

CUSTOMER CARE REGISTRY

PROJECT REPORT

Submitted by:

TEAM ID:PNT2022TMID49581

P.Anantha Krishnan(950019104001)

V.Maharaja(950019104023)

A.Manickavasagam(950019104026)

S.Mariselvam(950019104028)

ANNA UNIVERSITY REGIONAL CAMPUS-TIRUNELVELI

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

AUGUST 2022-NOVEMBER 2022

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
 - IP address

1. INTRODUCTION

1.1 Project Overview:

Customer Complaint is important information reflecting customers sound and is a primary measure of customer dissatisfaction. An Effective and Efficient response to these complaints is an essential index of organization's performance. The presented model for the Customer Care Registry will have the ability to minimize customers' dissatisfaction and on the other hand it can encourage customers to participate in controlling the quality of the service provided. It also reduces recurring complaints. It will Improve standards of service to the community. Eventually, it raises standards of administrative decision-making. In today's competitive environment, product and service innovations are re-defining accepted levels of performance.

1.2 Purpose:

While customer service and support teams interact with customers on an as-needed basis -- using various chat channels such as phone, website chat applications and social media messaging -- those moments of contact are critical to keeping an organization running successfully. If customer service teams provide a positive customer experience and customer support teams successfully assist customers with questions or problems, then those customers are likely to be happy and do business again with the organization. Satisfied customers may also choose to leave positive reviews or tell friends and family about an organization with good customer service and support, which can bring in more customers to an organization.

2.LITERATURE SURVERY

2.1 Existing Problem:

1. Help Desk:

Help desk software is a tool that serves a wide range of customer support activities. You can organize messages, give assistance, and exchange information with customers at a single point of contact. The help desk uses tickets for communication that's why it's also known as a ticketing system.

Advantages:

- Improves the security of your IT network.
- Improves the service desk as well as the products and services offered by the organization.
- Helps you ensure that the service quality is as defined in the SLA.

Disadvantage:

- Focuses on improving customer service efficiency over excellence.
- Dishevels flow of communication.
- Does not focus on building long term customer relationships.

2.Fresh Desk:

Online cloud-based customer service software providing helpdesk support with all smart automations to get things done faster.

Advantage:

- Modern and user-friendly design.
- A feature-rich customer service platform.
- Freshdesk marketplace with many integrations (Salesforce, Slack, etc.).
- The free plan available.

Disadvantage:

- Limited automation and rules (plus, hard to configure).
- Poor dashboard (Note: Freshdesk Analytics is in the beta version. The current dashboard offers fundamental features only).
- The dark mode is native Sometimes it seems the support team of Freshdesk is not easily reachable.

2.2 References:

<https://freshdesk.com/>

<https://www.helpdesk.com/>

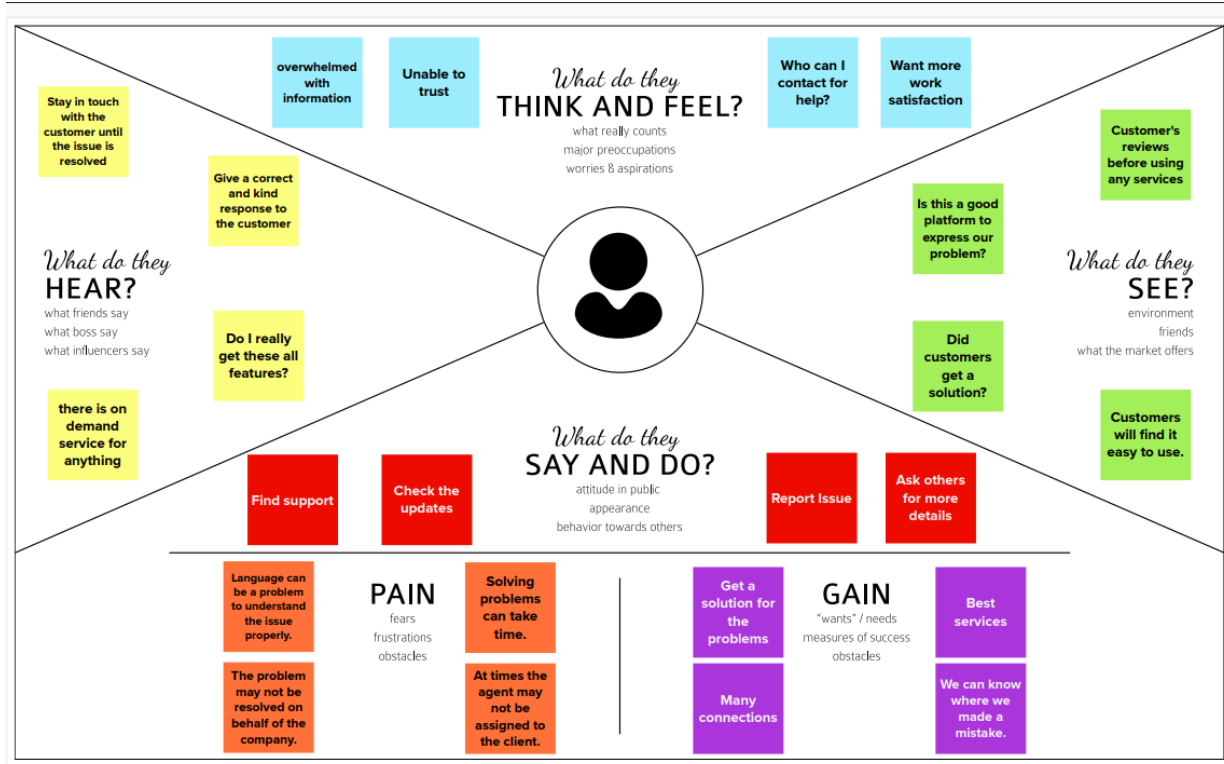
2.3 Problem Statement Definition:

