**DISTRICT WISE COMPLAINT MANAGEMENT**

**ABSTRACT**

This project has been developed to find the crime complaint at anywhere through public peoples. This project will be useful to monitor the crimes and voice, image based Crime complaint. The User upload the complaints also and the admin can download and view the Complaints. The system keeps logs of a case which includes case status, people involved, disputes, Items recovered on scene and other details. So this website allows admin to update the status of investigation by adding the case details. This allows authorized officers to check case status and look into its status online when needed.

Officer can login into this website by using their username and password and view the case details and can check the status of complaint by using complaint type, area, district name etc...Then admin can view different complaint details, and take the action according to the complaint details. Then admin can update each case detail in this website by using complaint no. and also admin can able to see the complaint records at any time. The system is proposed to help officers to speed up complaint process and track status of multiple complaint at a time. The system is designed to aid investigation teams to work collectively on complaint, coordinate and also speed up the process by suggesting logical solution based on data provided.

**INTRODUCTION**

This project District Wise Complaint Management system for the purposes of fast and efficient sharing of critical information across all district across the state. The system is implemented across Cities and Towns so that a admin can access information across all records in the state thus helping speedy and successful completion to complaints. The System would also be used to generate information for pro-active and preventive measures for fighting complaints.

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work. The existing system has several disadvantages and many more difficulties to work well. The proposed system tries to eliminate or reduce these difficulties up to some extent. The proposed system will help the user to reduce the workload and mental conflict. The proposed system helps the user to work user friendly and he can easily do his jobs without time lagging. By using this application people who are afraid or don’t have enough time to go public service office for complaint about their personal issues here they can give their complaint through online to register any type of complaint.

The project has been planned to be having the view of distributed architecture, with centralized storage of the database. The application for the storage of the data has been planned. Using the constructs of MYSQL server and all the user interfaces have been designed using the PYTHON technologies

**SYSTEM STUDY**

**Existing System**

In the existing system only we can see the details of particular information about the police stations in our state, the existing system has more workload for the authorized person, but in the case of Proposed System, the user can registered in our site and send the complaint report and voice, image complaint about a particular city or person.

**Disadvantages of existing system**

* More man power.
* Time consuming.
* Consumes large volume of pare work.
* Needs manual calculations.
* No direct role for the higher officials.
* Damage of machines due to lack of attention.

To avoid all these limitations and make the working more accurately the system needs to be computerized.

**Proposed System**

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work. The existing system has several disadvantages and many more difficulties to work well. The proposed system tries to eliminate or reduce these difficulties up to some extent. The proposed system will help the user to reduce the workload and mental conflict. The proposed system helps the user to work user friendly and he can easily do his jobs without time lagging. By using this application people who are afraid or don’t have enough time to go public service office for complaint about their personal issues here they can give their complaint through online to register any type of complaint.

**Features**

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

* Ensure data accuracy’s.
* Proper control of the higher officials.
* Reduce the damages of the machines.
* Minimize manual data entry.
* Minimum time needed for the various processing.
* Greater efficiency.
* Better service.
* User friendliness and interactive.
* Minimum time required.
* Voice, image based Compliant

**Modules**

* User Registration
* Admin login
* Compliant Details
* Upload files
* Compliant status

**User Registration**

Once the registered user details have been provided to the complaints he/she can login into this website and access the system. The case details can also be view by the users.

**Admin login**

In this login section, the main role of the administrator is to manage all the complaint and have the access to view the complaint details and has the right to add the case officer to a particular complaint. The role of Administrator is to manage officer, add complaint and description details and update the status of complaint. Here, the admin can also view the details of the complaints.

**Compliant Details**

This module help the user to report online complaint, the user adds the complaint details like place, complaint information etc…depending upon this complaint details only respected person can take the action.

**Upload Documents**

This module helps the user to report online complaints user adds the complaint details like place, complaint, image, video information etc…

**Compliant status**

This module allows us to view the status of all complaint that you have posted earlier. Helps the officer to view and check the status of the complaint that is added by the admin. This helps the officer to take action based on the complaint details registered by admin. The action details contains no of time response, reply details, complaint type etc ...

SOFTWARE SPECIFICATIONS:

**HARDWARE SPECIFICATION**

* PROCESSOR : PENTIUM III 866 MHz
* RAM : 256 MD SD RAM
* MONITOR : 15” COLOR
* HARD DISK : 40 GB
* CDDRIVE : LG 52X
* KEYBOARD : STANDARD 102 KEYS

**SOFTWARE SPECIFICATION**

* FRONT END : PYTHON
* BACK END : MYSQL
* OPERATING SYSTEM : WIN Family.

**SYSTEM ENVIRONMENT**

**LANGUAGE FEATURES**

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

**Statements and control flow**

The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.

The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if). The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block. The while statement, which executes a block of code as long as its condition is true.

The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits. The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming. The def statement, which defines a function or method. The with statement (from Python 2.5), which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior. The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block. The assert statement, used during debugging to check for conditions that ought to apply. The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.

The import statement, which is used to import modules whose functions or variables can be used in the current program. There are four ways of using import: import <module name> or from <module name> import \* or import numpy as np or from numpy import pi as Pie. The print statement was changed to the print() function in Python 3. Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

**Flask:**

Python Flask is a lightweight and flexible web framework for building web applications with Python. It is known for its simplicity and ease of use, making it a popular choice for developers of all levels. Flask allows developers to quickly create web applications without the complexity of a full-fledged framework, while still providing the necessary tools and features for building powerful applications.

Flask is a micro web framework written in Python. It was developed by Armin Ronacher and released in 2010. Flask is based on the Werkzeug WSGI toolkit and the Jinja2 template engine. Unlike other web frameworks like Django, Flask does not enforce any particular way of doing things, allowing developers the freedom to structure their applications as they see fit.

Lightweight: Flask is minimalistic and has a small footprint, making it fast and efficient.

Easy to Use: Flask has a simple and intuitive API, making it easy for developers to get started quickly.

Extensible: Flask is highly extensible, with a large ecosystem of extensions available for adding additional functionality.

Flexible: Flask allows developers to use any ORM (Object-Relational Mapping) library, database, or other third-party libraries of their choice.

Built-in Development Server: Flask comes with a built-in development server, making it easy to test and debug applications locally.

### Once you have developed your Flask application, you can deploy it to a production server for public access. There are several deployment options available for Flask applications, including traditional web servers like Apache or Nginx, cloud platforms like Heroku or AWS, and containerization platforms like Docker.

Python Flask is a versatile web framework that provides developers with the tools and flexibility to build web applications quickly and efficiently. By following best practices and leveraging advanced features like routing, templates, and extensions, you can create robust and scalable Flask applications for a variety of use cases.

### Distutils Basics and Design Flaws

Distutils contains commands, each of which is a class with a run method that can be called with some options. Distutils also provides a Distribution class that contains global values every command can look at.

To use Distutils, a developer adds a single Python module to a project, conventionally called setup.py. This module contains a call to Distutils' main entry point: the setup function. This function can take many options, which are held by a Distribution instance and used by commands. Here's an example that defines a few standard options like the name and version of the project, and a list of modules it contains:

from distutils.core import setup

setup(name='MyProject', version='1.0', py\_modules=['mycode.py'])

This module can then be used to run Distutils commands like sdist, which creates a source distribution in an archive and places it in a dist directory:

$ python setup.py sdist

Using the same script, you can install the project using the install command:

$ python setup.py install

Distutils provides other commands such as:

* upload to upload a distribution into an online repository.
* register to register the metadata of a project in an online repository without necessary uploading a distribution,
* bdist to creates a binary distribution, and
* bdist\_msi to create a .msi file for Windows.

