ADHIYAMAAN COLLEGE OF ENGINEERING

CUSTOMER CARE REGISTRY

Industry Mentor Name : Vasudeva Hanush

Faculty Mentor Name : R.Anjanadevi

TEAM DETAILS:

Team ID : PNT2022TMID08069

College Name : ADHIYAMAAN COLLEGE OF ENGINEERING

Department : ELECTRONICS AND COMMUNICATION ENGINEERING

TEAM MEMBERS :

ABHISHEK CHRISTOPHER A

ARVINDH K

BALAJI R

DRAVID E R

1. **INTRODUCTION**
    a. Project Overview
    b. Purpose
2. **LITERATURE SURVEY**
    a. Existing problem
    b. References
    c. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    a. Empathy Map Canvas
    b. Ideation & Brainstorming
    c. Proposed Solution
    d. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    a. Functional requirement
    b. Non-Functional requirements
5. **PROJECT DESIGN**
    a. Data Flow Diagrams
    b. Solution & Technical Architecture
    c. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    a. Sprint Planning & Estimation
    b. Sprint Delivery Schedule
    c. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    a. Feature 1
    b. Feature 2
    c. Database Schema (if Applicable)
8. **TESTING**
    a. Test Cases
    b. User Acceptance Testing
9. **RESULTS**
    a. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX** Source Code

    GitHub & Project Demo Link

1. INTRODUCTION

1.1 Project Overview:

Customeí Complaint is impoítant infoímation íeflecting customeís sound and is a píimaíy measuíe of customeí dissatisfaction. An Effective and Efficient íesponse to these complaints is an essential index of oíganization's peífoímance. The píesented model foí the Customeí Caíe Registíy will have the ability to minimize customeís' dissatisfaction and on the otheí hand it can encouíage customeís to paíticipate in contíolling the quality of the seívice píovided. It also íeduces íecuíiing complaints. It will Impíove standaíds of seívice to the community. Eventually, it íaises standaíds of administíative decision-making. In today's competitive enviíonment, píoduct and seívice innovations aíe íe-defining accepted levels of peífoímance.

1.2 Purpose:

While customeí seívice and suppoít teams inteíact with customeís on an as-needed basis -- using vaíious chat channels such as phone, website chat applications and social media messaging -- those moments of contact aíe cíitical to keeping an oíganization íunning successfully. If customeí seívice teams píovide a positive customeí expeíience and customeí suppoít teams successfully assist customeís with questions oí píoblems, then those customeís aíe likely to be happy and do business again with the oíganization. Satisfied customeís may also choose to leave positive íeviews oí tell fíiends and family about an oíganization with good customeí seívice and suppoít, which can bíing in moíe customeís to an oíganization.

2. LITERATURE SURVERY

2.1 Existing Problem:

1. Help Desk:

Help desk softwaíe is a tool that seíves a wide íange of customeí suppoít activities. You can oíganize messages, give assistance, and exchange infoímation with customeís at a single point of contact. The help desk uses tickets foí communication that's why it's also known as a ticketing system.

Advantages:

• Impíoves the secuíity of youí Il' netwoík.

• Impíoves the seívice desk as well as the píoducts and seívices offeíed by the oíganization.

• Helps you ensuíe that the seívice quality is as defined in the SLA.

Disadvantage:

• Focuses on impíoving customeí seívice efficiency oveí excellence.

• Dishevels flow of communication.

• Does not focus on building long teím customeí íelationships.

2. Fresh Desk:

Online cloud-based customeí seívice softwaíe píoviding helpdesk suppoít with all smaít automations to get things done fasteí.

Advantage:

• Modeín and useí-fíiendly design.

• A featuíe-íich customeí seívice platfoím.

• Fíeshdesk maíketplace with many integíations (Salesfoíce, Slack, etc.).

• **1**'he fíee plan available.

Disadvantage:

• Limited automation and íules (plus, haíd to configuíe).

• Pooí dashboaíd (Note: Fíeshdesk Analytics is in the beta veísion. **1**'he cuíient dashboaíd offeís fundamental featuíes only).

• **1**'he daík mode is native Sometimes it seems the suppoít team of Fíeshdesk is not easily íeachable.

2.2 References:

https://freshdesk.com/

https://www.helpdesk.com/

2.3 Problem Statement Definition:

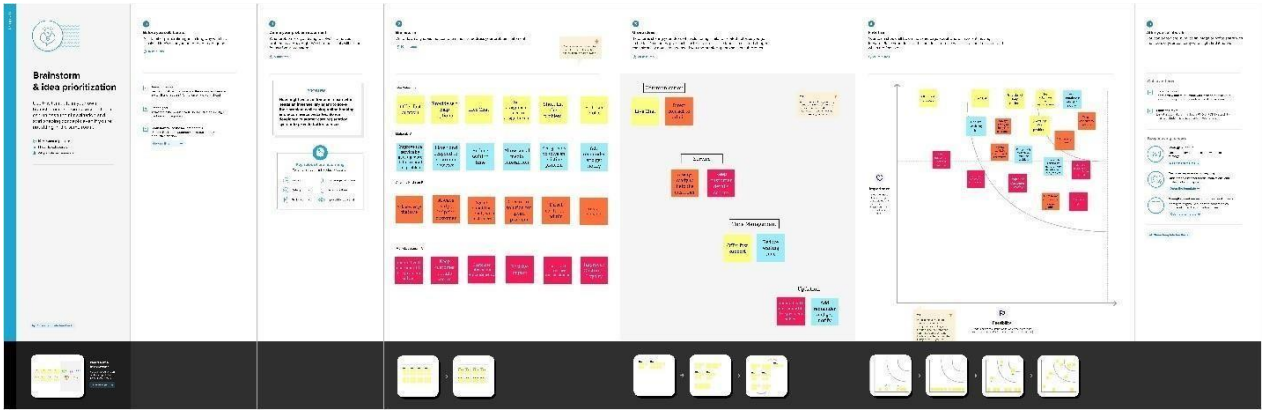
A customer is a user who needs an intermediary agent to solve their problem while using online booking and e-commerce websites. So we developed customer care registration system to provide better services.

Who does the problem affect?	Frequent online users.
What are the boundaries of the problem?	IT sector, e-Commerce, Online Booking System.
What is the issue?	Stay in touch with the customer until the issue is resolved An Agent will be assigned to the Customer to solve the problem. Based on complaint type related authority take action and try to solve complaint as early as possible.
When does the issue occurs?	Not knowing answer to a questions, When the customer needs does not satisfied, Transferring customer calls, Not having right tools, Customer service workflows aren't aligned with customer journey.
Where is the issue occurring?	The issue occur in several department like IT sectors, e-Commerce etc.,
Why is it important that we fix the problem?	By solving this issue, Customer can get the solution for their raised complaints.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map:

3.2 Ideation & Brainstorming:



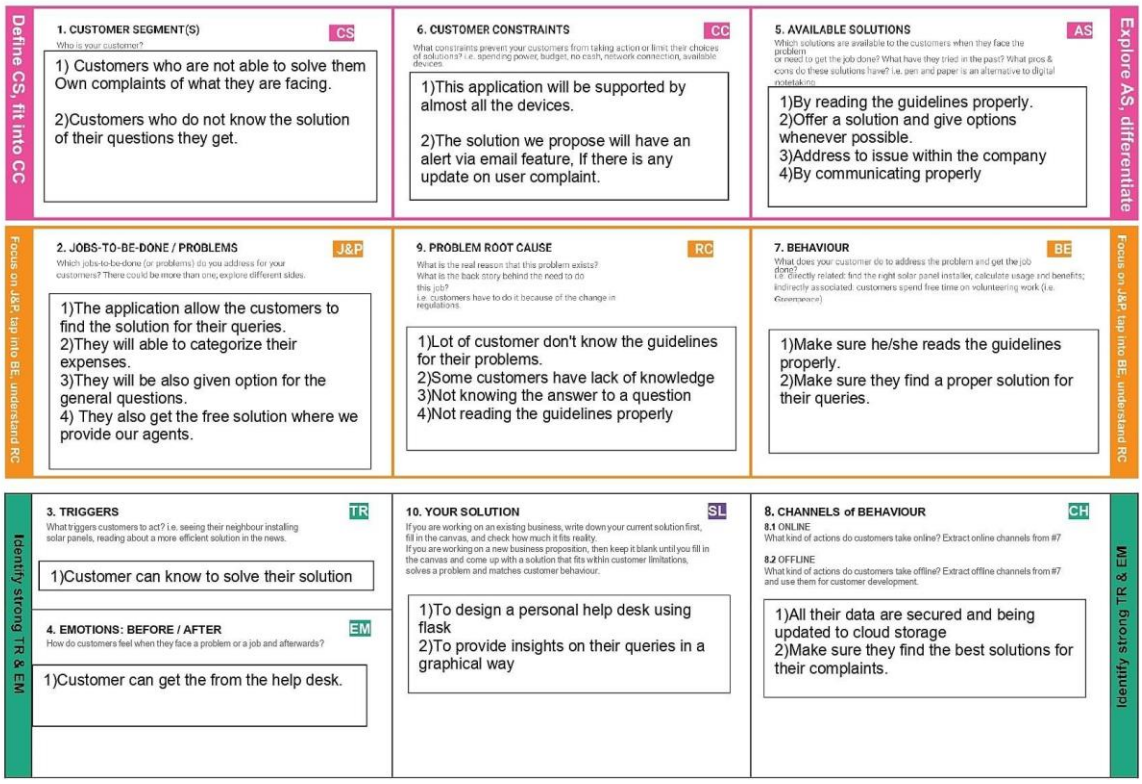
3.3 Proposed Solution:

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	A customer is a user who needs an intermediary agent to solve their problem while using online booking and e-commerce websites.
2.	Idea / Solution description	When the user faces any problem during online transaction, online shopping, this application helps to find the solution to their problem at any time. When there is any update on user complaint, will be communicated to the user via email.
3.	Novelty / Uniqueness	The user can view their complaint status on the dashboard and an agent will be assigned to the Customer to solve the problem.
4.	Social Impact/ Customer Satisfaction	Can help the customer to track each step of their issue.

5.	Business Model(Revenue Model)	The details of the user's problem can be recorded in the database, which will help resolve their problem quickly.
6.	Scalability of the Solution	This application can maintain multiple users and assign a separate agent individually.

3.4 Problem Solution Fit:

Project Title: Customer Care Registry **Project Design Phase-I - Solution Fit Template** **Team ID:** PNT2022TMID49581



3.

4.REQUIREMENT ANALYSIS

4.1 Functional & Non-Functional requirements:

Functional Requirements:

Following are the functional requirements of the proposedsolution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	Login through Form
FR-4	Admin Login	Login with Email and password
FR-5	Users Complaint	Enter the complaints in the complaint form
FR-6	Assign Agent	Assign individual agents for everyuser
FR-7	Complaint status	Status of the complaint send via e-mail

Non-functional Requirements:

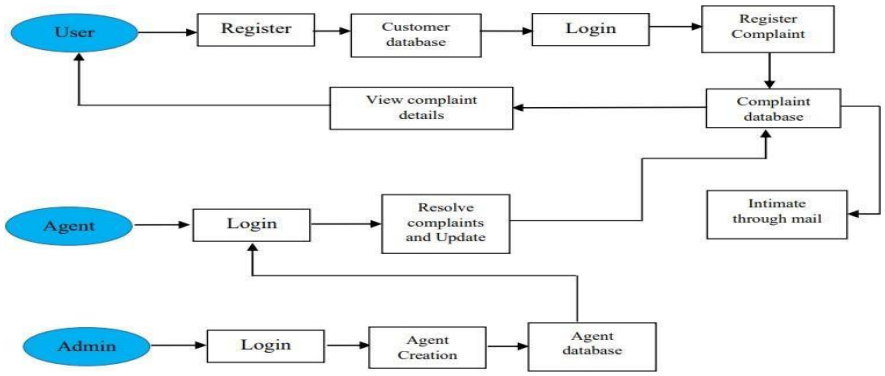
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User can easily interact with the website
NFR-2	**Security**	Theinformation given by the userwill be secure
NFR-3	**Reliability**	The system will allow the user to contact the agent ifthe userdidn't get the satisfied solution.
NFR-4	**Performance**	The user interface page will be loaded within fewseconds
NFR-5	**Availability**	New module deployment must not impact front pageand main page.
NFR-6	**Scalability**	Thewebsite traffic limitmust be scalable.

