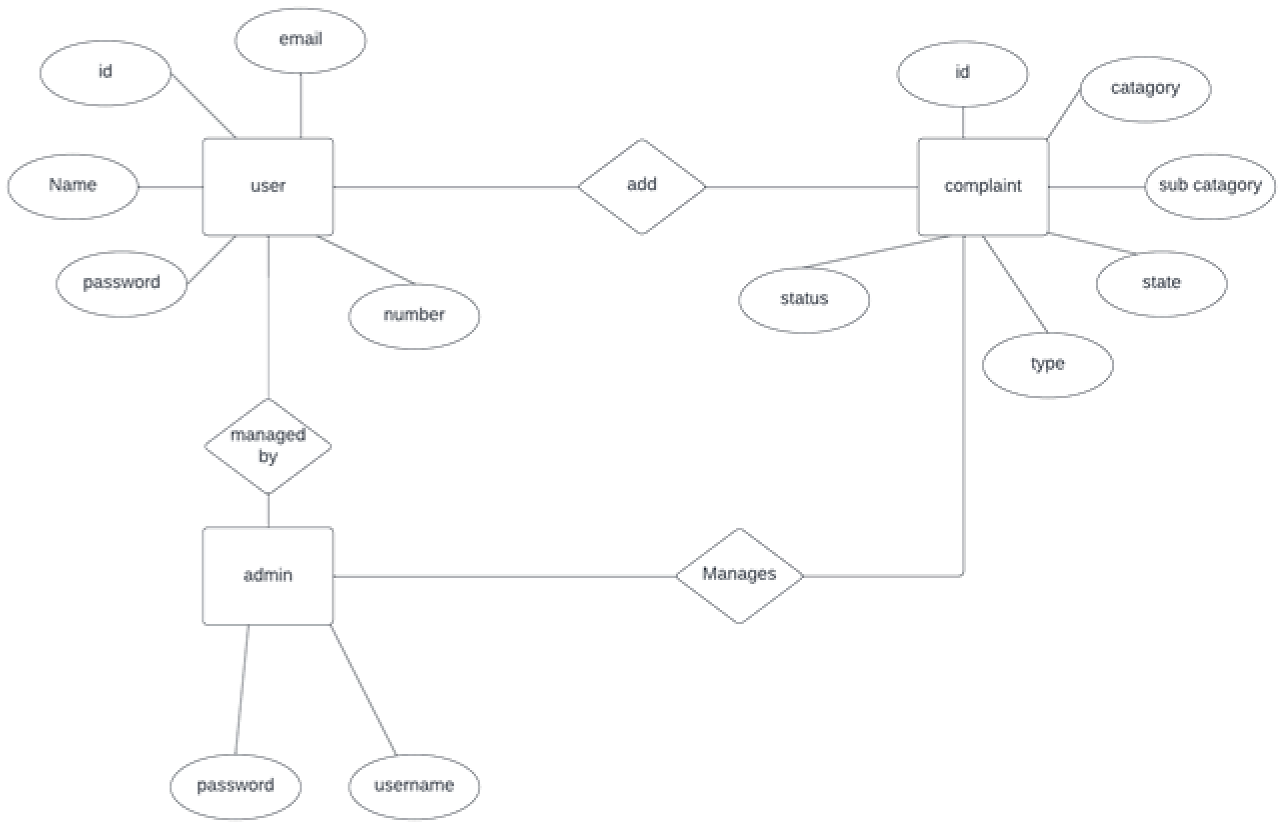
•Properties are characteristics of an entity also called as attributes.

•A key is a single attribute or combination of 2 or more attributes of an entity set is used to identify one or more instances of the set.

•In relational model we represent the entity by a relation and use tuples to represent an instance of the entity.

• Relationship is used in data modeling to represent in association between an entity set.

•An association between two attributes indicates that the values of the associated attributes are independent.



## 3.3 PROCESS DESIGN

### 3.3.1 DATA FLOW DIAGRAM

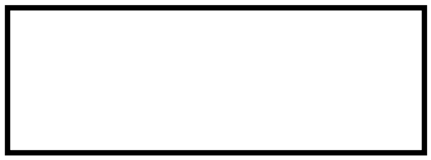
A Data Flow Diagram (DFD) is a graphical representation of the “flow of data” through an information system, modeling its process aspects. A DFD is often used preliminary step to create an overview of the system, which can be later be elaborated. DFDs can also be used for the visualization data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequential or parallel (which is shown on a flow chart). It is common practice to draw the context level. Dataflow flow diagram first, which shows the interaction between the system and external agents which act as data sources and data sinks. This helps to create an accurate drawing in the context diagram. The system’s interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system has a single process, and gives no clues as to its internal normalization. These context level DFD is next exploded to produce a level1 DFD that shows some of details of the system being modeled.

The level1 DFD shows how the system is divided into sub-systems each of which deals with one or more of the data flows to or from an external agent and which together provide all the functionality of the system as a whole. It also identifies internal data stores that must be present in order of the system to its job, and shows the flow of data between the various parts of the system. With its data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. Most data flowing models use four kinds of symbols. These symbols are used to represent four kinds of system components. Processes are Data flow represented by a thin line in the DFD and each data store a unique name, square or rectangle represented external entities. Data flow diagram does not support detailed description of the modules but graphically describes a system’s data and how the data interact with the system. Data flow diagrams are categorized either as logical diagram focuses on the businesses and how the business operates. It is not concerned with how the system will be constructed. Instead it describes events that takes place and the data required and produced by each event. Conversely a physical data flow diagram shows how the system will be implemented including the hardware, software, files and peoples involved in the system. A DFD shows what kind of information will be input to and from the system. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel.

There are different notifications to draw data flow diagram, defining different visual representations for processes, data stores, data flow and external entities.

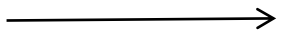
**External entity**

An external entity therefore resides outside the boundaries of the system, meaning that the system has no formal control over data once it has been received by one.



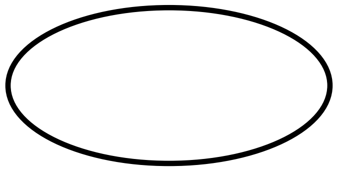
**Data flow**

A dataflow is route, which enables packets of data to travel from one point to another and arrow identifies data flow data in motion. It is a pipe through which information flows.



**Process**

A process represents transformation where incoming data flows are changed into outgoing data flows. A circle or a bubble represents a process that transforms incoming data flows into outgoing data flows.



**Data store**

A data store represents a repository of data that is to be stored for use by one or more processes and may be simple has buffer or queue or sophisticated as a relational database. They should have clear names.

**A source or sink**

A source or sink or person or pan of organizations, which enters or receives information from the system but is considered to be outside the context of data flow model. The DFD methodology is quite effectiveness, especially when the required design is unclear and the user analyst need a notational language for communication. The DFD is easy to understand after a brief orientation. Several rules of thump are used during drawing DFDs:

- Process should be named and numbered for easy reference.
- The direction of flow is from top to bottom and left to right.
- When a process is exploited into lower level details they are numbered.
- The names of data stores, sources and destinations are written in capital letters.

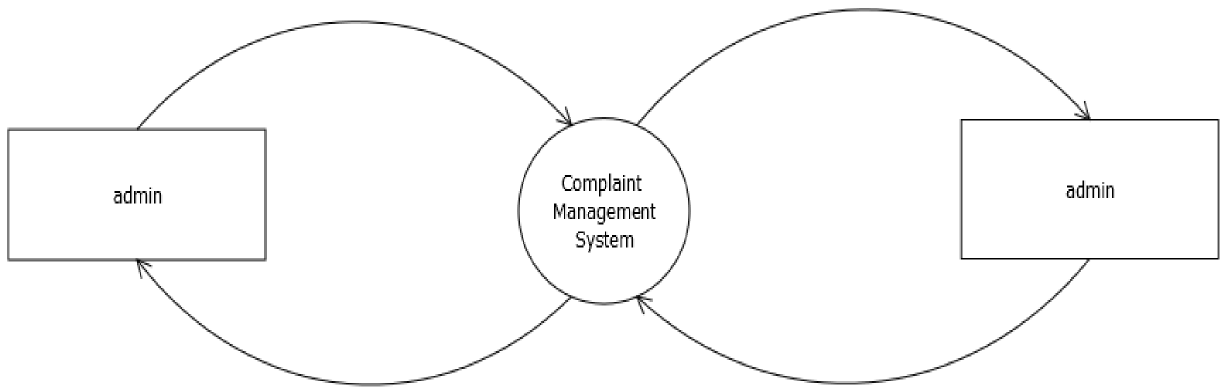
Basic rules for drawing DFD:

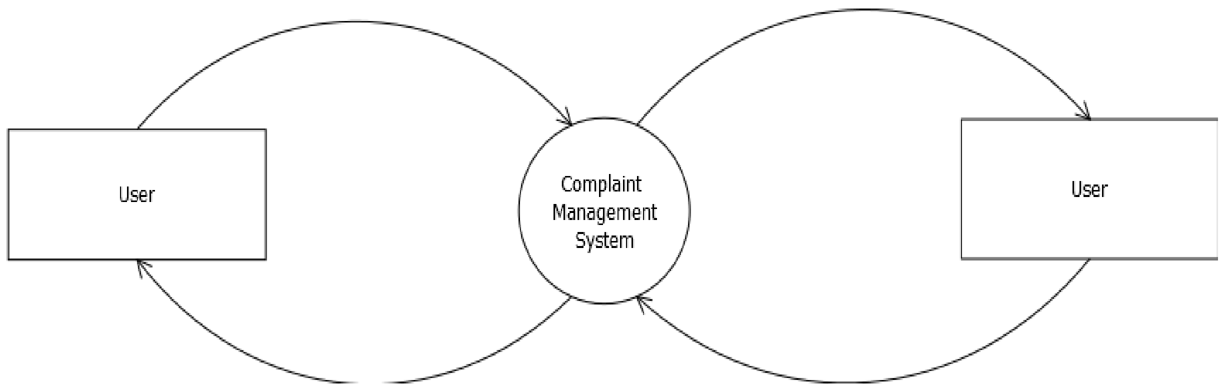
- Process should be name and easy for reference and each process must have a minimum of one data flowing going onto it and one data flowing out it.