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.

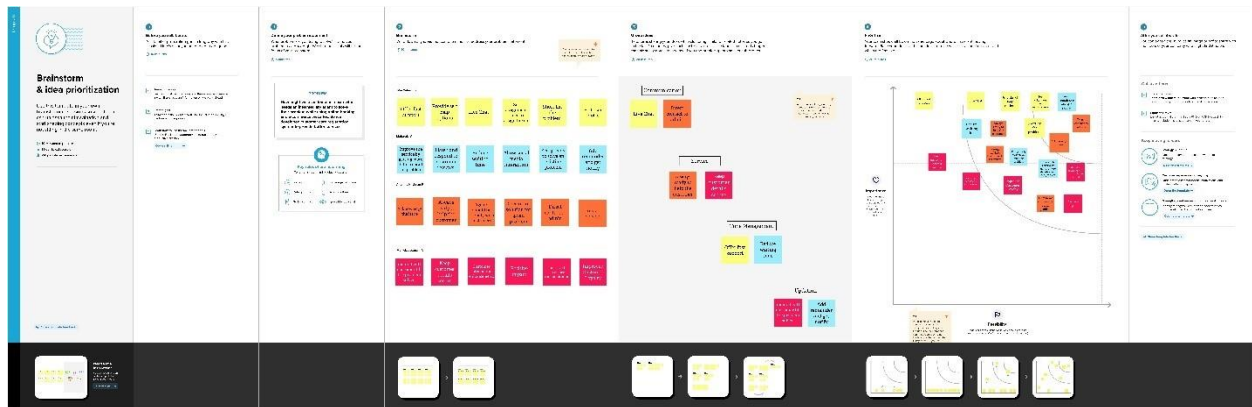
Who does the problem affect?	Frequent online users.
What are the boundaries of the problem?	IT sector, e-Commerce, Online Booking System.
What is the issue?	Stay in touch with the customer until the issue is resolved An Agent will be assigned to the Customer to solve the problem. Based on complaint type 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.IDEAATION & 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: PNT2022TMD49581