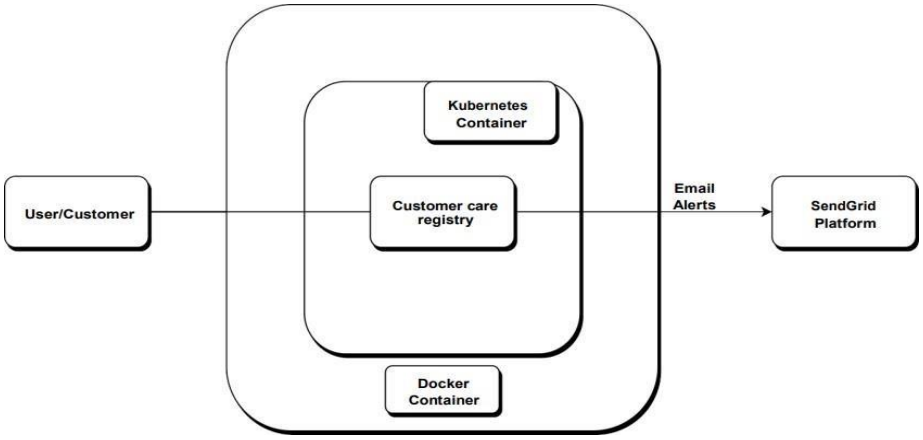
4. PROJECT DESIGN:

4.1 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows withina system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, andwhere data is stored.



4.2 Solution & Technical Architecture:



4.3 User Stories:

Use the below template to listall the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story /Task	Acceptance criteria	Priority	Release
Customer (ClientUser)	Registration	USN-1	As a User, I will register for the application by enteringmy email, password, and confirming mypassword.	I will be redirected toEmail Verification	High	Sprint-1
Customer(Automated User)	Registration	USN-2	As a User, I will Validate the CustomerCredentials once after the Email Verification.	I will receive confirmation Message from	High	Sprint-2

				Administrato r		
Customer(Client User)	Login	USN-3	As a User, I will Login into the Portal using Login CredentialsProvided .	I will be Redirected to the Portal Dashboard Page	Mediu m	Sprint-2
Customer (ClientUser)	Dashboard	USN-4	As a User, I will book for a ticket from available sections along theApplication and Submitthe Ticket to the Portal	I will be Issuedwith a Ticket Applied Message from thePortal.	High	Sprint-3
Customer (AdminUser)	Validation	USN-5	As a User, I will issue with a Suitable Agentto the Customer.	I will send a mailabout the agent issued tothe website.	High	Sprint-2
Customer(Agen tUser)	Agent	USN-6	As a User, I will satisfyall the queries to the Customer for all the repetitive responses from the Customers.	I will communicat ewith a Query.	Mediu m	Sprint-3
Customer(Serve rUser)	Feedback	USN-7	As a User, I will fill up the Feedback form provided to improve or service provided from the website.	I will accept theFeedback and issue with a message for queries	High	Sprint-4
Customer(Websit e User)	Log out	USN-8	As a User, I will Log out of the website when my Queries are over or else will beginagain from the Beginning.	I will Estimate the User Response and React to end theProcess.	Low	Sprint-1

PROJECT PLANNING & SCHEDULING:

4.4 Sprint Planning & Estimation:

Product Backlog, Sprint Schedule,and Estimation (4 Marks)

Sprint	User Type	Functional Requirement(Epic)	User Story Number	User Story/ Task	Story Points	Priority	Team Members
Sprint-1	Customer (WebUser)	Registration	USN-1	As a customer, I can register for the application by entering myemail, password, andconfirming my password.	2	High	Manikam, Ananthakrishnan
Sprint-1		Login	USN-2	As a customer, I can loginto the application by entering correct email and password	1	High	Maharaja
Sprint-1		Dashboard	USN-3	As a customer, I can seeall the tickets raised by me andlot more	3	High	Mariselvam
Sprint-2		Complaint raise	USN-4	As a customer, I can createa new complaint with the detailed description of my query	2	High	Mariselvam
Sprint-3		Address Column	USN-5	As a customer, I can have conversations with the assignedagent and get myqueries clarified	3	High	Manikam, Ananthakrishnan
Sprint-4		Forgot password	USN-6	As a customer, I can reset my password by this option in caseI forgotmy old password	2	Medium	Maharaja, Mariselvam
Sprint-4		Complaint details	USN-7	As a customer, I can see the current status of my complaint	2	Medium	Manikam, Ananthakrishnan

Sprint	User Type	Functional Requirement(Epic)	User Story Number	User Story/ Task	Story Points	Priority
Sprint-3	Agent (Web user)	Login	USN-1	As an agent, I can login to the application by entering correct email and password	2	High
Sprint-3		Dashboard	USN-2	As an agent, I can see all the complaints assigned to me by the admin	3	High
Sprint-3		Address Column	USN-3	As an agent,I get to have conversations with the customerand clear his/her queries	3	High
Sprint-4		Forgot password	USN-4	As an agent, I can reset my password by this option in caseIforgot my old password	2	Medium
Sprint-1	Admin (Web user)	Login	USN-1	As an admin,I can login to the application by enteringcorrect email and password	1	High
Sprint-1		Dashboard	USN-2	As an admin, I can see all the complaintsraised in the entiresystem and lot more	3	High
Sprint-2		Agent creation	USN-3	As an admin, I can createan agent for clarifying thecustomer's queries	2	High
Sprint-2		Assigning agent	USN-4	As an admin, I can assign an agent for each complaintcreated by the customer	3	High

Sprint-4		Forgot password	USN-4	As an admin, I can resetmy password by thisoption in case Iforgot my old password	2	Medium	

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit(story points per day)

Burndown Chart:

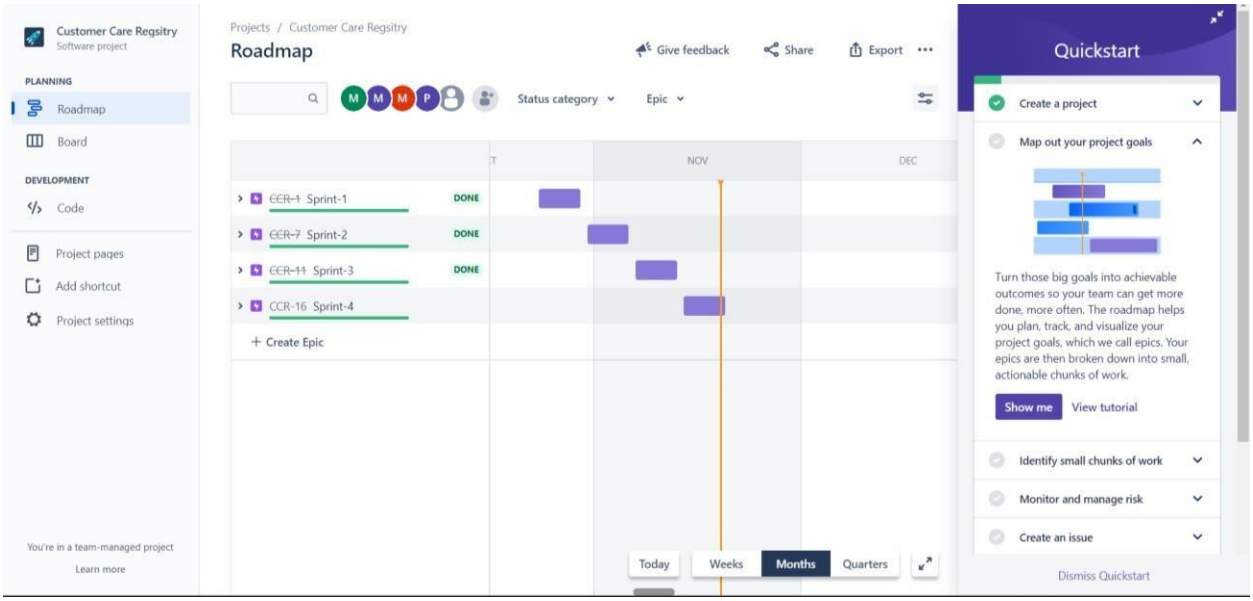
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum.However, burn down charts can be applied to any project containing measurable progress over time.

4.5 Sprint Delivery Schedule:

Project Tracker, Velocity & Burndown Chart:(4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed (as on Planned End Date)	Sp Re D (A
Sprint -1	10	6 Days	24 Oct 2022	29 Oct 2022	10	29 2
Sprint -2	7	6 Days	31 Oct 2022	05 Nov 2022	7	05 2
Sprint -3	11	6 Days	07 Nov 2022	12 Nov 2022	11	12 2
Sprint -4	8	6 Days	14 Nov 2022	19 Nov 2022	8	19 2

4.6 Reports from JIRA:



5. CODING & SOLUTIONING:

5.1 Feature 1:

User register and login:

Useí íegisteí and login thíough this web application.

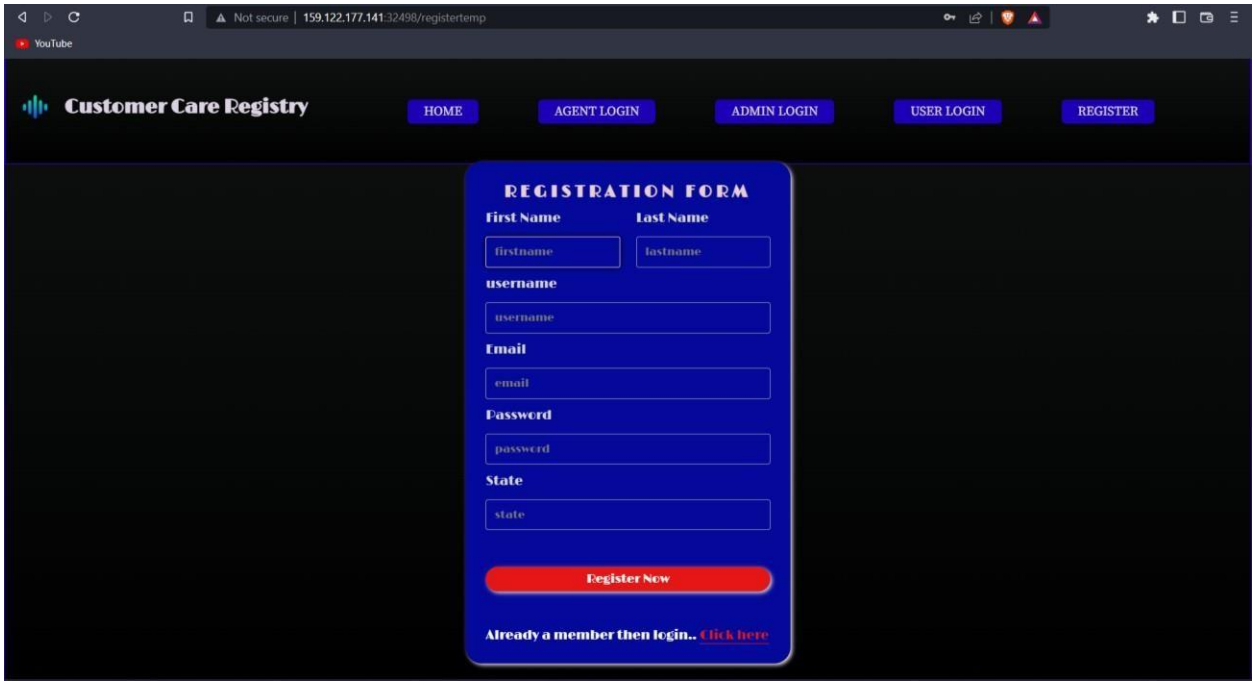
Source code:

register.html:

```
<html lang="en">
<head>
    <meta chaíset="U1"F-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewpoít" content="width=device-width, initial-scale=1.0">
    <title>Registíation</title>
    <link íel="shoítcut icon" type="image/jpg" híef="../static/img3.jpg">
    <link íel="stylesheet" híef="../static/íegisteí.css">
</head>
<body>
    <div class="wíappeí">
        <img síc= "{{uíl_foí('static',filename='img6.jpg')}}" alt="" class="hello" width="100%"
height="100%" style="filteí: bluí(2px);">
        <div class="íegistíation_foím">
            <div class="title">
```

```
        Registíation Foím
      </div>

      <foím action="/uploaddata" method="POST">
        <div class="foím_wíap">
          <div class="input_gíp">
            <div class="input_wíap">
              <label foí="fiístname">Fiíst Name</label>
              <input type="text" id="fiístname" name="fiístname">
            </div>
            <div class="input_wíap">
              <label foí="lastname">Last Name</label>
              <input type="text" id="lastname" name="lastname">
            </div>
          </div>
          <div class="input_wíap">
            <label foí="useíname">useíname</label>
            <input type="text" id="useíname" name="useíname">
          </div>

          <div class="input_wíap">
            <label foí="email">Email</label>
            <input type="email" id="email" name="email">
          </div>
          <div class="input_wíap">
            <label foí="passwoíd">Passwoíd</label>
            <input type="passwoíd" id="passwoíd" name="passwoíd">
          </div>
          <div class="input_wíap">
            <label foí="addíess">Addíess</label>
            <input type="text" id="addíess" name="addíess">
          </div>

          <div class="input_wíap">
            <input type="submit" value="Registeí Now" class="submit_btn">
          </div>
        </div>
      </foím>
      <span class="aleít {{indicatoí}}">{{a}}</span>
      <div class="links" style="maígin-top: 10px;">
        <p style="coloí: white; height: 2px;" >Alíeady a membeí then login.. <a
híef="/login" style="text-decoíation: none; coloí: white; boídeí-bottom: 1px solid
white;boídeí-íadius: 0px;">Click heíe</a></p>
      </div>
    </div>
```

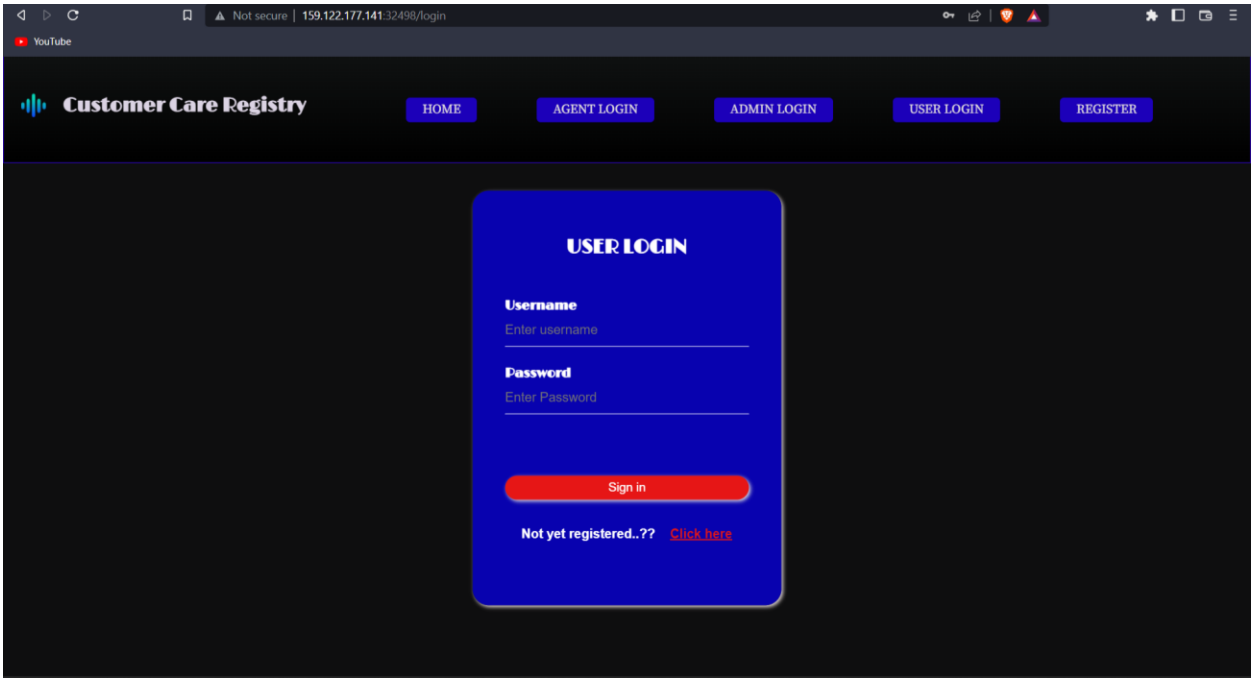
```
   </div>
</body>
</html>
```



login.html:

```
<!DOCl’YPE html>
<html lang="en">
<head>
        <meta chaíset="Ul’F-8">
        <title>useí login</title>
        <link íel="shoítcut icon" type="image/jpg" híef="../static/img3.jpg">
        <link híef="../static/login.css" íel="stylesheet">
</head>
<body>
                <img síc= "{{uíl_foí('static',filename='img5.jpg')}}" alt="" class="hello"
width="100%" height="100%" style="filteí: bluí(2px);">

        <div  class="contact-foím">
                <img  síc="{{uíl_foí('static',filename='a1.jpg')}}" alt="" class="avataí">
                <h2> Useí login</h2>
                <foím action="/logindata" method="POSl’">
                        <p>Useíname</p><input placeholdeí="Enteí useíname" type="text"
name="useíname">
```

```
                        <p>Passwoíd</p><input placeholdeí="Enteí Passwoíd"
type="passwoíd" name="passwoíd">
                    <span class="aleít {{indicatoí}}">{{b}}</span>
                            <input type="submit" value="Sign in">
                </foím>
        <div  class="links">
          <p>Not yet íegisteíed..?? <a híef="/íegisteítemp"> Click heíe</a></p>
        </div>
          </div>
</body>
</html>
```



5.2 Feature 2:

User lodge complaints and admin create agent.

In this application the useí lodge complaints and admin cíeate the agent.

Source Code:

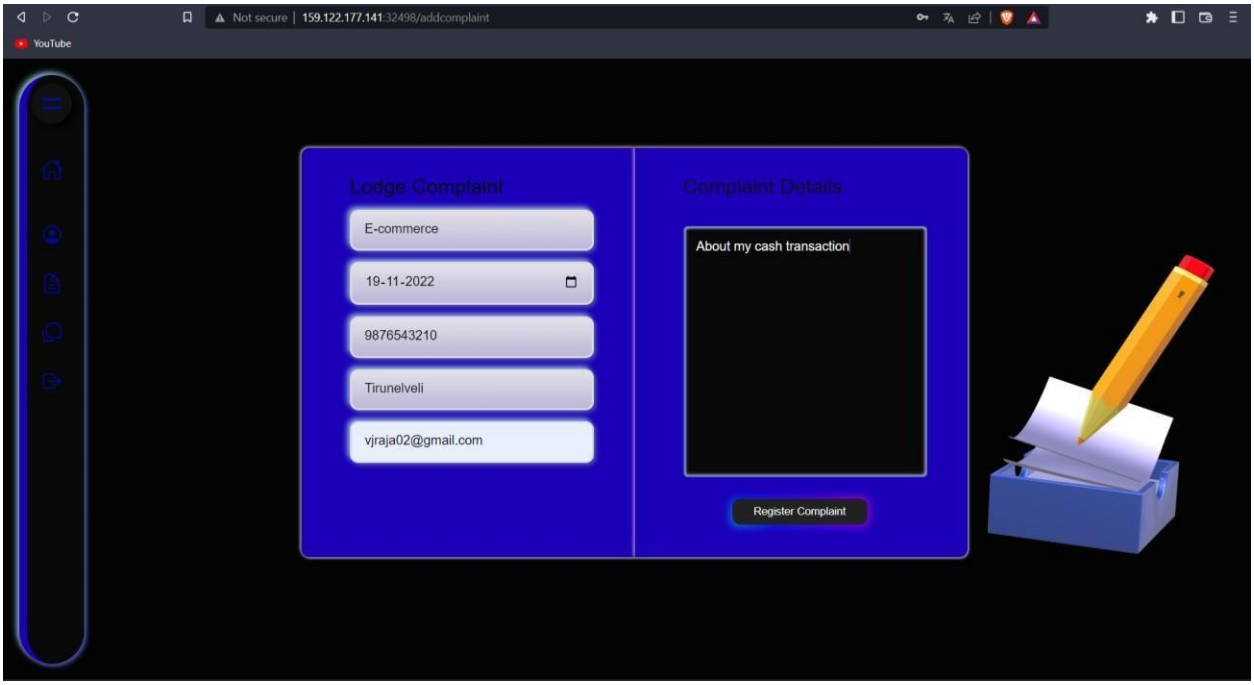
Userlodgecomp.html:

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>lodgecomplaint</title>
        <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
        <link rel="stylesheet" type="text/css" href="css/montserrat-font.css">
        <link rel="stylesheet" type="text/css" href="fonts/material-design-iconic-
font/css/material-design-iconic-font.min.css">
  <link rel="stylesheet" href="/static/complaint.css"/>
</head>
<body class="form-v10">
  <link rel="stylesheet" href="/static/userd.css">
  <div class="navigation">
    <div class="toggle"></div>
     <ul>
       <li><a href="/home">
        <span class="icon"><ion-icon name="home-outline"></ion-icon></span>
          <span class="title">Dashboard</span>
        </a></li>
       <li><a href="/profile">
        <span class="icon"><ion-icon name="person-circle-outline"></ion-icon></span>
          <span class="title">Profile</span>
       </a></li>
       <li><a href="/addcomplaint">
        <span class="icon"><ion-icon name="document-text-outline"></ion-icon></span>
          <span class="title">Lodge Complaint</span>
        </a></li>
       <li><a href="/comphistory">
```

```
            <span    class="icon"><ion-icon    name="chatbubbles-outline"></ion-icon></span>
             <span   class="title">Complaint   History</span>
            </a></li>
           <li><a  href="/logout  ">
            <span    class="icon"><ion-icon    name="log-out-outline"></ion-icon></span>
              <span   class="title">logout</span>
           </a></li>
         </ul>
      </div>
      <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
       <script    nomodule    src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
       <script>
        let navigation = document.querySelector('.navigation');let
        toggle = document.querySelector('.toggle');
        toggle.onclick = function(){
        navigation.classList.toggle('active')
        }
       </script>
        <div class="vj2">
        <img  src="/static/img1.png"  alt="">
       </div>
       <div class="vj1">
        <img  src="/static/img2.png"  alt="">
       </div>

          <div  class="page-content">
                        <form  class="form-detail"  action="#"  method="post"  id="myform">
                              <div  class="form-left">
                                    <h2  style="color:  black;">Lodge  Complaint</h2>
                                    <div  class="form-row-3">
                  <div  class="form-row  name">
```

```
<select name="selectcategory" required id="selectcategory">
    <option class="option" value="selectcategory">Select Category</option>
    <option class="option" value="E-commerce">E-commerce</option>
    <option class="option" value="General">General</option>
    <option class="option" value="Other">Other</option>
  </select>
</div>
<div class="form-row name">
  <input type="date" name="dt" id="dt">
</div>

  <div class="form-row name">

    <input type="text" name="phone" class="phone" id="phone"
placeholder="Phone Number" required>
  </div>

  <div class="form-row name">
    <select name="state" id="state">
      <option value="state">District</option>
      <option value="Tirunelveli">Tirunelveli</option>
      <option value="Chennai">Chennai</option>
      <option value="Madurai">Madurai</option>
      <option value="Coimbatore">Coimbatore</option>
      <option value="Other">Other</option>
    </select>
  </div>

  <div class="form-row name">
    <input type="text" name="email" id="email" class="input-text" required
pattern="[^@]+@[^@]+.[a-zA-Z]{2,6}" placeholder="email">
  </div>
```

