

**SYNOPSIS**

**Title: “HELPHUB” SOCIAL REQUISITION AND COMPLAINT MANAGEMENT SYSTEM**

**OBJECTIVE:**

This project named as helphub it is built using vb.Net Framework Visual Basic C# desktop application software and SQlite, which mainly focuses on receiving complaints and requests for services based on social issues like road , sewage or water, electricity, domestic, and other issues from the general public and can also check status of their complaints and requests made by them, there are root admin , supervisor, state admin and admin who monitor the complaints and requests update the status regarding the same to its users.

The objective of the HelpHub project is to create a comprehensive system for receiving, processing, and responding to complaints and service requests from the general public. The system should provide a user-friendly interface, allowing users to submit and monitor their requests, and should provide administrators with the tools to effectively manage the complaints and requests. Additionally, the system should be secure and reliable, utilizing the latest database technology to ensure data integrity and system stability.

In this project we are using Visual Basic C# 2022 to mainly design the front end of the project and the back end is implemented on SQLite

C# is a general and modern object-oriented programming (OOP) language provided by Microsoft that runs on .Net Framework. C# is pronounced as “C-Sharp”. C# is specially designed and developed to work with Microsoft’s.Net platform.

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day and provides a good graphical user interface to the user of the system.

**List of Modules:**

**User:** Can file complaints and requests, check the status of complaints and requests, and update their profile details. They can also download a consolidated report of complaints and requests.

**State Admin:** Can check complaints and requests within their specific state, update the status of complaints and requests, filter complaints and requests by type, update their profile details, and download a consolidated report of complaints and requests.

**Supervisor:** Can check complaints and requests from all states, update the status of complaints and requests, filter complaints and requests by state and type, update their profile details, and download a consolidated report of complaints and requests.

**Admin:** Can check complaints and requests from all states, update the status of complaints and requests, filter complaints and requests by state and type, delete complaints and requests, add new supervisors and state admin, ban unwanted users, update their profile details, and download a consolidated report of complaints and requests.

**Super Admin:** Has all the privileges of an admin, can access user and admin logs, ban and unban logs, user, supervisor, admin, and state admin databases, perform adding, deleting, and modification of data, download the whole data in pdf format, access the super admin database in read-only mode, and view statistics of user and complaint/request databases.

**Root:** Has all the privileges of a Super Admin, can access super admin logs, and perform adding, deleting, and modification of data.

**Complain:** Users can file a complaint regarding their social and domestic issues, Track updates of their complaint.

**Request:** Users can request a service regarding their social and domestic issues, Track updates of their request from the concerned authorities.

*Requirement*

*Specification*

**Hardware Requirements:**

* Processor : 32 BIT, Pentium-4 or higher
* RAM : 2 GB RAM or higher
* Hard disk : 20 GB (Minimum)
* Monitor : 1024x768(Resolution)
* Keyboard
* Mouse
* Printer

**Software requirements:**

* Operating system : Windows 8 or above
* Front end : VB.NET C# 2022
* Back end : SQLite (serverless)

*Introduction*

**INTRODUCTION**

**Introduction:**

Introducing Helphub, a powerful desktop application built using VB C#, designed to address social issues faced by the general public. This innovative platform allows citizens to submit complaints and requests for services related to roads, sewage, water, electricity, domestic issues and more. With Helphub, citizens can rest assured that their concerns will be heard and addressed on time. The platform is overseen by a team of root admin, supervisor, state admin and admin who work diligently to monitor and resolve complaints and requests. With Helphub, we are committed to improving the quality of life for our citizens and building a better community.

Helphub is a one-stop solution for citizens looking to report and resolve social issues in their community. The user-friendly interface and easy-to-use features make it simple for anyone to submit a complaint or request for service. Whether it's a pothole on a local street, a water leak, or an electrical issue, Helphub is here to help. Our team of dedicated professionals works tirelessly to ensure that complaints and requests are handled efficiently and effectively. With real-time tracking and updates, citizens can stay informed about the status of their complaints or request.

Furthermore, Helphub is equipped with advanced analytics and reporting tools that allow our team to identify and address recurring issues in the community. This helps us to take proactive measures and improve our services over time. With Helphub, we are committed to building a more responsive and accountable government that truly serves the needs of the people.

In summary, Helphub is a powerful and innovative desktop application that empowers citizens to report and resolve social issues in their community. With a user-friendly interface, advanced tracking and reporting features, and a dedicated team of professionals, Helphub is the perfect solution for anyone looking to make a positive impact in their community.

*Objective*

**OBJECTIVE:**

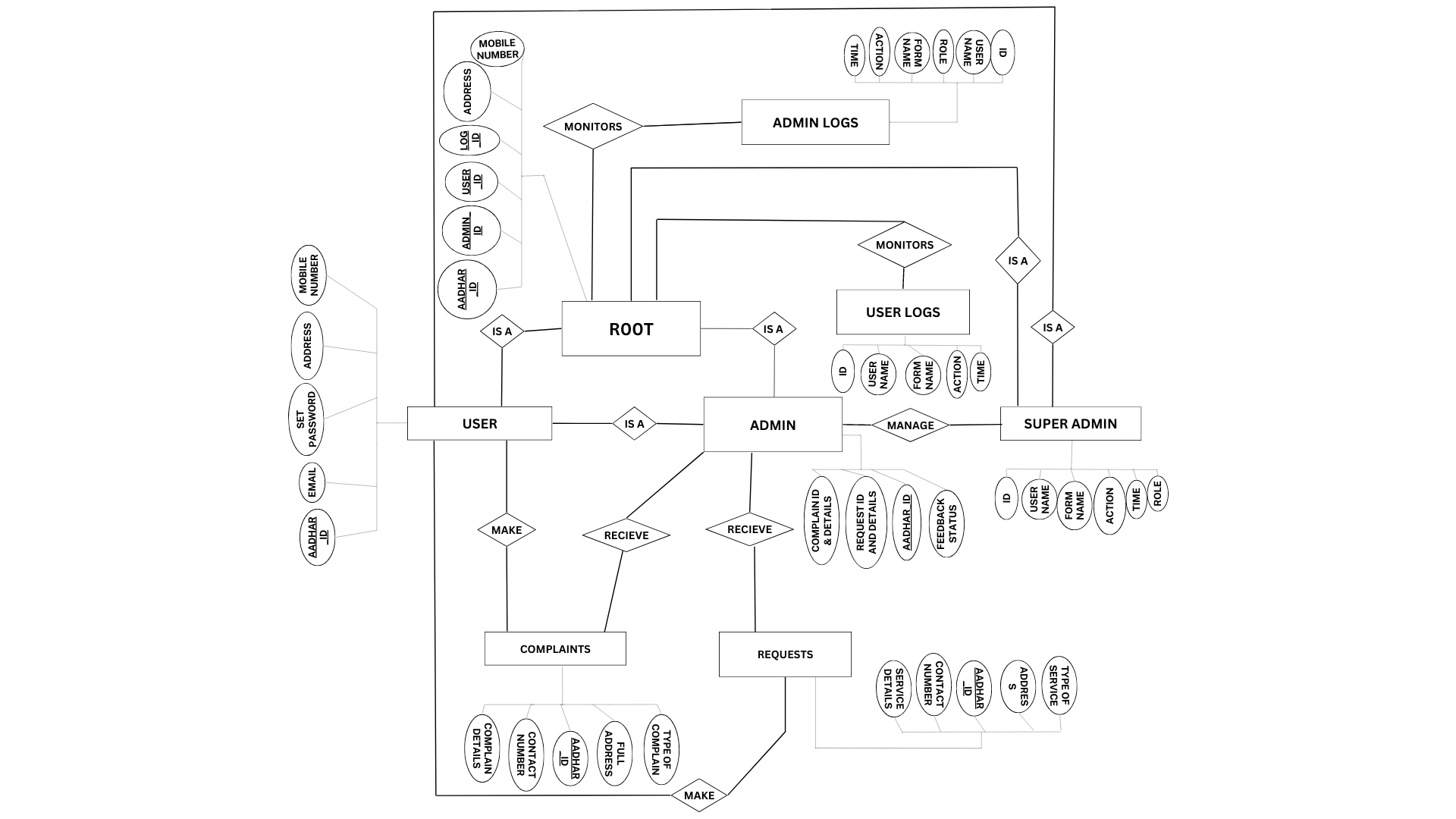
The main objective of Helphub is to provide an easy-to-use and efficient platform for citizens to report and track issues in their community. The application is designed to streamline the process of reporting and tracking issues, making it easier for citizens to get the help they need.

The objective of HelpHub is to create a comprehensive system for receiving, processing, and responding to complaints and service requests from the general public. HelpHub will be built using the .NET Framework, Visual Basic, C#, and SQLite, and will provide a user-friendly interface for submitting, tracking, and managing complaints and service requests. The system will also be secure and reliable, utilizing the latest database technology to ensure data integrity and system stability. Additionally, HelpHub will provide administrators with the tools to effectively manage the complaints and requests and track the status of each request. HelpHub will allow users to submit complaints and service requests based on issues such as road, sewer, water, electricity, domestic, and other social issues, and will enable users to easily track the status of their complaints and requests.

The system should provide a user-friendly interface, allowing users to submit and monitor their requests, and should provide administrators with the tools to effectively manage the complaints and requests. Additionally, the system should be secure and reliable, utilizing the latest database technology to ensure data integrity and system stability.

*E-R diagram*

**E-R Diagram**



*Tool description*

**Visual Basic 2022**

Microsoft Visual Basic 2022 is the next version of the popular Visual Basic programming language. Visual Basic 2022 will include support for the latest language features and tools, along with improved productivity and performance. The language will include support for the latest versions of .NET, as well as support for 64-bit applications. Visual Basic 2022 provides improved support for data types, including the new BigInteger data type and the improved DateTime data type. It also includes a new collection of control properties for better performance and improved user experience. Additionally, Visual Basic 2022 includes improved debugging tools, such as the new IntelliTrace feature, which enables developers to quickly identify and diagnose errors in their code. Finally, Visual Basic 2022 includes improved IntelliSense features and improved productivity features, such as the new refactoring tools.

Like Visual Basic 2019 and Visual Basic Express 2013 is also a full-fledged Object-Oriented Programming (OOP) Language, so it has caught up with other OOP languages such as C++, Java, C# and others.

**Visual Basic 2022 Data Types:**

Visual Basic 2013 classifies the information mentioned above into two major data types, they are the numeric data types and the non-numeric data types.

**Numeric Data Types**

Numeric data types are types of data that consist of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide and so on. Examples of numeric data types are your examination marks, your height, your weight, the number of students in a class, share values, price of goods, monthly bills, fees, etc. In Visual Basic 2008, numeric data are divided into 7 types, depending on the range of values they can store. Calculations that only involve round figures or data that don't need precision can use Integer or Long integer in the computation. Programs that require high precision calculation need to use Single and Double decision data types; they are also called floating point numbers. For currency calculation, you can use the currency data types. Lastly, if even more precision is required to perform calculations that involve a many decimal points, we can use the decimal data types

**Non-numeric Data Types**

Non-Numeric data types are data that cannot be manipulated mathematically using standard arithmetic operators. The non-numeric data comprises text or string data types, the Date data types, the Boolean data types that store only two values (true or false), Object data type and Variant data type.

**Visual Studio 2022 Project Page:**

The New Project Page comprises three templates, Visual Basic, Visual C# and Visual C++. We shall select Visual Basic. Visual Basic 2022 offers you four types of projects that you can create. As we are going to learn to create Windows Applications, we will select Windows Forms Application.

At the bottom of this dialog box, you can change the default project name WindowsApplication1 to some other name you like, for example, MyFirstProgram. After you have renamed the project, click OK to continue.

Controls in Visual Basic 2022 are objects that can be placed on the form to perform various tasks. We can use them to create all kinds of Windows applications. The diagram below shows the toolbox that contains the controls of Visual Basic 2022. They are categorized into Common Controls, Containers, Menus, Toolbars, Data, Components, Printings and Dialogs. Now, we will focus on the common controls. Some of the most frequently used common controls are Button, Label, ComboBox, ListBox, PictureBox, Text Box etc.

To insert a control into your form in Visual Basic 2022 IDE, you just need to drag the control from the toolbox and drop it into the form. You can reposition and resize it as you like.

**The Control Properties in Visual Basic 2022:**

Before writing an event procedure for a control in Visual Basic 2022 to respond to a user’s input or action, you have to set certain properties for the control to determine its appearance and how it will work with the event procedure. You can set the properties of the controls in the properties window of Visual Basic 2022 IDE at design time or at run time.

**SQLite (serverless)**

SQLite was first designed and implemented by D. Richard Hipp in the year 2000, as a part of a project for a client, but later Hipp decided to make it open-source. Since then, SQLite has become one of the most widely used and popular database engines in the world, with a wide range of applications, including mobile phones, embedded systems, web browsers, and many other software applications.

One of the most significant features of SQLite is its small size and minimalistic design. The entire library, including the database engine and the command-line interface, is less than 1MB in size. This makes it easy to distribute and use in embedded systems and other environments where disk space is limited.

SQLite is also known for its reliability and robustness, as it is built on a transactional storage engine, which ensures that all data changes are atomic, consistent, isolated, and durable. Additionally, it supports advanced features like multi-threading, transactions, and rollbacks, which are essential for high-performance and fault-tolerance.

SQLite is also widely supported by many programming languages, including C, C++, Java, Python, Perl, Ruby, and many others. This has made it easy to integrate SQLite into various software applications, and as a result, it has become one of the most widely used and popular database engines in the world.

SQLite is a relational database management system (RDBMS) contained in a C library. It is a self-contained, serverless, zero-configuration, and transactional SQL database engine. SQLite is commonly used in mobile apps and web browsers, as well as in embedded systems and the Internet of Things (IoT) devices.