It will also let you get information about the project via other command line options.

So installing a project or getting information about it is always done by invoking Distutils through this file. For example, to find out the name of the project:

$ python setup.py --name

MyProject

setup.py is therefore how everyone interacts with the project, whether to build, package, publish, or install it. The developer describes the content of his project through options passed to a function, and uses that file for all his packaging tasks. The file is also used by installers to install the project on a target system.

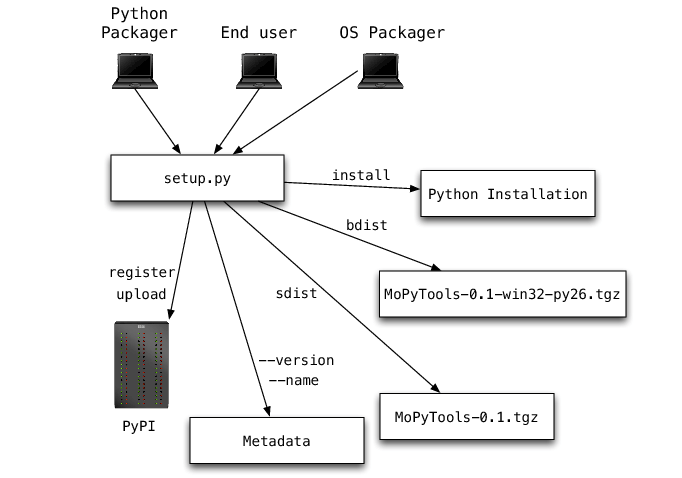


Figure 14.1: Setup

## *What is Python?*

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

**Introduction**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to www browsers to games.

## History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## Python Features

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.
* Apart from the above-mentioned features, Python has a big list of good features, few are listed below −
* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

## Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

* Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
* Win 9x/NT/2000
* Macintosh (Intel, PPC, 68K)
* OS/2
* DOS (multiple versions)
* PalmOS
* Nokia mobile phones
* Windows CE
* Acorn/RISC OS
* BeOS
* Amiga
* VMS/OpenVMS
* QNX
* VxWorks
* Psion
* Python has also been ported to the Java and .NET virtual machines.

### Integrated Development Environment

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

* **Unix** − IDLE is the very first Unix IDE for Python.
* **Windows** − PythonWin is the first Windows interface for Python and is an IDE with a GUI.
* **Macintosh** − The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine.

# My-SQL

# MySQL is a simple, yet powerful Open Source Software relational database management system that uses SQL. MySQL is a true multi-user, multithreaded SQL database server. MySQL is fast and flexible enough to store logs and pictures in it. Its main goals are speed, robustness, and ease of use. Most likely MySQL and PHP combination is encountered today and probably for the years to come.

This basic MySQL explains some of the basic SQL statements. If this is the first time you have used a relational database management system, this tutorial gives you everything you need to know to work with MySQL such as querying data, updating data, managing databases, and creating tables. If you are already familiar with other relational database management systems such as PostgreSQL, Oracle, and Microsoft SQL Server. you can use this tutorial to refresh your knowledge and understand how SQL dialect of MySQL is different from other systems.

**Strengths of MySQL**

* MySQL has the biggest market share of any open source database. Almost any web-hosting company can provide.
* After setting database and access to it, managing the database is straightforward. Initial access needs to be configured by a database administrator. Tools such as MySQL Administrator or phpMyAdmin help to manage database.
* SQL has always been relatively fast, much due to its simplicity. In the last few years, MySQL has gained foothold in the enterprise market due to new “enterprise class” features and general maturity without compromising performance for simple usage.

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. This tutorial will give you a quick start to MySQL and make you comfortable with MySQL programming.

* A Relational DataBase Management System (RDBMS) is a software that −
* Enables you to implement a database with tables, columns and indexes.
* Guarantees the Referential Integrity between rows of various tables.
* Updates the indexes automatically.
* Interprets an SQL query and combines information from various tables.
* RDBMS Terminology
* Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.
* Database − A database is a collection of tables, with related data.
* Table − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
* Column − One column (data element) contains data of one and the same kind, for example the column postcode.
* Row − A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
* Redundancy − Storing data twice, redundantly to make the system faster.
* Primary Key − A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
* Foreign Key − A foreign key is the linking pin between two tables.
* Compound Key − A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
* Index − An index in a database resembles an index at the back of a book.

Referential Integrity − Referential Integrity makes sure that a foreign key value always points to an existing row.

What is a Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

* MySQL Database
* MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −
* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

## Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, **Manpower** and **manpower** are two different identifiers in Python.

Here are naming conventions for Python identifiers −

* Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
* Starting an identifier with a single leading underscore indicates that the identifier is private.
* Starting an identifier with two leading underscores indicates a strongly private identifier.
* If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

## *Standard Data Types*

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

# Flask

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

What is Web Framework?

Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

What is Flask?

Flask is a web application framework written in Python. It is developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

A **Flask** application is started by calling the **run()** method. However, while the application is under development, it should be restarted manually for each change in the code. To avoid this inconvenience, enable **debug support**. The server will then reload itself if the code changes. It will also provide a useful debugger to track the errors if any, in the application.

The **Debug** mode is enabled by setting the **debug** property of the **application** object to **True** before running or passing the debug parameter to the **run()** method.

app.debug = True

app.run()

app.run(debug = True)

**SYSTEM DESIGN & DEVELOPMENT**

**FILE DESIGN**

At the highest level a ﬁle system is a way to organize, store, retrieve, and manage information on a permanent storage medium such as a disk. File systems manage permanent storage and form an integral part of all operating systems. There are many different approaches to the task of managing permanent storage. At one end of the spectrum are simple ﬁle systems that impose enough restrictions to inconvenience users and make using the ﬁle system difficult. At the other end of the spectrum are persistent object stores and object-oriented databases that abstract the whole notion of permanent storage so that the user and programmer never even need to be aware of it. The problem of storing, retrieving, and manipulating information on a computer is of a general-enough nature that there are many solutions to the problem.

The primary functionality that all ﬁle systems must provide is a way to store a named piece of data and to later retrieve that data using the name given to it. We often refer to a named piece of data as a ﬁle. A ﬁle provides only the most basic level of functionality in a ﬁle system. A ﬁle is where a program stores data permanently. In its simplest form a ﬁle stores a single piece of information. A piece of information can be a bit of text (e.g., a letter, program source code, etc.), a graphic image, a database, or any collection of bytes a user wishes to store permanently. The size of data stored may range from only a few bytes to the entire capacity of a volume. A ﬁle system should be able to hold a large number of ﬁles, where “large” ranges from tens of thousands to millions.

The name of a ﬁle is metadata because it is a piece of information about the ﬁle that is not in the stream of bytes that make up the ﬁle. There are several other pieces of metadata about a ﬁle as well—for example, the owner, security access controls, date of last modiﬁcation, creation time, and size. The ﬁle system needs a place to store this metadata in addition to storing the ﬁle contents.