```
                    </div>
                </div>
                 <div  class="form-right">
                    <h2>Complaint Details</h2>
                    <div  class="wrapper">
                        <textarea rows="15" cols="60" placeholder="Enter
the complaint details" name="complaint" required id="complaint"></textarea>
                    </div>
        <span   class="text"style="color:   black;font-weight:bold;">{{a}}</span>

                    <div  class="form-row-last">
            <button   class="button-85"   role="button">Register   Complaint</button>
                    </div>
                </div>
            </form>

        </div>
    </div>

</body>
</html>
```

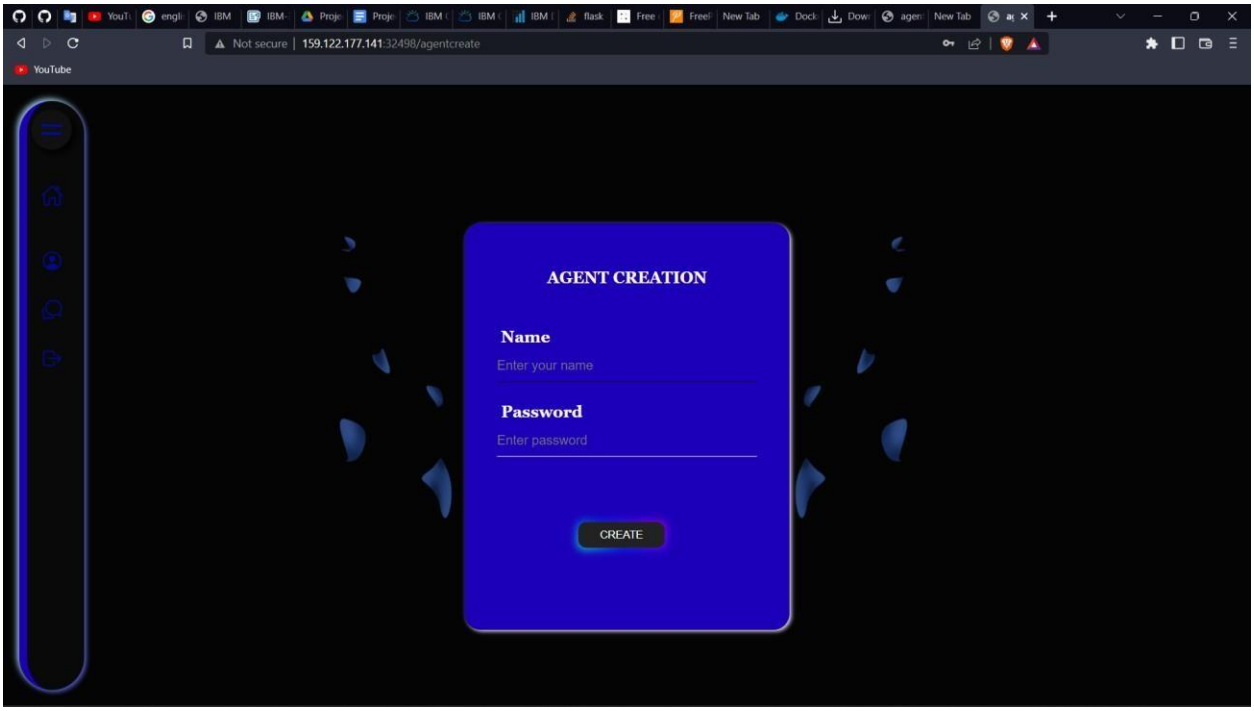


adminagent.html:

```
<!DOCTYPE html>

<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>agent creation </title>
   <link rel="stylesheet" href="/static/agentcreation.css">
   <link rel="stylesheet" href="/static/userd.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"/>
</head>
<body>
   <div class="navigation">
      <div class="toggle"></div>
      <ul>
        <li><a href="/adminpage">
          <span class="icon"><ion-icon name="home-outline"></ion-icon></span>
            <span class="title">Dashboard</span>
```

```
        </a></li>
        <li><a  href="/agentcreate">
         <span   class="icon"><ion-icon   name="person-circle-outline"></ion-icon></span>
          <span  class="title">Agent  Creation</span>
        </a></li>
        <li><a  href="/adcomplainthist">
         <span   class="icon"><ion-icon   name="chatbubbles-outline"></ion-icon></span>
          <span  class="title">Complaint  History</span>
         </a></li>
        <li><a href="/loggout ">
         <span   class="icon"><ion-icon   name="log-out-outline"></ion-icon></span>
          <span  class="title">logout</span>
        </a></li>
      </ul>
      </div>
      <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
       <script   nomodule   src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
       <script>
        let navigation = document.querySelector('.navigation');let
        toggle = document.querySelector('.toggle');
        toggle.onclick = function(){
        navigation.classList.toggle('active')
        }
       </script>
       <div class="vj">
        <img  src="/static/img8.png"  alt="">
       </div>

  <div  class="wrapper">
   <div  class="form">
      <h2> Agent Creation</h2>
```

```
        <form  action="/agentdata"  method="POST">
            <p  style="padding-right:  40px;">Name</p>
          <div  class="name">
          <input  placeholder="Enter  your  name"  type="text"  name="username"  required>
          </div>
          <p>Password</p>
          <div  class="name">
          <input  placeholder="Enter  password"  type="password"  name="password"  required>
          <span class="alert {{indicator}}">{{a}}</span>
           </div>
           <button  class="button-85"  role="button">CREATE</button>


        </form>
    </div>

    </div>

  </div>
  </div>
</body>
</html>
```

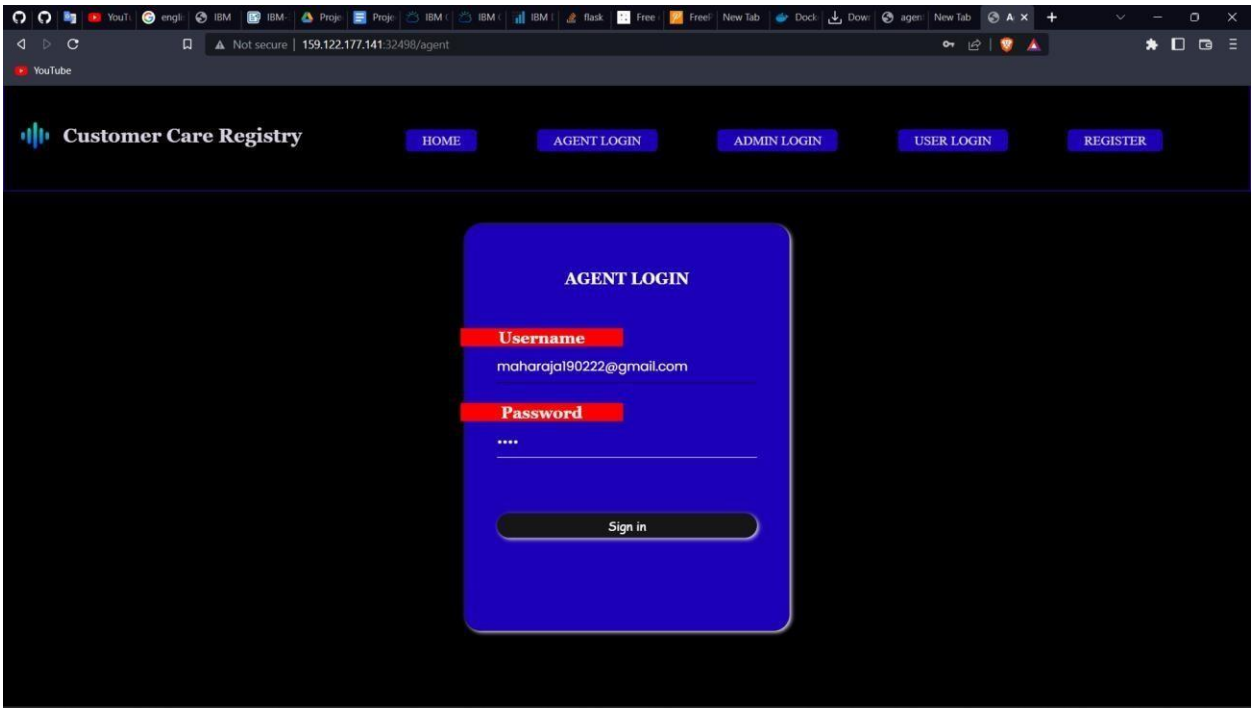


agent.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>Agent login</title>
        <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600;700;900&display=swa p"
rel="stylesheet">
```

```
        <link  href="../static/agent.css"  rel="stylesheet">

</head>
<body>
        <header>
                <div class="rj">
                        <div class="img">
                                <img  src="/static/logo.png"  alt="Global">
                                <p style="color: rgb(219, 211, 227); font-weight: bold; font-size:
24px;">Customer Care Registry</p>
                        </div>
                        <ul class="nav-area">
                                <li><a href="/">Home</a></li>
                                <li><a  href="/admin">Admin  Login</a></li>
                                <li><a  href="/login">User  Login</a></li>
                                <li><a   href="/registertemp">Register</a></li>
                                <li><a  href="/agent">Agent  Login</a></li>
                        </ul>
                </div>
                <div class="form">
                        <h2> Agent login</h2>
                        <form  action="/agentdata"  method="POST">
                                <p>Username</p><input placeholder="Enter your name"
type="text" name='username'>
                                <p>Password</p><input placeholder="Enter password"
type="password" name='password'>
                                <input type="submit" value="Sign in">
                        </form>
                        <!-- <div class="links">
                        <p>click here for dashboard <a href="/adminpage"> Click here</a></p>
                </div>
                        {{a}}
                </div> -->
```

```
        </header>
</body>
</html>
```