It is also often used as a testing and development database, due to its simplicity and ease of use. Some of the features of SQLite include:

ACID (Atomicity, Consistency, Isolation, Durability) compliance

Support for most SQL92 standard

Simple and easy to use API.

No setup or administration required.

Small footprint and minimal resource requirements

Cross-platform compatibility

Open-source, Public-Domain license

SQLite stores data in tables, which can be queried using SQL (Structured Query Language). It supports most SQL commands, including SELECT, INSERT, UPDATE, DELETE, and CREATE.

SQLite is a powerful, open-source, serverless, zero-configuration, transactional SQL database engine. It is self-contained, meaning it does not require a separate server process or system to operate. It is lightweight, fast and reliable, making it an ideal choice for applications requiring a small footprint and high performance. SQLite supports most of the standard SQL features as well as additional features such as triggers, views and full-text search. It is also highly portable, making it easy to transfer data between systems. SQLite is free and open source, making it an ideal choice for many applications.

SQLite is a popular choice for embedded systems and mobile devices because of its small size and minimal resource requirements. It can run on a wide range of platforms, including iOS, Android, and Windows, and it can be integrated into a variety of programming languages, including C, C++, Python, and Java.

SQLite is also a popular choice for web browsers because it can store data locally in the browser, making it available offline. This is useful for creating web apps that can work offline and for storing user data locally for improved performance.

SQLite is also a popular choice for testing and development because it is easy to use and requires no setup or administration. It can be used as a test database for testing the functionality of an application without the need to set up a separate test environment.

SQLite is an open-source software, and it is in public domain, meaning it is free to use, distribute and modify. It is also actively developed and maintained, with new releases and updates regularly.

Features:

features of the SQLite database:

1. Serverless –

SQLite does not require a server or any other additional software to run, making it easy to install and maintain.

1. Cross-platform –

SQLite is compatible with Linux, Windows, MacOS and other operating systems.

1. Lightweight –

SQLite is a lightweight database, meaning it does not require many resources to run.

1. Flexible –

SQLite can be used with multiple programming languages, making it easy to integrate with existing applications.

1. Secure –

SQLite includes a built-in security system to protect against malicious attacks.

1. Full-featured –

SQLite offers a full suite of features, including support for data types, triggers, and views.

1. Portable –

SQLite is portable, meaning it can be used across multiple platforms without any compatibility issues.

1. Scalable –

SQLite is designed to be scalable, allowing it to handle increasing amounts of data without any performance issues.

1. Free –

SQLite is open-source and free to use.

1. Reliable –

SQLite is designed to be reliable and robust, ensuring data integrity and system stability.

*System analysis*

**Feasibility Study**

Feasibility study is the measure of how beneficial or practical the development of an information system will be to an organization. The Feasibility analysis is a cross life cycle activity and should be continuously performed throughout the system life cycle. Feasibility study lets the developer foresee the future of the project and the usefulness. The study on feasibility is done on the basis of a few factors. They are:

**Operational feasibility:**

The operational feasibility of the HelpHub project is high due to its simple design and the use of existing technologies. The project is built using Visual Basic CSharp 2022, which is a popular programming language with a wide range of features and tools. Additionally, the project uses SQLite, which is a reliable and widely-used database engine. This ensures that the project will be able to store and manage large amounts of data with minimal effort. Furthermore, the project is designed to be user-friendly and can easily be used by people with limited technical knowledge. Therefore, the operational feasibility of HelpHub is high. Considering all these factors we can conclude that all the users and end users will be satisfied by the system.

**Technical feasibility:**

The technical feasibility of the project is also high. The project is being built using VB C# as the front-end development tool and SQLite as the back-end database. SQLite is a widely used, open-source database engine that is known for its small size, high performance, and reliability. It is also easy to use and requires minimal setup and administration. The use of SQLite as the back-end database for the project will help to ensure that the system can handle large amounts of data efficiently and provide fast and accurate results to the users.

The use of VB C# to design the front-end of the project will provide a good graphical user interface to the users of the system. VB C# is a popular programming language that is known for its ease of use and its ability to create visually appealing and user-friendly interfaces.

Overall, the project "Helphub" appears to be technically and operationally feasible, with a well-defined scope, clear objectives, and a solid plan for development and implementation.

**Schedule feasibility:**

The duration of time required for the project has been planned appropriately and it is the same as the duration of time expected by the customer. Therefore the product can be delivered to the customer within the expected time duration, satisfying the customer. Hence the project is feasible in scheduling.

**Economic feasibility:**

According to the resources available and the project scheduling process it is estimated that the expenses allocated for the software to be developed, by the customer is sufficient enough. Hence the economical factor has been considered feasible.

**Behavioural Feasibility:**

This includes the following questions:

* Is there sufficient support for the users?
* Will the proposed system cause harm?

This project would be beneficial because it satisfies the objectives when developed and installed. All behavioural aspects are considered carefully and conclude that the project is behaviourally feasible.

*Data flow diagram*

**DATA FLOW DIAGRAM**

**Data Flow diagram:**

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD)

The idea behind the explosion of a process into more process is that understanding at one level of details is exploded into greater detailed at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical form, this lead to modular design.

A DFD is known as a “bubble chart” has the purpose of clarifying system requirements and identifying major transformation they will become program in system design. So it is the starting point of the design to lowest level of details. A DFD consists of series of bubbles joined by data flows in the system.

**DFD Symbols**

In DFD, there are four Symbols

1. A square-defines a source or destination system data
2. An arrow identified data flow. Its is the pipeline through which the information flow
3. A circle or a bubble represents a process that transforms
4. Incoming data flow into outgoing data flows
5. An open rectangle is a data source, data at rest or a temporary of data