SYSTEM DESIGN

INPUT DESIGN

During input design, determine how data will be captured and entered into the system. Data capture is the identification and recording of source data. Data entry is the process of converting source data into computer-readable form and entering it into the information system. Input design has six main points. They are,

1. To select a suitable input and data entry method

2. To reduce input volume

3. To design attractive data entry screens

4. To use validation checks to reduce input errors

5. To design required source documents

6. To develop effective input control

Receive user details from the user in registration phase. Username and password are received from the user for login phase. Content to display in the user web page are get as a input from the user in design time phase.

**OUTPUT DESIGN**

During the output design after receiving the data how the response is send to the user is designed. User details are stored in the database. If the username and password is valid then user is redirected to his index page otherwise to the login page. Contents got from the user and validation is done for verifying whether the data is valid or not. If the data is a valid data then BPEL engine process the data and web service is called and data updating is done and presented to the user as a web page.

DATABASE DESIGN

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system.

**SYSTEM DEVELOPMENT**

**Development Approach**

Share complaints Web Portal was designed and developed based on the Waterfall Model. This model particularly expresses the interaction between subsequent phases. Testing software is not an activity, which strictly follows the implementation phase. In each phase of the software development process has to compare the results obtained against that which is required. In all phases quality has to be assessed and controlled.

**SYSTEM TESTING AND IMPLEMENTATION**

**SYSTEM TESTING:**

**Testing methodologies:**

The term System Testing can be used in a number of ways. In a general sense, the term ‘system testing’ refers to the testing of the system in artificial condition to ensure that it should perform as expected and as required.

From a system development perspective, system testing refers to the testing performed by the development team (the programmers and other technicians) to ensure that the system works module by module (‘unit testing’) and also as a whole. System Testing should ensure that each function of the system works as expected and that any errors (bugs) are noted and analyzed. It should additionally ensure that interface for export and import routines, function as required. System Testing does not concern itself with the functionality of the system and whether this is appropriate to meet the needs of the users. Having met the criteria of the Test Plan the software may then be passed for User Acceptance Testing.

The various testing methodologies performed for this system is:

* Unit Testing
* Integration Testing
* White Box Testing
* Black Box Testing

***Unit testing***

In computer programming, a unit test is a procedure used to validate that a particular module of source code is working properly. The idea about unit test is to write test cases for all functions and methods so that whenever a change causes a regression, it can be quickly identified and fixed. Ideally, each test case is separate from the others; constructs such as mock object can assist in separating unit tests. This type of testing is mostly done by the developers and not by end-users.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Unit testing provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Unit testing provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits and allowed to correct the following errors.

1. Mixed mode operations
2. Incorrect initialization
3. Incorrect symbolic representation of the expression
4. Simplified integration
5. Facilitated for the various changes made to the system

***Integration Testing:***

Integration testing can proceed in a number of different ways, which can be broadly characterized as **top down** or **bottom up**. In **top down integration testing** the high level control routines is tested first, possibly with the middle level control structure present only as **stubs**. Subprogram **stubs** are incomplete subprograms which are only present to allow the higher level control routines to be tested.

Top down testing can proceed in a **depth-first** or **breadth-first** manner. For depth-first integration each module is tested in increasing detail, replacing more and more levels of details with actual code rather than stubs. Alternatively breadth-first would proceed by refining all the modules at the same level of the control throughout the application.

In practice a combination of the two techniques would be used. At the initial stage all the modules might be only partly functional, possibly being implemented only to deal with non-erroneous data. These would be tested in breadth-first manner, but over a period of time each would be replaced with successive refinements which were closer to the full functionality. This allows depth-first testing of a module to be performed simultaneously with breadth-first testing of all the modules.

The other major category of integration testing is **bottom up integration testing** where an individual module is tested from a test harness. Once a set of individual modules have been tested they are then combined into a collection of modules, know as **builds**, which are then by a second test harness. This process can continue until the build consists of the entire application.

This second approach is used in this project where the individual modules that are-Mobile Call Status, Mobile Time Retrieval and Internet connectivity are first developed and then later they were integrated into one application and tested for the results.

***White Box Testing:***

White box testing is testing from the inside--tests that go in and test actual program structure.

Basis path testing: Very simply, test every statement in the program at least once. You’ll note that the testing department at FCC chose test cases that did this; the entire execution tree was covered.

Basis path testing is MANDATORY--so much so that there are software products written especially to assist in it.

* Profiling: there are a lot of tools--often included with compilers--which show where the CPU is spending most of its time in a program. Naturally, the busiest parts of the program are the ones you want to test most.
* Loop tests: Exercise each DO, WHILE, FOR and other repeating statements several times.
* Input tests: as the old saying goes-- garbage out, garbage out. If a procedure receives the wrong data, it’s not going to work. Each procedure should be tested to make certain that the procedure actually received the data you sent to it. This will spot type mismatches, bad pointers, and other such bugs.

Here in this project each decision path is check and all the loops are executed separately to ensure that the program is logically correct and has exited right time

***Black Box Testing:***

Black box testing, concrete box or functional testing is used in computer programming, software engineering and software testing to check that the outputs of a program, given certain inputs, conform to the functional specification of the program.

The term black box indicates that the internal implementation of the program being executed is not examined by the tester. For this reason black box testing is not normally carried out by the programmer. In most real-world engineering firms, one group does design work while a separate group does the testing.

Boundary value analysis is a technique of black box testing in which input values at the boundaries of the input domain are tested. it has been widely recognized that input values at the extreme ends of, and just outside of, input domains tend to the cause errors in system functionality.

In boundary value analysis, value at and just beyond boundaries of the input domain are used to generate test cases to ensure proper functionality of the system.

***Advantages of Black Box Testing***

* More effective on larger units of code than glass box testing
* Testing needs no knowledge of implementation, including specific programming language
* Tester and programmer are independent of each other
* Test are done from a user’s point of view
* Will help to expose any ambiguities or inconsistencies in the specifications
* Test cases can be designed as soon as the specifications are complete

In this project all the function are tested to check whether all of them are working properly. The performance rate is verified by considering response time and speed. Hence the error are identified and corrected.

**QUALITY ASSURANCE**

Quality assurance comprises all those planned and systematic

actions necessary to provide confidence that a structure, system or component will perform satisfactorily is service.

Quality assurance includes formal view of care, problem definition, corrective actions to remedy any deficiencies and evaluation of actions that to be taken.

The function of software quality that assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented. This is an “umbrella activity” that is applied throughout the engineering process. Quality software is reasonably bug-free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable.

The system is developed such that it ensures all the level of quality. It checks whether a user friendly environment is provided to the users and that there is a reliable, accurate and efficient flow of data within the system. The system also checks that due it contains the level of security required for the user. Hence as long as there is no hardware complaint, there is no problem with the software.

**SYSTEM IMPLEMENTATION:**

***Plan:***

Implementation is the state in the project where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and giving confidence on the new system for the users that will work efficiently and effectively. The system is implemented only after thorough testing is done and if it is found to work according to the specification.

It involves careful planning, investigation of the current system and is constraints on implementation, design of methods to achieve changeover, and evaluation of the changeover methods apart from planning. Two major tasks for preparing the implementation are educating, training the users and testing the system.

***Implementation plan preparation***

The implementation process begins with the preparation of plan for implementation. According to this plan other activities are carried out. In this plan discussion has been made regarding the equipment, resources and how to test the activities. Thus a clear planner prepared for the activities.