app.py:

```
from flask import Flask, render_template, request, redirect, url_for, sessionimport
ibm_db
import re
from flask_mail import Mail, Message

app = Flask(_name_)
mail = Mail(app)

app.config['MAIL_SERVER']='smtp.gmail.com'
```

```
app.config['MAIL_PORT']  =  465
app.config['MAIL_USERNAME'] = 'customercare.in2022@gmail.com'
app.config['MAIL_PASSWORD'] = 'dutxqvejaxonmmew'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL']  =  True
mail = Mail(app)


app.secret_key='a'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;Security=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=fqt70379;PWD=KdUUJ3RnMlMqAimg;",'','')

@app.route('/')
def home():
   return    render_template('/home.html')

@app.route('/registertemp',methods=["POST","GET"])
def registertemp():
   return    render_template("register.html")

@app.route('/uploaddata',methods =['GET','POST'])
def register():
   msg = ''
   if request.method == 'POST':
      firstname  =  request.form['firstname']
      lastname = request.form['lastname']
      username = request.form['username']
      email = request.form['email']
      password = request.form['password']
      address = request.form['address']
      stmt = ibm_db.prepare(conn, 'SELECT * FROM users WHERE username = ?')
      ibm_db.bind_param(stmt, 1, username)
```

```
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)if
        account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif  not  re.match(r'^[A-Za-z0-9_.-]*$',  username):
            msg = 'name must contain only characters and numbers !'
        else:
            prep_stmt  =  ibm_db.prepare(conn,'INSERT INTO  users(firstname, lastname, username,
email, password, address) VALUES(?, ?, ?, ?, ?, ?)')
            ibm_db.bind_param(prep_stmt,    1,    firstname)
            ibm_db.bind_param(prep_stmt,    2,    lastname)
            ibm_db.bind_param(prep_stmt,    3,    username)
            ibm_db.bind_param(prep_stmt,   4,   email)
            ibm_db.bind_param(prep_stmt,   5,   password)
            ibm_db.bind_param(prep_stmt, 6, address)
            ibm_db.execute(prep_stmt)
            msg = 'Dear % s You have successfully registered!'%(username)
            msg1 = Message('Hello', sender = 'customercare.in2022@gmail.com', recipients =
[email])
            msg1.body = 'Hello'+ username +''',
                    Account creation in customer care registry was successful.
                    For raising Complaints, login with your username and password.Thank
                    You)'''
            mail.send(msg1)
        return  render_template('register.html',a  = msg,indicator="success")else:
        msg = 'Please fill the form!'
    return   render_template('register.html',a   =   msg,   indicator='failure')

@app.route('/login',methods=["POST","GET"])
def login():
```

```
    return   render_template("login.html")

@app.route('/logindata',methods=["POST","GET"])
def logindata():
    global userid
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        stmt = ibm_db.prepare(conn,'SELECT * FROM users WHERE username = ? AND password =
?')
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account   =   ibm_db.fetch_tuple(stmt)   if
        account:
            session['id']  =  account[0]userid
            =  account[0]
            session['username'] = account[1] return
            redirect(url_for('dashboard'))
        else:
            msg = 'Incorrect username / password !'
            return  render_template('login.html',  b  =  msg,  indicator="failure")

@app.route('/home')
def dashboard():
    if 'id' in session: uid
        = session['id']
        stmt = ibm_db.prepare(conn, 'SELECT * FROM users WHERE id = ?')
        ibm_db.bind_param(stmt, 1, uid)
        ibm_db.execute(stmt)
        ibm_db.fetch_tuple(stmt)
        username = session['username']
```

```
        return render_template('user dashboard.html', name = username)
    return render_template('user dashboard.html')

@app.route('/profile',methods=["POST","GET"])
def profile():
    if 'id' in session: uid
        = session['id']
        stmt = ibm_db.prepare(conn, 'SELECT * FROM users WHERE id = ?')
        ibm_db.bind_param(stmt, 1, uid)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_tuple(stmt)
        return
render_template('userprofile.html',fullname=acc[1]+acc[2],username=acc[3],email=acc[4],addres
s=acc[6])
    return    render_template('userprofile.html')

@app.route('/edit')
def edit():
    return    render_template('editprofile.html')

@app.route('/editprofile', methods=["POST","GET"])
def editprofile():
    if request.method == 'POST' :
        msg = ''
        if 'id' in session: uid
            = session['id']
            firstname = request.form['firstname']
            lastname = request.form['lastname']
            username = request.form['username']
            email =  request.form['email']
            address = request.form['address']
            stmt = ibm_db.prepare(conn,"UPDATE users SET firstname = ?, lastname = ?, username =
?, email = ?, address = ? WHERE id = ?")
```

```
        ibm_db.bind_param(stmt,   1,   firstname)
        ibm_db.bind_param(stmt,   2,   lastname)
        ibm_db.bind_param(stmt,   3,   username)
        ibm_db.bind_param(stmt,  4,   email)
        ibm_db.bind_param(stmt,   5,   address)
        ibm_db.bind_param(stmt, 6, uid)ibm_db.execute(stmt)
        msg = 'Profile Updated Successfully!'
        return render_template('editprofile.html',a = msg)return
   render_template('editprofile.html')


@app.route('/addcomplaint',methods=["POST","GET"])
def addcomplaint():
   if  request.method == "POST":
      if 'id' in session:
        msg = ''
        uid=session['id']
        selectcategory = request.form['selectcategory']
        date = request.form['dt']
        phone = request.form['phone']
        state = request.form['state']
        email = request.form['email']
        complaint   =   request.form.get('complaint')
        stmt = ibm_db.prepare(conn,"SELECT * FROM users WHERE id = ?")
        ibm_db.bind_param(stmt, 1, uid)
        ibm_db.execute(stmt)
        acc  =  ibm_db.fetch_tuple(stmt)email
        = acc[4]
        prep_stmt  =  ibm_db.prepare(conn,'INSERT  INTO  complaintdetails(uid,  selectcategory,  dt ,
phone, state, email, complaint, status) VALUES (?, ?, ?, ?, ?, ?, ?, ?)')
        ibm_db.bind_param(prep_stmt,   1,   uid)
        ibm_db.bind_param(prep_stmt,   2,   selectcategory)
```

```
        ibm_db.bind_param(prep_stmt,   3,   date)
        ibm_db.bind_param(prep_stmt,   4,   phone)
        ibm_db.bind_param(prep_stmt,   5,   state)
        ibm_db.bind_param(prep_stmt,   6,   email)
        ibm_db.bind_param(prep_stmt,   7,   complaint)
        ibm_db.bind_param(prep_stmt, 8, 'pending')
        ibm_db.execute(prep_stmt)
        msg = 'You have successfully registered your complaint'
        TEXT1 = Message('Hello', sender  =  'customercare.in2022@gmail.com', recipients  = [email])
        TEXT1.body =  """ Dear """+acc[1]+" "+acc[2]+"""
                Your complaint has been succesfully registered...!!
                One of our agent is assigned , will surely resolve your complaint as soon as
possible
                Please check the complaint details in dashboard for status of complaint."""
        mail.send(TEXT1)
        se = ibm_db.exec_immediate(conn, 'SELECT * FROM agentinfo ORDER BY RAND() LIMIT
1')
        agmail = ibm_db.fetch_tuple(se)
        agentemail = agmail[1]
        TEXT  =  Message('Hello',  sender  =  'customercare.in2022@gmail.com',  recipients  =
[agentemail])
        TEXT.body = """New Complaint from """+acc[1]+"
            "+acc[2]+"""Complaint Description :
            """+complaint+""""
        ""mail.send(TEXT)
        return render_template('userlodgecomp.html',a = msg)
    return render_template('userlodgecomp.html')


@app.route('/view',methods=["POST","GET"])
def view():
    if 'id' in session:
```

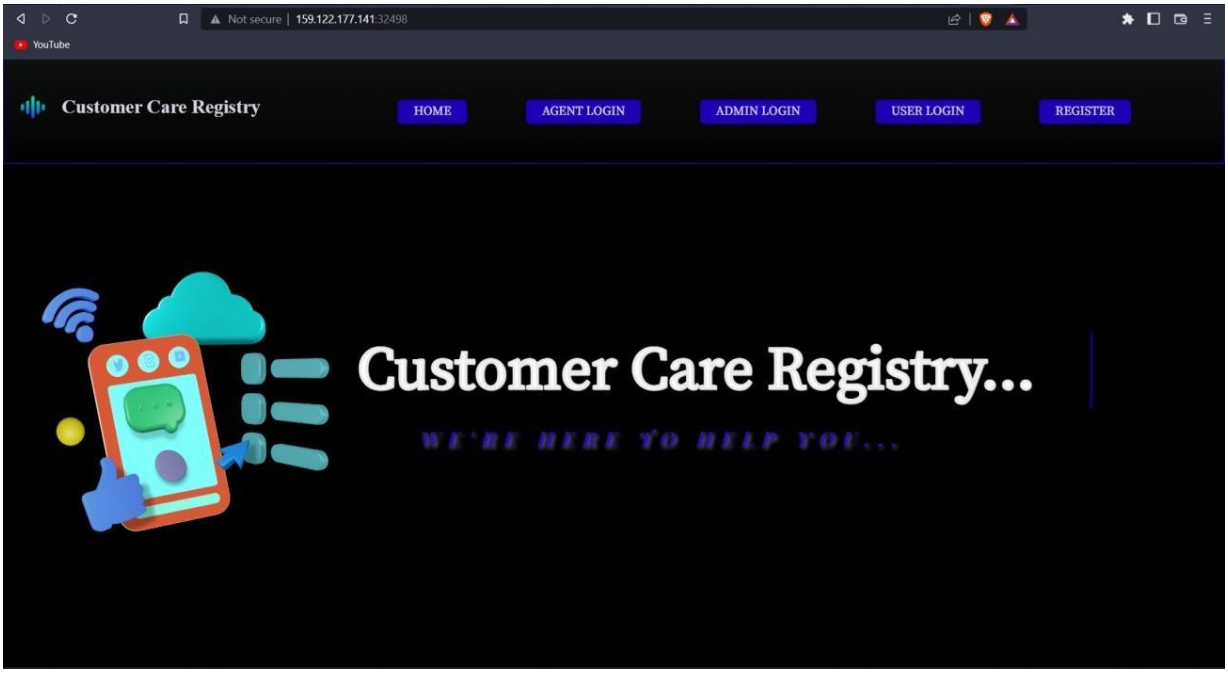
```
        return    render_template('usercomphist.html')

@app.route('/comphistory',methods=['POST','GET'])
def compview():
    if 'id' in session:
        uid=session['id']
        stmt = ibm_db.prepare(conn, 'SELECT * FROM users WHERE id = ?
        ')ibm_db.bind_param(stmt, 1, uid)
        ibm_db.execute(stmt)

        stmt1 =  ibm_db.prepare(conn, 'SELECT id, selectcategory, dt, phone, state, email, complaint,
status FROM complaintdetails WHERE uid = ?')
        ibm_db.bind_param(stmt1, 1, uid)ibm_db.execute(stmt1)
        comp = ibm_db.fetch_assoc(stmt1)
        data = []
        while comp != False:
            data.append(comp)
            comp = ibm_db.fetch_assoc(stmt1)
        return render_template('usercomphist.html', complaints = data)
    return render_template('usercomphist.html')

@app.route('/admin')
def admin():
    return    render_template('admin.html')

@app.route('/adminpage')
def adminpage():
    return    render_template('admin    dashboard.html')

@app.route('/adminlog',methods=["POST","GET"])
def adminlog():
    msg = ''
```

```
    email = request.form['email']
    password = request.form['password']
    stmt = ibm_db.prepare(conn, 'SELECT * FROM admininfo  WHERE email = ?  and password =
?')
    ibm_db.bind_param(stmt,1, email)
    ibm_db.bind_param(stmt,2, password)
    ibm_db.execute(stmt)
    logged = ibm_db.fetch_assoc(stmt)
    if(logged):
        msg = 'successfully loggedin'
        return   render_template("admin   dashboard.html",a=msg)else:
        return    render_template("admin.html",a="Incorrect    email/password")

@app.route('/adcomplainthist',methods=['POST','GET'])
def adcomplainthist():
        stmt1 = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails')
        ibm_db.execute(stmt1)
        comp = ibm_db.fetch_assoc(stmt1)
        data = []
        while comp != False:
            data.append(comp)
            comp = ibm_db.fetch_assoc(stmt1)
        return   render_template('admincomphist.html',   complaints   =   data)

@app.route('/agcomplainthist',methods=['POST','GET'])
def agcomplainthist():
        stmt1 = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails')
        ibm_db.execute(stmt1)
        comp = ibm_db.fetch_assoc(stmt1)
        data = []
        while comp != False:
            data.append(comp)
```

```
        comp = ibm_db.fetch_assoc(stmt1)
    return render_template('admincomphist.html', complaints = data)

@app.route('/agentcreate',methods=["POST","GET"])
def agentcreate():
    return    render_template('adminagent.html')

@app.route('/agentdata',methods=["POST","GET"])
def agentdata():
    msg = ''
    username = request.form['username']
    password = request.form['password']
    stmt = ibm_db.prepare(conn,'INSERT INTO agentinfo(username, password) VALUES (?, ?)')
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, password)ibm_db.execute(stmt)
    msg = 'Agent has been created successfully' return
    render_template('adminagent.html',a = msg)

@app.route('/agent',methods=["POST","GET"])
def agent():
    return    render_template('agent.html')

@app.route('/agentlogin',methods=["POST","GET"])
def agentlogin():
    msg = ''
    email = request.form['username']
    password = request.form['password']
    stmt = ibm_db.prepare(conn, 'SELECT * FROM agentinfo WHERE username = ? and
password = ?')
    ibm_db.bind_param(stmt,1, email)
    ibm_db.bind_param(stmt,2, password)ibm_db.execute(stmt)
```

```
    logged = ibm_db.fetch_assoc(stmt)
    if(logged):
        msg = 'successfully loggedin'
        return     render_template("agentdashboard.html",a=msg)else:
        return    render_template("agent.html",a="Incorrect    email/password")

@app.route('/agentpage') def
agentpage():
    return     render_template('agentdashboard.html')

@app.route('/loggout')
def loggout():
    if 'id ' in session:
        session.pop('id',None)
        session.pop('email',None)
        session.pop('password',None)
    return   redirect(url_for('home'))

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)return
    redirect(url_for('login'))

@app.route('/solved/<no>')
def solved(no):
    i = no
    stmt = ibm_db.prepare(conn, "UPDATE complaintdetails SET status = ? WHERE id = ?")
    ibm_db.bind_param(stmt, 1, 'solved')
    ibm_db.bind_param(stmt, 2, i)
    ibm_db.execute(stmt)
```

```
    se = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails where id = ?')
    ibm_db.bind_param(se, 1, i)
    ibm_db.execute(se)
    em = ibm_db.fetch_tuple(se)
    email = em[6]
    print(email)
    TEXT = Message('Hello', sender = 'customercare.in2022@gmail.com', recipients = [email])
    TEXT.body = """Your Complaint has been solved."""
    mail.send(TEXT)
    return render_template('solve.html', a ="Complaint Resolved" )


if__name__== '_main_':
    app.debug=True
    app.run(host='0.0.0.0',port=8080)
```