- Each data store must have at least one data flowing into it and one data flowing it.
- A data flow out of a process should have some relevance to one or more of the data flows into a process.
- Data stored in system must go through a process data cannot be filled directly from an external entity or retrieved directly by an external entity. This would be classified as a breach of security! A Process or business outside another business should have direct access to another business filling system.

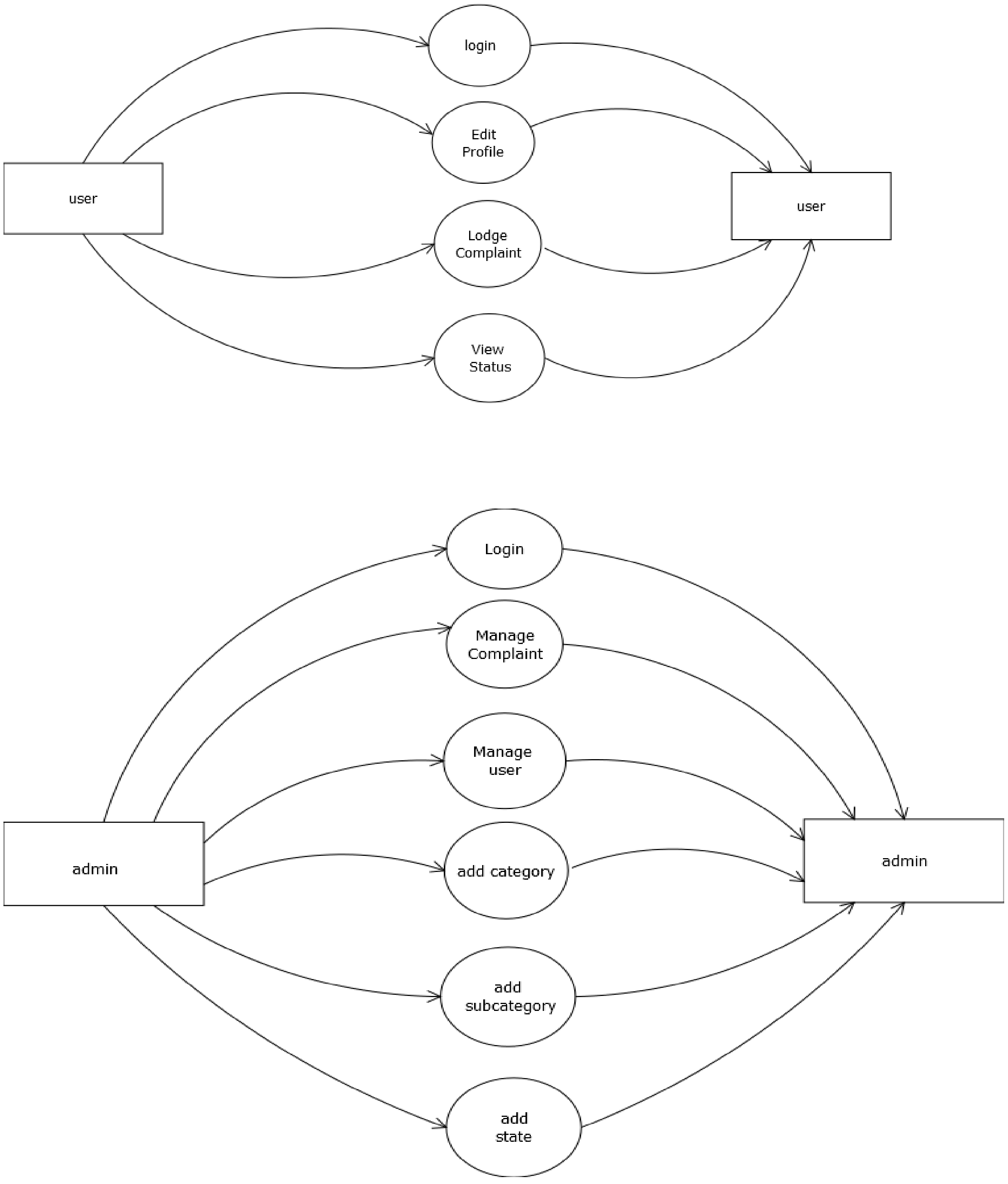
Similarly, filling system within an organization cannot logically communicate with one another unless there is a process involved. One data source cannot have a direct link to another data store.

**<u>CONTEXT LEVEL DFD</u>**



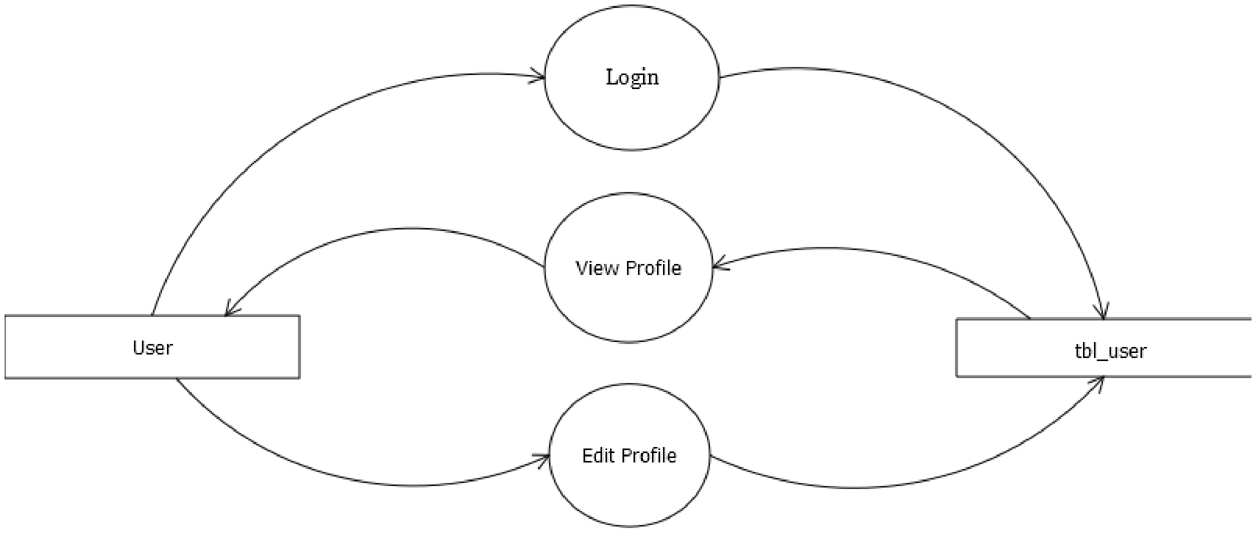


## LEVEL 1 DFD

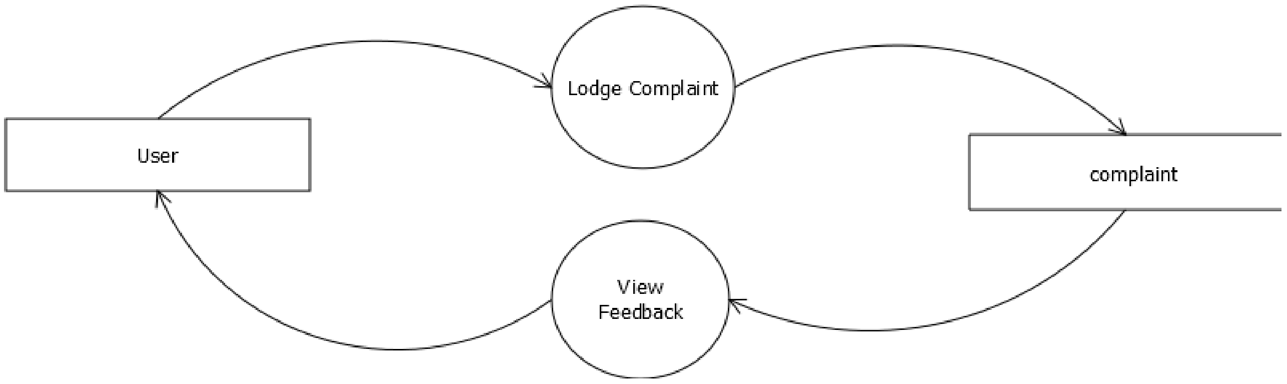

login
Edit Profile
user
user
Lodge Complaint
View Status
Login
Manage Complaint
Manage user
admin
add category
admin
add subcategory
add state