***Equipment Acquisition***

According to the above plan the necessary equipment have to be acquired to implement the new system, which would include all the requirements for installing and maintaining Flask framework, python, My SQL.

***Program code preparation***

One of the most important development activities is coding or programming. The system flowcharts and other charts are converted into modular programs. They have to be compiled, tested and debugged.

***User training and documentation***

Once the planning has been completed the major effort in the computer department is that the user department must consist of educated and trained staff as the system becomes more complex. The success of the system depends upon how they are operated and used the system.

Thus the quality of training the personnel is connected to the success of the system. Implementation depends upon the right people being trained at the right time. Education involves creating the right atmosphere and motivating the user. Staff education should encourage the participation of all the staff.

***Changeover***

Changeover is the change of moving over from the old system to the new computerized system. In order that this is done all the files have to be converted to the new format. The accuracy of the conversion is of utmost importance both to user confidence in the system and to effective operation. When the files have been set up on the computer, the changeover can take place. There are several possible methods of doing this.

E.g. direct changeover, parallel running, pilot running, and staged changeover.

This method is the complete replacement of the old system by new, in one move. When direct changeover is planned, system tests and training should be comprehensive and changeover itself is planned in detail.

***Parallel Running:***

Parallel running or operation means processing current data by both the old and new systems to cross check the results.

The old system is kept alive and operational until the system has been proved for at least one system cycle, using full live data in the operational environment of place, people, equipment and time. It allows the result of the new system to be compared with the old system before the acceptance by the user. Parallel operation does not allow much time or learning and testing activities.

***Staged Changeover:***

A staged changeover involves a series of limited size direct changeovers. The new system being introduced piece by piece. A complete start, a logical section is committed to the new system while the remaining parts or sections will be processed by the old system.

In this project, direct changeover is applied where the entire system is implemented directly after it has been developed.

***SYSTEM MAINTENANCE:***

***Maintenance***

The term “Software Maintenance” is used to describe software engineering activities. Maintenance activities involve making enhancements to software products, adapting to new environments and correcting problems. Software product enhancements may involve providing new functional capabilities, improving user displays and nodes of interaction, upgrading external documents and internal documentation or upgrading the performance characteristics of a system. Adaptation of software to a new environment may involve moving the software to a different machine, or for instance, modifying the software to accommodate a new telecommunication protocol or an additional disk drives. Problem correction involves modification and revalidation of software to correct errors.

Many activities performed during software development enhance the maintainability of a software product. They are:-

***Analysis activities:***

The analysis phase of software development is concerned with determining customer requirements and constraints and establishing feasibility of the product.

* Develop standards and guidelines
* Set milestones for supporting documents
* Specify quality assurance procedures
* Identify likely product enhancements
* Determine resources required for maintenance
* Estimate maintenance costs

***Architectural Design Activities:***

* Emphasize clarity and modularity as design criteria
* Design to ease likely enhancement
* Use standardized notations to document, data flows, functions, structure and interconnections
* Observe the principles of information hiding, data abstraction and top-down hierarchical decomposition

***Detailed Design Activities***

* Use standardized notations to specify algorithms, data structures and procedure interface specifications
* Specify side effects and exception handling for each routine

***Implementation activities***

* Use single entry, single exit constructs
* Use standard indentation of constructs
* Use simple, clear coding style
* Use symbolic constants to parameterize routines
* Provide margins on resources
* Provide standard documentation
* Follow standard internal commenting guidelines

***Other activities:***

* Develop a maintenance guide
* Develop a test suite
* Provide test suite documentation

**CONCLUSION**

It is concluded that the application works well and satisfy the end users. The application is tested very well and errors are properly debugged. The application is simultaneously accessed from more than one system. Simultaneous login from more than one place is tested.

This system is user friendly so everyone can use easily. Proper documentation is provided. The end user can easily understand how the whole system is implemented by going through the documentation. The system is tested, implemented and the performance is found to be satisfactory. All necessary output is generated. Thus, the project is completed successfully.

Further enhancements can be made to the application, so that the application functions very attractive and useful manner than the present one. The speed of the transactions become more enough now.

SCOPE FOR FUTURE ENHANCMENT:

Every application has its own merits and demerits. The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Changing the existing modules or adding new modules can append improvements. Further enhancements can be made to the application, so that the future enhancement is we develop the application through website and useful manner than the present one.

**BIBILIOGRPHY**

* Python Essential Reference: Developer's Library Paperback – 9 Jul 2009 by David Beazley (Author)
* Python Programming: An Introduction to Computer Science 2nd Edition by John Zelle (Author)
* Automate the Boring Stuff with Python: Practical Programming for Total Beginners 1st Edition by Al Sweigart (Author)
* Python Programming for the Absolute Beginner, 3rd Edition 3rd Edition by Michael Dawson (Author)
* Python Programming for Beginners: An Introduction to the Python Computer Language and Computer Programming 1st Edition by Jason Cannon (Author)
* Python: Python Programming For Beginners - The Comprehensive Guide To Python Programming: Computer Programming, Computer Language, Computer Science Paperback – August 9, 2016 by Adam Stark (Author)

**Data Flow diagram**

level 0

Complaint Management

User

Administrator

UserID, file Complaints

Solutions

**Level 1**

User Details

People

Complaints Details

**Level 2**

Admin

Complaint Details

Complaint Status

**TABLES**

**Admin Login**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| user | varchar(50) | User name |
| Psw | int | Password |

**People Details**

**Uid: Primary Key**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| Uid | varchar(5) | User id |
| name | varchar(50) | Full Name |
| age | int | User Age |
| dob | date | Date of Birth |
| add | varchar(250) | Residential Address |
| pno | int | phone Number |
| Email | varchar(50) | Email id |

**Complaint Details**

Sid**: Primary Key**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| uid | varchar(50) | User id |
| uname | int | User name |
| Complaints | Varchar(50) | Complaints Details |