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS Who is your customer? 1) Customers who are not able to solve them Own complaints of what they are facing. 2) Customers who do not know the solution of their questions they get.	6. CUSTOMER CONSTRAINTS CC What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices. 1) This application will be supported by almost all the devices. 2) The solution we propose will have an alert via email feature, If there is any update on user complaint.	5. AVAILABLE SOLUTIONS AS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notepad. 1) By reading the guidelines properly. 2) Offer a solution and give options whenever possible. 3) Address to issue within the company 4) By communicating properly	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one, explore different sides. 1) The application allow the customers to find the solution for their queries. 2) They will be able to categorize their expenses. 3) They will be also given option for the general questions. 4) They also get the free solution where we provide our agents.	9. PROBLEM ROOT CAUSE RC What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations. 1) Lot of customer don't know the guidelines for their problems. 2) Some customers have lack of knowledge 3) Not knowing the answer to a question 4) Not reading the guidelines properly	7. BEHAVIOUR BE What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; Indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace) 1) Make sure he/she reads the guidelines properly. 2) Make sure they find a proper solution for their queries.	
Focus on J&P, tap into BE, understand RC	3. TRIGGERS TR What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news. 1) Customer can know to solve their solution	10. YOUR SOLUTION SL If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour. 1) To design a personal help desk using flask 2) To provide insights on their queries in a graphical way	8. CHANNELS of BEHAVIOUR CH 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. 1) All their data are secured and being updated to cloud storage 2) Make sure they find the best solutions for their complaints.	Focus on BE, tap into RC, understand J&P
	4. EMOTIONS: BEFORE / AFTER EM How do customers feel when they face a problem or a job and afterwards? 1) Customer can get the from the help desk.			

Identify strong TR & EM

Identify strong TR & EM

4.REQUIREMENT ANALYSIS

4.1 Functional & Non-Functional requirements:

Functional Requirements:

Following are the functional requirements of the proposed solution.

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 every user
FR-7	Complaint status	Status of the complaint send via e-mail

Non-functional Requirements:

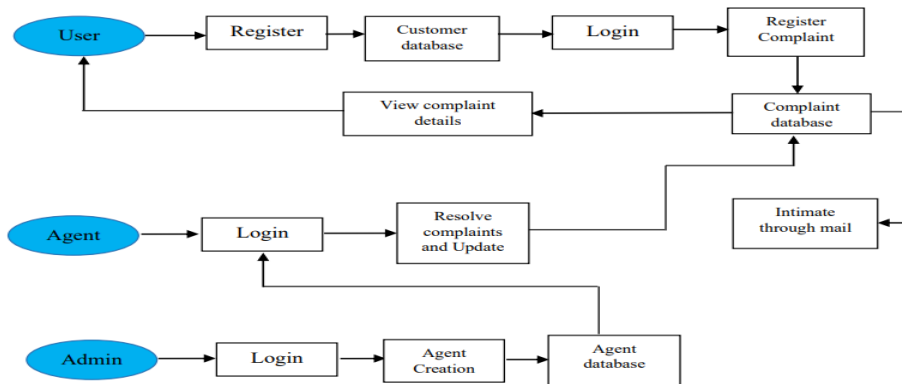
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User can easily interact with the website
NFR-2	Security	The information given by the user will be secure
NFR-3	Reliability	The system will allow the user to contact the agent if the user didn't get the satisfied solution.
NFR-4	Performance	The user interface page will be loaded within few seconds
NFR-5	Availability	New module deployment must not impact front page and main page.
NFR-6	Scalability	The website traffic limit must be scalable.

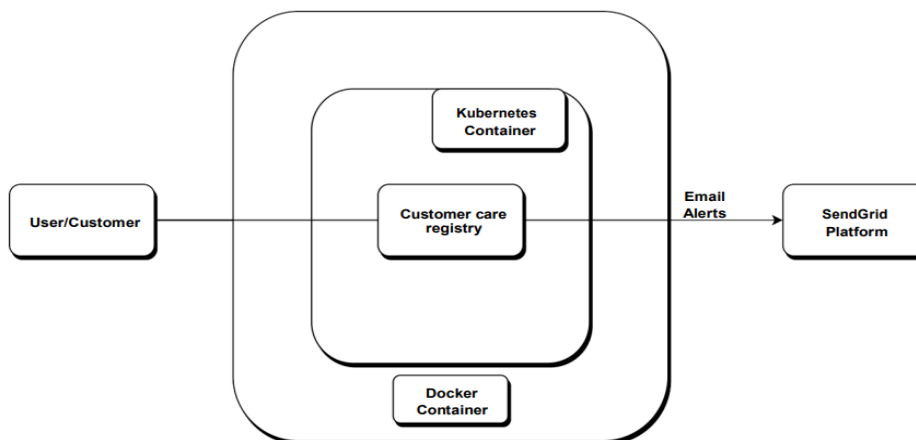
5.PROJECT DESIGN:

5.1 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a 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, and where data is stored.