**LEVEL 2 DFD**

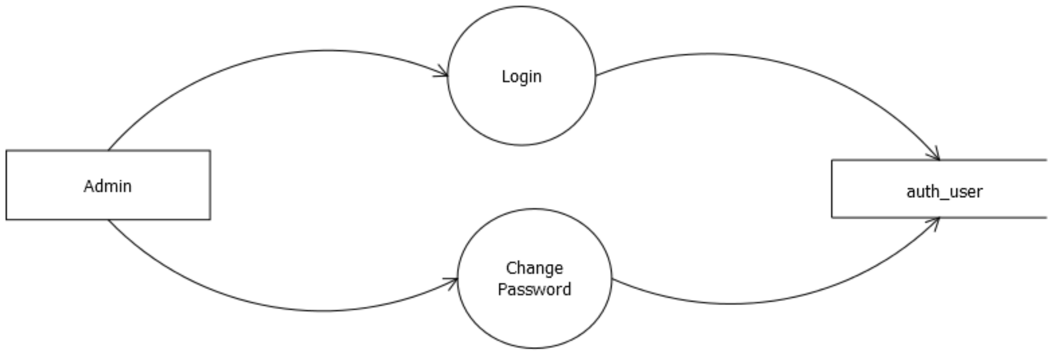
**2.2 Lodge Complaint**



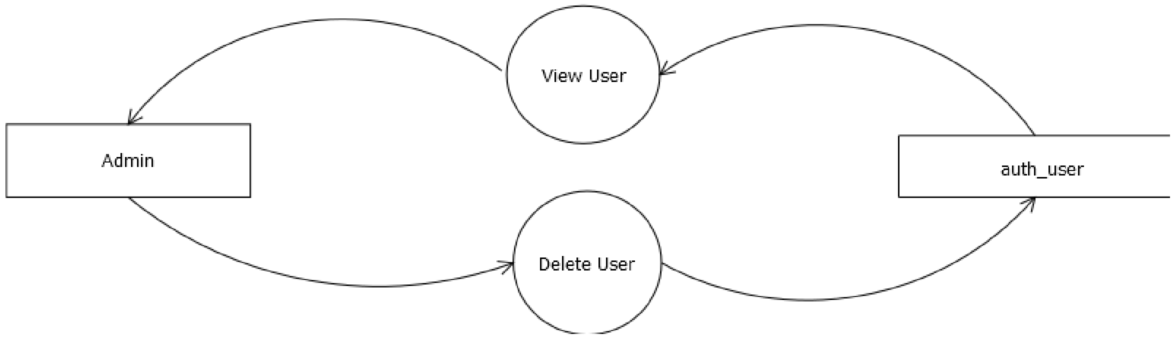
**2.1 Manage Profile**



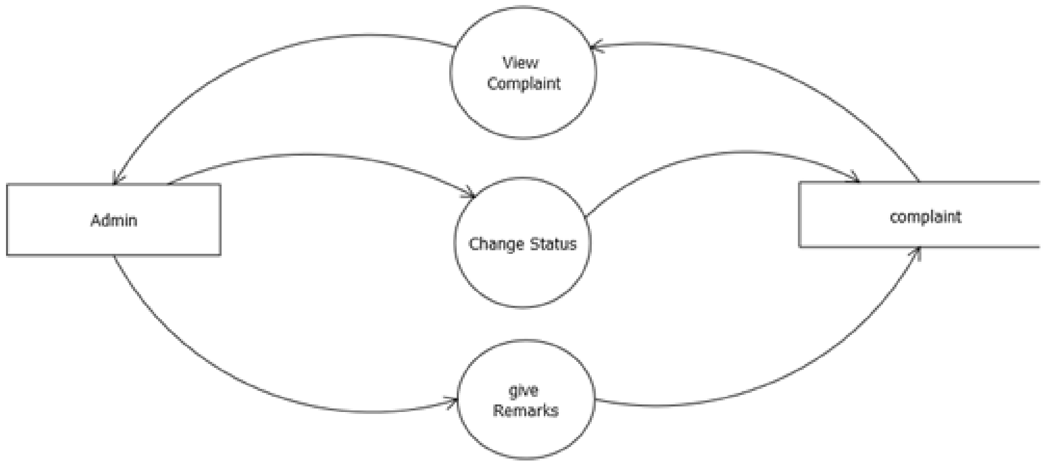
## 2.3 Manage Admin Profile



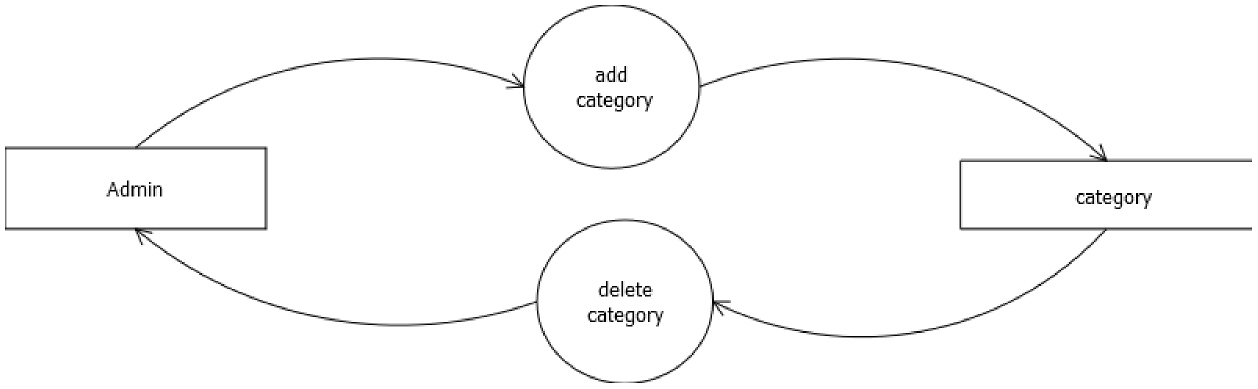
## 2.4 Mange User



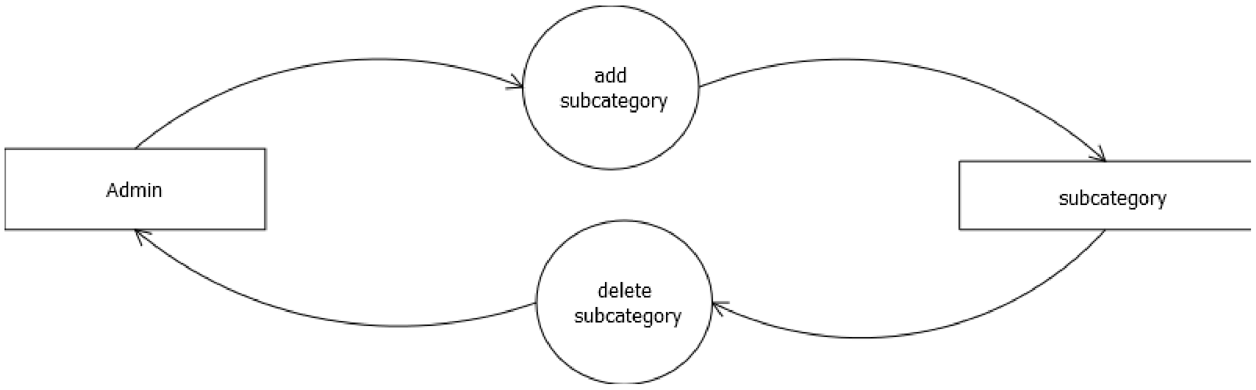
## 2.5 Manage Complaint



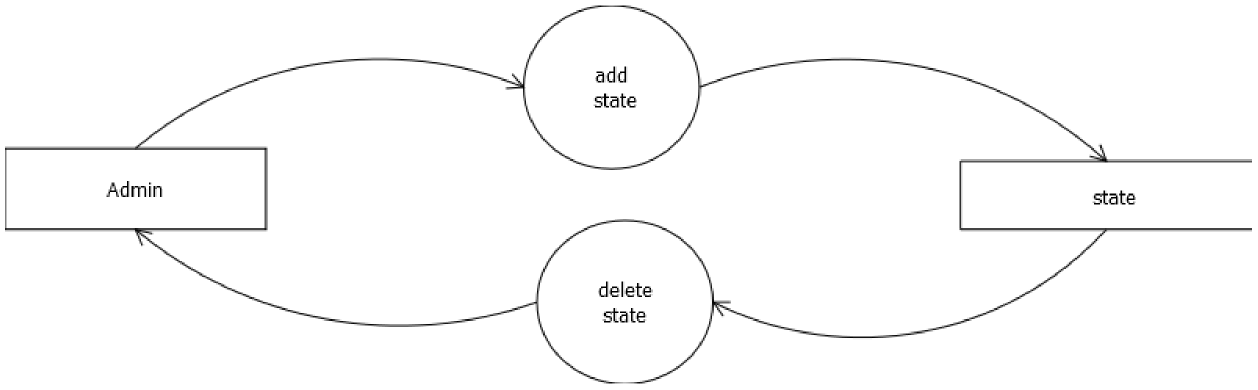
## 2.6 Add Category



## 2.6 Add Subcategory



## 2.7 Add state



## 3.4 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things that are what data should be given as an input, how the data should be arranged or coded, dialog to guide the operating personnel in providing input and methods for preparing input validations and steps to follow when error occur.

## 3.5 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the Information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the systems relationship to help user decision-making.

- Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
- Select methods for presenting information.
- Select methods for presenting information.

# CHAPTER 4

# DEVELOPMENT PHASE

## 4.1 INTRODUCTION

In the phase the organization actually purchases and installs the respective software and hardware in order to support the IT infrastructure. Following this, the creation of the database and actual code can begin to complete the system on the basis of given specifications. In development phase development phase, all the documents from the previous phase are transformed into the actual system.

During the Development Phase, the system developer takes the detailed logical information documented in the previous phase and transforms it into machine executable form, and ensures that all of the individual components of the automated system/application function correctly and interface properly with other components within the system/application. As necessary and appropriate, system hardware, networking and telecommunications equipment, and software components is acquired and configured. New custom-software programs are developed, databases are built, and software components are integrated. Test data and test case specifications are finalized.