**Complaint Status**

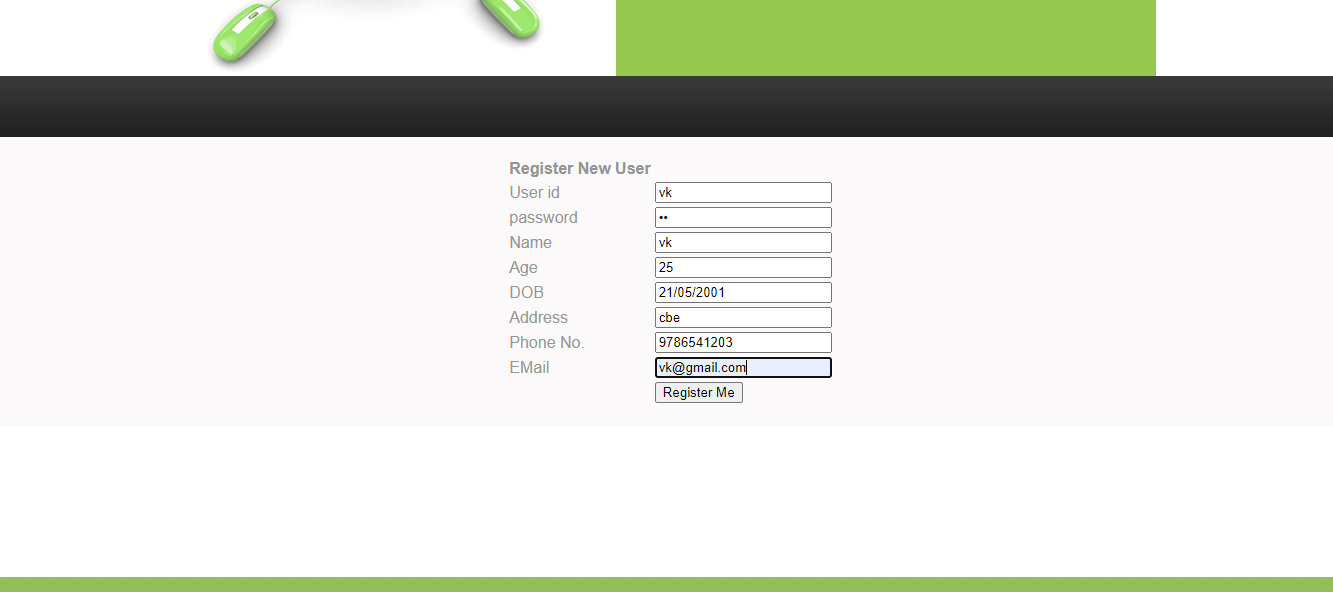
**Sid: Foreign Key**

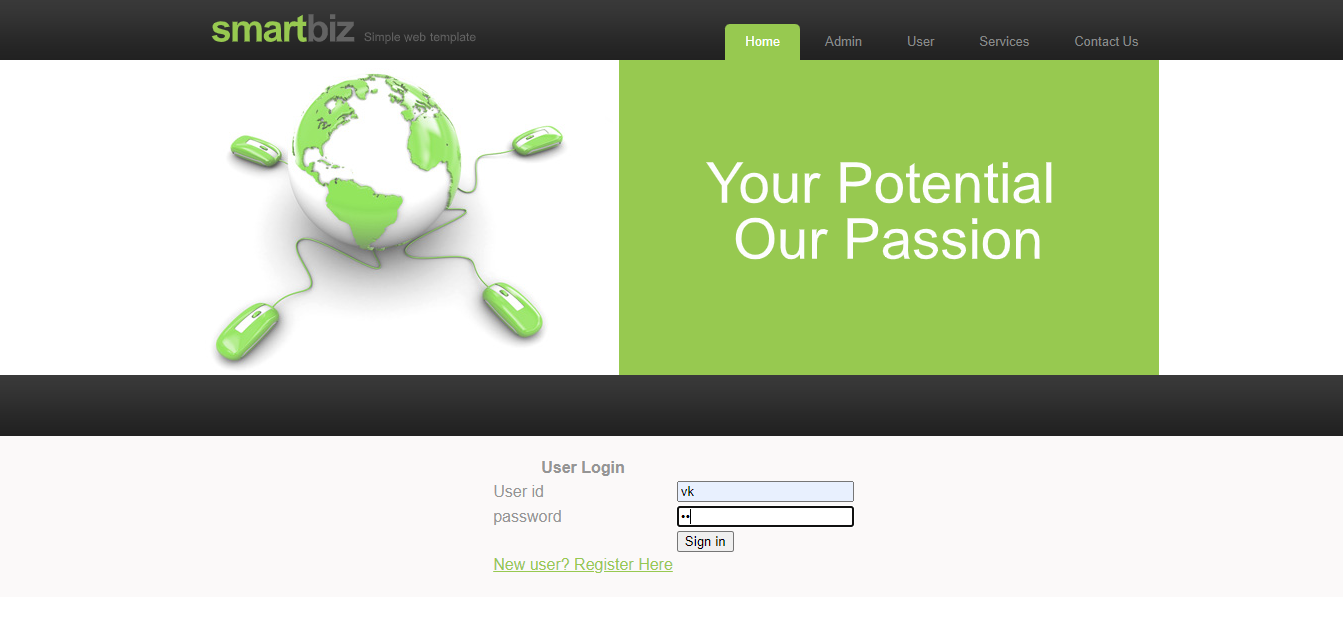
**Uid : Foreign Key**

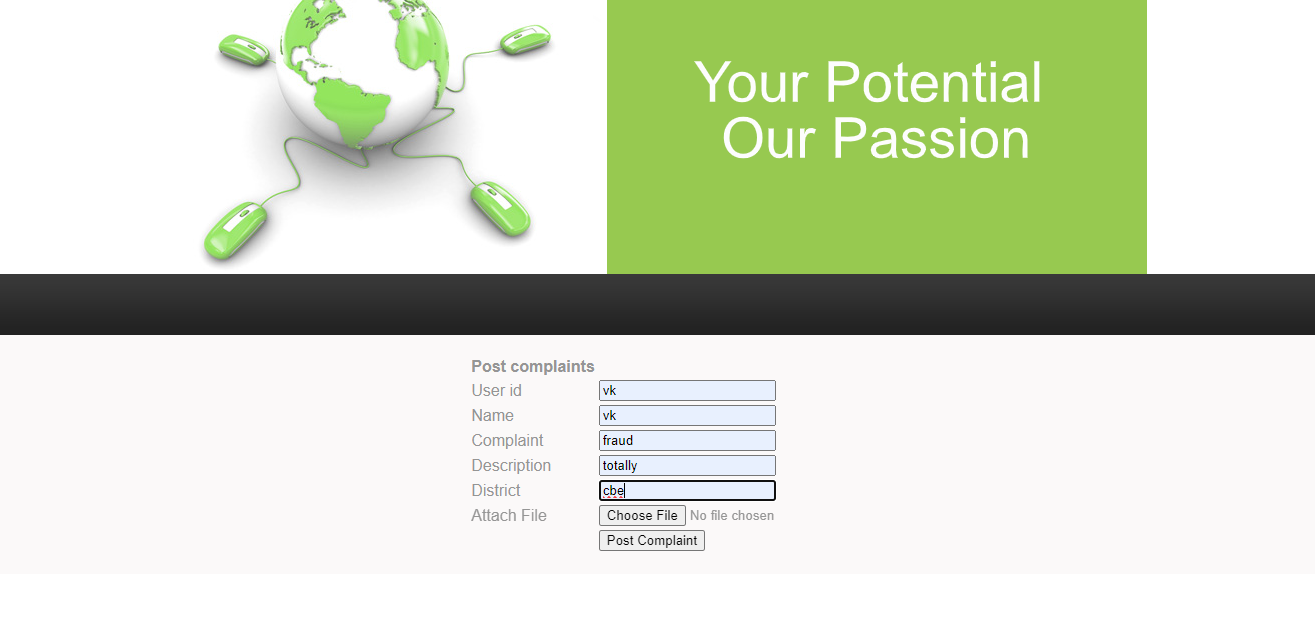
|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| uid | varchar(50) | User id |
| uname | int | User name |
| Complaints Status | Varchar(50) | Complaints Details Update |