5.2 Solution & Technical Architecture:



5.3 User Stories:

Use the below template to list all 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 entering my email, password, and confirming my password.	I will be redirected to Email Verification	High	Sprint-1
Customer(Automated User)	Registration	USN-2	As a User, I will Validate the Customer Credentials once after the Email Verification.	I will receive confirmation Message from Administrator	High	Sprint-2
Customer(Client User)	Login	USN-3	As a User, I will Login into the Portal using Login Credentials Provided .	I will be Redirected to the Portal Dashboard Page	Medium	Sprint-2
Customer (ClientUser)	Dashboard	USN-4	As a User, I will book for a ticket from available sections along the Application and Submit the Ticket to the Portal	I will be Issued with a Ticket Applied Message from the Portal.	High	Sprint-3
Customer (AdminUser)	Validation	USN-5	As a User, I will issue with a Suitable Agent to the Customer.	I will send a mail about the agent issued to the website.	High	Sprint-2
Customer(Agent User)	Agent	USN-6	As a User, I will satisfy all the queries to the Customer for all the repetitive responses from the Customers.	I will communicate with a Query.	Medium	Sprint-3
Customer(Server User)	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 the Feedback and issue with a message for queries	High	Sprint-4

Customer(Web site User)	Log out	USN-8	As a User, I will Log out of the website when my Queries are over or else will begin again from the Beginning.	I will Estimate the User Response and React to end the Process.	Low	Sprint-1
-------------------------	---------	-------	--	---	-----	----------

6. PROJECT PLANNING & SCHEDULING:

6.1 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 (Web User)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	2	High	Manikam, Ananthakrishnan
Sprint-1		Login	USN-2	As a customer, I can log into the application by entering correct email and password	1	High	Maharaja
Sprint-1		Dashboard	USN-3	As a customer, I can see all the tickets raised by me and lot more	3	High	Mariselvam
Sprint-2		Complaint raise	USN-4	As a customer, I can create a 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 assigned agent and get my queries clarified	3	High	Manikam, Ananthakrishnan	
Sprint-4		Forgot password	USN-6	As a customer, I can reset my password by this option in case I forgot my 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	Team Members	
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	Mariselvam	
Sprint-3		Dashboard	USN-2	As an agent, I can see all the complaints assigned to me by the admin	3	High	Maharaja	
Sprint-3		Address Column	USN-3	As an agent, I get to have conversations with the customer and clear his/her queries	3	High	Maharaja, Mariselvam	
Sprint-4		Forgot password	USN-4	As an agent, I can reset my password by this option in case I forgot my old password	2	Medium	Maharaja, Mariselvam	

Sprint-1	Admin (Web user)	Login	USN-1	As an admin,I can login to the application by entering correct email and password	1	High	Manikam, Ananthakrishnan	
Sprint-1		Dashboard	USN-2	As an admin, I can see all the complaints raised in the entire system and lot more	3	High	Mariselvam	
Sprint-2		Agent creation	USN-3	As an admin, I can create an agent for clarifying the customer's queries	2	High	Manikam	
Sprint-2		Assigning agent	USN-4	As an admin, I can assign an agent for each complaint created by the customer	3	High	Mariselvam , Ananthakrishnan	
Sprint-4		Forgot password	USN-4	As an admin, I can reset my password by this option in case I forgot my old password	2	Medium	Manikam	Maharaja

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:

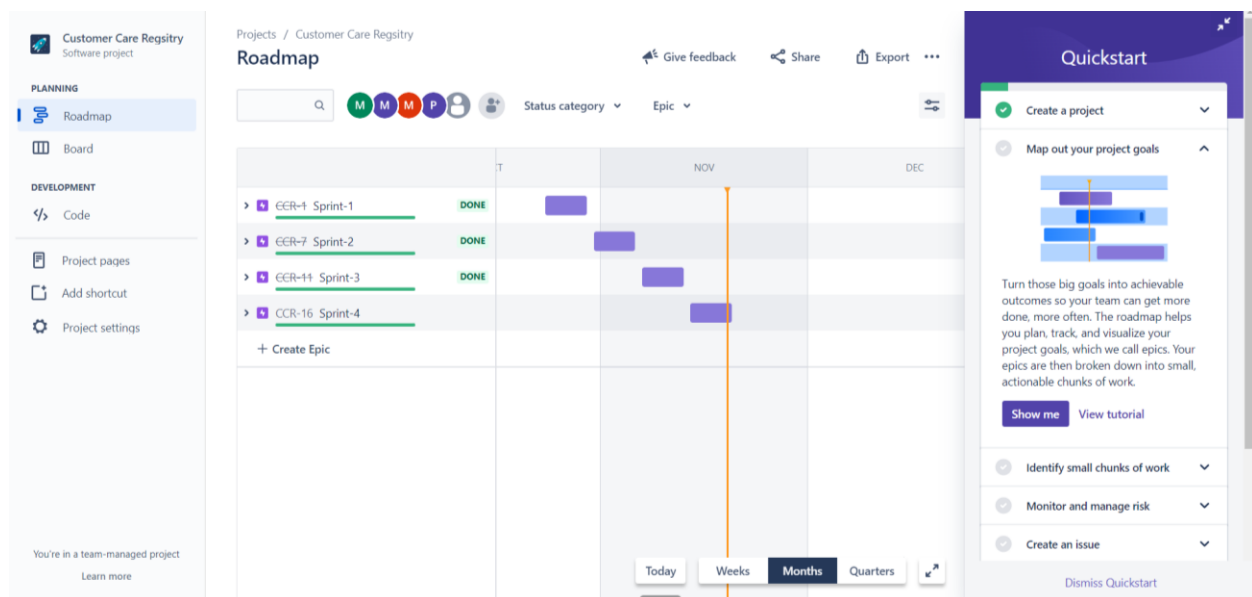
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

6.2 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)	Sprint Release Date (Actual)	
Sprint-1		10	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022	
Sprint-2		7	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022	
Sprint-3		11	6 Days	07 Nov 2022	12 Nov 2022	11	12 Nov 2022	
Sprint-4		8	6 Days	14 Nov 2022	19 Nov 2022	8	19 Nov 2022	

6.3 Reports from JIRA:



7. CODING & SOLUTIONING:

7.1 Feature 1:

User register and login:

User register and login through this web application.

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">
</head>
<body>
  <div class="wrapper">
    
    <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">
            </div>
            <div class="input_wrap">
              <label for="lastname">Last Name</label>
              <input type="text" id="lastname" name="lastname">
            </div>
          </div>
          <div class="input_wrap">
            <label for="username">username</label>
            <input type="text" id="username" name="username">
          </div>

          <div class="input_wrap">
            <label for="email">Email</label>
            <input type="email" id="email" name="email">
          </div>
        </div>
      </form>
    </div>
  </div>
```

```

</div>
<div class="input_wrap">
  <label for="password">Password</label>
  <input type="password" id="password" name="password">
</div>
<div class="input_wrap">
  <label for="address">Address</label>
  <input type="text" id="address" name="address">
</div>