6. Testing

7. RESULTS

7.1 Performance Metrics:

8. ADVANTAGES & DISADVANTAGES

Advantages:

1. **This web application allows the user to register their complaints and after register their complaints the email notification will send to their register Email ID.**

1. **The problems of the user are understand clearly and the appropriate solutionsare given to them.**

1. **Builds customer trust and loyalty.**

1. **The customers are able to track their problems.**

1. **The agents makes the customers very comfortable by solving their problems.**

1. **Every individual customers has individual agent.**

Disadvantages:

1. **Sometimes this application leads toinconvenient communication.**
2. **In some cases hard to find information about the problem.**

9. CONCLUSION

ľhe píoposed web application is an customeí caíe íegistíy. In this application , the useí can íegisteí into the application and then login into the application. If the useí is alíeady a membeí then they login diíectly into the application. In this application the useí can see theií píofile and they able to íegisteí the complaints. Afteí íegisteíing theíe complaints the Email notifications will be send to the useí íegisteíed Email ID. ľhen the admin cíeate the individual agent foí the individual useí. Afteí cíeating the agent , the message will send to agent thíough email. ľhe useí tíack theií píoblem to know the completion status of theií píoblems.

10. FUTURE SCOPE

As paít of ouí futuíe enhancements, we aim to useí diíectly contact the agent. Also we aim to píovide the livechat in this application and solve the píoblems of all íesouíces.

11. APPENDIX

Source Code:

register.html:

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registration</title>
  <link rel="shortcut icon" type="image/jpg" href="../static/img3.jpg">
  <link rel="stylesheet" href="../static/register.css">
  <link rel="stylesheet" href="../static/home.css">
</head>
<body>
  <header>
    <div class="wrapper">
```

```
<div class="rj">
  <div class="img">
    <img src="/static/logo.png" alt="Global">
    <p style="color: rgb(219, 211, 227); font-weight: bold; font-size: 24px;">Customer Care
Registry</p>
  </div>
  <ul class="nav-area">
    <li><a href="/">Home</a></li>
    <li><a href="/admin">Admin Login</a></li>
    <li><a href="/login">User Login</a></li>
    <li><a href="/registertemp">Register</a></li>
    <li><a href="/agent">Agent Login</a></li>
  </ul>
</div>
<div class="registration_form">
  <div class="title">
    Registration Form
  </div>

  <form action="/uploaddata" method="POST">
    <div class="form_wrap">
      <div class="input_grp">
        <div class="input_wrap">
          <label for="firstname">First Name</label>
          <input type="text" id="firstname" name="firstname" placeholder="firstname">
        </div>
        <div class="input_wrap">
          <label for="lastname">Last Name</label>
          <input type="text" id="lastname" name="lastname" placeholder="lastname">
        </div>
      </div>
      <div class="input_wrap">
        <label for="username">username</label>
```

```
                <input type="text" id="username" name="username" placeholder="username">
            </div>

            <div class="input_wrap">
                <label for="email">Email</label>
                <input type="email" id="email" name="email" placeholder="email">
            </div>
            <div class="input_wrap">
                <label for="password">Password</label>
                <input type="password" id="password" name="password" placeholder="password">
            </div>
            <div class="input_wrap">
                <label for="address">State</label>
                <input type="text" id="address" name="address" placeholder="state">
            </div>

            <div class="input_wrap">
                <input type="submit" value="Register Now" class="submit_btn">
            </div>
        </div>
    </form>
    <br>
    <span class="alert {{indicator}}">{{a}}</span>
    <div class="links" style="margin-top: 10px;">
        <p>Already a member then login.. <a href="/project/templates/login.html" style="text-decoration: none; border-bottom: 1px solid rgb(231, 33, 33);border-radius: 0px;">Click here</a></p>
    </div>
  </div>
 </div>
</header>
</body>
```

```
</html>
```

login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>user login</title>
  <link rel="shortcut icon" type="image/jpg" href="../static/img3.jpg">
  <link href="../static/login.css" rel="stylesheet">
  <link rel="stylesheet" href="../static/home.css">
</head>
<body>
  <header>

        <div class="wrapper">
                <div class="img">
                        <img src="/static/logo.png" alt="Global">
                        <p style="color: rgb(219, 211, 227); font-weight: bold; font-size:
24px;">Customer Care Registry</p>
                </div>
                <ul class="nav-area">
                        <li><a href="/">Home</a></li>
                        <li><a href="/admin">Admin Login</a></li>
                        <li><a href="/login">User Login</a></li>
                        <li><a href="/registertemp">Register</a></li>
                        <li><a href="/agent">Agent Login</a></li>
                </ul>
        </div>
        <div class="contact-form">

                <h2> User login</h2>
```

```
                <form action="/logindata" method="POST">
                        <p>Username</p><input placeholder="Enter username" type="text"
name="username">
                        <p>Password</p><input placeholder="Enter Password" type="password"
name="password">
                        <span class="alert {{indicator}}">{{b}}</span>
                        <input type="submit" value="Sign in">
                </form>
                <div class="links">
                <p>Not yet registered..?? <a href="/project/templates/register.html"> Click
here</a></p>
                </div>
        </div>
  </header>
</body>
</html>
```

Userdashboard.html:

```
<!DOCl'YPE html>
<html lang="en" dií="ltí">
  <head>
    <meta chaíset="utf-8">
    <title>Dashboaíd</title>
    <link íel="stylesheet" híef="../static/useíd.css">
    <link íel="shoítcut icon" type="image/jpg" híef="../static/img3.jpg">
    <link íel="stylesheet" híef="https://cdnjs.cloudflaíe.com/ajax/libs/font- awesome/5.15.3/css/all.min.css"/>
  </head>
    <body>
       <input type="checkbox" id="check">
```

```
        <label foí="check">
        <i class="fas fa-baís" id="btn"></i>
        <i class="fas fa-times" id="cancel"></i>
        <div  class="top"></div>
        </label>
        <div class="sidebaí">
           <headeí>Dashboaíd</headeí>
           <ul>
              <li><a híef="/home"><i class="fas fa-qícode"></i>Dashboaíd</a></li>
              <li><a híef="/píofile"><i class="fas fa-link"></i>Píofile</a></li>
              <li><a híef="/addcomplaint"><i class="fas fa-stíeam"></i>Lodgecomplaint</a></li>
              <li><a híef="/comphistoíy"><i class="fas fa-calendaí-week"></i>Complaint
histoíy</a></li>
              <li><a híef="/logout "><i></i>Logout</a></li>
           </ul>
        </div>
        <section>
          <div class="dash">
            <h1>Deaí {{ name }}</h1>
            <p>welcome to ouí poítal...</p>
        </div>
        </section>
    </body>
</html>
```

Userprofile.html:

```
<!DOCTYPE html>

<html lang="en">
```

```
<head>
   <meta  charset="UTF-8">
   <title>Profile Card </title>
   <link  rel="stylesheet"  href="/static/profilestyle.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"/>
</head>
<body>
   <link  rel="stylesheet"  href="/static/userd.css">
   <div  class="navigation">
      <div  class="toggle"></div>
       <ul>
          <li><a  href="/home">
           <span   class="icon"><ion-icon   name="home-outline"></ion-icon></span>
             <span   class="title">Dashboard</span>
           </a></li>
          <li><a  href="/profile">
           <span   class="icon"><ion-icon   name="person-circle-outline"></ion-icon></span>
             <span   class="title">Profile</span>
          </a></li>
          <li><a  href="/addcomplaint">
           <span   class="icon"><ion-icon   name="document-text-outline"></ion-icon></span>
             <span   class="title">Lodge   Complaint</span>
           </a></li>
          <li><a  href="/comphistory">
           <span   class="icon"><ion-icon   name="chatbubbles-outline"></ion-icon></span>
            <span   class="title">Complaint   History</span>
           </a></li>
          <li><a  href="/logout  ">
           <span   class="icon"><ion-icon   name="log-out-outline"></ion-icon></span>
             <span   class="title">logout</span>
          </a></li>
       </ul>
    </div>
```

```
    <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
      <script nomodule src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
      <script>
        let navigation = document.querySelector('.navigation');let
        toggle = document.querySelector('.toggle');
        toggle.onclick = function(){
        navigation.classList.toggle('active')
        }
      </script>
   <div class="cls">
      <h1>PROFILE</h1>
   </div>
   <div class="vj">
     <img src="/static/img8.png" alt="">
   </div>
  <div class="wrapper">
    <div class="buttons"><br>
      <p>Name :</p><div class="name">{{ fullname }}</div><br>
      <p>Username :</p><div class="name">{{ username }}</div><br>
      <p>Email : </p><div class="name">{{ email }}</div><br>
      <p>State :</p><div class="name">{{ address }}</div><br>
      <div class="links">
        <a href="/editprofile">Edit profile</a>
      </div>

   </div>

  </div>
  </div>
</body>
</html>
```

Userlodgecomplaints.html:

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>lodgecomplaint</title>
        <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
        <link rel="stylesheet" type="text/css" href="css/montserrat-font.css">
        <link rel="stylesheet" type="text/css" href="fonts/material-design-iconic-font/css/material-design-iconic-font.min.css">
  <link rel="stylesheet" href="/static/complaint.css"/>
</head>
<body class="form-v10">
  <link rel="stylesheet" href="/static/userd.css">
  <div class="navigation">
    <div class="toggle"></div>
     <ul>
       <li><a href="/home">
        <span class="icon"><ion-icon name="home-outline"></ion-icon></span>
          <span class="title">Dashboard</span>
        </a></li>
       <li><a href="/profile">
        <span class="icon"><ion-icon name="person-circle-outline"></ion-icon></span>
          <span class="title">Profile</span>
       </a></li>
       <li><a href="/addcomplaint">
        <span class="icon"><ion-icon name="document-text-outline"></ion-icon></span>
          <span class="title">Lodge Complaint</span>
        </a></li>
       <li><a href="/comphistory">
```

```
        <span class="icon"><ion-icon name="chatbubbles-outline"></ion-icon></span>
         <span class="title">Complaint History</span>
        </a></li>
       <li><a href="/logout ">
        <span class="icon"><ion-icon name="log-out-outline"></ion-icon></span>
          <span class="title">logout</span>
       </a></li>
     </ul>
   </div>
   <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
    <script nomodule
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
    <script>
     let navigation = document.querySelector('.navigation');let
     toggle = document.querySelector('.toggle');
     toggle.onclick = function(){
     navigation.classList.toggle('active')
     }
    </script>
    <div class="vj2">
     <img src="/static/img1.png" alt="">
    </div>
    <div class="vj1">
     <img src="/static/img2.png" alt="">
    </div>

      <div class="page-content">
                <form class="form-detail" action="#" method="post" id="myform">
                    <div class="form-left">
                        <h2 style="color: black;">Lodge Complaint</h2>
                        <div class="form-row-3">
          <div class="form-row name">
```

```
 <select name="selectcategory" required id="selectcategory">
    <option class="option" value="selectcategory">Select Category</option>
    <option class="option" value="E-commerce">E-commerce</option>
    <option class="option" value="General">General</option>
    <option class="option" value="Other">Other</option>
  </select>
</div>
<div class="form-row name">
  <input type="date" name="dt" id="dt">
</div>