**Process that transforms data flow**

**Source or destination of the data**

**Data flow**

**Process for data store.**

**CONTEXT LEVEL :**

**Diagram

Description automatically generated**

**LEVEL 1 :**

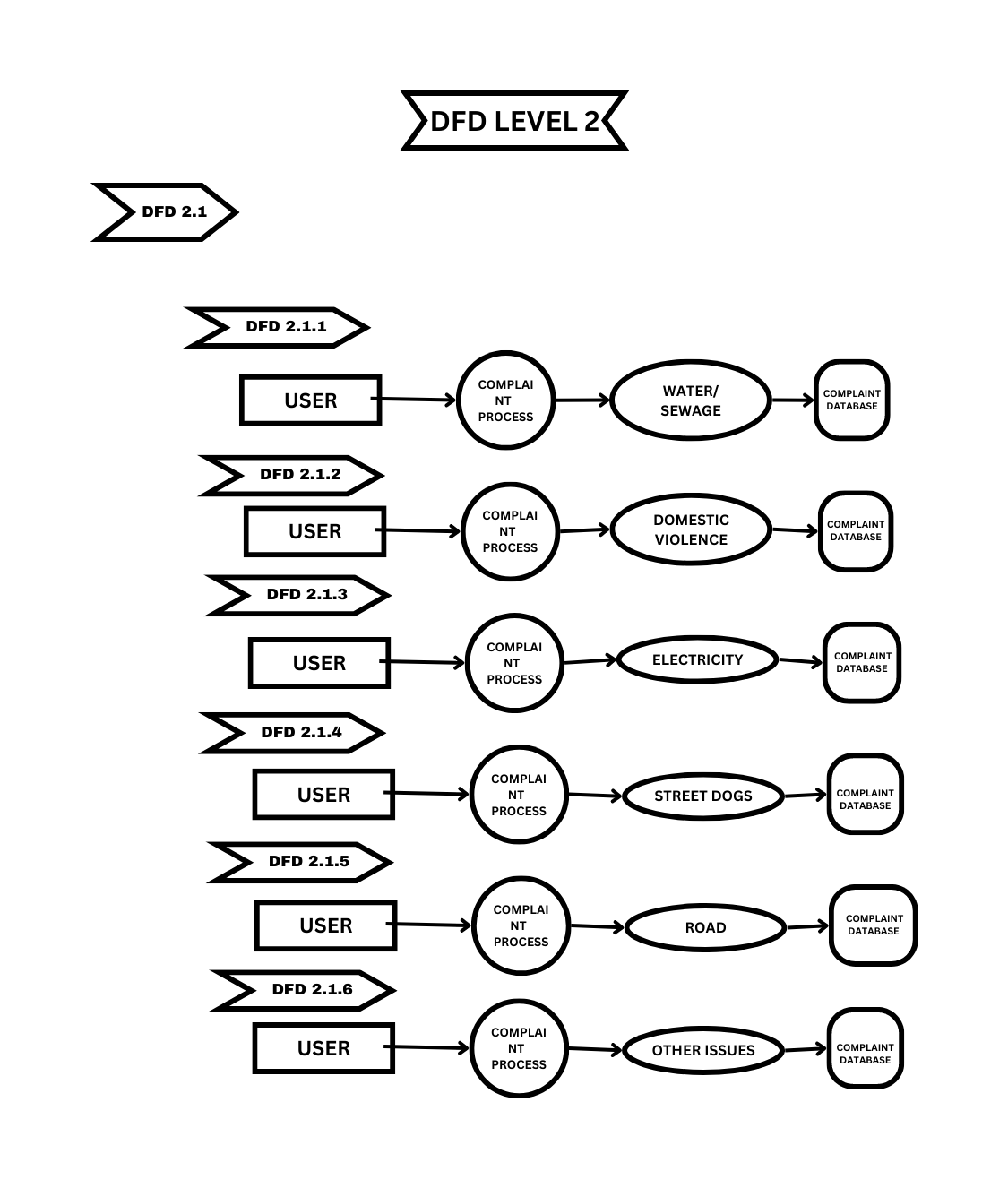
Diagram

Description automatically generated

Diagram

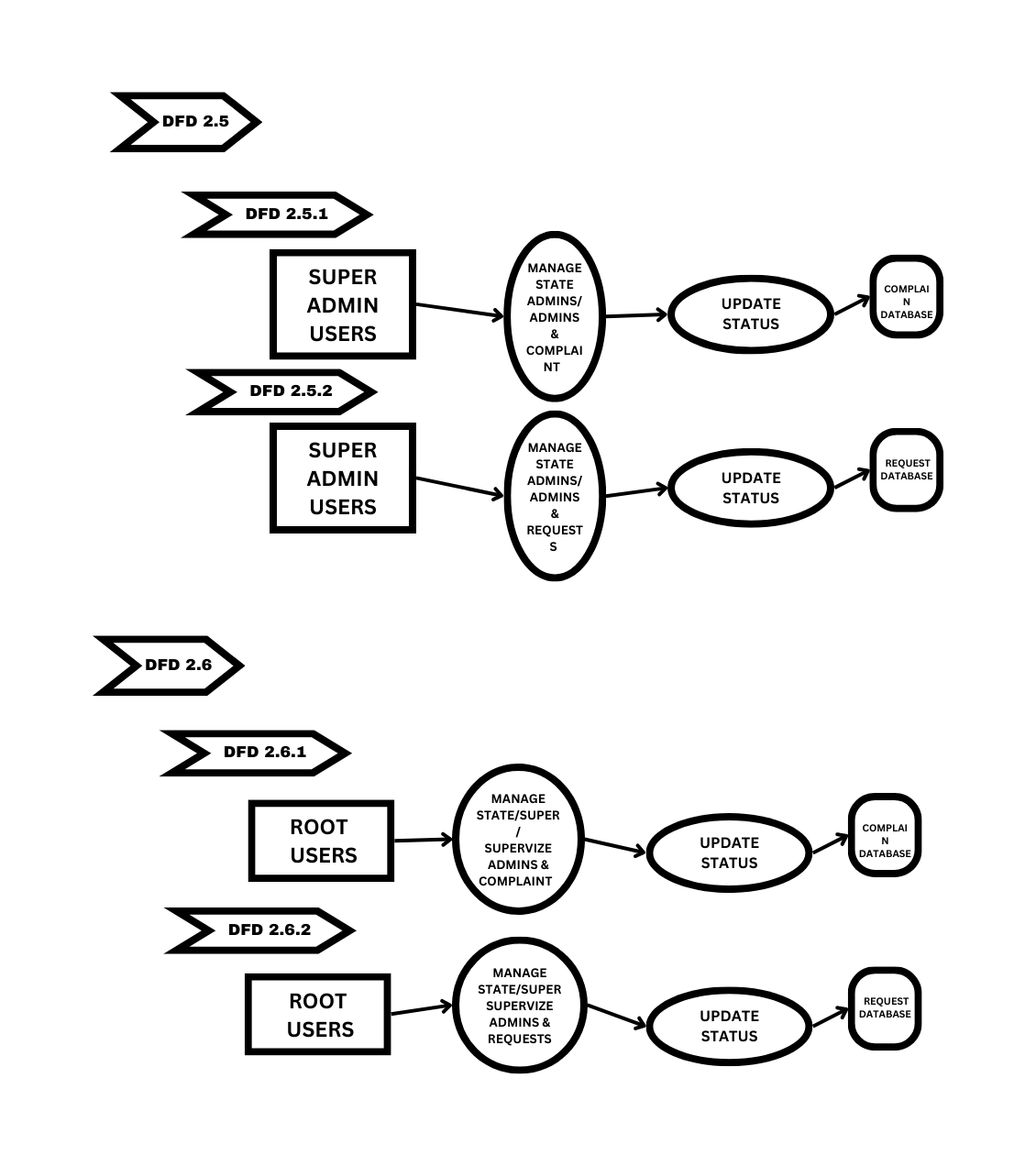
Description automatically generated

**LEVEL 2:**

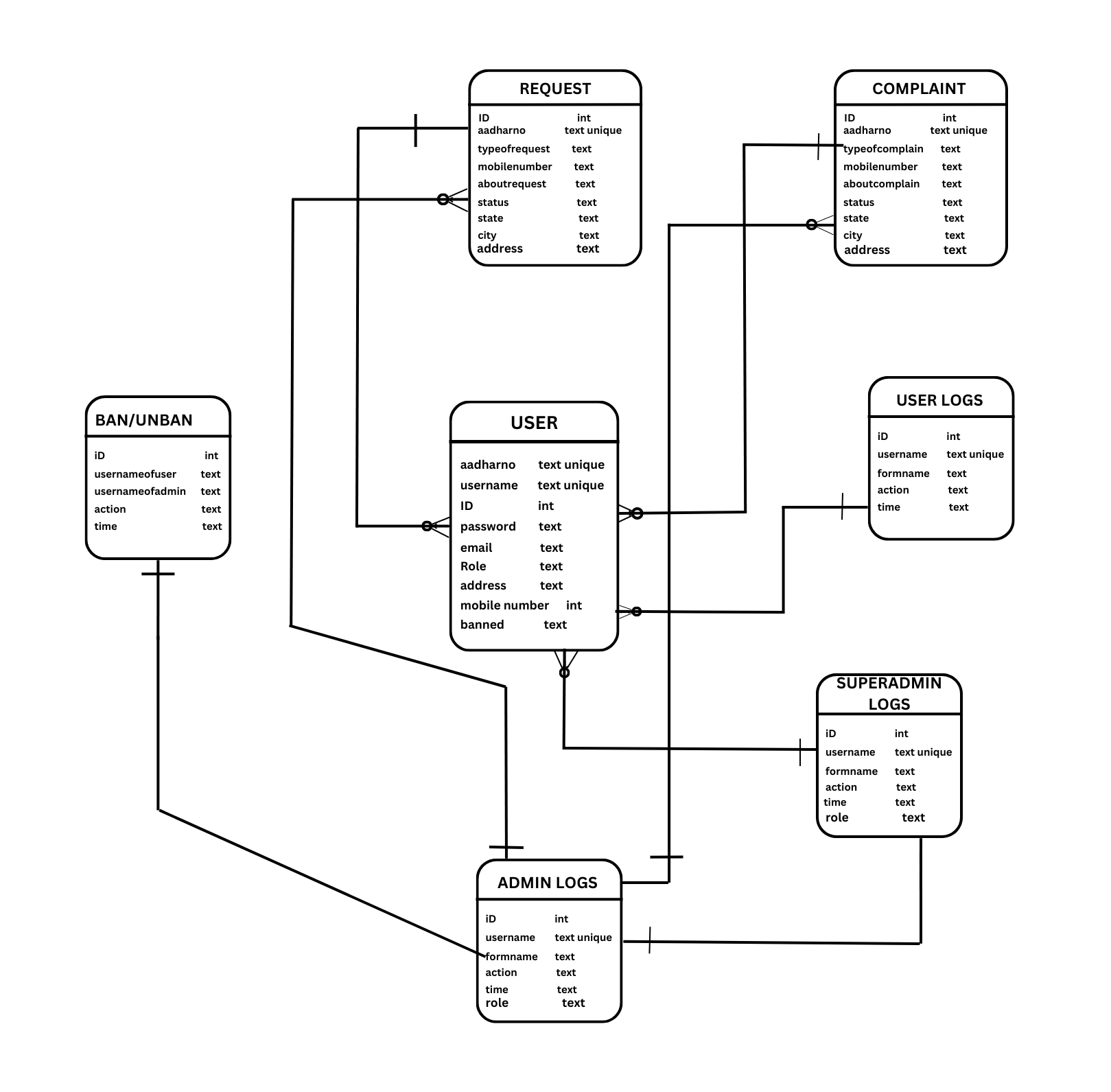


Diagram

Description automatically generated



*Schema*

****

*Source code*

**SOURCE CODE**

UserData.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace helphub

{

   public class UserData

   {

       public static string ID = "";

       public static string username = "";

       public static string aadharno = "";

       public static string mobilenumber = "";

       public static string password = "";

       public static string email = "";

       public static string role = "";

       public static string address = "";

   }

}

Database.cs

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;

namespace helphub

{

   public class Database

   {

       // staic obj variable of class so it can be accessed without create object

       public static Database databaseobj = new Database();

       public DataTable dt = new DataTable();

       public DataTable dt1 = new DataTable();

       string SQLitecnStr = @"Data Source=.\helphub.db";

       SQLiteConnection SQLiteConn = new SQLiteConnection();

       SQLiteCommand SQLitecmd = new SQLiteCommand();

       SQLiteCommand SQLitecmd1 = new SQLiteCommand();

       public Database()

       {

           SQLiteConn.ConnectionString = SQLitecnStr;

           SQLiteConn.Open();

           SQLitecmd.Connection = SQLiteConn;

       }

       void checkconn()

       {

           if(SQLiteConn.State == ConnectionState.Closed)

           {

               SQLiteConn.Open();

           }

       }

       // state list

       public List<string> StateList = new List<string> {

           "AP|Andhra Pradesh",

           "AR|Arunachal Pradesh",

           "AS|Assam",

           "BR|Bihar",

           "CT|Chhattisgarh",

           "GA|Goa",

           "GJ|Gujarat",

           "HR|Haryana",

           "HP|Himachal Pradesh",

           "JK|Jammu and Kashmir",

           "JH|Jharkhand",

           "KA|Karnataka",

           "KL|Kerala",

           "MP|Madhya Pradesh",

           "MH|Maharashtra",

           "MN|Manipur",

           "ML|Meghalaya",

           "MZ|Mizoram",

           "NL|Nagaland",

           "OR|Odisha",

           "PB|Punjab",

           "RJ|Rajasthan",

           "SK|Sikkim",

           "TN|Tamil Nadu",

           "TG|Telangana",

           "TR|Tripura",

           "UT|Uttarakhand",

           "UP|Uttar Pradesh",

           "WB|West Bengal",

           "AN|Andaman and Nicobar Islands",

           "CH|Chandigarh",

           "DN|Dadra and Nagar Haveli",

           "DD|Daman and Diu",

           "DL|Delhi",

           "LD|Lakshadweep",

           "PY|Puducherry"};

       // clear function

       public void clear()

       {

           dt.Clear();

           dt.Columns.Clear();

           dt1.Clear();

           dt1.Columns.Clear();

       }

       // common functions

       public Boolean check\_username\_exist(string username)

       {

           clear();

           checkconn();

           SQLitecmd.CommandText = "SELECT \* FROM user WHERE username='" + username + "'";

           SQLiteDataAdapter da = new SQLiteDataAdapter(SQLitecmd);

           da.Fill(dt);

           if (dt.Rows.Count == 1)

           {

               return true;

           }

           return false;

       }

       public int deleteaccount(string username,string formname)

       {

           try

           {

               clear();

               checkconn();

               SQLitecmd.CommandText = "DELETE FROM user WHERE ID=" + UserData.ID + "";

               int status = SQLitecmd.ExecuteNonQuery();

               if (status == 0)

               {

                   CreateLogs.createlogobj.userlog(username, "Unable to delete account", formname);

                   MessageBox.Show("Unable to Delete Account");

                   return 0;

               }

               else

               {

                   CreateLogs.createlogobj.userlog(username, "Account Deleted", formname);

                   MessageBox.Show("Account Successfully Deleted", "Account Deleted");

                   LOGIN login = new LOGIN();

                   login.Show();

                   return 1;

               }

           }

           catch (Exception ex)

           {

               CreateLogs.createlogobj.userlog(username, "Unable to delete account " + ex.Message, formname);

               MessageBox.Show("Unable to Delete Account", ex.Message);

           }

           return 0;

       }

       public void FetchData(DataGridView complaindataview, DataGridView requestdataview)

       {

           clear();

           checkconn();

           try

           {

               // connection + query

               SQLitecmd.Connection = SQLiteConn;

               SQLitecmd.CommandText = "SELECT \* FROM complaint WHERE aadharno='" + UserData.aadharno + "'";

               // connection + query

               SQLitecmd1.Connection = SQLiteConn;

               SQLitecmd1.CommandText = "SELECT \* FROM request WHERE aadharno='" + UserData.aadharno + "'";

               // data adapter

               SQLiteDataAdapter da = new SQLiteDataAdapter(SQLitecmd);

               SQLiteDataAdapter da1 = new SQLiteDataAdapter(SQLitecmd1);

               // filling data in datatable with adapter

               da.Fill(dt);

               da1.Fill(dt1);

               if (dt.Rows.Count > 0 || dt1.Rows.Count > 0)

               {

                   complaindataview.DataSource = dt;

                   requestdataview.DataSource = dt1;

               }

               else

               {

                   MessageBox.Show("No records found", "STATUS", MessageBoxButtons.OK, MessageBoxIcon.Error);

               }

               SQLiteConn.Close();

           }

           catch (Exception ex)

           {

               MessageBox.Show("Unable to Load Data", ex.Message);

               Console.WriteLine(ex);

           }

       }

       // specific function

       public void register(REGISTER passedFunction)

       {

           clear();

           checkconn();

           SQLitecmd.CommandText = "insert into user(aadharno,username,mobilenumber,password,email,address) VALUES('" + passedFunction.Aadhar.Text + "','" + passedFunction.username.Text + "','" + passedFunction.Contact.Text + "','" + passedFunction.Password.Text + "','" + passedFunction.Email.Text + "','" + passedFunction.Address.Text + "')";

           try

           {

               CreateLogs.createlogobj.userlog(passedFunction.username.Text, "User Registered", passedFunction.Name);

               SQLitecmd.ExecuteNonQuery();

               MessageBox.Show("Registered Successfully", "Register", MessageBoxButtons.OK, MessageBoxIcon.Information);

               LOGIN login = new LOGIN();

               login.Show();

               passedFunction.Close(); //Close Form1,the current open form.

           }

           catch (SQLiteException ex)

           {

               int code = ex.ErrorCode;

               if (code == 19)

               {

                   CreateLogs.createlogobj.userlog(passedFunction.username.Text, "Already Registered Username/Aadhar Number", passedFunction.Name);

                   MessageBox.Show("Already Registered Username/Aadhar Number", "Register", MessageBoxButtons.OK, MessageBoxIcon.Information);

               }

               else

               {

                   CreateLogs.createlogobj.userlog(passedFunction.username.Text, "Database Error " + ex.Message, passedFunction.Name);

                   MessageBox.Show("Database Error: Error code:- " + code + ",Error message:- " + ex.Message + "", "Register", MessageBoxButtons.OK, MessageBoxIcon.Error);

               }

           }

           catch (Exception ex)

           {

               CreateLogs.createlogobj.userlog(passedFunction.username.Text, "Registration Failed " + ex.Message, passedFunction.Name);

               MessageBox.Show("Registration Failed: " + ex.Message + "", "Register", MessageBoxButtons.OK, MessageBoxIcon.Error);

           }

       }

       public void login(LOGIN passedFunction)

       {

           clear();

           checkconn();

           try

           {

               SQLitecmd.CommandText = "SELECT \* FROM user WHERE username='" + passedFunction.Username.Text + "' AND password='" + passedFunction.Password.Text + "'";

               SQLiteDataAdapter da = new SQLiteDataAdapter(SQLitecmd);

               da.Fill(dt);

               if (dt.Rows.Count == 1)

               {

                   if (dt.Rows[0]["banned"].ToString() == "YES")

                   {

                       CreateLogs.createlogobj.userlog(passedFunction.Username.Text, "Banned user tried login", passedFunction.Name);

                       MessageBox.Show("Account Banned", "SECURITY SYSTEM HELPHUB", MessageBoxButtons.OK, MessageBoxIcon.Error);

                       return;

                   }

                   foreach (DataRow row in dt.Rows)

                   {

                       UserData.ID = row["ID"].ToString();

                       UserData.username = row["username"].ToString();

                       UserData.aadharno = row["aadharno"].ToString();

                       UserData.mobilenumber = row["mobilenumber"].ToString();

                       UserData.password = row["password"].ToString();

                       UserData.email = row["email"].ToString();

                       UserData.role = row["Role"].ToString();

                       UserData.address = row["address"].ToString();

                   }

                   if (UserData.role == "root")

                   {

                       ROOTDASHBOARD rootadmin = new ROOTDASHBOARD();

                       rootadmin.Show();

                       passedFunction.Close();

                       return;

                   }

                   if (UserData.role == "SUPERADMIN")

                   {

                       SUPERADMIN superadmin = new SUPERADMIN();

                       superadmin.Show();

                       passedFunction.Close();

                       return;

                   }

                   if (UserData.role == "ADMIN" || UserData.role == "SUPERVISOR" || StateList.Contains(UserData.role))

                   {

                       CreateLogs.createlogobj.userlog(passedFunction.Username.Text, UserData.role + " Logged in", passedFunction.Name);

                       CreateLogs.createlogobj.adminlog(passedFunction.Text, "Logged in", passedFunction.Name, UserData.role);

                       ADMIN admin = new ADMIN();

                       admin.Show();

                       passedFunction.Close(); //Close Form1,the current open form.

                       return;

                   }

                   else

                   {

                       MessageBox.Show("Logged in Successfully", "LOGIN", MessageBoxButtons.OK, MessageBoxIcon.Information);

                       CreateLogs.createlogobj.userlog(passedFunction.Username.Text, UserData.role + " Logged in", passedFunction.Name);

                       DASHBOARD dashboard = new DASHBOARD();

                       dashboard.Show();

                       passedFunction.Close(); //Close Form1,the current open form.