<div class="input_wrap">
  <input type="submit" value="Register Now" class="submit_btn">
</div>
</div>
</form>
<span class="alert {{indicator}}">{{a}}</span>
<div class="links" style="margin-top: 10px;">
  <p style="color: white; height: 2px;" >Already a member then login.. <a href="/login"
style="text-decoration: none; color: white; border-bottom: 1px solid white;border-radius:
0px;">Click here</a></p>
</div>
</div>
</div>
</body>
</html>

```

The screenshot shows a web browser window with the address bar displaying "Not secure | 159.122.177.141:32498/registertemp". The page features a dark blue header with the "Customer Care Registry" logo and navigation buttons: HOME, AGENT LOGIN, ADMIN LOGIN, USER LOGIN, and REGISTER. The main content area is a light blue box titled "REGISTRATION FORM". It contains the following fields and labels:

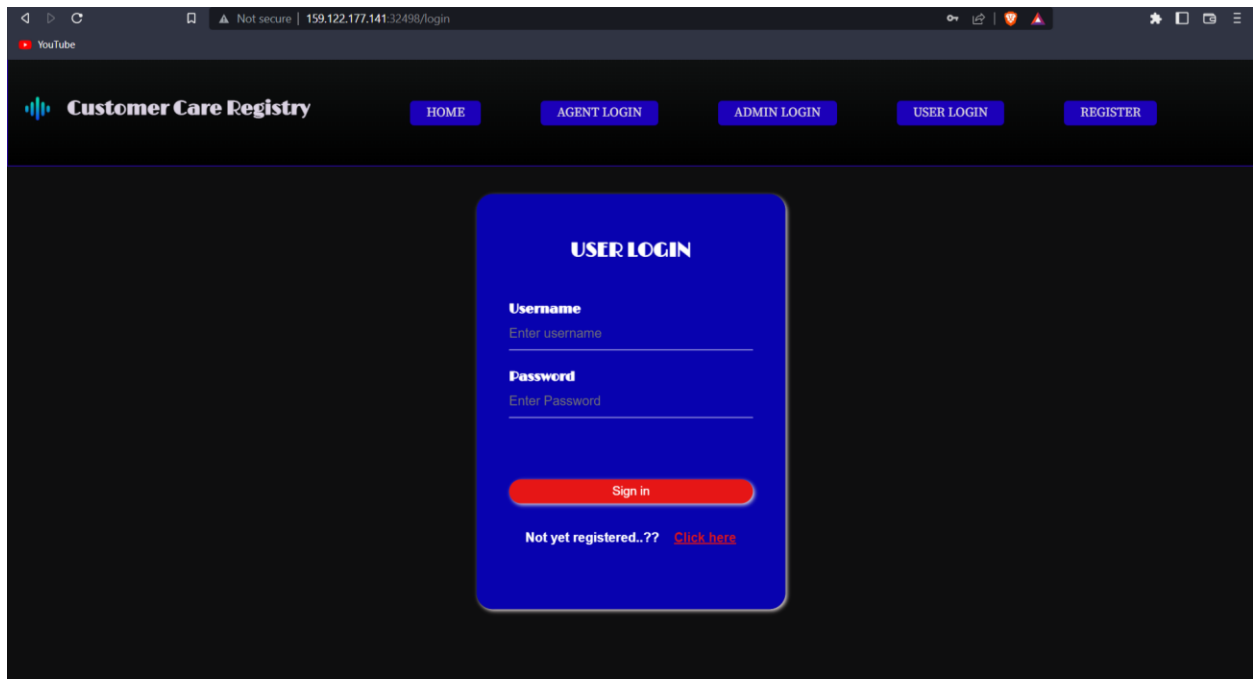
- First Name**: Input field with placeholder "firstname"
- Last Name**: Input field with placeholder "lastname"
- username**: Input field with placeholder "username"
- Email**: Input field with placeholder "email"
- Password**: Input field with placeholder "password"
- State**: Input field with placeholder "state"

Below the form fields is a red button labeled "Register Now". At the bottom of the form box, there is a link: "Already a member then login.. [Click here](#)".

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">
</head>
<body>
  

  <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="/registertemp"> Click here</a></p>
    </div>
  </div>
</body>
</html>
```



7.2 Feature 2:

User lodge complaints and admin create agent.

In this application the user lodge complaints and admin create 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">
    
</div>
<div class="vj1">
    
</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>
```

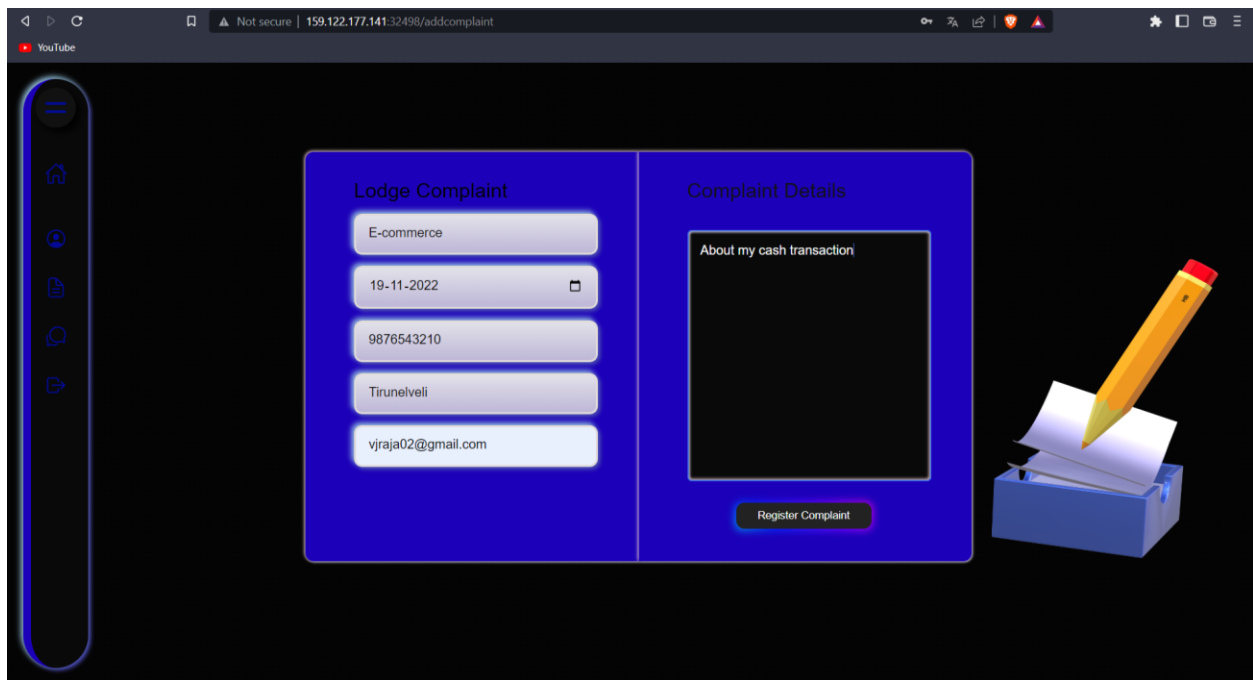
```