Unit and integration testing is performed by the developer with test results appropriately documented. Data conversion and training plans are finalized and user procedures are baselined, while operations, office and maintenance procedures are also initially developed. The Development Phase ends with a Stage Gate Review to determine readiness to proceed to the Test Phase.

The Development Phase includes several activities that are the responsibility of the developer. The developer places the outputs under configuration control and performs change control. The developer also documents and resolves problems and non-conformances found in the software products and tasks. The developer selects, tailors, and uses those standards, methods, tools, and computer programming languages that are documented, appropriate, and established by the organization for performing the activities in the Development Phase. Plans for conducting the activities of the Development Phase are developed, documented and executed. The plans include specific standards, methods, tools, actions, and responsibility associated with the development and qualification.

## 4.2 SOFTWARE REQUIREMENT SPECIFICATION

| | | |
|---|---|---|
| Tool Used | : | Python |
| Database Used | : | Sqlite3 |
| Operating system | : | Microsoft Window 10 |

## 4.3Hardware Requirement Specification

| | | |
|---|---|---|
| Processor | : | Intel Core i3/equivalent |
| RAM | : | 4 GB |
| Hard Disk Drive | : | 500 GB SATA |

## 4.4 Tools ,Platform

### 4.4.1 FRONT END

**HTML**

**HTML** stands for **Hyper Text Markup Language**, which is the most widely used language on Web to develop web pages. **HTML** was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012. Originally, **HTML** was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers. Now, HTML is being widely used to format web pages with the help of different tags available in HTML language. **HTML** is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page. The markup tells the Web browser how to display a Web page's words and images for the user. Each individual markup code is referred to as an element (but many people also refer to it as a tag). Some elements come in pairs that indicate when some display effect is to begin and when it is to end.

**CASCADING STYLE SHEET (CSS)**

Cascading Style Sheets (CSS) are a collection of rules we use to define and modify web pages. CSS are similar to styles in Word. CSS allow Web designers to have much more control over their pages look and layout. For instance, you could create a style that defines the body text to be Verdana, 10 point. Later on, you may easily change the body text to Times New Roman, 12 point by just changing the rule in the CSS. Instead of having to change the font on each page of your website, all you need to do is redefine the style on the style sheet, and it will instantly change on all of the pages that the style sheet has been applied to. With HTML styles, the font change would be applied to each instance of that font and have to be changed in each spot. CSS can control the placement of text and objects on your pages as well as the look of those objects. HTML information creates the objects (or gives objects meaning), but styles describe how the objects should appear. The HTML gives your page structure, while the CSS creates the “presentation”. An external CSS is really just a text file with a .css extension. These files can be created with Dreamweaver, a CSS editor, or even Notepad. The best practice is to design your web page on paper first so you knowwhere you will want to use styles on your page. Then you can create the styles and apply them to your page.

## JAVASCRIPT

JavaScript is a programming language commonly used in web development. It was originally developed by Netscape as a means to add dynamic and interactive elements to websites. While JavaScript is influenced byJava, the syntax is more similar to C and is based on ECMAScript, a scripting language developed by Sun Microsystems. JavaScript is a client-side scripting language, which means the source code is processed by the client's web browser rather than on the web server. This means JavaScript functions can run after a webpage has loaded without COMMUNICATING with the server. For example, a JavaScript function may check a web form before it is submitted to make sure all the required fields have been filled out. The JavaScript code can produce an error message before any information is actually transmitted to the server. Like server-side scripting languages, such as PHP and ASP, JavaScript code can be inserted anywhere within the HTML of a webpage. However, only the output of server-side code is displayed in the HTML, while JavaScript code remains fully visible in the source of the webpage. It can also be referenced in a separate .JS file, which may also be viewed in a browser.

## BACK END

### PYTHON 3.9.1

**Python** is a general purpose, dynamic, High Level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures. Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. Python supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles. Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python was invented by **Guido van Rossum** in 1991 at CWI in Netherland. The idea of Python programming language has taken from the ABC programming language or we can say that ABC is a predecessor of Python language. There is also a fact behind the choosing name Python. Guido van Rossum was a fan of the popular BBC comedy show of that time, **"Monty Python's Flying Circus"**. So he decided to pick the name **Python** for his newly created programming language. Python has the vast community across the world and releases its version within the short period.

- Easy to use and Learn
- Expressive Language
- Interpreted Language
- Object-Oriented Language
- Open Source Language
- Extensible
- Learn Standard Library
- GUI Programming Support
- Integrated
- Embeddable
- Dynamic Memory Allocation
- Wide Range of Libraries and Frameworks

Python is a general-purpose, popular programming language and it is used in almost every technical field. The various areas of Python use are given below.

- Data Science
- Date Mining
- Desktop Applications

- Console-based Applications
- Mobile Applications
- Software Development
- Artificial Intelligence
- Web Applications
- Enterprise Applications
- 3D CAD Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions

## ADVANTAGES OF PYTHON

**1. Easy to Read, Learn and Write**

Python is a high-level programming language that has English-like syntax. This makes it easier to read and understand the code. Python is really easy to pick up and learn, that is why a lot of people recommend Python to beginners. You need less lines of code to perform the same task as compared to other major languages like C/C++ and Java.

**2. Improved Productivity**

Python is a very productive language. Due to the simplicity of Python, developers can focus on solving the problem. They don't need to spend too much time in understanding the syntax or behavior of the programming language. You write less code and get more things done.

**3. Interpreted Language**

Python is an interpreted language which means that Python directly executes the code line by line. In case of any error, it stops further execution and reports back the error which has occurred. Python shows only one error even if the program has multiple errors. This makes debugging easier.

**4. Dynamically Typed**

Python doesn't know the type of variable until we run the code. It automatically assigns the data type during execution. The programmer doesn't need to worry about declaring variables and their data types.

**5. Free and Open-Source**

Python comes under the OSI approved open-source license. This makes it free to use and distribute. You can download the source code, modify it and even distribute your version of Python. This is useful for organizations that want to modify some specific behavior and use their version for development.

**6. Vast Libraries Support**

The standard library of Python is huge, you can find almost all the functions needed for your task. So, you don’t have to depend on external libraries. But even if you do, a Python package manager (pip) makes things easier to import other great packages from the Python package index (PyPi). It consists of over 200,000 packages.

**7. Portability**

In many languages like C/C++, you need to change your code to run the program on different platforms. That is not the same with Python. You only write once and run it anywhere.

## DJANGO

Django is a Python framework that makes it easier to create web sites using Python. Django takes care of the difficult stuff so that you can concentrate on building your web applications. Django emphasizes reusability of components, also refereed to as DRY (Don't Repeat Yourself), and comes with ready-to-use features like login system, database connection and CRUD operations (Create Read Update Delete). Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement. This framework uses a famous tag line: **The web framework for perfectionists with deadlines.**

## SQLITE3

**SQLite** is a database engine written in the C language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. It is the most widely deployed database engine, as it is used by several of the top web browsers, operating systems, mobile phones, and other embedded systems. SQLite has bindings to many programming languages. It generally follows PostgreSQL syntax but does not enforce type checking This means that one can, for example, insert a string into a column defined as an integer. Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. Linking may be static or dynamic. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication.

SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

# CHAPTER 5

# SYSTEM IMPLEMENTATION

## 5.1 Introduction

### 6.1 SYSTEM IMPLEMENTATION

A crucial phase in the system life cycle is the successful implementation of the new system design. Implementation simply means converting a new system design into operation. This involves creating computer-compatible files, training the operating staff, and installing hardware, terminals and telecommunication network before the system is up and running.

In system implementation, user training is crucial for minimizing resistance to change and giving the new system a chance to prove its worth. Training aids, such as user-friendly manuals, a data dictionary, job performance aids that communicate information about the new system, and "HELP" screens, provide the user with a good start on the new system.

The term implementation has different meanings, ranging from the conversion of a basic application to a complete replacement of a computer system. Implementation is used here to mean the process of converting a new or a revised system design into an operational one. Conversion is one aspect of implementation.

Here, implementation of a modified application to replace an existing one, using the same computer is done. The implementation stage involves following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of method to achieve change over.
- Evaluation of the changeover method.