                   }

               }

               else if (dt.Rows.Count >= 2)

               {

                   CreateLogs.createlogobj.userlog(passedFunction.Username.Text, "Multiple accounts error", passedFunction.Name);

                   MessageBox.Show("Multiple Username Error, Kindly Contact Admin....", "LOGIN", MessageBoxButtons.OK, MessageBoxIcon.Error);

               }

               else

               {

                   if (!check\_username\_exist(passedFunction.Username.Text))

                   {

                       CreateLogs.createlogobj.userlog(passedFunction.Username.Text, "Username Doesnt Exists", passedFunction.Name);

                       MessageBox.Show("Username Doesn't Exists, Kindly Register now", "LOGIN", MessageBoxButtons.OK, MessageBoxIcon.Warning);

                       REGISTER register = new REGISTER();

                       register.Show();

                       passedFunction.Close(); //Close Form1,the current open form.

                   }

                   else

                   {

                       CreateLogs.createlogobj.userlog(passedFunction.Username.Text, UserData.role + " Wrong Credentials", passedFunction.Name);

                       MessageBox.Show("Wrong Credentials", "LOGIN", MessageBoxButtons.OK, MessageBoxIcon.Warning);

                   }

               }

           }

           catch (Exception ex)

           {

               CreateLogs.createlogobj.userlog(passedFunction.Username.Text, UserData.role + " login failed", passedFunction.Name);

               MessageBox.Show("Login Failed", ex.Message);

           }

       }

       public void complaint(COMPLAINT passedFunction)

       {

           clear();

           checkconn();

           try

           {

               String typeofcomplain = passedFunction.ComboBox1.SelectedItem.ToString();

               String state = passedFunction.comboBox2.SelectedItem.ToString();

               SQLitecmd.CommandText = "insert into complaint(aadharno,typeofcomplain,mobilenumber,aboutcomplain,address,state,city,username) VALUES('" + passedFunction.Aadhar.Text + "','" + typeofcomplain + "','" + passedFunction.Contact.Text + "','" + passedFunction.Dcomplaint.Text + "','" + passedFunction.Address.Text + "','" + state + "','" + passedFunction.city.Text + "','" + UserData.username + "')";

               try

               {

                   CreateLogs.createlogobj.userlog(UserData.username, "Complain filled", passedFunction.Name);

                   SQLitecmd.ExecuteNonQuery();

                   MessageBox.Show("complain Filled, Checkout in status section", "Complain", MessageBoxButtons.OK);

                   STATUS status = new STATUS();

                   status.Show();

                   passedFunction.Close(); //Close Form1,the current open form.

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.userlog(UserData.username, "Unable to file complain: " + ex.Message, passedFunction.Name);

                   MessageBox.Show("Unable to file complain: " + ex.Message + "", "Complain", MessageBoxButtons.OK, MessageBoxIcon.Error);

               }

           }

           catch (Exception ex)

           {

               CreateLogs.createlogobj.userlog(UserData.username, "Unable to file complain: " + ex.Message, passedFunction.Name);

               MessageBox.Show("Unable to File Complain", ex.Message);

               Console.WriteLine(ex);

           }

       }

       public void request(REQUEST passedFunction)

       {

           clear();

           checkconn();

           try

           {

               String typeofcomplain = passedFunction.ComboBox1.SelectedItem.ToString();

               String state = passedFunction.comboBox2.SelectedItem.ToString();

               SQLitecmd.CommandText = "insert into request(aadharno,typeofrequest,mobilenumber,aboutrequest,address,state,city,username) VALUES('" + passedFunction.Aadhar.Text + "','" + typeofcomplain + "','" + passedFunction.Contact.Text + "','" + passedFunction.DREQUEST.Text + "','" + passedFunction.Address.Text + "','" + state + "','" + passedFunction.city.Text + "','" + UserData.username + "')";

               try

               {

                   SQLitecmd.ExecuteNonQuery();

                   MessageBox.Show("Request received, Checkout in status section", "Request", MessageBoxButtons.OK);

                   CreateLogs.createlogobj.userlog(UserData.username, "User sent a request", passedFunction.Name);

                   STATUS status = new STATUS();

                   status.Show();

                   passedFunction.Close(); //Close Form1,the current open form.

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.userlog(UserData.username, "Can't Register Your Request: " + ex.Message, passedFunction.Name);

                   MessageBox.Show("Can't Register Your Request: " + ex.Message + "", "Request", MessageBoxButtons.OK, MessageBoxIcon.Error);

               }

               SQLiteConn.Close();

           }

           catch (Exception ex)

           {

               CreateLogs.createlogobj.userlog(UserData.username, "Can't Register Your Request: " + ex.Message, passedFunction.Name);

               MessageBox.Show("Unable to Record your Request: " + ex.Message + "", "Request", MessageBoxButtons.OK, MessageBoxIcon.Error);

               Console.WriteLine(ex);

           }

       }

   }

}

LOGIN.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace helphub

{

   public partial class LOGIN : Form

   {

       public LOGIN()

       {

           InitializeComponent();

           Username.Focus();

           Username.Select();

       }

       private void Button2\_Click(object sender, EventArgs e)

       {

           if (Username.Text.Trim() == "" || Password.Text.Trim() == "")

           {

               MessageBox.Show("Empty Fields", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

           }

           else

           {

               Database.databaseobj.login(this);

           }

       }

       private void label3\_Click(object sender, EventArgs e)

       {

           REGISTER register = new REGISTER();

           register.Show();

           this.Close(); //Close Form1,the current open form.

       }

       private void username\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Password.Focus();

           }

       }

       private void password\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Button2\_Click(null, EventArgs.Empty);

           }

       }

   }

}

REGISTER.cs

using Microsoft.VisualBasic;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Reflection.Metadata;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using System.Data.SQLite;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

using FluentValidation;

namespace helphub

{

   public partial class REGISTER : Form

   {

       public class RegisterUser

       {

           public string Address { get; set; }

           public string Contact { get; set; }

           public string username { get; set; }

           public string Password { get; set; }

           public string Aadhar { get; set; }

           public string Email { get; set; }

       }

       public class RegisterValidator : AbstractValidator<RegisterUser>

       {

           public RegisterValidator()

           {

               RuleFor(RegisterUser => RegisterUser.Email).EmailAddress().NotNull();

               RuleFor(RegisterUser => RegisterUser.Contact).Matches("^[\\+]?[(]?[0-9]{3}[)]?[-\\s\\.]?[0-9]{3}[-\\s\\.]?[0-9]{4,6}$").NotNull();

               RuleFor(RegisterUser => RegisterUser.username).NotNull();

               RuleFor(RegisterUser => RegisterUser.Address).NotNull();

               RuleFor(RegisterUser => RegisterUser.Password).NotNull();

               RuleFor(RegisterUser => RegisterUser.Aadhar).Matches("^[2-9]{1}[0-9]{3}\\s[0-9]{4}\\s[0-9]{4}$").NotNull();

           }

       }

       public REGISTER()

       {

           InitializeComponent();

           username.Select();

       }

       private async void Button2\_Click(object sender, EventArgs e)

       {

           RegisterUser ruser = new RegisterUser();

           RegisterValidator validator = new RegisterValidator();

           ruser.Aadhar = Aadhar.Text;

           ruser.Contact = Contact.Text;

           ruser.Email = Email.Text;

           ruser.Password = Password.Text;

           ruser.username = username.Text;

           ruser.Address = Address.Text;

           var result = validator.Validate(ruser);

           if (!result.IsValid)

           {

               String errors ="Kindly Solve Below Errors\n";

               int i = 1;

               foreach (var failure in result.Errors)

               {

                   errors = ""+ errors + " "+i+") " + failure.ErrorMessage + "\n";

                   i++;

               }

               MessageBox.Show(errors, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

               return;

           }

           if (username.Text.Trim() == "" && Password.Text.Trim() == "" && Aadhar.Text.Trim() == "" && Contact.Text.Trim() == "" && Email.Text.Trim() == "")

           {

               MessageBox.Show("Empty Fields", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

           }

           else

           {

               Database.databaseobj.register(this);

           }

       }

       private void pictureBox2\_Click(object sender, EventArgs e)

       {

           LOGIN login = new LOGIN();

           login.Show();

           this.Close(); //Close Form1,the current open form.

       }

       // change focus

       private void username\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Aadhar.Focus();

           }

       }

       private void aadhar\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Email.Focus();

           }

       }

       private void email\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Contact.Focus();

           }

       }

       private void contact\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Address.Focus();

           }

       }

       private void address\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Password.Focus();

           }

       }

       private void password\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Button2\_Click(null, EventArgs.Empty);

           }

       }

       private void pictureBox1\_Click(object sender, EventArgs e)

       {

       }

   }

}

DASHBOARD.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace helphub

{

   partial class DASHBOARD : Form

   {

       public List<string> StateList = new List<string> {

           "AP|Andhra Pradesh",

           "AR|Arunachal Pradesh",

           "AS|Assam",

           "BR|Bihar",

           "CT|Chhattisgarh",

           "GA|Goa",

           "GJ|Gujarat",

           "HR|Haryana",

           "HP|Himachal Pradesh",

           "JK|Jammu and Kashmir",

           "JH|Jharkhand",

           "KA|Karnataka",

           "KL|Kerala",

           "MP|Madhya Pradesh",

           "MH|Maharashtra",

           "MN|Manipur",

           "ML|Meghalaya",

           "MZ|Mizoram",

           "NL|Nagaland",

           "OR|Odisha",

           "PB|Punjab",

           "RJ|Rajasthan",

           "SK|Sikkim",

           "TN|Tamil Nadu",

           "TG|Telangana",

           "TR|Tripura",

           "UT|Uttarakhand",

           "UP|Uttar Pradesh",

           "WB|West Bengal",

           "AN|Andaman and Nicobar Islands",

           "CH|Chandigarh",

           "DN|Dadra and Nagar Haveli",

           "DD|Daman and Diu",

           "DL|Delhi",

           "LD|Lakshadweep",

           "PY|Puducherry"

       };

       public DASHBOARD()

       {

           InitializeComponent();

           if (UserData.role == "ADMIN" || UserData.role == "SUPERVISOR" || StateList.Contains(UserData.role) || UserData.role == "SUPERADMIN" || UserData.role == "root")

           {

               this.pictureBox3.Visible = true;

           }

           else

           {

               this.pictureBox3.Visible = false;

           }

       }

       private void pictureBox1\_Click(object sender, EventArgs e)

       {

           LOGIN login = new LOGIN();

           login.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private void Button1\_Click(object sender, EventArgs e)

       {

           COMPLAINT complaint = new COMPLAINT();

           complaint.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private void Button2\_Click(object sender, EventArgs e)

       {

           REQUEST request = new REQUEST();

           request.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private void Button3\_Click(object sender, EventArgs e)

       {

           STATUS status = new STATUS();

           status.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private void pictureBox3\_Click(object sender, EventArgs e)

       {

           if(UserData.role == "ADMIN" || UserData.role == "SUPERVISOR" || StateList.Contains(UserData.role) || UserData.role == "SUPERADMIN" || UserData.role == "root")

           {

               ADMIN admin = new ADMIN();

               admin.Show();

               this.Hide(); //Close Form1,the current open form.

           }

       }

       private void pictureBox4\_Click(object sender, EventArgs e)

       {

           ABOUT about = new ABOUT();

           about.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private void pictureBox5\_Click(object sender, EventArgs e)

       {

           PROFILE profile = new PROFILE();

           profile.Show();

           this.Hide();

       }

   }

}

COMPLAINT.cs

using FluentValidation;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static helphub.COMPLAINT;

using static helphub.REQUEST;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace helphub

{

   public partial class COMPLAINT : Form

   {

       public class Complaint

       {

           public string DCOMPLAIN { get; set; }

           public string Address { get; set; }

           public string Contact { get; set; }

           public string City { get; set; }

       }

       public class ComplaintValidator : AbstractValidator<Complaint>

       {

           public ComplaintValidator()

           {

               RuleFor(Complaint => Complaint.DCOMPLAIN).NotNull().WithMessage("Kindly Provide Proper Details about Complain");

               RuleFor(RegisterUser => RegisterUser.Contact).NotNull().Matches("^[\\+]?[(]?[0-9]{3}[)]?[-\\s\\.]?[0-9]{3}[-\\s\\.]?[0-9]{4,6}$");

               RuleFor(Complaint => Complaint.Address).NotNull();

               RuleFor(Complaint => Complaint.City).NotNull();

           }

       }

       public COMPLAINT()

       {

           InitializeComponent();

           Aadhar.Text = UserData.aadharno;

           Contact.Text = UserData.mobilenumber;

           ComboBox1.SelectedItem = "DOMESTIC VOILENCE";

           comboBox2.SelectedItem = "AP|Andhra Pradesh";

           ComboBox1.Select();

           Address.Text = UserData.address;

       }

       private void pictureBox2\_Click(object sender, EventArgs e)

       {

           LOGIN login = new LOGIN();

           login.Show();

           this.Close(); //Close Form1,the current open form.

       }

       private void Button2\_Click(object sender, EventArgs e)

       {

           Complaint complain = new Complaint();

           ComplaintValidator validator = new ComplaintValidator();

           complain.Address = Address.Text;

           complain.Contact = Contact.Text;

           complain.DCOMPLAIN = Dcomplaint.Text;

           complain.City = city.Text;

           var result = validator.Validate(complain);

           if (!result.IsValid)

           {

               String errors = "Kindly Solve Below Errors\n";

               int i = 1;

               foreach (var failure in result.Errors)

               {

                   errors = "" + errors + " " + i + ") " + failure.ErrorMessage + "\n";

                   i++;

               }

               MessageBox.Show(errors, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

               return;

           }

           if (Dcomplaint.Text.Trim() == "" && Address.Text.Trim() == "" && Aadhar.Text.Trim() == "" && Contact.Text.Trim() == "" && ComboBox1.SelectedItem.ToString() == "")

           {

               MessageBox.Show("Empty Fields", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

           }

           else

           {

               Database.databaseobj.complaint(this);

           }

       }

       private void pictureBox3\_Click(object sender, EventArgs e)

       {

           DASHBOARD dashboard = new DASHBOARD();

           dashboard.Show();

           this.Close(); //Close Form1,the current open form.