        </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="/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="vj">
    

```

</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>

{{ a }}

</div>

<button class="button-85" role="button">CREATE</button>

</form>

</div>

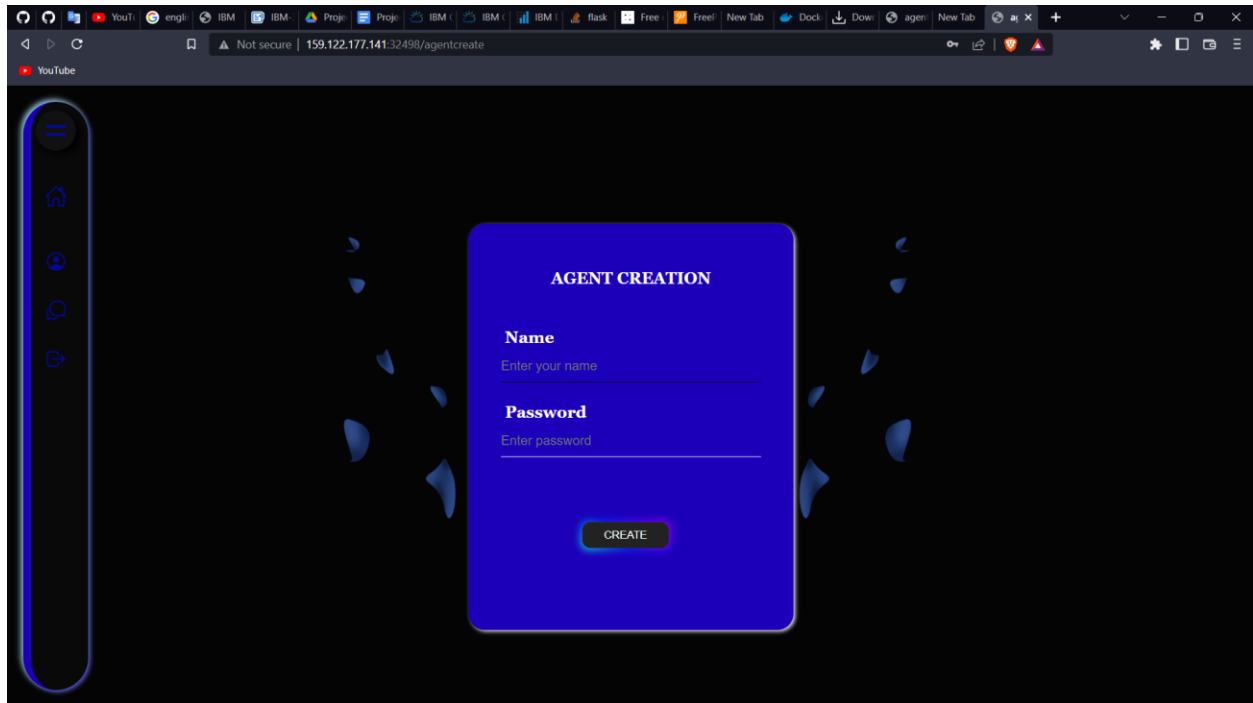
</div>

</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=swap"
```

```
    rel="stylesheet">
```