## 5.2 Coding

### 5.2.1 Sample code

**Views.py**

```
from django.db.models import Q

from django.shortcuts import render,redirect

from .models import *

from django.contrib.auth.models import User

from django.contrib.auth import login,logout,authenticate

from django.contrib import messages
from datetime import date

from datetime import datetime, timedelta, time

import random


def index(request):

    return render(request, 'user_login.html')


def admin_login(request):

    error = ""

    if request.method == 'POST':

        u = request.POST['username']

        p = request.POST['password']

        user = authenticate(username=u, password=p)

        try:

            if user.is_staff:

                login(request,user)

                error = "no"

            else:

                error = "yes"


        except:

            error = "yes"

    d = {'error': error}

    return render(request, 'admin_login.html',d)
```

```
def changepassword_admin(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    error = ""
    if request.method == "POST":
        o = request.POST['password']
        n = request.POST['newpassword']
        try:
            u = User.objects.get(id=request.user.id)
            if u.check_password(o):
                u.set_password(n)
                u.save()
                error = "no"
            else:
                error = "not"
        except:
            error = "yes"
    d = {'error': error}
    return render(request,'changepassword_admin.html',d)



def add_category(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    error = ""
    category = Category.objects.all()
    if request.method=="POST":
        cat = request.POST['category']
        des = request.POST['description']
        try:
            Category.objects.create(catname=cat,catdes=des)
            error = "no"
        except:
            error = "yes"
    d = {'error':error,'category':category}
    return render(request, 'add_category.html', d)



def edit_category(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    category = Category.objects.get(id=pid)
    error = ""
    if request.method == 'POST':
        cn = request.POST['category']
        cd = request.POST['description']
        category.catname = cn
        category.catdes = cd
        try:
            category.save()
            error = "no"
        except:
            error = "yes"
    d = {'error': error,'category':category}
    return render(request, 'edit_category.html',d)



def delete_category(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    category = Category.objects.get(id=pid)
    category.delete()
    return redirect('add_category')

def add_subcategory(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    error = ""
    category = Category.objects.all()
    subcategory = SubCategory.objects.all()
    if request.method=="POST":
        cat = request.POST['category']
        subcat = request.POST['subcategory']
        categoryid = Category.objects.get(id=cat)
        try:
            SubCategory.objects.create(categoryid=categoryid,subcategoryname=subcat)
            error = "no"
        except:
            error = "yes"
    d = {'error':error,'category':category,'subcategory':subcategory}
    return render(request, 'add_subcategory.html', d)



def edit_subcategory(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    category = Category.objects.all()
    subcategory = SubCategory.objects.get(id=pid)
    error = ""
    if request.method == 'POST':
        catid = request.POST['category']
        subcat = request.POST['subcategory']
        categoryid = Category.objects.get(id=catid)
        subcategory.categoryid = categoryid
        subcategory.subcategoryname = subcat
        try:
            subcategory.save()
            error = "no"
        except:
            error = "yes"
    d = {'error': error,'category':category,'subcategory':subcategory}
    return render(request, 'edit_subcategory.html',d)


def delete_subcategory(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    subcategory = SubCategory.objects.get(id=pid)
    subcategory.delete()
    return redirect('add_subcategory')



def add_state(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    error = ""
    state = State.objects.all()

    if request.method=="POST":
        s = request.POST['state']

        try:
            State.objects.create(statename=s)
            error = "no"
        except:
            error = "yes"
    d = {'error':error,'state':state}
    return render(request, 'add_state.html', d)



def edit_state(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    state = State.objects.get(id=pid)
    error = ""
    if request.method == 'POST':
        s = request.POST['state']
        state.statename = s

        try:
            state.save()
            error = "no"
        except:
            error = "yes"
    d = {'error': error,'state':state}
    return render(request, 'edit_state.html',d)



def delete_state(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    state = State.objects.get(id=pid)
    state.delete()
    return redirect('add_state')


def Logout(request):
    logout(request)
    return redirect('index')


def registration(request):
    error = ""
    if request.method=="POST":
        fn = request.POST['fullname']
        em = request.POST['email']
        pwd = request.POST['password']
        cn = request.POST['contactno']
        try:
            user = User.objects.create_user(first_name=fn, username=em, password=pwd,last_name=cn)
            tblUser.objects.create(user=user,regdate=date.today())
            error = "no"
        except:
            error = "yes"
    d = {'error':error}
    return render(request, 'registration.html', d)
```

```
def user_login(request):
    error = ""
    if request.method == 'POST':
        u = request.POST['emailid']
        p = request.POST['password']
        user = authenticate(username=u, password=p)

        if user:
            login(request, user)
            error = "no"
        else:
            error = "yes"

    d = {'error': error}
    return render(request, 'user_login.html', d)


def user_home(request):
    if not request.user.is_authenticated:
        return redirect('user_login')
    user = request.user
    tbluser = tblUser.objects.get(user=user)
    count1 = Complaint.objects.filter(status=None,userid=tbluser).count()
    count2 = Complaint.objects.filter(status="in process",userid=tbluser).count()
    count3 = Complaint.objects.filter(status="closed",userid=tbluser).count()
    d = {'count1': count1,'count2': count2,'count3': count3}
    return render(request, 'user_home.html',d)


def profile(request):
    if not request.user.is_authenticated:
        return redirect('user_login')
    user = request.user
    state = State.objects.all()
    userinfo = tblUser.objects.get(user=request.user)
    error = ""
    if request.method == 'POST':
        fn = request.POST['fullname']
        cn = request.POST['contactno']
        addr = request.POST['address']
        st = request.POST['state']
        ct = request.POST['country']
        pc = request.POST['pincode']
        user.first_name = fn
        user.last_name = cn
        userinfo.address = addr
        userinfo.state = st
        userinfo.country = ct
        userinfo.pincode = pc
        try:
            userinfo.save()
            user.save()
            error = "no"
        except:
            error = "yes"
    d = {'state': state,'userinfo':userinfo,'error':error}
    return render(request, 'profile.html',d)



def update_image(request):
    if not request.user.is_authenticated:
        return redirect('user_login')
    user = request.user
    userinfo = tblUser.objects.get(user=request.user)
    error = ""
    if request.method == 'POST':
        img = request.FILES['image']
        userinfo.userimage = img
        try:
            userinfo.save()
            user.save()
            error = "no"
        except:
            error = "yes"
    d = {'userinfo':userinfo,'error':error}
    return render(request, 'update_image.html',d)
```

```
def changepwd_user(request):
    if not request.user.is_authenticated:
        return redirect('user_login')
    error = ""
    if request.method == "POST":
        o = request.POST['password']
        n = request.POST['newpassword']
        try:
            u = User.objects.get(id=request.user.id)
            if u.check_password(o):
                u.set_password(n)
                u.save()
                error = "no"
            else:
                error = "not"
        except:
            error = "yes"
    d = {'error': error}
    return render(request,'changepwd_user.html',d)


def register_complaint(request):
    if not request.user.is_authenticated:
        return redirect('user_login')
    error = ""
    complaint=None
    category = Category.objects.all()
    state = State.objects.all()
    user = request.user
    tbluser = tblUser.objects.get(user=user)
    if request.method=="POST":
        ct = request.POST['category']
        sct = request.POST['subcategory']
        comptype = request.POST['complainttype']
        s = request.POST['state']
        noc = request.POST['noc']
        cd = request.POST['complaindetails']
        cat = Category.objects.get(id=ct)
        scat = SubCategory.objects.get(id=sct)

        complaint = Complaint.objects.create(userid=tbluser,category=cat,subcategory=scat,complainttype=comptype,state=s,complaintdetail=cd,noc=noc,regdate=date.today())
        error = "no"
        try:
            cf = request.FILES['compfile']
            complaint.complaintfile=cf
            complaint.save()
        except:
            pass
    d = {'error':error,'category':category,'state':state,'complaint':complaint}
    return render(request, 'register_complaint.html', d)


def load_subcategory(request):
    category_id = request.GET.get('category')
    subcategory = SubCategory.objects.filter(categoryid=category_id).order_by('subcategoryname')
    return render(request, 'subcategory_list.html', {'subcategory': subcategory})



def complaint_history(request):
    if not request.user.is_authenticated:
        return redirect('user_login')
    user = request.user
    tbluser = tblUser.objects.get(user=user)
    complaint = Complaint.objects.filter(userid=tbluser)
    d = {'complaint':complaint}
    return render(request, 'complaint_history.html', d)


def complaint_details(request,pid):
    if not request.user.is_authenticated:
        return redirect('user_login')


    complaint = Complaint.objects.get(id=pid)
    d = {'complaint':complaint}
    return render(request, 'complaint_details.html', d)



def notprocess_complaint(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    complaint = Complaint.objects.filter(status=None)
    d = {'complaint':complaint}
    return render(request, 'notprocess_complaint.html', d)

def complaint_detailsadmin(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    complaint = Complaint.objects.get(id=pid)

    d = {'complaint':complaint}
    return render(request,'complaint_detailsadmin.html', d)

def update_complaint(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    complaint = Complaint.objects.get(id=pid)
    error = ""
    if request.method == 'POST':
        st = request.POST['status']
        rem = request.POST['remark']
        complaint.status = st
        complaint.remark = rem
        complaint.remarkdate = date.today()
        complaint.lastupdationdate = date.today()
        try:
            complaint.save()
            error = "no"
        except:
            error = "yes"
    d = {'complaint':complaint,'error':error}
    return render(request,'update_complaint.html', d)


def user_profile(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    user = User.objects.get(id=pid)
    tbluser = tblUser.objects.get(user=user)
    d = {'tbluser':tbluser}
    return render(request,'user_profile.html', d)



def manage_users(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    tbluser = tblUser.objects.all()
    d = {'tbluser':tbluser}
    return render(request,'manage_users.html', d)


def delete_user(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    user = User.objects.get(id=pid)
    user.delete()
    return redirect('manage_users')


def inprocess_complaint(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    complaint = Complaint.objects.filter(status="in process")
    d = {'complaint':complaint}
    return render(request, 'inprocess_complaint.html', d)


def closed_complaint(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    complaint = Complaint.objects.filter(status="closed")
    d = {'complaint':complaint}
    return render(request, 'closed_complaint.html', d)
```

### 5.2.2 Code Validation and Optimization

Validation is the status of the project when the theoretical designs turned into a working system. It is used to reduce the number of loops in the program. If the number of loops increases no. of executions also increases. Then there may be a chance for the program to get stuck. It is verified whether the input data entered gets a corresponding and valid and trustable result . On entering the symptom corresponding disease is predicted. Meanwhile the recommendation system will take in the input that the user entered and performs the analyses and the result is passed onto to the result page. Optimization is the last part of the system development life cycle. Code optimization, as the name indicates can be explained as the method that is used for the modification of codes in order to improve the efficiency and quality of the code. As a result of optimization, a program may become lesser in size, consume lesser memory, execute more rapidly, or performs fewer operations (input/output). An optimized program should possess exactly the same outputs and side effects as that of its non-optimized program. However, the benefits are given more weight in comparison to alteration in program behaviour. Optimization can also be referred to as a program transformation technique, performed either by optimizers or programmers. An optimizer is a specialized software or an inbuilt unit of a compiler which is also called as an optimizing compiler. Modern processors are also used to optimize the order of execution of code instructions. Code optimization is generally performed at the end of the developmental stage as it reduces the readability and adds code in order to increase the performance.