       }

       private void Label4\_Click(object sender, EventArgs e)

       {

       }

       private void combobox\_KeyDown(object sender, KeyEventArgs e)

       {

           if (e.KeyCode == Keys.Enter)

           {

               Dcomplaint.Focus();

           }

       }

   }

}

REQUEST.cs

using FluentValidation;

using Microsoft.VisualBasic.ApplicationServices;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static helphub.COMPLAINT;

using static helphub.REGISTER;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace helphub

{

   public partial class REQUEST : Form

   {

       public class Request

       {

           public string DREQUEST { get; set; }

           public string Address { get; set; }

           public string Contact { get; set; }

           public string City { get; set; }

       }

       public class RequestValidator : AbstractValidator<Request>

       {

           public RequestValidator()

           {

               RuleFor(Request => Request.DREQUEST).NotNull().WithMessage("Kindly Provide Proper Details about Request");

               RuleFor(Request => Request.Address).NotNull();

               RuleFor(Request => Request.Contact).NotNull().Matches("^[\\+]?[(]?[0-9]{3}[)]?[-\\s\\.]?[0-9]{3}[-\\s\\.]?[0-9]{4,6}$");

               RuleFor(Complaint => Complaint.City).NotNull();

           }

       }

       public REQUEST()

       {

           InitializeComponent();

           Aadhar.Text = UserData.aadharno;

           Contact.Text = UserData.mobilenumber;

           ComboBox1.SelectedItem = "MEDICAL EMERGENCY";

           comboBox2.SelectedItem = "AP|Andhra Pradesh";

           ComboBox1.Select();

           Address.Text = UserData.address;

       }

       private void Button2\_Click(object sender, EventArgs e)

       {

       }

       private void pictureBox2\_Click(object sender, EventArgs e)

       {

           LOGIN login = new LOGIN();

           login.Show();

           this.Close(); //Close Form1,the current open form.

       }

       private void Button2\_Click\_1(object sender, EventArgs e)

       {

           Request request = new Request();

           RequestValidator validator = new RequestValidator();

           request.Address = Address.Text;

           request.Contact = Contact.Text;

           request.DREQUEST = DREQUEST.Text;

           request.City = city.Text;

           var result = validator.Validate(request);

           if (!result.IsValid)

           {

               String errors = "Kindly Solve Below Errors\n";

               int i = 1;

               foreach (var failure in result.Errors)

               {

                   errors = "" + errors + " " + i + ") Property " + failure.PropertyName + " failed validation. Error was: " + failure.ErrorMessage + "\n";

                   i++;

               }

               MessageBox.Show(errors, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

               return;

           }

           if (DREQUEST.Text.Trim() == "" && Address.Text.Trim() == "" && Aadhar.Text.Trim() == "" && Contact.Text.Trim() == "" && ComboBox1.SelectedItem.ToString() == "")

           {

               MessageBox.Show("Empty Fields", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

           }

           else

           {

               Database.databaseobj.request(this);

           }

       }

       private void pictureBox3\_Click(object sender, EventArgs e)

       {

           DASHBOARD dashboard = new DASHBOARD();

           dashboard.Show();

           this.Close(); //Close Form1,the current open form.

       }

   }

}

STATUS.cs

using iTextSharp.text.pdf;

using iTextSharp.text;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace helphub

{

   public partial class STATUS : Form

   {

       public STATUS()

       {

           InitializeComponent();

           Database.databaseobj.FetchData(complaindataview,requestdataview);

       }

       private void pictureBox3\_Click(object sender, EventArgs e)

       {

           DASHBOARD dashboard = new DASHBOARD();

           dashboard.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private void pictureBox2\_Click(object sender, EventArgs e)

       {

           LOGIN login = new LOGIN();

           login.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private void button2\_Click(object sender, EventArgs e)

       {

           DownloadPdf.downloadpdf("COMPLAINS REPORT",complaindataview);

       }

       private void button3\_Click(object sender, EventArgs e)

       {

           DownloadPdf.downloadpdf("REQUESTS REPORT", requestdataview);

       }

   }

}

PROFILE.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace helphub

{

   public partial class PROFILE : Form

   {

       public PROFILE()

       {

           InitializeComponent();

           username.Text = UserData.username;

           email.Text = UserData.email;

           mobilenumber.Text = UserData.mobilenumber;

           aadharnumber.Text = UserData.aadharno;

           address.Text = UserData.address;

       }

       private void button1\_Click(object sender, EventArgs e)

       {

           UPDATEDETAILS updatedetails = new UPDATEDETAILS();

           updatedetails.Show();

           this.Close();       }

       private void button2\_Click(object sender, EventArgs e)

       {

           int x = this.Left + (this.Width / 2) - 200;

           int y = this.Top + (this.Height / 2) - 100;

           string confirm = Microsoft.VisualBasic.Interaction.InputBox("Type 'confirm' to delete account", "DELETE ACCOUNT", "", x, y);

           if (confirm == "" || confirm != "confirm")

           {

               CreateLogs.createlogobj.userlog(username.Text, "Wrong Confirm spelling", this.Name);

               MessageBox.Show("Wrong Input");

               return;

           }

           if (confirm == "confirm")

           {

               if (Database.databaseobj.deleteaccount(username.Text, "PROFILE") == 0)

               {

                   return;

               }

               else

               {

                   this.Close();

               }

           }

       }

       private void pictureBox2\_Click(object sender, EventArgs e)

       {

           DASHBOARD dashboard = new DASHBOARD();

           dashboard.Show();

           this.Close();

       }

   }

}

UPDATEDETAILS.cs

using FluentValidation;

using Microsoft.VisualBasic.ApplicationServices;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static helphub.REGISTER;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace helphub

{

   public partial class UPDATEDETAILS : Form

   {

       public class UpdateUser

       {

           public string Address { get; set; }

           public string Contact { get; set; }

           public string username { get; set; }

           public string oldPassword { get; set; }

           public string newPassword { get; set; }

           public string Aadhar { get; set; }

           public string Email { get; set; }

       }

       public class UpdateValidator : AbstractValidator<UpdateUser>

       {

           public UpdateValidator()

           {

               RuleFor(UpdateUser => UpdateUser.Email).EmailAddress().NotNull();

               RuleFor(UpdateUser => UpdateUser.Contact).Matches("^[\\+]?[(]?[0-9]{3}[)]?[-\\s\\.]?[0-9]{3}[-\\s\\.]?[0-9]{4,6}$").NotNull();

               RuleFor(UpdateUser => UpdateUser.username).NotNull();

               RuleFor(UpdateUser => UpdateUser.Address).NotNull();

               RuleFor(UpdateUser => UpdateUser.newPassword).NotNull();

               RuleFor(UpdateUser => UpdateUser.oldPassword).NotNull();

               RuleFor(UpdateUser => UpdateUser.Aadhar).Matches("^[2-9]{1}[0-9]{3}\\s[0-9]{4}\\s[0-9]{4}$").NotNull();

           }

       }

       public UPDATEDETAILS()

       {

           InitializeComponent();

           Aadhar.Text = UserData.aadharno;

           Contact.Text = UserData.mobilenumber;

           Email.Text = UserData.email;

           username.Text = UserData.username;

           Address.Text = UserData.address;

       }

       PROFILE profile = new PROFILE();

       private void Button2\_Click(object sender, EventArgs e)

       {

           UpdateUser ruser = new UpdateUser();

           UpdateValidator validator = new UpdateValidator();

           ruser.Aadhar = Aadhar.Text;

           ruser.Contact = Contact.Text;

           ruser.Email = Email.Text;

           ruser.oldPassword = oldpassword.Text;

           ruser.newPassword = newpassword.Text;

           ruser.username = username.Text;

           ruser.Address = Address.Text;

           var result = validator.Validate(ruser);

           if (!result.IsValid)

           {

               String errors = "Kindly Solve Below Errors\n";

               int i = 1;

               foreach (var failure in result.Errors)

               {

                   errors = "" + errors + " " + i + ") " + failure.ErrorMessage + "\n";

                   i++;

               }

               MessageBox.Show(errors, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

               return;

           }

           else

           {

               string SQLitecnStr = @"Data Source=.\helphub.db";

               SQLiteConnection SQLiteConn = new SQLiteConnection();

               SQLiteCommand SQLitecmd = new SQLiteCommand();

               SQLiteConn.ConnectionString = SQLitecnStr;

               SQLiteConn.Open();

               SQLitecmd.Connection = SQLiteConn;

               SQLitecmd.CommandText = "SELECT \* FROM user WHERE username='" + UserData.username + "' AND password='" + oldpassword.Text + "'";

               SQLiteDataAdapter da = new SQLiteDataAdapter(SQLitecmd);

               DataTable dt = new DataTable();

               da.Fill(dt);

               if (dt.Rows.Count != 1)

               {

                   MessageBox.Show("Wrong Old Password", "Update Deatils", MessageBoxButtons.OK, MessageBoxIcon.Error);

                   return;

               }

               else

               {

                   SQLitecmd.CommandText = "UPDATE user SET aadharno = '" + Aadhar.Text + "',username = '" + username.Text + "',mobilenumber = '" + Contact.Text + "',password = '" + newpassword.Text + "',email = '" + Email.Text + "',address='" + Address.Text + "' WHERE ID = " + UserData.ID + "";

                   try

                   {

                       int status = SQLitecmd.ExecuteNonQuery();

                       if (status == 0)

                       {

                           MessageBox.Show("Update Failed", "Update Deatils", MessageBoxButtons.OK, MessageBoxIcon.Error);

                           profile.Show();

                           this.Hide(); //Close Form1,the current open form.

                       }

                       else

                       {

                           MessageBox.Show("Deatils Updated", "Update Deatils", MessageBoxButtons.OK, MessageBoxIcon.Information);

                           SQLitecmd.CommandText = "SELECT \* FROM user WHERE username='" + UserData.username + "' AND password='" + oldpassword.Text + "'";

                           da = new SQLiteDataAdapter(SQLitecmd);

                           dt = new DataTable();

                           da.Fill(dt);

                           if (dt.Rows.Count == 1)

                           {

                               foreach (DataRow row in dt.Rows)

                               {

                                   UserData.ID = row["ID"].ToString();

                                   UserData.username = row["username"].ToString();

                                   UserData.aadharno = row["aadharno"].ToString();

                                   UserData.mobilenumber = row["mobilenumber"].ToString();

                                   UserData.password = row["password"].ToString();

                                   UserData.email = row["email"].ToString();

                                   UserData.role = row["Role"].ToString();

                                   UserData.address = row["address"].ToString();

                               }

                           }

                           profile.Show();

                           this.Hide(); //Close Form1,the current open form.

                       }

                   }

                   catch (SQLiteException ex)

                   {

                       int code = ex.ErrorCode;

                       if (code == 19)

                       {

                           MessageBox.Show("Already Registered Username/Aadhar Number", "Update Deatils", MessageBoxButtons.OK, MessageBoxIcon.Information);

                       }

                       else

                       {

                           MessageBox.Show("Database Error: Error code:- " + code + ",Error message:- " + ex.Message + "", "Register", MessageBoxButtons.OK, MessageBoxIcon.Error);

                       }

                       profile.Show();

                       this.Hide(); //Close Form1,the current open form.

                   }

                   catch (Exception ex)

                   {

                       MessageBox.Show("Update Failed: " + ex.Message + "", "Update Deatils", MessageBoxButtons.OK, MessageBoxIcon.Error);

                       profile.Show();

                       this.Hide(); //Close Form1,the current open form.

                   }

               }

               SQLiteConn.Close();

           }

       }

       private void pictureBox1\_Click(object sender, EventArgs e)

       {

           profile.Show();

           this.Hide(); //Close Form1,the current open form.