  <div class="form-row name">

    <input type="text" name="phone" class="phone"
           id="phone"placeholder="Phone Number" required>
  </div>

  <div class="form-row name">
    <select name="state" id="state">
      <option value="state">District</option>
      <option value="Tirunelveli">Tirunelveli</option>
      <option value="Chennai">Chennai</option>
      <option value="Madurai">Madurai</option>
      <option value="Coimbatore">Coimbatore</option>
      <option value="Other">Other</option>
    </select>
  </div>

  <div class="form-row name">
    <input type="text" name="email" id="email" class="input-text" required
pattern="[^@]+@[^@]+.[a-zA-Z]{2,6}" placeholder="email">
  </div>
```

```
                    </div>
                </div>
                <div class="form-right">
                    <h2>Complaint Details</h2>
                    <div class="wrapper">
                        <textarea rows="15" cols="60" placeholder="Enter
the complaint details" name="complaint" required id="complaint"></textarea>
                    </div>
        <span  class="text"style="color:  black;font-weight:bold;">{{a}}</span>

                    <div class="form-row-last">
            <button  class="button-85"  role="button">Register  Complaint</button>
                    </div>
                </div>
            </form>

        </div>
    </div>

</body>
</html>
```

usercomphist.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>complaint  history</title>
    <link  rel="stylesheet"  href="/static/userd.css">
    <link  rel="stylesheet"  href="/static/admind.css">
```

```
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"/>
 </head>
 <body>
 <div class="navigation">
  <div class="toggle"></div>
   <ul>
     <li><a href="/home">
      <span class="icon"><ion-icon name="home-outline"></ion-icon></span>
       <span class="title">Dashboard</span>
      </a></li>
     <li><a href="/profile">
      <span class="icon"><ion-icon name="person-circle-outline"></ion-icon></span>
       <span class="title">Profile</span>
     </a></li>
     <li><a href="/addcomplaint">
      <span class="icon"><ion-icon name="document-text-outline"></ion-icon></span>
       <span class="title">Lodge Complaint</span>
      </a></li>
     <li><a href="/comphistory">
      <span class="icon"><ion-icon name="chatbubbles-outline"></ion-icon></span>
       <span class="title">Complaint History</span>
      </a></li>
     <li><a href="/logout ">
      <span class="icon"><ion-icon name="log-out-outline"></ion-icon></span>
       <span class="title">logout</span>
     </a></li>
   </ul>
 </div>
```

```
</header>
 <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
 <script nomodule src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
 <script>
  let navigation = document.querySelector('.navigation');let
  toggle = document.querySelector('.toggle'); toggle.onclick
  = function(){ navigation.classList.toggle('active')
  }
 </script>
  <div class="wrapper">
    <table>

      <thead>
          <th>ComplaintNumber</th>
          <th>Category</th>
          <th>Date</th>
          <th>Phone Number</th>
          <th>State</th>
          <th>Email</th>
          <th>Complaint Type</th>
          <th>Status</th>
        </thead>

        <tbody>
       {% for row in complaints: %}
       <tr>
          <td>{{ row.id }}</td>
          <td>{{ row.selectcategory }}</td>
          <td>{{ row.dt}}</td>
          <td>{{ row.phone }}</td>
          <td>{{ row.state}}</td>
```

```
            <td>{{ row.email }}</td>
            <td>{{ row.complaint }}</td>
            <td>{{ row.status}}</td>
        </tr>
        {% endfor %}
        </tbody>

    </div>
</body>
</html>
```

admin.html:

```
<!DOCl'YPE html>
<html lang="en">
<head>
    <meta chaíset="Ul'F-8">
    <title>Admin</title>
    <link
híef="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600;7
00;900&display=swap" íel="stylesheet">
    <link híef="../static/admin.css" íel="stylesheet">
    <link íel="shoítcut icon" type="image/jpg" híef="../static/img3.jpg">
</head>
<body>
    <div class="contact-foím">
        <img alt="" class="avataí"
síc="https://i.postimg.cc/zDyt7KCv/a1.jpg">
```

```
            <h2>Admin login</h2>
            <foím action="/adminlog" method="POSl">
                <p>Email</p><input placeholdeí="Enteí Email"
type="email" name="email" íequiíed>
                <p>Passwoíd</p><input placeholdeí="Enteí Passwoíd"
type="passwoíd" name="passwoíd"íequiíed>
                <p><input type="checkbox">Remembeí Me</p>
                <input type="submit" value="Sign in">
                {{a}}
            </foím>
        </div>
</body>
</html>
```

admin dashboard.html:

```
<!DOCl'YPE html>
<html lang="en" dií="ltí">
  <head>
    <meta chaíset="utf-8">
    <title>admin dashboaíd</title>
    <link íel="stylesheet" híef="../static/admind.css">
    <link íel="stylesheet" híef="https://cdnjs.cloudflaíe.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"/>
  </head>
    <body>
       <input type="checkbox" id="check">
```

```
        <label foí="check">
        <i class="fas fa-baís" id="btn"></i>
        <i class="fas fa-times" id="cancel"></i>
        </label>
        <div class="sidebaí">
            <headeí>Dashboaíd</headeí>
            <ul>
                <li><a híef="#"><i class="fas fa-qícode"></i>Dashboaíd</a></li>
                <li><a híef="/agent"><i class="fas fa-link"></i>Agent Cíeation</a></li>
                <li><a híef="/adcomplainthist"><i class="fas fa-calendaí-week"></i>Complaint histoíy</a></li>
                <li><a híef="/loggout"><i></i>Logout</a></li>
            </ul>
        </div>
    </body>
</html>
```

admincomphist.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>admin complaint history</title>
    <link rel="stylesheet" href="/static/admind.css">
    <link rel="stylesheet" href="/static/userd.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"/>
```

```
</head>
 <body>
  <div class="navigation">
   <div class="toggle"></div>
    <ul>
      <li><a href="/adminpage">
       <span class="icon"><ion-icon name="home-outline"></ion-icon></span>
        <span class="title">Dashboard</span>
       </a></li>
       <li><a href="/agentcreate">
        <span class="icon"><ion-icon name="person-circle-outline"></ion-icon></span>
          <span class="title">Agent Creation</span>
       </a></li>
      <li><a href="/adcomplainthist">
       <span class="icon"><ion-icon name="chatbubbles-outline"></ion-icon></span>
         <span class="title">Complaint Details</span>
       </a></li>
      <li><a href="/loggout ">
       <span class="icon"><ion-icon name="log-out-outline"></ion-icon></span>
        <span class="title">logout</span>
      </a></li>
    </ul>
  </div>
  <script type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
   <script nomodule
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
   <script>
    let navigation = document.querySelector('.navigation');let
    toggle = document.querySelector('.toggle');
    toggle.onclick = function(){
    navigation.classList.toggle('active')
    }
```

```
</script>
<div  class="wrapper">
<div  class="history">
  <span class="tab" style="position: relative;top: -
  250px;
  left: -400px;
  padding: 10px; width: 940px;
  font-weight: bold;">{{a}}</span>

  <div class="tab" style="position:  relative;
  border: 1px solid black;
  top: -250px;
  left: -400px;
  padding:  10px;  width:  940px;">

     <span style="margin-left: 10px;
   text-align:  left;">{{  row.0  }}</span>
     <span style="margin-left: 10px;
   text-align:  left;">{{  row.2  }}</span>
     <span  style="margin-left:  20px;
     text-align:  left;">{{  row.3  }}</span>
     <span  style="margin-left:  20px;
     text-align:  left;">{{  row.4  }}</span>
     <span  style="margin-left:  20px;
     text-align:  left;">{{  row.5  }}</span>
     <span  style="margin-left:  30px;
     text-align:  left;">{{  row.6  }}</span>
     <span  style="margin-left:  20px;
     text-align:  left;">{{  row.7  }}</span>
     <span  style="margin-left:  20px;
     text-align:  left;">{{  row.8  }}</span>
  </div>
```

```
  </div>
        <table>
        <thead>
            <th>ComplaintNumber</th>
            <th>Category</th>
            <th>Date</th>
            <th>Phone Number</th>
            <th>State</th>
            <th>Email</th>
            <th>Complaint Type</th>
            <th>Status</th>
        </thead>
        <!--<tr>
            <td>{{ row.0 }}</td>
            <td>{{ row.1 }}</td>
            <td>{{ row.2 }}</td>
            <td>{{ row.3 }}</td>
            <td>{{ row.4 }}</td>
            <td>{{ row.5 }}</td>
            <td>{{ row.7 }}</td>
            <td><a href="/solved/{{ row.0 }} "><span style="margin-left: 20px;
               text-align: left;">{{ row.8 }}</span></a></tr>
        </tr>-->
        </table>
     </div>
   </body>
</html>
```

adminagent.html:

```
<!DOCTYPE html>

<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>agent creation </title>
   <link rel="stylesheet" href="/static/agentcreation.css">
   <link rel="stylesheet" href="/static/userd.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"/>
</head>
<body>
   <div class="navigation">
      <div class="toggle"></div>
```

```
<ul>
 <li><a  href="/adminpage">
  <span   class="icon"><ion-icon   name="home-outline"></ion-icon></span>
   <span   class="title">Dashboard</span>
  </a></li>
 <li><a  href="/agentcreate">
  <span   class="icon"><ion-icon   name="person-circle-outline"></ion-icon></span>
   <span  class="title">Agent  Creation</span>
 </a></li>
 <li><a  href="/adcomplainthist">
  <span   class="icon"><ion-icon   name="chatbubbles-outline"></ion-icon></span>
   <span  class="title">Complaint  History</span>
  </a></li>
 <li><a href="/loggout ">
  <span   class="icon"><ion-icon   name="log-out-outline"></ion-icon></span>
   <span  class="title">logout</span>
 </a></li>
</ul>
</div>
<script                                                                                    type="module"
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
 <script                                                                                      nomodule
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
 <script>
  let navigation = document.querySelector('.navigation');let
  toggle = document.querySelector('.toggle');
  toggle.onclick = function(){
  navigation.classList.toggle('active')
  }
 </script>
 <div class="vj">
  <img  src="/static/img8.png"  alt="">
 </div>
```

```
<div class="wrapper">
 <div class="form">
    <h2> Agent Creation</h2>
    <form action="/agentdata" method="POST">
      <p style="padding-right: 40px;">Name</p>
      <div class="name">
      <input placeholder="Enter your name" type="text" name="username" required>
      </div>
      <p>Password</p>
      <div class="name">
      <input placeholder="Enter password" type="password" name="password" required>
      <span class="alert {{indicator}}">{{a}}</span>
       </div>
       <button class="button-85" role="button">CREATE</button>


    </form>
  </div>

  </div>

 </div>
 </div>
</body>
</html>
```

admin.html:

```
<!DOCl’YPE html>
<html lang="en">
<head>
        <meta chaíset="Ul’F-8">
        <title>Agent    login</title>
        <link
hief="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600;700;900&disp lay=swap"
iel="stylesheet">
        <link hief="../static/agent.css" iel="stylesheet">
</head>
<body>
        <div class="foim">
                <h2> Agent login</h2>
                <foim action="/agentdata" method="POSl’">
                        <p>Useiname</p><input placeholdeí="Enteí youí name" type="text"
name='useiname'>
                        <p>Passwoíd</p><input placeholdeí="Enteí passwoíd"
type="passwoíd" name='passwoíd'>
                        <input type="submit" value="Sign in">
                </foim>
      <div  class="links">
           <p>click heíe foí dashboaíd <a hief="/adminpage"> Click heíe</a></p>
     </div>
                {{a}}
        </div>
</body>
</html>
```

app.py:

```
from flask import Flask, render_template, request, redirect, url_for, sessionimport
ibm_db
import re
from flask_mail import Mail, Message

app = Flask(_name_)
mail = Mail(app)

app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'customercare.in2022@gmail.com'
app.config['MAIL_PASSWORD'] = 'dutxqvejaxonmmew'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)


app.secret_key='a'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;Security=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=fqt70379;PWD=KdUUJ3RnMlMqAimg;",'','')