**Sample Output:**

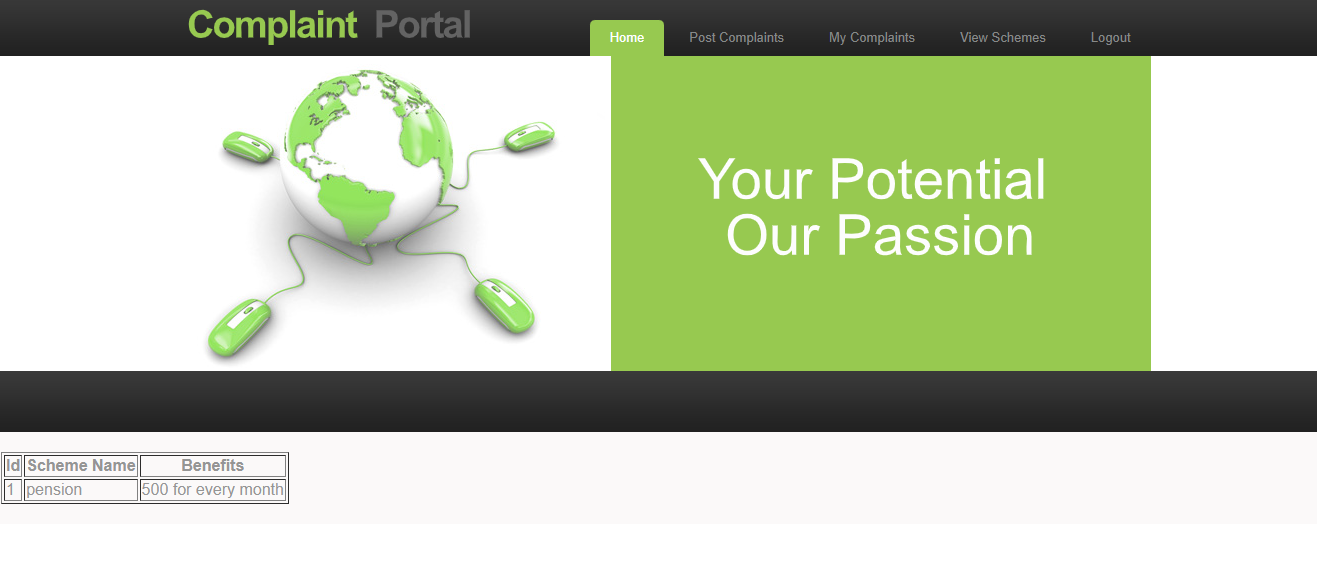


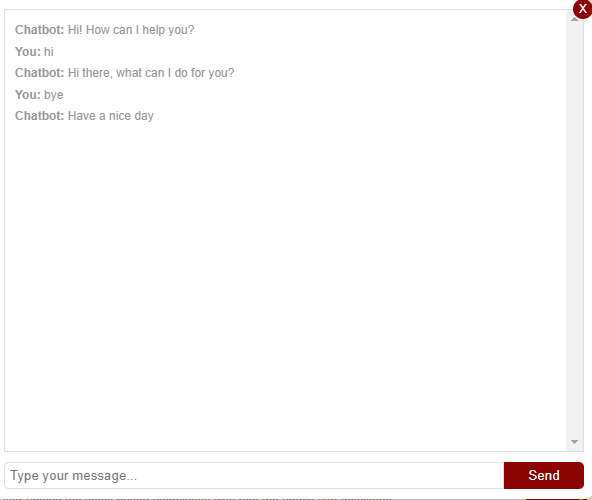


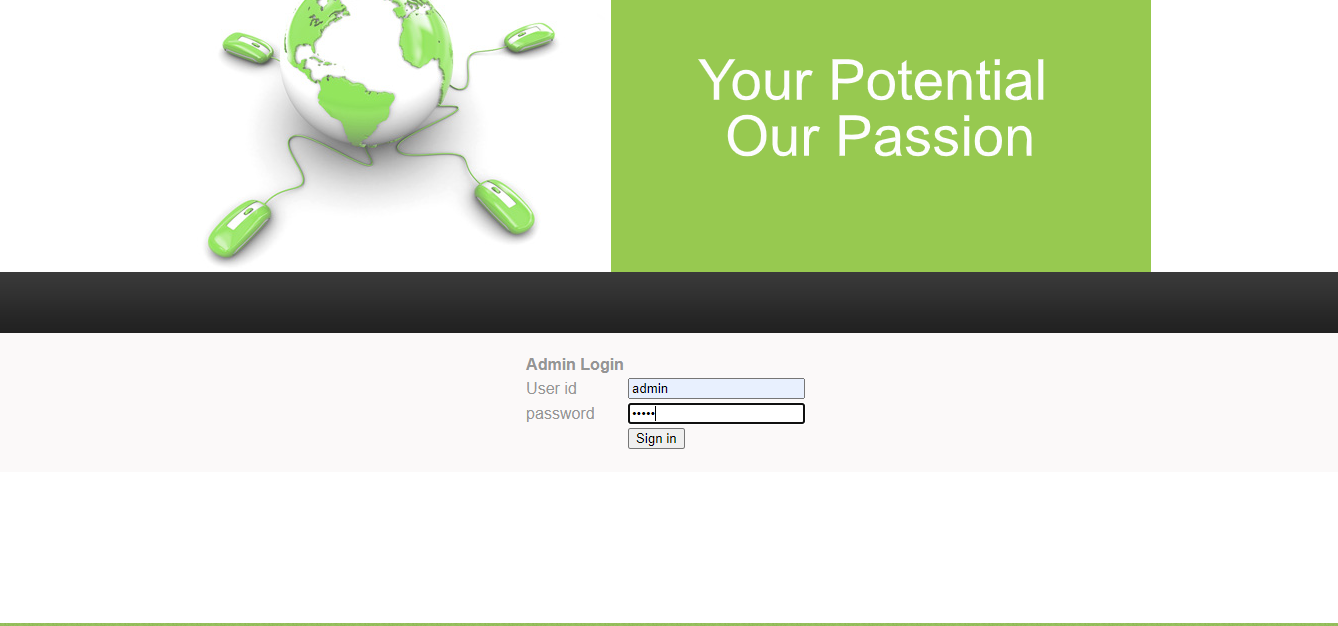


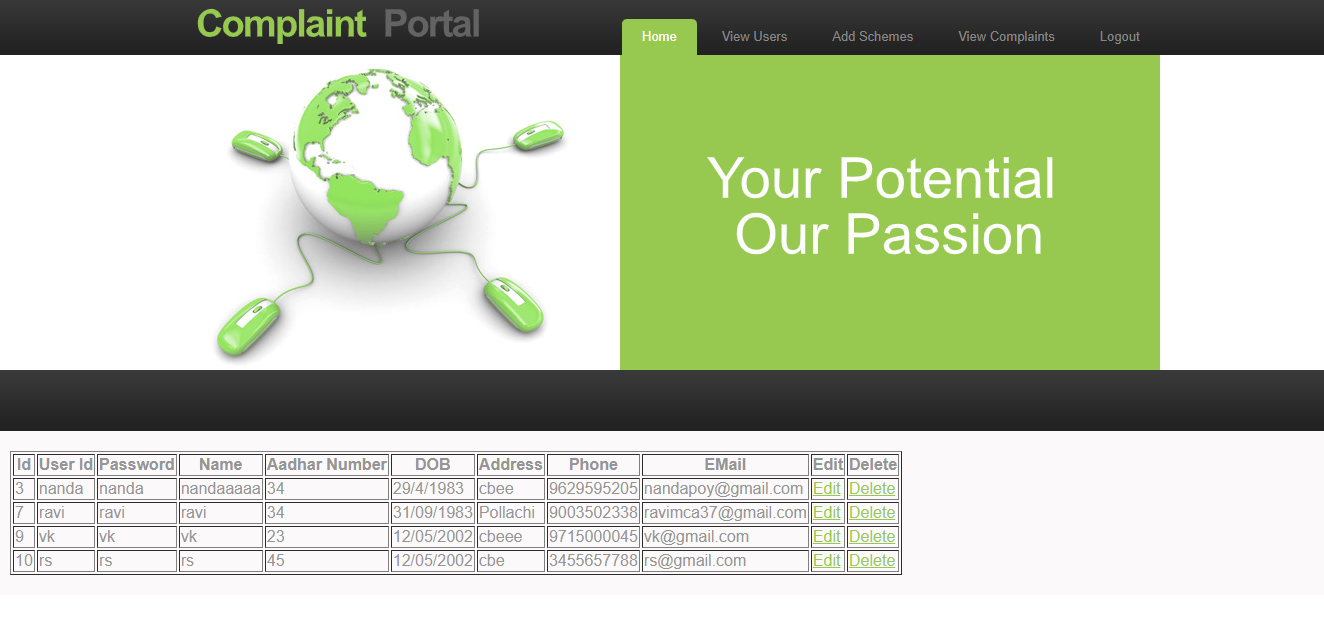


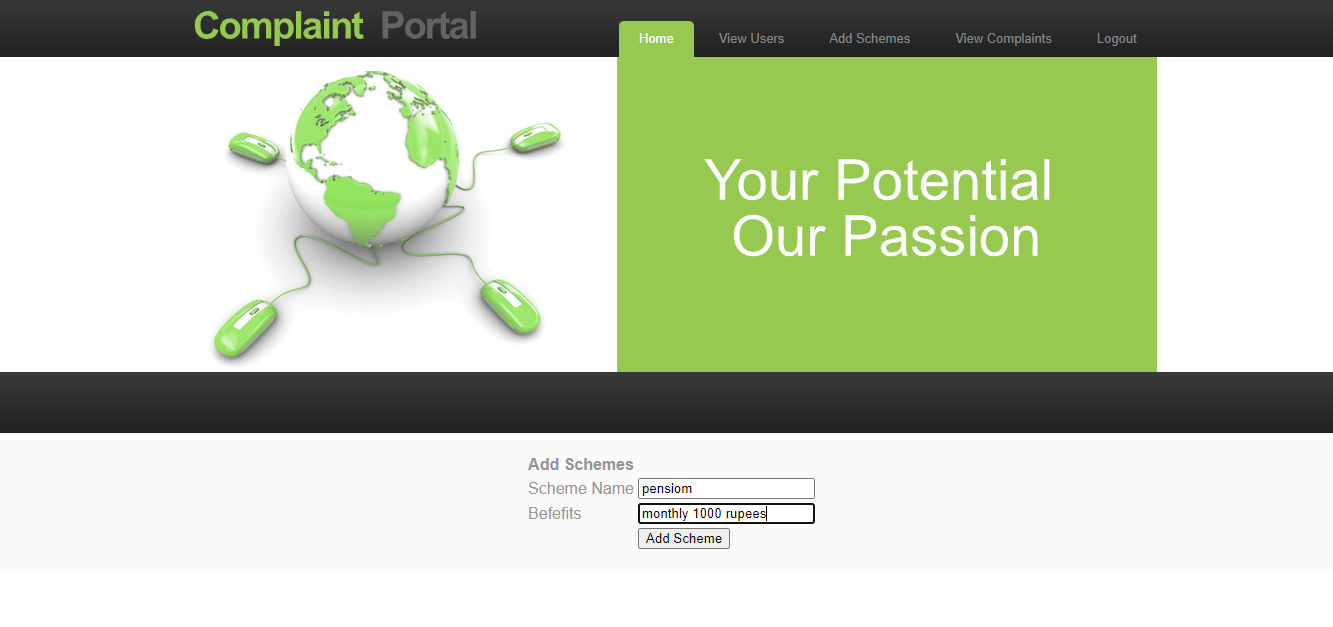


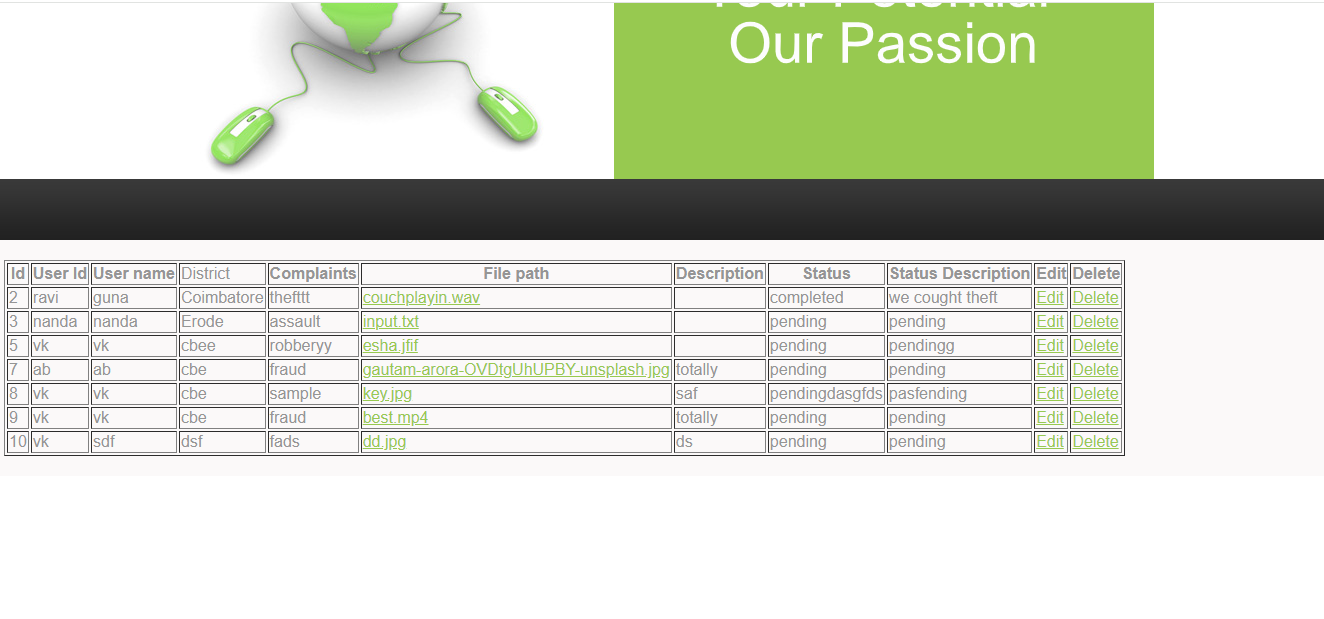












**Code**

from flask import Flask, render\_template, request

from flask import Flask, render\_template, json, request

from flask\_mysqldb import MySQL

#from werkzeug import generate\_password\_hash, check\_password\_hash

import pymysql

import sys

import pymysql.cursors

from flask import Flask, render\_template, request, redirect

from flask import Flask, render\_template, request

from flask import flash

#from werkzeug import secure\_filename

from flask import Flask, session, redirect, url\_for, escape, request

#from settings import PROJECT\_ROOT

import os

#import mysql.connector

app = Flask(\_\_name\_\_)