       }

   }

}

ADMIN.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

using System.Windows.Forms;

using System.Drawing;

using System.Diagnostics;

namespace helphub

{

   public partial class ADMIN : Form

   {

       public ADMIN()

       {

           InitializeComponent();

           if (UserData.role != "SUPERADMIN" && UserData.role != "root")

           {

               superadmin.Hide();

               pictureBox4.Hide();

           }

           if (UserData.role != "ADMIN" && UserData.role != "SUPERADMIN" && UserData.role != "root")

           {

               pictureBox5.Hide();

               pictureBox6.Hide();

               pictureBox7.Hide();

               button3.Hide();

               button4.Hide();

               pictureBox1.Width = 682;

               pictureBox1.Height = 736;

               pictureBox1.Location = new Point(-168, -139);

           }

           if (UserData.role != "ADMIN" && UserData.role != "SUPERVISOR" && UserData.role != "SUPERADMIN" && UserData.role != "root")

           {

               label5.Hide();

               comboBox3.Hide();

               label6.Hide();

               comboBox4.Hide();

           }

           ComboBox1.SelectedItem = "ALL REQUEST";

           comboBox2.SelectedItem = "ALL COMPLAINS";

           comboBox3.SelectedItem = "ALL STATE";

           comboBox4.SelectedItem = "ALL STATE";

           fetchData();

           CreateLogs.createlogobj.adminlog(UserData.username, "Checking Admin Panel", this.Name, UserData.role);

       }

       public void fetchData()

       {

           try

           {

               string SQLitecnStr = @"Data Source=./helphub.db";

               SQLiteConnection SQLiteConn = new SQLiteConnection();

               SQLiteCommand SQLitecmd = new SQLiteCommand();

               SQLiteCommand SQLitecmd1 = new SQLiteCommand();

               SQLiteConn.ConnectionString = SQLitecnStr;

               SQLiteConn.Open();

               SQLitecmd.Connection = SQLiteConn;

               SQLitecmd1.Connection = SQLiteConn;

               SQLitecmd.CommandText = "";

               SQLitecmd1.CommandText = "";

               CreateLogs.createlogobj.adminlog(UserData.username, "Fetching Data of complain and request", this.Name, UserData.role);

               if (UserData.role == "ADMIN" || UserData.role == "SUPERVISOR" || UserData.role == "SUPERADMIN" || UserData.role == "root")

               {

                   if (comboBox2.SelectedItem == "ALL COMPLAINS" && comboBox3.SelectedItem == "ALL STATE")

                   {

                       SQLitecmd.CommandText = "SELECT \* FROM complaint";

                   }

                   else if (comboBox2.SelectedItem == "ALL COMPLAINS" && comboBox3.SelectedItem != "ALL STATE")

                   {

                       SQLitecmd.CommandText = "SELECT \* FROM complaint WHERE state = '" + comboBox3.SelectedItem + "'";

                   }

                   else if (comboBox2.SelectedItem != "ALL COMPLAINS" && comboBox3.SelectedItem == "ALL STATE")

                   {

                       SQLitecmd.CommandText = "SELECT \* FROM complaint WHERE typeofcomplain = '" + comboBox2.SelectedItem + "'";

                   }

                   else

                   {

                       SQLitecmd.CommandText = "SELECT \* FROM complaint WHERE typeofcomplain = '" + comboBox2.SelectedItem + "' AND state = '" + comboBox3.SelectedItem + "'";

                   }

                   if (ComboBox1.SelectedItem == "ALL REQUEST" && comboBox4.SelectedItem == "ALL STATE")

                   {

                       SQLitecmd1.CommandText = "SELECT \* FROM request";

                   }

                   else if (ComboBox1.SelectedItem == "ALL REQUEST" && comboBox4.SelectedItem != "ALL STATE")

                   {

                       SQLitecmd1.CommandText = "SELECT \* FROM request WHERE state = '" + comboBox4.SelectedItem + "'";

                   }

                   else if (ComboBox1.SelectedItem != "ALL REQUEST" && comboBox4.SelectedItem == "ALL STATE")

                   {

                       SQLitecmd1.CommandText = "SELECT \* FROM request WHERE typeofrequest = '" + ComboBox1.SelectedItem + "'";

                   }

                   else

                   {

                       SQLitecmd1.CommandText = "SELECT \* FROM request WHERE typeofrequest = '" + ComboBox1.SelectedItem + "' AND state = '" + comboBox4.SelectedItem + "'";

                   }

               }

               else

               {

                   if (comboBox2.SelectedItem == "ALL COMPLAINS")

                   {

                       SQLitecmd.CommandText = "SELECT \* FROM complaint WHERE state = '" + UserData.role + "'";

                   }

                   else

                   {

                       SQLitecmd.CommandText = "SELECT \* FROM complaint WHERE state = '" + UserData.role + "' AND typeofcomplain = '" + comboBox2.SelectedItem + "'";

                   }

                   if (ComboBox1.SelectedItem == "ALL REQUEST")

                   {

                       SQLitecmd1.CommandText = "SELECT \* FROM request WHERE state = '" + UserData.role + "'";

                   }

                   else

                   {

                       SQLitecmd1.CommandText = "SELECT \* FROM request WHERE state = '" + UserData.role + "' AND typeofrequest = '" + ComboBox1.SelectedItem + "'";

                   }

               }

               // data adapter

               SQLiteDataAdapter da = new SQLiteDataAdapter(SQLitecmd);

               // data table

               DataTable dt = new DataTable();

               // filling data in datatable with adapter

               da.Fill(dt);

               complaindataview.DataSource = dt;

               SQLiteDataAdapter da1 = new SQLiteDataAdapter(SQLitecmd1);

               DataTable dt1 = new DataTable();

               da1.Fill(dt1);

               requestdataview.DataSource = dt1;

               SQLiteConn.Close();

           }

           catch (Exception ex)

           {

               MessageBox.Show("Unable to Load Data", ex.Message);

               Console.WriteLine(ex);

           }

       }

       private void pictureBox3\_Click\_1(object sender, EventArgs e)

       {

           DASHBOARD dashboard = new DASHBOARD();

           dashboard.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private void pictureBox2\_Click\_1(object sender, EventArgs e)

       {

           LOGIN login = new LOGIN();

           login.Show();

           this.Hide(); //Close Form1,the current open form.

       }

       private Boolean check\_complaint\_exist(string ID)

       {

           string SQLitecnStr = @"Data Source=./helphub.db";

           SQLiteConnection SQLiteConn = new SQLiteConnection();

           SQLiteCommand SQLitecmd = new SQLiteCommand();

           SQLiteConn.ConnectionString = SQLitecnStr;

           SQLiteConn.Open();

           SQLitecmd.Connection = SQLiteConn;

           SQLitecmd.CommandText = "SELECT \* FROM complaint WHERE ID=" + ID + "";

           SQLiteDataAdapter da = new SQLiteDataAdapter(SQLitecmd);

           DataTable dt = new DataTable();

           da.Fill(dt);

           if (dt.Rows.Count == 1)

           {

               return true;

           }

           return false;

       }

       private void button2\_Click(object sender, EventArgs e)

       {

           int x = this.Left + (this.Width / 2) - 200;

           int y = this.Top + (this.Height / 2) - 100;

           string ID = "";

           ID = Microsoft.VisualBasic.Interaction.InputBox("Enter ID of Complaint", "Update Complaint status", "", x, y);

           if (ID == "")

           {

               MessageBox.Show("ID field is empty/Updating status aborted");

               return;

           }

           if (check\_complaint\_exist(ID))

           {

               string Status = Microsoft.VisualBasic.Interaction.InputBox("Enter status of Complaint", "Update Complaint status", "", x, y);

               if (Status == "")

               {

                   MessageBox.Show("Status field is empty/Updating status aborted");

                   return;

               }

               try

               {

                   string SQLitecnStr = @"Data Source=./helphub.db";

                   SQLiteConnection SQLiteConn = new SQLiteConnection();

                   SQLiteCommand SQLitecmd = new SQLiteCommand();

                   SQLiteConn.ConnectionString = SQLitecnStr;

                   SQLiteConn.Open();

                   SQLitecmd.Connection = SQLiteConn;

                   int tempid = int.Parse(ID);

                   SQLitecmd.CommandText = "UPDATE complaint SET status = '" + Status + "' WHERE ID = " + tempid + ";";

                   try

                   {

                       SQLitecmd.ExecuteNonQuery();

                       MessageBox.Show("Status Updated Succesfully");

                       CreateLogs.createlogobj.adminlog(UserData.username, "Complain Id:- " + ID + " Status updated to " + Status + "", this.Name, UserData.role);

                       fetchData();

                       this.Refresh();

                   }

                   catch (Exception ex)

                   {

                       MessageBox.Show("Update Failed", ex.Message);

                       CreateLogs.createlogobj.adminlog(UserData.username, "Complain Id:- " + ID + " Status update failed (updated status:- " + Status + ") - " + ex.Message + "", this.Name, UserData.role);

                   }

                   SQLiteConn.Close();

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.adminlog(UserData.username, "Complain Id:- " + ID + " Status update failed (updated status:- " + Status + ") - " + ex.Message + "", this.Name, UserData.role);

                   MessageBox.Show("Update Failed", ex.Message);

                   Console.WriteLine(ex);

               }

           }

           else

           {

               CreateLogs.createlogobj.adminlog(UserData.username, "Id doesn't exists - failed attempt to update complain status", this.Name, UserData.role);

               MessageBox.Show("ID doesn't Exists");

           }

       }

       private Boolean check\_request\_exist(string ID)

       {

           string SQLitecnStr = @"Data Source=./helphub.db";

           SQLiteConnection SQLiteConn = new SQLiteConnection();

           SQLiteCommand SQLitecmd = new SQLiteCommand();

           SQLiteConn.ConnectionString = SQLitecnStr;

           SQLiteConn.Open();

           SQLitecmd.Connection = SQLiteConn;

           SQLitecmd.CommandText = "SELECT \* FROM request WHERE ID=" + ID + "";

           SQLiteDataAdapter da = new SQLiteDataAdapter(SQLitecmd);

           DataTable dt = new DataTable();

           da.Fill(dt);

           if (dt.Rows.Count == 1)

           {

               return true;

           }

           return false;

       }

       private void button1\_Click(object sender, EventArgs e)

       {

           int x = this.Left + (this.Width / 2) - 200;

           int y = this.Top + (this.Height / 2) - 100;

           string ID = Microsoft.VisualBasic.Interaction.InputBox("Enter Id of Request", "Update Request Status", "", x, y);

           if (ID == "")

           {

               MessageBox.Show("ID field is empty/Updating status aborted");

               return;

           }

           if (check\_request\_exist(ID))

           {

               string Status = Microsoft.VisualBasic.Interaction.InputBox("Enter status of Complaint", "Update Complaint status", "", x, y);

               if (Status == "")

               {

                   MessageBox.Show("Status field is empty/Updating status aborted");

                   return;

               }

               try

               {

                   string SQLitecnStr = @"Data Source=./helphub.db";

                   SQLiteConnection SQLiteConn = new SQLiteConnection();

                   SQLiteCommand SQLitecmd = new SQLiteCommand();

                   SQLiteConn.ConnectionString = SQLitecnStr;

                   SQLiteConn.Open();

                   SQLitecmd.Connection = SQLiteConn;

                   int tempid = int.Parse(ID);

                   SQLitecmd.CommandText = "UPDATE request SET status = '" + Status + "' WHERE ID = " + tempid + ";";

                   try

                   {

                       SQLitecmd.ExecuteNonQuery();

                       MessageBox.Show("Status Updated Succesfully");

                       CreateLogs.createlogobj.adminlog(UserData.username, "Request Id:- " + ID + " Status updated to " + Status + "", this.Name, UserData.role);

                       fetchData();

                       this.Refresh();

                   }

                   catch (Exception ex)

                   {

                       MessageBox.Show("Update Failed", ex.Message);

                       CreateLogs.createlogobj.adminlog(UserData.username, "Request Id:- " + ID + " Status update failed (updated status:- " + Status + ") - " + ex.Message + "", this.Name, UserData.role);

                   }

                   SQLiteConn.Close();

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.adminlog(UserData.username, "Request Id:- " + ID + " Status update failed (updated status:- " + Status + ") - " + ex.Message + "", this.Name, UserData.role);

                   MessageBox.Show("Update Failed", ex.Message);

                   Console.WriteLine(ex);

               }

           }

           else

           {

               CreateLogs.createlogobj.adminlog(UserData.username, "Id doesn't exists - failed attempt to update request status", this.Name, UserData.role);

               MessageBox.Show("ID doesn't Exists");

           }

       }

       private Boolean check\_username\_exist(string username)

       {

           string SQLitecnStr = @"Data Source=./helphub.db";

           SQLiteConnection SQLiteConn = new SQLiteConnection();

           SQLiteCommand SQLitecmd = new SQLiteCommand();

           SQLiteConn.ConnectionString = SQLitecnStr;

           SQLiteConn.Open();

           SQLitecmd.Connection = SQLiteConn;

           SQLitecmd.CommandText = "SELECT \* FROM user WHERE username='" + username + "'";

           SQLiteDataAdapter da = new SQLiteDataAdapter(SQLitecmd);

           DataTable dt = new DataTable();

           da.Fill(dt);

           if (dt.Rows.Count == 1)

           {

               if ((dt.Rows[0]["role"].ToString() == "SUPERADMIN" || dt.Rows[0]["role"].ToString() == "SUPERVISOR" || dt.Rows[0]["role"].ToString() == "root" || dt.Rows[0]["role"].ToString() == "ADMIN" || Database.databaseobj.StateList.Contains(dt.Rows[0]["role"].ToString())) && UserData.role == "ADMIN")

               {

                   MessageBox.Show("You can't change details of a " + dt.Rows[0]["role"].ToString() + "");

                   CreateLogs.createlogobj.adminlog(UserData.username, "You can't change details of a " + dt.Rows[0]["role"].ToString() + "", this.Name, UserData.role);

                   throw new Exception("You can't change details of a " + dt.Rows[0]["role"].ToString() + "");

               }

               else

               {

                   return true;

               }

           }

           return false;

       }

       private void pictureBox4\_Click(object sender, EventArgs e)

       {

           int x = this.Left + (this.Width / 2) - 200;

           int y = this.Top + (this.Height / 2) - 100;

           string username = Microsoft.VisualBasic.Interaction.InputBox("Enter username of user to make admin", "Update user role", "", x, y);

           if (username != "")

           {

               try

               {

                   if (check\_username\_exist(username))

                   {

                       string SQLitecnStr = @"Data Source=./helphub.db";

                       SQLiteConnection SQLiteConn = new SQLiteConnection();

                       SQLiteCommand SQLitecmd = new SQLiteCommand();

                       SQLiteConn.ConnectionString = SQLitecnStr;

                       SQLiteConn.Open();

                       SQLitecmd.Connection = SQLiteConn;

                       SQLitecmd.CommandText = "UPDATE user SET role = 'ADMIN' WHERE username = '" + username + "';";

                       try

                       {

                           SQLitecmd.ExecuteNonQuery();

                           CreateLogs.createlogobj.adminlog(UserData.username, "New admin added :- " + username + " by  " + UserData.username + "", this.Name, UserData.role);

                           MessageBox.Show("New admin added Succesfully");

                           fetchData();

                           this.Refresh();

                       }

                       catch (Exception ex)

                       {

                           CreateLogs.createlogobj.adminlog(UserData.username, "Can't Add New Admin " + ex.Message, this.Name, UserData.role);

                           MessageBox.Show("Can't Add New Admin", ex.Message);