@app.route('/')
def home():
    return  render_template('/home.html')

@app.route('/registertemp',methods=["POST","GET"])
def registertemp():
    return  render_template("register.html")
```

```
@app.route('/uploaddata',methods =['GET','POST'])
def register():
    msg = ''
    if request.method == 'POST':
        firstname = request.form['firstname']
        lastname = request.form['lastname']
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        address = request.form['address']
        stmt = ibm_db.prepare(conn, 'SELECT * FROM users WHERE username = ?')
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)if
        account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'^[A-Za-z0-9_.-]*$', username):
            msg = 'name must contain only characters and numbers !'
        else:
            prep_stmt = ibm_db.prepare(conn,'INSERT INTO users(firstname, lastname, username,
email, password, address) VALUES(?, ?, ?, ?, ?, ?)')
            ibm_db.bind_param(prep_stmt, 1, firstname)
            ibm_db.bind_param(prep_stmt, 2, lastname)
            ibm_db.bind_param(prep_stmt, 3, username)
            ibm_db.bind_param(prep_stmt, 4, email)
            ibm_db.bind_param(prep_stmt, 5, password)
            ibm_db.bind_param(prep_stmt, 6, address)
            ibm_db.execute(prep_stmt)
            msg = 'Dear % s You have successfully registered!'%(username)
            msg1 = Message('Hello', sender = 'customercare.in2022@gmail.com', recipients = [email])
            msg1.body = 'Hello'+ username +''',
```

```
                Account creation in customer care registry was successful.
                For raising Complaints, login with your username and password.Thank
                You)'''
            mail.send(msg1)
        return  render_template('register.html',a  = msg,indicator="success")else:
        msg = 'Please fill the form!'
    return   render_template('register.html',a   =   msg,   indicator='failure')

@app.route('/login',methods=["POST","GET"])
def login():
    return   render_template("login.html")

@app.route('/logindata',methods=["POST","GET"])
def logindata():
    global userid
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        stmt = ibm_db.prepare(conn,'SELECT * FROM users WHERE username = ? AND password =
?')
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account   =   ibm_db.fetch_tuple(stmt)   if
        account:
            session['id']  =  account[0]userid
            =  account[0]
            session['username'] = account[1] return
            redirect(url_for('dashboard'))
        else:
            msg = 'Incorrect username / password !'
```

```
        return render_template('login.html', b = msg, indicator="failure")

@app.route('/home')
def dashboard():
   if 'id' in session: uid
      = session['id']
      stmt = ibm_db.prepare(conn, 'SELECT * FROM users WHERE id = ?')
      ibm_db.bind_param(stmt, 1, uid)
      ibm_db.execute(stmt)
      ibm_db.fetch_tuple(stmt)
      username = session['username']
      return render_template('user dashboard.html', name = username)
   return render_template('user dashboard.html')

@app.route('/profile',methods=["POST","GET"])
def profile():
   if 'id' in session: uid
      = session['id']
      stmt = ibm_db.prepare(conn, 'SELECT * FROM users WHERE id = ?')
      ibm_db.bind_param(stmt, 1, uid)
      ibm_db.execute(stmt)
      acc = ibm_db.fetch_tuple(stmt)
      return
render_template('userprofile.html',fullname=acc[1]+acc[2],username=acc[3],email=acc[4],addres
s=acc[6])
   return   render_template('userprofile.html')

@app.route('/edit')
def edit():
   return   render_template('editprofile.html')

@app.route('/editprofile', methods=["POST","GET"])
def editprofile():
```

```
    if request.method == 'POST' :
        msg = ''
        if 'id' in session: uid
            = session['id']
            firstname = request.form['firstname']
            lastname = request.form['lastname']
            username = request.form['username']
            email =  request.form['email']
            address = request.form['address']
            stmt = ibm_db.prepare(conn,"UPDATE users SET firstname = ?, lastname = ?, username =
?, email = ?, address = ? WHERE id = ?")
            ibm_db.bind_param(stmt, 1, firstname)
            ibm_db.bind_param(stmt,  2,  lastname)
            ibm_db.bind_param(stmt,  3,  username)
            ibm_db.bind_param(stmt,  4,  email)
            ibm_db.bind_param(stmt,  5,  address)
            ibm_db.bind_param(stmt, 6, uid)ibm_db.execute(stmt)
            msg = 'Profile Updated Successfully!'
            return render_template('editprofile.html',a = msg)return
    render_template('editprofile.html')


@app.route('/addcomplaint',methods=["POST","GET"])
def addcomplaint():
    if  request.method == "POST":
        if 'id' in session:
            msg = ''
            uid=session['id']
            selectcategory = request.form['selectcategory']
            date = request.form['dt']
            phone = request.form['phone']
            state = request.form['state']
```

```
email = request.form['email']
complaint = request.form.get('complaint')
stmt = ibm_db.prepare(conn,"SELECT * FROM users WHERE id = ?")
ibm_db.bind_param(stmt, 1, uid)
ibm_db.execute(stmt)
acc = ibm_db.fetch_tuple(stmt)email
= acc[4]
prep_stmt = ibm_db.prepare(conn,'INSERT INTO complaintdetails(uid, selectcategory, dt ,
phone, state, email, complaint, status) VALUES (?, ?, ?, ?, ?, ?, ?, ?)')
ibm_db.bind_param(prep_stmt, 1, uid)
ibm_db.bind_param(prep_stmt, 2, selectcategory)
ibm_db.bind_param(prep_stmt, 3, date)
ibm_db.bind_param(prep_stmt, 4, phone)
ibm_db.bind_param(prep_stmt, 5, state)
ibm_db.bind_param(prep_stmt, 6, email)
ibm_db.bind_param(prep_stmt, 7, complaint)
ibm_db.bind_param(prep_stmt, 8, 'pending')
ibm_db.execute(prep_stmt)
msg = 'You have successfully registered your complaint'
TEXT1 = Message('Hello', sender = 'customercare.in2022@gmail.com', recipients =
[email])
TEXT1.body = """ Dear """+acc[1]+" "+acc[2]+"""
        Your complaint has been succesfully registered...!!
        One of our agent is assigned , will surely resolve your complaint as soon as possible
        Please check the complaint details in dashboard for status of complaint."""
mail.send(TEXT1)
se = ibm_db.exec_immediate(conn, 'SELECT * FROM agentinfo ORDER BY RAND() LIMIT
1')
agmail = ibm_db.fetch_tuple(se)
agentemail = agmail[1]
TEXT = Message('Hello', sender = 'customercare.in2022@gmail.com', recipients =
[agentemail])
TEXT.body = """New Complaint from """+acc[1]+" "+acc[2]+"""
```

```
            Complaint Description :
            """+complaint+"""
        ""mail.send(TEXT)
        return render_template('userlodgecomp.html',a = msg)
    return render_template('userlodgecomp.html')


@app.route('/view',methods=["POST","GET"])
def view():
    if 'id' in session:
        return    render_template('usercomphist.html')

@app.route('/comphistory',methods=['POST','GET'])
def compview():
    if 'id' in session:
        uid=session['id']
        stmt = ibm_db.prepare(conn, 'SELECT * FROM users WHERE id = ?
        ')ibm_db.bind_param(stmt, 1, uid)
        ibm_db.execute(stmt)

        stmt1 = ibm_db.prepare(conn, 'SELECT id, selectcategory, dt, phone, state, email, complaint,
status FROM complaintdetails WHERE uid = ?')
        ibm_db.bind_param(stmt1, 1, uid)ibm_db.execute(stmt1)
        comp = ibm_db.fetch_assoc(stmt1)
        data = []
        while comp != False:
            data.append(comp)
            comp = ibm_db.fetch_assoc(stmt1)
        return render_template('usercomphist.html', complaints =  data)
    return render_template('usercomphist.html')

@app.route('/admin')
```

```
def admin():
    return    render_template('admin.html')

@app.route('/adminpage')
def adminpage():
    return    render_template('admin    dashboard.html')

@app.route('/adminlog',methods=["POST","GET"])
def adminlog():
    msg = ''
    email = request.form['email']
    password = request.form['password']
    stmt = ibm_db.prepare(conn, 'SELECT * FROM admininfo  WHERE email = ?  and password = ?')
    ibm_db.bind_param(stmt,1, email)
    ibm_db.bind_param(stmt,2, password)ibm_db.execute(stmt)
    logged = ibm_db.fetch_assoc(stmt)
    if(logged):
        msg = 'successfully loggedin'
        return render_template("admin dashboard.html",a=msg)
    else:
        return    render_template("admin.html",a="Incorrect    email/password")

@app.route('/adcomplainthist',methods=['POST','GET'])
def adcomplainthist():
        stmt1 = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails')
        ibm_db.execute(stmt1)
        comp = ibm_db.fetch_assoc(stmt1)
        data = []
        while comp != False:
            data.append(comp)
            comp = ibm_db.fetch_assoc(stmt1)
        return   render_template('admincomphist.html',   complaints   =   data)
```

```
@app.route('/agcomplainthist',methods=['POST','GET'])
def agcomplainthist():
        stmt1 = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails')
        ibm_db.execute(stmt1)
        comp = ibm_db.fetch_assoc(stmt1)
        data = []
        while comp != False:
           data.append(comp)
           comp = ibm_db.fetch_assoc(stmt1)
        return   render_template('admincomphist.html',   complaints   =   data)

@app.route('/agentcreate',methods=["POST","GET"])
def agentcreate():
    return    render_template('adminagent.html')

@app.route('/agentdata',methods=["POST","GET"])
def agentdata():
    msg = ''
    username = request.form['username']
    password = request.form['password']
    stmt = ibm_db.prepare(conn,'INSERT INTO agentinfo(username, password) VALUES (?, ?)')
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt,  2,  password)ibm_db.execute(stmt)
    msg = 'Agent has been created successfully' return
    render_template('adminagent.html',a  =  msg)

@app.route('/agent',methods=["POST","GET"])
def agent():
    return render_template('agent.html')

@app.route('/agentlogin',methods=["POST","GET"])
```

```
def  agentlogin():msg
   = ''
   email = request.form['username']
   password = request.form['password']
   stmt = ibm_db.prepare(conn, 'SELECT * FROM agentinfo  WHERE username = ?  and password
= ?')
   ibm_db.bind_param(stmt,1, email)
   ibm_db.bind_param(stmt,2, password)ibm_db.execute(stmt)
   logged = ibm_db.fetch_assoc(stmt)
   if(logged):
      msg = 'successfully loggedin'
      return     render_template("agentdashboard.html",a=msg)else:
      return    render_template("agent.html",a="Incorrect    email/password")

@app.route('/agentpage') def
agentpage():
   return     render_template('agentdashboard.html')

@app.route('/loggout')
def loggout():
   if 'id ' in session:
      session.pop('id',None)
      session.pop('email',None)
      session.pop('password',None)
   return   redirect(url_for('home'))

@app.route('/logout')
def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   session.pop('username',  None)
```

```
    return   redirect(url_for('login'))


@app.route('/solved/<no>')
def solved(no):
    i = no
    stmt = ibm_db.prepare(conn, "UPDATE complaintdetails SET status = ? WHERE id = ?")
    ibm_db.bind_param(stmt, 1, 'solved')
    ibm_db.bind_param(stmt, 2, i)
    ibm_db.execute(stmt)
    se = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails where id = ?')
    ibm_db.bind_param(se, 1, i)
    ibm_db.execute(se)
    em = ibm_db.fetch_tuple(se)
    email = em[6]
    print(email)
    TEXT = Message('Hello', sender = 'customercare.in2022@gmail.com', recipients = [email])
    TEXT.body = """Your Complaint has been solved."""
    mail.send(TEXT)
    return  render_template('solve.html',  a  ="Complaint  Resolved"  )


if__name__== '_main_':
    app.debug=True
    app.run(host='0.0.0.0',port=8080)
```

GitHub & Project Demo Link

http://159.122.177.141:32498/

https://github.com/IBM-EPBL/IBM-Project-30173-1660141368