#UPLOAD\_FOLDER = url\_for('static',='/uploads')

UPLOAD\_FOLDER = '/static/uploads/'

ALLOWED\_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif', 'wav', 'mp3'])

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

@app.route('/')

def student():

return render\_template('index.html')

@app.route('/showusersignin',methods = ['POST', 'GET'])

def showusersignin():

return render\_template("usersignin.html",result = result)

@app.route('/services',methods = ['POST', 'GET'])

def services():

return render\_template("services.html",result = result)

@app.route('/contactus',methods = ['POST', 'GET'])

def contactus():

return render\_template("contactus.html",result = result)

@app.route('/showuserhome',methods = ['POST', 'GET'])

def showuserhome():

return render\_template("userhome.html",result = result)

@app.route('/showuserregister',methods = ['POST', 'GET'])

def showuserregister():

return render\_template("userregister.html",result = result)

@app.route('/ushow1',methods = ['POST', 'GET'])

def ushow1():

try:

return render\_template("postcomplaints.html")

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/logout',methods = ['POST', 'GET'])

def logout():

try:

return render\_template("index.html")

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/ushow2',methods = ['POST', 'GET'])

def ushow2():

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

sql = "select \* from complaints where uid = %s"

cursor.execute(sql,session['uid']) #,session['uid']

data = cursor.fetchall()

return render\_template("userviewcomplaints.html",data=data,eid=0)

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/ashow1',methods = ['POST', 'GET'])

def ashow1():

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

sql = "select \* from users"

cursor.execute(sql)

data = cursor.fetchall()

return render\_template("adminviewusers.html",data=data,eid=0)

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/ashow1/<int:id>',methods = ['POST', 'GET'])

def aedituser(id):

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

sql = "select \* from users"

cursor.execute(sql)

data = cursor.fetchall()

return render\_template("adminviewusers.html",data=data,eid=id)

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/adeleteuser/<int:id>',methods = ['POST', 'GET'])

def adeleteuser(id):

try:

if request.method == 'GET':

# validate the received values

if id:

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

# Read a single record

sql = "delete from users where id=%s"

cursor.execute(sql, (id))

connection.commit()

sql = "select \* from users"

cursor.execute(sql)

data = cursor.fetchall()

#return render\_template("adminviewusers.html", data=data, eid=0)

except Exception as e:

print(str(e))

finally:

#connection.close()

return render\_template("adminviewusers.html", data=data, eid=0)

#return "Update successfully done."

#data = cursor.fetchall()

else:

return json.dumps({'html': '<span>Enter the required fields</span>'})

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/adeletecomp/<int:id>',methods = ['POST', 'GET'])

def adeletecomp(id):

try:

if request.method == 'GET':

# validate the received values

if id:

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

# Read a single record

sql = "delete from complaints where id=%s"

cursor.execute(sql, (id))

connection.commit()

sql = "select \* from complaints"

cursor.execute(sql)

data = cursor.fetchall()

except Exception as e:

print(str(e))

finally:

return render\_template("adminviewcomplaints.html", data=data, eid=0)

#return "Update successfully done."

#data = cursor.fetchall()

else:

return json.dumps({'html': '<span>Enter the required fields</span>'})

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/udeletecomp/<int:id>',methods = ['POST', 'GET'])

def udeletecomp(id):

try:

if request.method == 'GET':

# validate the received values

if id:

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

# Read a single record

sql = "delete from complaints where id=%s"

cursor.execute(sql, (id))

connection.commit()

sql = "select \* from complaints where uid=%s"

cursor.execute(sql,session['uid'])

data = cursor.fetchall()

except Exception as e:

print(str(e))

finally:

return render\_template("userviewcomplaints.html", data=data, eid=0)

#return "Update successfully done."

#data = cursor.fetchall()

else:

return json.dumps({'html': '<span>Enter the required fields</span>'})

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/aupdclickuser',methods = ['POST', 'GET'])

def aupdclickuser():

try:

if request.method == 'POST':

p1 = request.form["p1"] #request.args.get('p1', '')

p2 = request.form["p2"]

p3 = request.form["p3"]

p4 = request.form["p4"]

p5 = request.form["p5"]

p6 = request.form["p6"]

p7 = request.form["p7"]

p8 = request.form["p8"]

p9 = request.form["p9"]

# validate the received values

if p1 and p2:

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

# Read a single record

sql = "update users set pwd=%s,name=%s,age=%s,dob=%s,addr=%s,phone=%s,email=%s where id=%s"

cursor.execute(sql, (p3,p4,p5,p6,p7,p8,p9,p1))

sql = "select \* from users"

cursor.execute(sql)

data = cursor.fetchall()

#return render\_template("adminviewusers.html", data=data, eid=0)

except Exception as e:

print(str(e))

finally:

#connection.close()

return render\_template("adminviewusers.html", data=data, eid=0)

#return "Update successfully done."

#data = cursor.fetchall()

else:

return json.dumps({'html': '<span>Enter the required fields</span>'})

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/aupdclickcomp',methods = ['POST', 'GET'])

def aupdclickcomp():

try:

if request.method == 'POST':

p1 = request.form["p1"] #request.args.get('p1', '')

p2 = request.form["p2"]

p3 = request.form["p3"]

# validate the received values

if p1 and p2:

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

# Read a single record

sql = "update complaints set status=%s,statusdesc=%s where id=%s"

cursor.execute(sql, (p2,p3,p1))

sql = "select \* from complaints"

cursor.execute(sql)

data = cursor.fetchall()

except Exception as e:

print(str(e))

finally:

#connection.close()

return render\_template("adminviewcomplaints.html", data=data, eid=0)

#return "Update successfully done."

#data = cursor.fetchall()

else:

return json.dumps({'html': '<span>Enter the required fields</span>'})

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/uupdclickcomp',methods = ['POST', 'GET'])

def uupdclickcomp():

try:

if request.method == 'POST':

p1 = request.form["p1"] #request.args.get('p1', '')

p2 = request.form["p2"]

# validate the received values

if p1 and p2:

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

# Read a single record

sql = "update complaints set complaints=%s where id=%s"

cursor.execute(sql, (p2,p1))

sql = "select \* from complaints where uid=%s"

cursor.execute(sql,session['uid'])

data = cursor.fetchall()

except Exception as e:

print(str(e))

finally:

#connection.close()

return render\_template("userviewcomplaints.html", data=data, eid=0)

#return "Update successfully done."

#data = cursor.fetchall()

else:

return json.dumps({'html': '<span>Enter the required fields</span>'})

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/ashow2',methods = ['POST', 'GET'])

def ashow2():

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

sql = "select \* from complaints"

cursor.execute(sql)

data = cursor.fetchall()

return render\_template("adminviewcomplaints.html",data=data,eid=0)

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/ashow2/<int:id>',methods = ['POST', 'GET'])

def aeditcomplaint(id):

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',

db='citizencomplaint',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

with connection.cursor() as cursor:

sql = "select \* from complaints"

cursor.execute(sql)

data = cursor.fetchall()

return render\_template("adminviewcomplaints.html",data=data,eid=id)

except Exception as e:

return json.dumps({'error': str(e)})

@app.route('/ushow2/<int:id>',methods = ['POST', 'GET'])

def ueditcomplaint(id):

try:

connection = pymysql.connect(host='localhost',

user='root',

password='',