                       }

                       SQLiteConn.Close();

                   }

                   else

                   {

                       CreateLogs.createlogobj.adminlog(UserData.username, "Username Doesnt Exists - adding new admin", this.Name, UserData.role);

                       MessageBox.Show("Username Doesn't Exists");

                   }

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.adminlog(UserData.username, "Can't Add New Admin " + ex.Message, this.Name, UserData.role);

                   MessageBox.Show("Can't Add New Admin", ex.Message);

               }

           }

           else

           {

               MessageBox.Show("Username field is empty");

           }

       }

       private void pictureBox5\_Click(object sender, EventArgs e)

       {

           int x = this.Left + (this.Width / 2) - 200;

           int y = this.Top + (this.Height / 2) - 100;

           string username = Microsoft.VisualBasic.Interaction.InputBox("Enter username of user to make supervisor", "Update user role", "", x, y);

           if (username != "")

           {

               try

               {

                   if (check\_username\_exist(username))

                   {

                       string SQLitecnStr = @"Data Source=./helphub.db";

                       SQLiteConnection SQLiteConn = new SQLiteConnection();

                       SQLiteCommand SQLitecmd = new SQLiteCommand();

                       SQLiteConn.ConnectionString = SQLitecnStr;

                       SQLiteConn.Open();

                       SQLitecmd.Connection = SQLiteConn;

                       SQLitecmd.CommandText = "UPDATE user SET role = 'SUPERVISOR' WHERE username = '" + username + "';";

                       try

                       {

                           SQLitecmd.ExecuteNonQuery();

                           MessageBox.Show("New Supervisor added Succesfully");

                           CreateLogs.createlogobj.adminlog(UserData.username, "New supervisor added :- " + username + " by  " + UserData.username + "", this.Name, UserData.role);

                           fetchData();

                           this.Refresh();

                       }

                       catch (Exception ex)

                       {

                           CreateLogs.createlogobj.adminlog(UserData.username, "Can't Add New Supervisor " + ex.Message, this.Name, UserData.role);

                           MessageBox.Show("Can't Add New Supervisor", ex.Message);

                       }

                       SQLiteConn.Close();

                   }

                   else

                   {

                       CreateLogs.createlogobj.adminlog(UserData.username, "Username Doesnt Exists - adding new supervisor", this.Name, UserData.role);

                       MessageBox.Show("Username Doesn't Exists");

                   }

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.adminlog(UserData.username, "Can't Add New Supervisor " + ex.Message, this.Name, UserData.role);

                   MessageBox.Show("Can't Add New Supervisor", ex.Message);

               }

           }

           else

           {

               MessageBox.Show("Username field is empty");

           }

       }

       private void button3\_Click(object sender, EventArgs e)

       {

           int x = this.Left + (this.Width / 2) - 200;

           int y = this.Top + (this.Height / 2) - 100;

           string ID = Microsoft.VisualBasic.Interaction.InputBox("Enter Id of Complain", "Delete Complain", "", x, y);

           if (ID == "")

           {

               MessageBox.Show("ID field is empty/Deleting complain aborted");

               return;

           }

           if (check\_complaint\_exist(ID))

           {

               try

               {

                   string SQLitecnStr = @"Data Source=./helphub.db";

                   SQLiteConnection SQLiteConn = new SQLiteConnection();

                   SQLiteCommand SQLitecmd = new SQLiteCommand();

                   SQLiteConn.ConnectionString = SQLitecnStr;

                   SQLiteConn.Open();

                   SQLitecmd.Connection = SQLiteConn;

                   int tempid = int.Parse(ID);

                   SQLitecmd.CommandText = "DELETE FROM complaint WHERE ID = " + tempid + ";";

                   try

                   {

                       SQLitecmd.ExecuteNonQuery();

                       MessageBox.Show("Complaint Deleted");

                       CreateLogs.createlogobj.adminlog(UserData.username, "Complain Id:- " + ID + " Deleted", this.Name, UserData.role);

                       fetchData();

                       this.Refresh();

                   }

                   catch (Exception ex)

                   {

                       CreateLogs.createlogobj.adminlog(UserData.username, "Complain Id:- " + ID + " Process failed " + ex.Message, this.Name, UserData.role);

                       MessageBox.Show("Process Failed", ex.Message);

                   }

                   SQLiteConn.Close();

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.adminlog(UserData.username, "Complain Id:- " + ID + " Process Failed " + ex.Message, this.Name, UserData.role);

                   MessageBox.Show("Process Failed", ex.Message);

               }

           }

           else

           {

               CreateLogs.createlogobj.adminlog(UserData.username, "Complain Id:- " + ID + " Doesnt Exists", this.Name, UserData.role);

               MessageBox.Show("ID doesn't Exists");

           }

       }

       private void pictureBox6\_Click(object sender, EventArgs e)

       {

           AddStateAdmin admin = new AddStateAdmin();

           admin.Show();

       }

       private void button4\_Click(object sender, EventArgs e)

       {

           int x = this.Left + (this.Width / 2) - 200;

           int y = this.Top + (this.Height / 2) - 100;

           string ID = Microsoft.VisualBasic.Interaction.InputBox("Enter Id of Request", "Delete Request", "", x, y);

           if (ID == "")

           {

               MessageBox.Show("ID field is empty/Deleting request aborted");

               return;

           }

           if (check\_complaint\_exist(ID))

           {

               try

               {

                   string SQLitecnStr = @"Data Source=./helphub.db";

                   SQLiteConnection SQLiteConn = new SQLiteConnection();

                   SQLiteCommand SQLitecmd = new SQLiteCommand();

                   SQLiteConn.ConnectionString = SQLitecnStr;

                   SQLiteConn.Open();

                   SQLitecmd.Connection = SQLiteConn;

                   int tempid = int.Parse(ID);

                   SQLitecmd.CommandText = "DELETE FROM request WHERE ID = " + tempid + ";";

                   try

                   {

                       SQLitecmd.ExecuteNonQuery();

                       MessageBox.Show("Request Deleted");

                       CreateLogs.createlogobj.adminlog(UserData.username, "Request Id:- " + ID + " Deleted", this.Name, UserData.role);

                       fetchData();

                       this.Refresh();

                   }

                   catch (Exception ex)

                   {

                       CreateLogs.createlogobj.adminlog(UserData.username, "Request Id:- " + ID + " Deleteion failed " + ex.Message, this.Name, UserData.role);

                       MessageBox.Show("Process Failed", ex.Message);

                   }

                   SQLiteConn.Close();

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.adminlog(UserData.username, "Request Id:- " + ID + " Deleteion failed " + ex.Message, this.Name, UserData.role);

                   MessageBox.Show("Process Failed", ex.Message);

               }

           }

           else

           {

               CreateLogs.createlogobj.adminlog(UserData.username, "Request Id:- " + ID + "Doesnt Exists", this.Name, UserData.role);

               MessageBox.Show("ID doesn't Exists");

           }

       }

       private void ComboBox1\_SelectedIndexChanged(object sender, EventArgs e)

       {

           fetchData();

           this.Refresh();

       }

       private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

       {

           fetchData();

           this.Refresh();

       }

       private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

       {

           fetchData();

           this.Refresh();

       }

       private void comboBox4\_SelectedIndexChanged(object sender, EventArgs e)

       {

           fetchData();

           this.Refresh();

       }

       private void label2\_Click(object sender, EventArgs e)

       {

       }

       private void pictureBox7\_Click(object sender, EventArgs e)

       {

           int x = this.Left + (this.Width / 2) - 200;

           int y = this.Top + (this.Height / 2) - 100;

           string username = Microsoft.VisualBasic.Interaction.InputBox("Enter username of user to BAN/UNBAN", "BAN/UNBAN USER", "", x, y);

           if (username != "")

           {

               try

               {

                   if (check\_username\_exist(username))

                   {

                       string option = Microsoft.VisualBasic.Interaction.InputBox("Want to ban user or unban?\nType 'ban' to Ban a user\nType 'unban' to UnBan a user\nNote:- Type in lowercase only(only users can be banned from admin panel)", "BAN/UNBAN USER", "", x, y);

                       if (option != "")

                       {

                           string banorunban = option == "ban" ? "YES" : "NO";

                           if (username != "")

                           {

                               string SQLitecnStr = @"Data Source=./helphub.db";

                               SQLiteConnection SQLiteConn = new SQLiteConnection();

                               SQLiteCommand SQLitecmd = new SQLiteCommand();

                               SQLiteConn.ConnectionString = SQLitecnStr;

                               SQLiteConn.Open();

                               SQLitecmd.Connection = SQLiteConn;

                               SQLitecmd.CommandText = "UPDATE user SET banned = '" + banorunban + "' WHERE username = '" + username + "' AND role='USER';";

                               try

                               {

                                   SQLitecmd.ExecuteNonQuery();

                                   MessageBox.Show("User succesfully " + option + "ned");

                                   CreateLogs.createlogobj.banunbanlog(username, UserData.username, option);

                                   CreateLogs.createlogobj.adminlog(UserData.username, "" + username + " " + option + "ned", this.Name, UserData.role);

                               }

                               catch (Exception ex)

                               {

                                   CreateLogs.createlogobj.adminlog(UserData.username, "Can't " + option + " user " + ex.Message, this.Name, UserData.role);

                                   MessageBox.Show("Can't " + option + " user", ex.Message);

                               }

                               SQLiteConn.Close();

                           }

                       }

                       else

                       {

                           MessageBox.Show("Option field is empty");

                       }

                   }

                   else

                   {

                       MessageBox.Show("Username Doesn't Exists");

                   }

               }

               catch (Exception ex)

               {

                   CreateLogs.createlogobj.adminlog(UserData.username, "Can't  ban/unban  user " + ex.Message, this.Name, UserData.role);

               }

           }

           else

           {

               MessageBox.Show("Username field is empty");

           }

       }

       private void superadmin\_Click(object sender, EventArgs e)

       {

           SUPERADMIN superadmin = new SUPERADMIN();

           superadmin.Show();

           this.Hide();

       }

       private void button6\_Click(object sender, EventArgs e)

       {

           DownloadPdf.downloadpdf("REQUESTS REPORT", complaindataview);

       }

       private void button5\_Click(object sender, EventArgs e)

       {

           DownloadPdf.downloadpdf("COMPLAINS REPORT", complaindataview);

       }

   }

}

**TESTING**

*Testing*

**SYSTEM TESTING**

**TESTING**

Testing goes through the various stages, during testing the program to be tested has to be executed with a set of test cases, and have the output of the program for the test case is evaluated to determine if the program is performing as expected. Due to its approach dynamic testing only ascertains the presence of error in the program. The exact nature of error is not usually decided by testing. Testing form is the first in determining error in the program.

Once the programs are tested individually then the system as a whole needs to be tested. During testing, the system is used experimentally to ensure that the software does not fail i.e. it will run according to its specification. The programs executed to check for any syntax or logical error. The error is corrected and test is made to determine whether the program is doing what it is supposed to do.

**Various types of testing**

**Unit testing**

Each component of the system is tested individually. The programmer does the testing. Testing is restrictive in nature i.e. programmer should try to test all individual conditions and see if the program breaks under any circumstance.

**System testing**

This is an integrated form of testing, which focuses on functionality and interface between units and team in a controlled environment does it.

**Acceptance Testing**

This is system testing done by the user of the application the only emphasis is functionally testing as the user is not aware of the technical aspect of the system. The testing is also done in a controlled environment with logging o all error based on the error found in the system, the user must accept or reject the system.

**Module Testing**

This is an optional form of testing, which is done only for large system, which has a large number of modules.

**Security Testing**

Security testing will be done as a specialized form of testing if there is a high risk exposure in that area. If the risk exposure is not very high, then it can be done as part of the system testing. Typically, security testing would involve trying to break in to the system, trying to execute transactions not allowed to person; to access areas on disk were the user is not allowed.

Testing is vital t the success of the system. If it on. This done successfully, this shows that the parts of the system are working correctly and all the goals are achieved.

**IMPLEMENTATION**

Implementation is used here to mean the process of converting a new or revised system design int operational one; conversion is one aspect of implementation. The other aspect is post implementation review and software and maintenance.

There are three types f implementation:

1. Implementation of a computer system
2. Implementation of a new computer system
3. Implementation of a modified application.

**MAINTAINANCE**

After the system has successfully been implemented maintenance activity may require continuous involvement of the developers. Provision must be made for environmental changes, which may affect either the computer, or other parts of the computer based system: such activity is normally called maintenance. It includes both the improvement of the system functions and the correction of faults that arise during the operation of a system.

Maintenance activity may require the continuing involvement of a large proportion of computer department resources. Maintenance works may arise due to two reasons:

1. Errors that creep up during normal running
2. Request for changes by the service providers. As part of the normal running of the system when errors are found.

This maintenance work will help to ensure that the system works smoothly as predicted in the open environment. Whenever a maintenance work arises, the work has to be properly carried out with proper documentation. This is to avoid any form of changes in the structure of the system.