# CHAPTER 6

# SYSTEM TESTING

## 6.1 Introduction

Software testing is an integral part of software development to ensure software quality. Some software organizations are reluctant to include in their software cycle, because they are afraid of the high cost associated with the software testing. There are several factors that attribute that cost of software testing. Creating and maintaining large number of tester cases is a time consuming process. Furthermore, it requires skilled and experienced testers to develop greater quality test cases. Even with the wide availability of automation tools for testing, the degree of automation mostly remains at the automated test script level, and generally, significant amount of human intervention is required in testing. Testing provides a good indication of software as a while. The debugging process is the most unpredictable part of testing process.

Testing begins at the module level and work towards the integration of the entire computer based system. No testing is complete without verification and validation part. The goal of verification and validation activities are to access and improve the quality of work products generated during the development and modification of the software. Testing plays a vital role in determining the reliability and efficiency of the software and hence is very important stage in software development. Tests are to be conducted on the software to evaluate its performance under a number of conditions. Ideally, it should do so at the level of each module and also when all of them are integrated to form the complete system.

In the project “INTERACTIVELEARNING” the testing has been successfully handled with the modules. The test data was given to each and every module in all respect and got the desired output. Each module that has been tested is found working properly.

The entire process can be divided into 3 phases

- Unit Testing
- ntegrated Testing
- Final/System Testing

## 6.2 <u>Unit Testing</u>

Here, we test each module individually and integrate the overall system. Unit test focuses verification efforts even in the smallest unit of software design in each module. This is known as ‘Module testing’. The modules of this project are tested separately. This testing is carried out in the programming style itself. In this testing each module is focused to work satisfactorily as regard to expected output from the module. There are some validation checks for the fields. Unit testing gives stress on the modules of ‘admission management system’ independently of one another, to find errors. Here different modules are tested against the specification produced during the design of the modules. Unit testing is done to test the working of individual modules with test SQL servers. Program unit is usually small enough that the programmer who developed it can test it in a great detail. Unit testing focuses first on the modules to locate errors. These errors are verified and corrected and so that the unit perfectly fits to the “Orphanage management system”.

## 6.3 Integration Testing

Data can be lost across an interface, one module can have an adverse effect on the other sub functions, when combined they may not perform the desired functions. Integrated testing is the systematic testing to uncover the errors within the interface. This testing is done with simple data and the developed system has run successfully with this simple data. The need of integrated system is to find the overall system performance. In this whole modules college admission management system are connected and tested.

## 6.4 System Testing

System testing focuses on testing the system as a whole. System testing is a crucial step in quality management process. In the software development life cycle, system testing is the first level where the system is tested as a whole. The system is tested to verify whether it meets the functional and technical requirements. The application/system is tested in an environment closely resembles production environment where the application where the application will be finally deployed.

## 6.5 Test Plan and Test Cases

6.5 Test Plan & Test Cases The objective of system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified and to ensure that the computer system and the associated clerical and other procedures work together. The initial phase of system testing is the responsibility of the analyst who determines what conditions are to be tested, generates test data, produced a schedule of expected results, runs the tests and compares the computer produced results with the expected results with the expected results. The analyst may also be involved in procedures testing. When the analyst is satisfied that the system is working properly, hands it over to the users for testing. The importance of system testing by the user must be stressed. Ultimately it is the user must verify the system and give the go ahead. During testing, the system is used experimentally to ensure that the software does not fail, i.e., that it will run according to its specifications and in the way, users expect it to. Special test data is input for processing (test plan) and the results are examined to locate unexpected results. A limited number of users may also be allowed to use the system so analysts can see whether they try to use it in unexpected ways. It is preferably to find these surprises before the organization implements the system and depends on it. In many organizations, testing is performed by person other than thos ewho write the original programs. Using persons who do not know how certain parts were designed or programmed ensures more complete and unbiased testing and more reliable software.

**CHAPTER 7**

# SYSTEM MAINTAINANCE

## 7.1 INTRODUCTION

System maintenance  is the act of ensuring that your software continues to perform at its peak. You can do this by updating your system's programs, deleting unnecessary files, and running virus scans. System maintenance includes everything from routine check-ups to upgrading hardware, replacing parts of the operating system, repairing software issues, and performing various other tasks. It is used for both personal computers and servers.

System maintenance helps to ensure that your software continues to function properly. By removing unnecessary files, updating the operating system, and running virus scans regularly, you can keep your system in top shape.

System maintenance is a term used by information technology (IT) specialists to describe any type of computer repair. It's not uncommon for data entry and maintenance to take hours, particularly if the work entails transferring unneeded or outdated information. To avoid giving more detailed explanations, pros frequently use broad terms instead of specifics.

The administrators of websites that are attacked by hackers, have technical difficulties or are being updated and repaired frequently display an image expressing regret for the outage and website downtime. Users will be able to comprehend that the website is unavailable and that the administrators are aware of it if you use this method.

Many IT professionals prefer to use scheduled maintenance instead of preventative maintenance because it appears less time-consuming. Maintenance that is scheduled runs smoothly when you can predict when it will be done. It's also good for things like emptying temp files, running defragmentation applications on hard drives, and clearing server caches

## 7.3 MAINTENANCE

System mainteanance is a going activity, which covers a wide variety of activities including, removing program and design errors, updating documentation and test data and updating user support system maintenance is a catch all term used to describe various forms of computer or server maintenance required to keep a computer system running properly, it can describe network maintenance which could mean that servers are being physical repaired, replaced or mode. For the purpose of convenience, maintenance may be categorized into three classes they are:

### 7.3.1 CORRECTIVE MAINTENANCE

This type of maintenance implies removing errors in a program, which might have kept in the system due to faulty design or wrong assumption.

### 7.3.2 ADAPTIVE MAINTENANCE

In adaptive maintenance program functions are changed to enable the information system to satisfy if the information needs of the user.

### 7.3.2 PERFECTIVE MAINTENANCE

In perfective maintenance means adding new programs or modifying the existing programs to enhance the performance of the information system. This type of maintenance under taken to respond to user addition needs which may be due to the changes within or outside of the organic- nation.

# CHAPTER 8

# FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT

## 8.1 INTRODUCTION

Customer complaints can be defined as the expression of customer expectations that have not been met in what business promises in terms of the product or services. It is no longer a secret that online customer reviews and a great online reputation are essential for your marketing success. It has become a common practice for people to check online reviews before buying a certain service or product. According to statistics,85% of consumers trust online reviews as much as personal recommendations.Moreover 74% of people who see customer reviews on a business website say they would contact the business. Besides bringing new clients, gathering customer feedback has many additional benefits and can open a lot of new possibilities for your company.

## 8.2 MERITS OF THE SYSTEM

1. Customer satisfaction enhancement
2. Product/service upgrade
3. Improvement of policies and procedures
4. Boost in customer communication
5. Positive impact on brand image

## 8.3 LIMITATIONS OF THE SYSTEM

1. Lack of Analytics
2. Inconsistent Complaint Handling
3. Changing Business Environment
4. Fragmented and Insufficient Information

## 8.4 FUTURE ENHANCEMENT OF THE SYSTEM

This system is found tested and examined for its successful processing. Future change in the environment or processing can be easily adopted by having simple change in coding. It is very user friendly, cost effective, feature rich and it provides very high level of security. It protects the unauthorized users. Moreover, the system coding is so well designed that new operations can be easily incorporated without much modification. A facility to inform through SMS or Email on landing of the consignment can be added in future.. This web application involves almost all the basic features of the online complaint management system. The future implementation will be online help for the users and chatting with website administrator.

# CHAPTER 9

# CONCLUSION

## CONCLUSION

A Complaint Management System is considered one of contemporary productivity enhancement gear extensively by means of all companies and management. It provides an online way of solving the problems faced by the public by saving time and eradicate corruption. The objective of complaint management system is to make complain easier to coordinate, monitor, track and resolve by tracking the status of complaint done by public to the department. This is a must have software for any organizations who value there clients and will help in improvising the communication between the customer and the organization.

# CHAPTER 10

# BIBILOGRAPHY

## Refereneces:

- https://phpgurukul.com/complaint-management-sytem/
- https://sourcecodehero.com/complaint-management-system-project-in-django-with-source-code/
- https://codeshoppy.com/shop/product/onlinecomplaint-django/
- https://codeprojectz.com/applications-and-web-projects/simple-complaint-management-system-in-python-with-source-code/https://codeprojectz.com/applications-and-web-projects/simple-complaint-management-system-in-python-with-source-code/

**APPENDIX**

# SOURCE CODES AND SCREENSHOTS

**CODE**

**Models.py**

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.

class Category(models.Model):
    catname = models.CharField(max_length=100)
    catdes = models.CharField(max_length=300)
    creationdate = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.catname

class SubCategory(models.Model):
    categoryid = models.ForeignKey(Category, on_delete=models.CASCADE, null=True)
    subcategoryname = models.CharField(max_length=100)
    creationdate = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.subcategoryname

class State(models.Model):
    statename = models.CharField(max_length=100)
    creationdate = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.statename

class tblUser(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    address = models.CharField(max_length=300,null=True)
    state = models.CharField(max_length=50,null=True)
    country = models.CharField(max_length=50,null=True)
    pincode = models.CharField(max_length=20,null=True)
    userimage = models.FileField(null=True)
    regdate = models.DateField()
    def __str__(self):
        return self.user.username



class Complaint(models.Model):
    userid = models.ForeignKey(tblUser, on_delete=models.CASCADE, null=True)
    category = models.CharField(max_length=100,null=True)
    subcategory = models.CharField(max_length=100,null=True)
    complainttype = models.CharField(max_length=100,null=True)
    state = models.CharField(max_length=100,null=True)
    noc = models.CharField(max_length=200, null=True)
    complaintdetail = models.CharField(max_length=300,null=True)
    complaintfile = models.FileField(null=True,blank=True)
    regdate = models.DateField()
    status = models.CharField(max_length=50, null=True)
    lastupdationdate = models.DateField(null=True)
    remark = models.CharField(max_length=200, null=True)
    remarkdate = models.DateField(null=True)
    def __str__(self):
        return self.id


class ComplaintRemark(models.Model):
    complaintid = models.ForeignKey(Complaint, on_delete=models.CASCADE, null=True)
    status = models.CharField(max_length=50, null=True)
    remark = models.CharField(max_length=200, null=True)
    remarkdate = models.DateField(null=True)
    def __str__(self):
        return self.id
```

**Registration.html**

```
{% load static %}

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="description" content="">

<meta name="author" content="Dashboard">

<meta name="keyword" content="Dashboard, Bootstrap, Admin, Template, Theme, Responsive, Fluid,
Retina">
```

```
<title>CMS | User Registration</title>

<link href="{% static 'user/assets/css/bootstrap.css' %}" rel="stylesheet">

<link href="{% static 'user/assets/font-awesome/css/font-awesome.css' %}" rel="stylesheet" />

<link href="{% static 'user/assets/css/style.css' %}" rel="stylesheet">

<link href="{% static 'user/assets/css/style-responsive.css' %}" rel="stylesheet">


</head>


<body>

<div id="login-page">

<div class="container">

<h3 align="center" style="color:#fff">Complaint Managent System</h3>

<hr />

<form class="form-login" method="post">

{% csrf_token %}

<h2 class="form-login-heading">User Registration</h2>

<p style="padding-left: 1%; color: green">


</p>

<div class="login-wrap">

<input type="text" class="form-control" placeholder="Full Name" name="fullname"
required="required" autofocus>

<br>

<input type="email" class="form-control" placeholder="Email ID" id="email"
onBlur="userAvailability()"  name="email" required="required">
```

```
<span id="user-availability-status1" style="font-size:12px;"></span>

<br>

<input type="password" class="form-control" placeholder="Password" required="required"
name="password"><br >

<input type="tel" class="form-control" maxlength="10" name="contactno" placeholder="Contact no"
pattern="^\d{10}$" title="use only digits" required autofocus>

<br>



<button class="btn btn-theme btn-block"  type="submit" name="submit" id="submit" ><i class="fa fa-
user"></i> Register</button>

<hr>



<div class="registration">

Already Registered<br/>

<a class="" href="{% url 'user_login' %}">

Sign in

</a>

</div>

<br>

<a class="" href="{% url 'admin_login' %}">

Admin Login

</a>



</div>
```

```
</form>


</div>
</div>


<!-- js placed at the end of the document so the pages load faster -->
<script src="{% static 'user/assets/js/jquery.js' %}"></script>
<script src="{% static 'user/assets/js/bootstrap.min.js' %}"></script>


<!--BACKSTRETCH-->
<!-- You can use an image of whatever size. This script will stretch to fit in any screen size.-->
<script type="text/javascript" src="{% static 'user/assets/js/jquery.backstretch.min.js' %}"></script>
<script>
$.backstretch("{% static 'user/assets/img/bg.jpg' %}", {speed: 500});
</script>



</body>
</html>


{% ifequal error "no" %}
<script>
alert('Signup Successfully');
window.location=('{% url 'user_login' %}');
```

```
</script>
{% endifequal %}
{% ifequal error "yes" %}
<script>
alert('Something went wrong,Try Again...');
</script>
{% endifequal %}

<script>
function phonenumber(inputtxt)
{
var phoneno = /^\d{10}$/;
if((inputtxt.value.match(phoneno))
{
return true;
}
else
{
alert("message");
return false;
}
}
</script>
```

**Register_complaint.html**

```
{% load static %}

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="description" content="">

<meta name="author" content="Dashboard">

<meta name="keyword" content="Dashboard, Bootstrap, Admin, Template, Theme, Responsive, Fluid,
Retina">


<title>CMS | User Register Complaint</title>


<!-- Bootstrap core CSS -->

<link href="{% static 'user/assets/css/bootstrap.css' %}" rel="stylesheet">

<!--external css-->

<link href="{% static 'user/assets/font-awesome/css/font-awesome.css' %}" rel="stylesheet" />

<link rel="stylesheet" type="text/css" href="{% static 'user/assets/js/bootstrap-
datepicker/css/datepicker.css' %}" />

<link rel="stylesheet" type="text/css" href="{% static 'user/assets/js/bootstrap-
daterangepicker/daterangepicker.css' %}" />

<link href="{% static 'user/assets/css/style.css' %}" rel="stylesheet">

<link href="{% static 'user/assets/css/style-responsive.css' %}" rel="stylesheet">



</head>
```

```
<body>


<section id="container" >

{% include 'header.html' %}

{% include 'sidebar.html' %}

<section id="main-content">

<section class="wrapper">

<h3><i class="fa fa-angle-right"></i> Register Complaint</h3>


<!-- BASIC FORM ELELEMNTS -->

<div class="row mt">

<div class="col-lg-12">

<div class="form-panel">


<form class="form-horizontal style-form" method="post" name="complaint" enctype="multipart/form-
data" id="complaint" data-subcategory-url="{% url 'ajax_load_subcategory' %}">

{% csrf_token %}

<div class="form-group">

<label class="col-sm-2 col-sm-2 control-label">Category</label>

<div class="col-sm-4">

<select name="category" id="category" class="form-control" required="">

<option value="">Select Category</option>

{% for i in category %}

<option value="{{i.id}}">{{i.catname}}</option>

{% endfor %}
```

```
</select>

</div>

<label class="col-sm-2 col-sm-2 control-label">Sub Category </label>

<div class="col-sm-4">

<select name="subcategory" id="subcategory" class="form-control" >


</select>

</div>

</div>


<div class="form-group">

<label class="col-sm-2 col-sm-2 control-label">Complaint Type</label>

<div class="col-sm-4">

<select name="complainttype" class="form-control" required="">

<option value="Complaint"> Complaint</option>

<option value="General Query">General Query</option>

</select>

</div>


<label class="col-sm-2 col-sm-2 control-label">State</label>

<div class="col-sm-4">

<select name="state" required="required" class="form-control">

<option value="">Select State</option>

{% for i in state %}

<option value="{{i.statename}}">{{i.statename}}</option>
```

```
{% endfor %}



</select>

</div>

</div>



<div class="form-group">

<label class="col-sm-2 col-sm-2 control-label">Nature of Complaint</label>

<div class="col-sm-4">

<input type="text" name="noc" required="required" value="" required="" class="form-control">

</div>


</div>


<div class="form-group">

<label class="col-sm-2 col-sm-2 control-label">Complaint Details (max 2000 words) </label>

<div class="col-sm-6">

<textarea  name="complaindetails" required="required" cols="10" rows="10" class="form-control"
maxlength="2000"></textarea>

</div>

</div>

<div class="form-group">

<label class="col-sm-2 col-sm-2 control-label">Complaint Related Doc(if any) </label>
```

```
<div class="col-sm-6">

<input type="file" name="compfile" class="form-control" >

</div>

</div>


<div class="form-group">

<div class="col-sm-10" style="padding-left:25% ">

<button type="submit" name="submit" class="btn btn-primary">Submit</button>

</div>

</div>


</form>

</div>

</div>

</div>


</section>

</section>

{% include 'footer.html' %}

</section>


<!-- js placed at the end of the document so the pages load faster -->

<script src="{% static 'user/assets/js/jquery.js' %}"></script>

<script src="{% static 'user/assets/js/bootstrap.min.js' %}"></script>
```

```
<script class="include" type="text/javascript" src="{% static 'user/assets/js/jquery.dcjqaccordion.2.7.js'
%}"></script>

<script src="{% static 'user/assets/js/jquery.scrollTo.min.js' %}"></script>




<!--common script for all pages-->

<script src="{% static 'user/assets/js/common-scripts.js' %}"></script>

<!--script for this page-->

<script src="{% static 'user/assets/js/jquery-ui-1.9.2.custom.min.js' %}"></script>


<!--custom switch-->

<script src="{% static 'user/assets/js/bootstrap-switch.js' %}"></script>


<!--custom tagsinput-->

<script src="{% static 'user/assets/js/jquery.tagsinput.js' %}"></script>


<!--custom checkbox & radio-->


<script type="text/javascript" src="{% static 'user/assets/js/bootstrap-datepicker/js/bootstrap-
datepicker.js' %}"></script>

<script type="text/javascript" src="{% static 'user/assets/js/bootstrap-daterangepicker/date.js'
%}"></script>

<script type="text/javascript" src="{% static 'user/assets/js/bootstrap-daterangepicker/daterangepicker.js'
%}"></script>


<script type="text/javascript" src="{% static 'user/assets/js/bootstrap-inputmask/bootstrap-
inputmask.min.js' %}"></script>
```

```
<script src="{% static 'user/assets/js/form-component.js' %}"></script>



<script>
//custom select box

$(function(){
$('select.styled').customSelect();
});