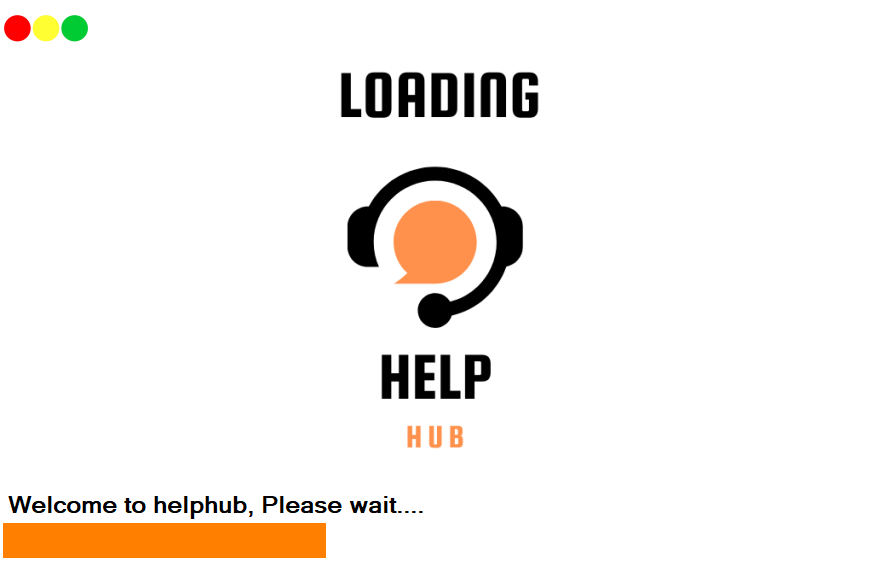
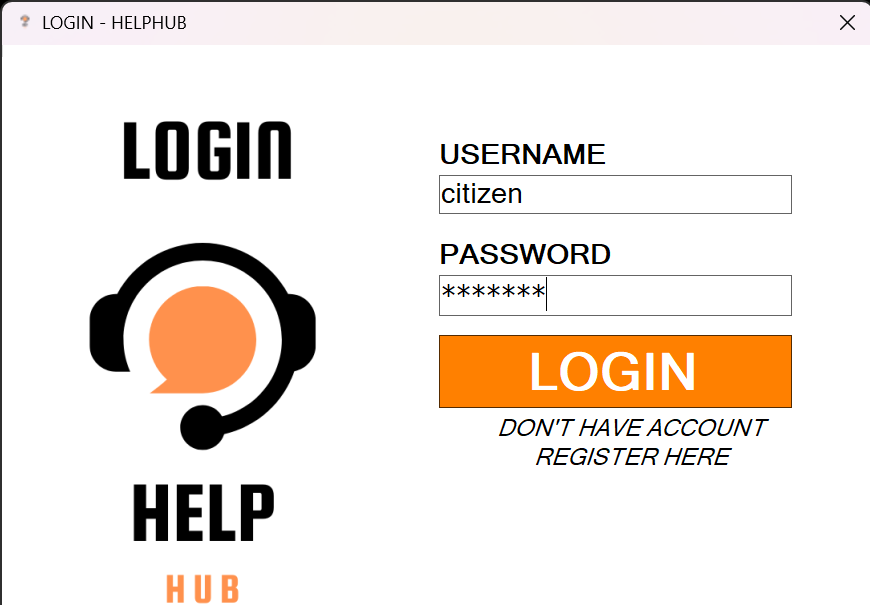
For every maintenance work an amendment notification is to be issued. This notification will have required changes and also authenticated. On the receipt of the amendment notification the amendment

Log is prepared which records these courses of action that has been planned to be taken. It also records the estimated and the actual completion of each activity.

**Sample Test Cases:**

* Invalid Login credentials are tested and appropriate error messages are displayed.
* The data capture part is thoroughly tested for entry of correct data in acceptable type and size.
* Appropriate interfaces are created to link various modules.

*Snapshots*

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, chat or text message

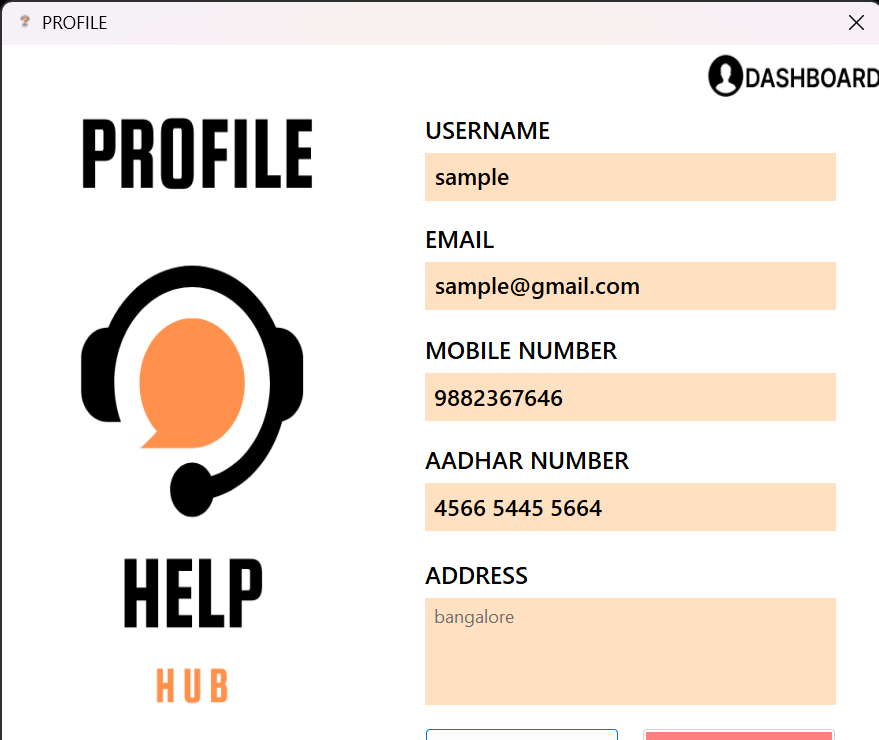
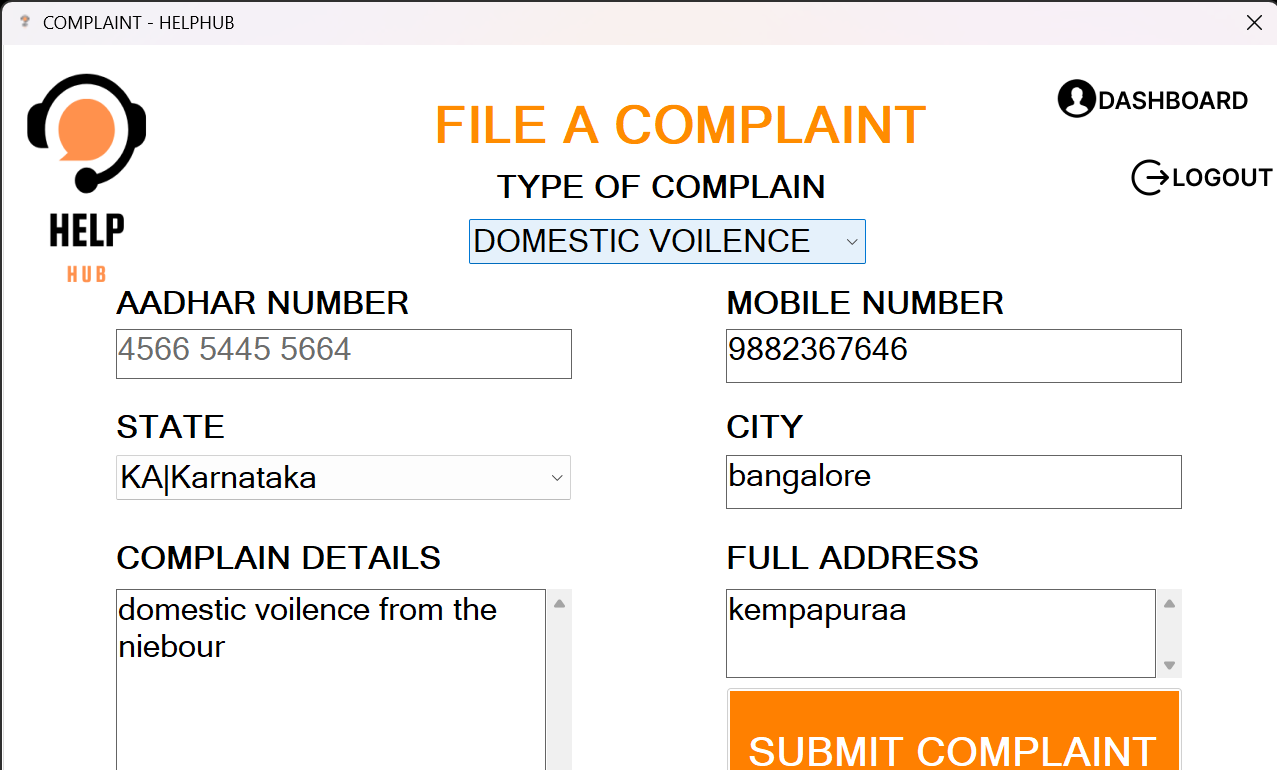
Description automatically generatedGraphical user interface, text, application, chat or text message

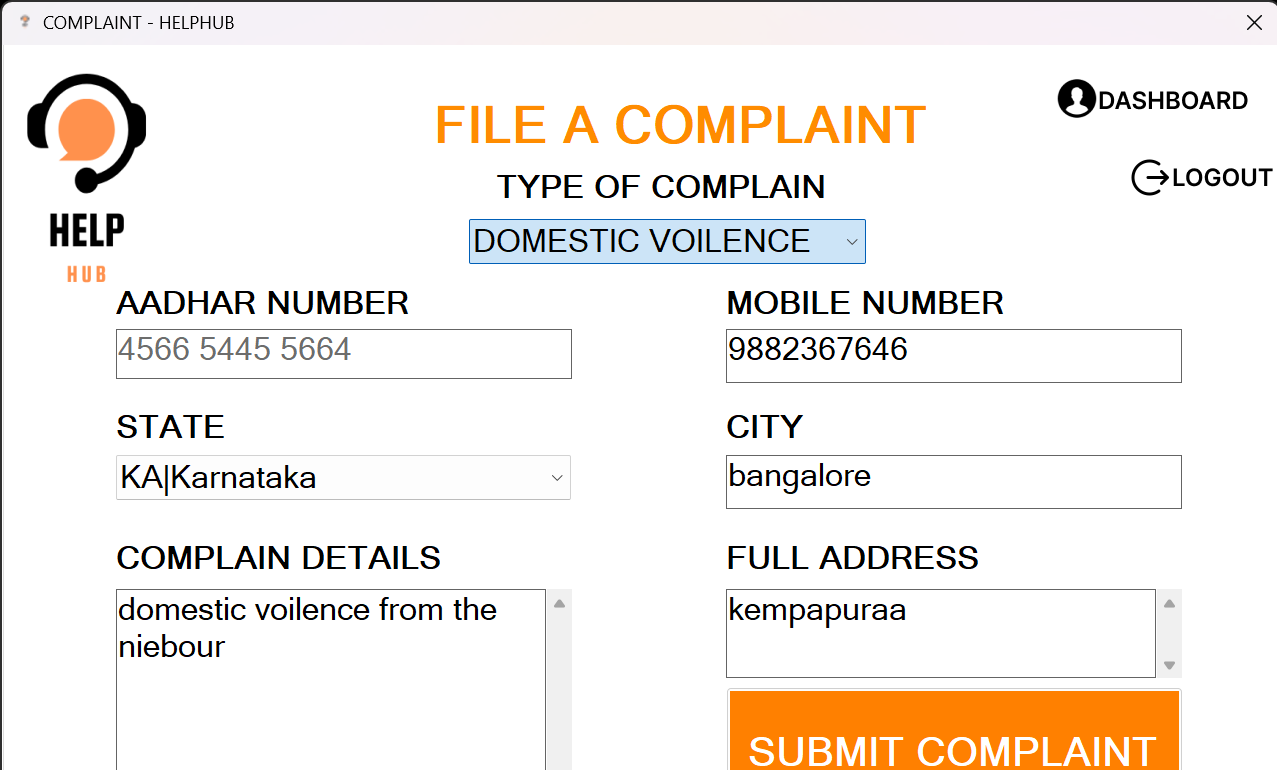
Description automatically generated

A picture containing graphical user interface

Description automatically generated A picture containing text

Description automatically generated

 Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generatedTable

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, application

Description automatically generated

*Future Enhancements*

**FUTURE ENHANCEMENTS**

Here are 10 potential enhancements for the HelpHub project:

* Implement a chatbot feature to automatically respond to complaints and service requests.
* Develop an interactive dashboard to track and monitor the progress of complaints and requests.
* Develop a mobile application to allow users to submit complaints and service requests on-the-go.
* Create a notification system to notify users of new updates and progress on their complaints and service requests.
* Utilize artificial intelligence and machine learning to enhance the accuracy and efficiency of the system.
* Create a system to digitally store and archive all complaints and service requests for future reference.
* Implement a feedback and rating system for users to provide reviews and feedback on the service they received.
* Allow users to submit complaints and service requests in multiple languages.
* Develop an automated system to monitor and analyze trends in complaints and service requests.
* Integrate the system with popular social media platforms to allow users to submit complaints and service requests directly through these platforms.

.

**MERITS**

* The Helphub project allows for easy and efficient management of complaints and requests related to various social issues.
* The use of VB C# for the front-end development allows for a user-friendly and visually appealing interface.
* The back-end implementation on SQLite provides a reliable and high-performing database that can handle large amounts of data.
* The SQLite database engine is widely used and known for its high-reliability and full-featured functionality.

*Conclusion*

# CONCLUSION

Helphub is a powerful and user-friendly application that allows citizens to easily report and track issues in their community. The application is designed to streamline the process of reporting and tracking issues, making it easier for citizens to get the help they need. With multiple levels of administration, Helphub ensures that complaints and requests are addressed in a timely manner, making it a valuable tool for any community. In summary, Helphub is a powerful and innovative desktop application that empowers citizens to report and resolve social issues in their community. With a user-friendly interface, advanced tracking and reporting features, and a dedicated team of professionals, Helphub is the perfect solution for anyone looking to make a positive impact in their community.

*Bibliography*

**Bibliography**

* SQLite

-   D. Richard Hipp

* .NET FRAMEWORK
* Matthew A. Stocker & Steven J. Stei
* <https://www.sqlite.org/>
* Visual basic .NET bible