</script>
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<script>
$("#category").change(function () {
var url = $("#complaint").attr("data-subcategory-url");  // get the url of the `load_subcategory` view
var categoryId = $(this).val();  // get the selected category ID from the HTML input

$.ajax({                       // initialize an AJAX request
url: url,                    // set the url of the request (= localhost:8000/load-courses/)
data: {
'category': categoryId       // add the category id to the GET parameters
},
success: function (data) {   // `data` is the return of the `load_subcategory` view function
$("#subcategory").html(data);  // replace the contents of the subcategory input with the data that came from the server
}
});


});
</script>


</body>
</html>


{% ifequal error "no" %}
<script>
alert('Your complain has been successfully filled and your complaintno. is  {{complaint.id}}');
window.location=('{% url 'register_complaint' %}');
</script>
{% endifequal %}
{% ifequal error "yes" %}
<script>
alert('Something Went Wrong. Please try again');
</script>
{% endifequal %}
```

**manage_user.html**

```
{% load static %}
<html lang="en">
```

```
<head>

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

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Admin| Manage Users</title>

<link type="text/css" href="{% static 'admin/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">

<link type="text/css" href="{% static 'admin/bootstrap/css/bootstrap-responsive.min.css' %}"
rel="stylesheet">

<link type="text/css" href="{% static 'admin/css/theme.css' %}" rel="stylesheet">

<link type="text/css" href="{% static 'admin/images/icons/css/font-awesome.css' %}" rel="stylesheet">

<link type="text/css"
href='http://fonts.googleapis.com/css?family=Open+Sans:400italic,600italic,400,600' rel='stylesheet'>

<script language="javascript" type="text/javascript">

var popUpWin=0;

function popUpWindow(URLStr, left, top, width, height)

{

if(popUpWin)

{

if(!popUpWin.closed) popUpWin.close();

}

popUpWin = open(URLStr,'popUpWin',
'toolbar=no,location=no,directories=no,status=no,menubar=no,scrollbars=yes,resizable=no,copyhistory=
yes,width='+600+',height='+600+',left='+left+', top='+top+',screenX='+left+',screenY='+top+'');

}



</script>

</head>

<body>
```

```
{% include 'admin_header.html' %}


<div class="wrapper">
<div class="container">
<div class="row">
{% include 'admin_sidebar.html' %}
<div class="span9">
<div class="content">


<div class="module">
<div class="module-head">
<h3>Manage Users</h3>
</div>
<div class="module-body table">


<table cellpadding="0" cellspacing="0" border="0" class="datatable-1 table table-bordered table-striped
display" width="100%">
<thead>
<tr>
<th>#</th>
<th> Name</th>
<th>Email </th>
<th>Contact no</th>
<th>Reg. Date </th>
<th>Action</th>
```

```
</tr>

</thead>

<tbody>


{% for i in tbluser %}

<tr>

<td>{{forloop.counter}}</td>

<td>{{i.user.first_name}}</td>

<td>{{i.user.username}}</td>

<td>{{i.user.last_name}}</td>


<td>{{i.regdate}}</td>


<td><a href="javascript:void(0);" onClick="popUpWindow('{% url 'user_profile' i.user.id %}');"
title="View Details">

<button type="button" class="btn btn-primary">View Detials</button>

</a>

<a href="{% url 'delete_user' i.user.id %}" title="Delete" onClick="return confirm('Do you really want
to delete ?')">

<button type="button" class="btn btn-danger">Delete</button></a>


</td>


{% endfor %}
```

```
</table>

</div>

</div>





</div><!--/.content-->

</div><!--/.span9-->

</div>

</div><!--/.container-->

</div><!--/.wrapper-->


{% include 'admin_footer.html' %}


<script src="{% static 'admin/scripts/jquery-1.9.1.min.js' %}" type="text/javascript"></script>

<script src="{% static 'admin/scripts/jquery-ui-1.10.1.custom.min.js' %}"
type="text/javascript"></script>

<script src="{% static 'admin/bootstrap/js/bootstrap.min.js' %}" type="text/javascript"></script>

<script src="{% static 'admin/scripts/flot/jquery.flot.js' %}" type="text/javascript"></script>

<script src="{% static 'admin/scripts/datatables/jquery.dataTables.js' %}"></script>

<script>

$(document).ready(function() {

$('.datatable-1').dataTable();

$('.dataTables_paginate').addClass("btn-group datatable-pagination");

$('.dataTables_paginate > a').wrapInner('<span />');
$('.dataTables_paginate > a:first-child').append('<i class="icon-chevron-left shaded"></i>');
$('.dataTables_paginate > a:last-child').append('<i class="icon-chevron-right shaded"></i>');
} );
</script>
</body>
```

**User_profile.html**

```
{% load static %}
<script language="javascript" type="text/javascript">
function f2()
{
window.close();
}ser
function f3()
{
window.print();
}
</script>
<!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=iso-8859-1" />
<title>User Profile</title>
</head>
```

```
<body>

div style="margin-left:50px;">

<form name="updateticket" id="updateticket" method="post">

{% csrf_token %}

<table width="100%" border="0" cellspacing="0" cellpadding="0">

<tr>

<td colspan="2"><b>{{tbluser.user.first_name}}'s profile</b></td>

</tr>

<tr>

<td  >&nbsp;</td>

<td >&nbsp;</td>

</tr>

<tr height="50">

<td><b>Reg Date:</b></td>

<td>{{tbluser.regdate}}</td>

</tr>

<tr height="50">

<td><b>User Email:</b></td>

<td>{{tbluser.user.username}}</td>

</tr>



<tr height="50">

<td><b>User Contact no:</b></td>

<td>{{tbluser.user.last_name}}</td>

</tr>
```

```
<tr height="50">

<td><b>Address:</b></td>

<td>{{tbluser.address}}</td>

</tr>


<tr height="50">

<td><b>State:</b></td>

<td>{{tbluser.state}}</td>

</tr>

<tr height="50">

<td><b>Country:</b></td>

<td>{{tbluser.country}}</td>

</tr>


<tr height="50">

<td><b>Pincode:</b></td>

<td>{{tbluser.pincode}}</td>

</tr>

<tr>

<td colspan="2">

<input name="Submit2" type="submit" class="txtbox4" value="Close this window " onClick="return
f2();" style="cursor: pointer;"  /></td>

</tr>

</table>

</form>
```

```
</div>


</body>
</html>
<?php } ?>
```

**Complaint_history.html**

```
{% load static %}
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="">
<meta name="author" content="Dashboard">
<meta name="keyword" content="Dashboard, Bootstrap, Admin, Template, Theme, Responsive, Fluid, Retina">


<title>CMS | Complaint History</title>


<!-- Bootstrap core CSS -->
<link href="{% static 'user/assets/css/bootstrap.css' %}" rel="stylesheet">
<!--external css-->
<link href="{% static 'user/assets/font-awesome/css/font-awesome.css' %}" rel="stylesheet" />


<!-- Custom styles for this template -->
```

```
<link href="{% static 'user/assets/css/style.css' %}" rel="stylesheet">
<link href="{% static 'user/assets/css/style-responsive.css' %}" rel="stylesheet">
<link href="{% static 'user/assets/css/table-responsive.css' %}" rel="stylesheet">

<!-- HTML5 shim and Respond.js IE8 support of HTML5 elements and media queries -->
<!--[if lt IE 9]>
<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
<script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
<![endif]-->
</head>

<body>

<section id="container" >
{% include 'header.html' %}
{% include 'sidebar.html' %}

<section id="main-content">
<section class="wrapper">
<h3><i class="fa fa-angle-right"></i>Your Complaint Hstory</h3>
<div class="row mt">
<div class="col-lg-12">
<div class="content-panel">
<section id="unseen">
<table class="table table-bordered table-striped table-condensed">
```

```
<thead>
<tr style="text-align: center">
<th style="text-align: center">Complaint Number</th>
<th style="text-align: center">Reg Date</th>
<th style="text-align: center">last Updation date</th>
<th style="text-align: center">Status</th>
<th style="text-align: center">Action</th>
</tr>
</thead>
<tbody>
{% for i in complaint %}
<tr>
<td align="center">{{i.id}}</td>
<td align="center">{{i.regdate}}</td>
<td align="center">{{i.lastupdationdate}}</td>
<td align="center">

{% if not i.status %}
<button type="button" class="btn btn-theme04">Not Process Yet</button>
{% endif %}

{% if i.status == 'in process' %}
<button type="button" class="btn btn-warning">In Process</button>
{% endif %}
{% if i.status == 'closed' %}
```

```
<button type="button" class="btn btn-success">Closed</button>

{% endif %}

<td align="center">

<a href="{% url 'complaint_details' i.id %}">

<button type="button" class="btn btn-primary">View Details</button></a>

</td>

</tr>

{% endfor %}


</tbody>

</table>

</section>

</div><!-- /content-panel -->

</div><!-- /col-lg-4 -->

</div><!-- /row -->




</section><! --/wrapper -->

</section><!-- /MAIN CONTENT -->

{% include 'footer.html' %}

</section>


<!-- js placed at the end of the document so the pages load faster -->

<script src="{% static 'user/assets/js/jquery.js' %}"></script>
```

```
<script src="{% static 'user/assets/js/bootstrap.min.js' %}"></script>

<script class="include" type="text/javascript" src="{% static 'user/assets/js/jquery.dcjqaccordion.2.7.js'
%}"></script>

<script src="{% static 'user/assets/js/jquery.scrollTo.min.js' %}"></script>

<!--common script for all pages-->

<script src="{% static 'user/assets/js/common-scripts.js' %}"></script>



<!--script for this page-->

</body>

</html>
```