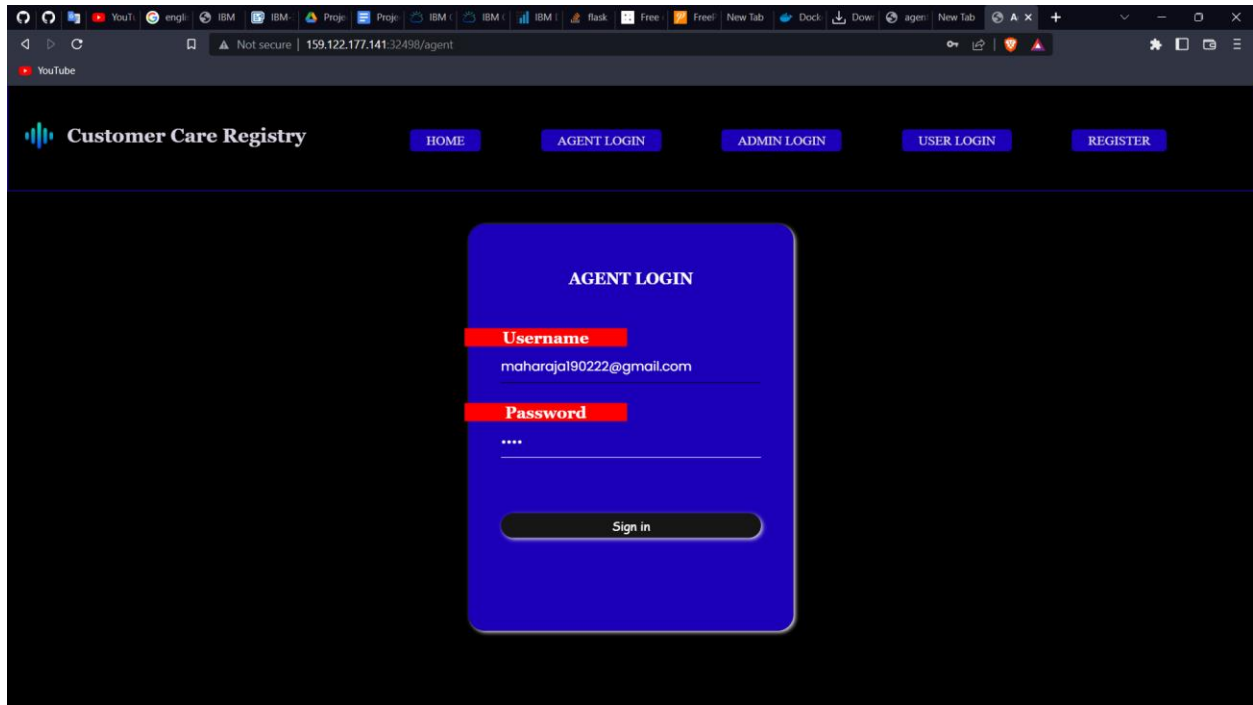
```
    <link href="../static/agent.css" rel="stylesheet">
```

```

</head>
<body>
    <header>
        <div class="rj">
            <div class="img">
                
                <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, session
import 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'^[^\s@]+@[^\s@]+\.[^\s@]+', 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(prepare_stmt, 1, firstname)
    ibm_db.bind_param(prepare_stmt, 2, lastname)
    ibm_db.bind_param(prepare_stmt, 3, username)
    ibm_db.bind_param(prepare_stmt, 4, email)
    ibm_db.bind_param(prepare_stmt, 5, password)
    ibm_db.bind_param(prepare_stmt, 6, address)
    ibm_db.execute(prepare_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],address=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(prepare_stmt, 1, uid)
```

```
            ibm_db.bind_param(prepare_stmt, 2, selectcategory)
```

```
            ibm_db.bind_param(prepare_stmt, 3, date)
```

```
            ibm_db.bind_param(prepare_stmt, 4, phone)
```

```
            ibm_db.bind_param(prepare_stmt, 5, state)
```

```
            ibm_db.bind_param(prepare_stmt, 6, email)
```

```
            ibm_db.bind_param(prepare_stmt, 7, complaint)
```

```
            ibm_db.bind_param(prepare_stmt, 8, 'pending')
```

```
            ibm_db.execute(prepare_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('/logout')
def logout():
    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.""""
```

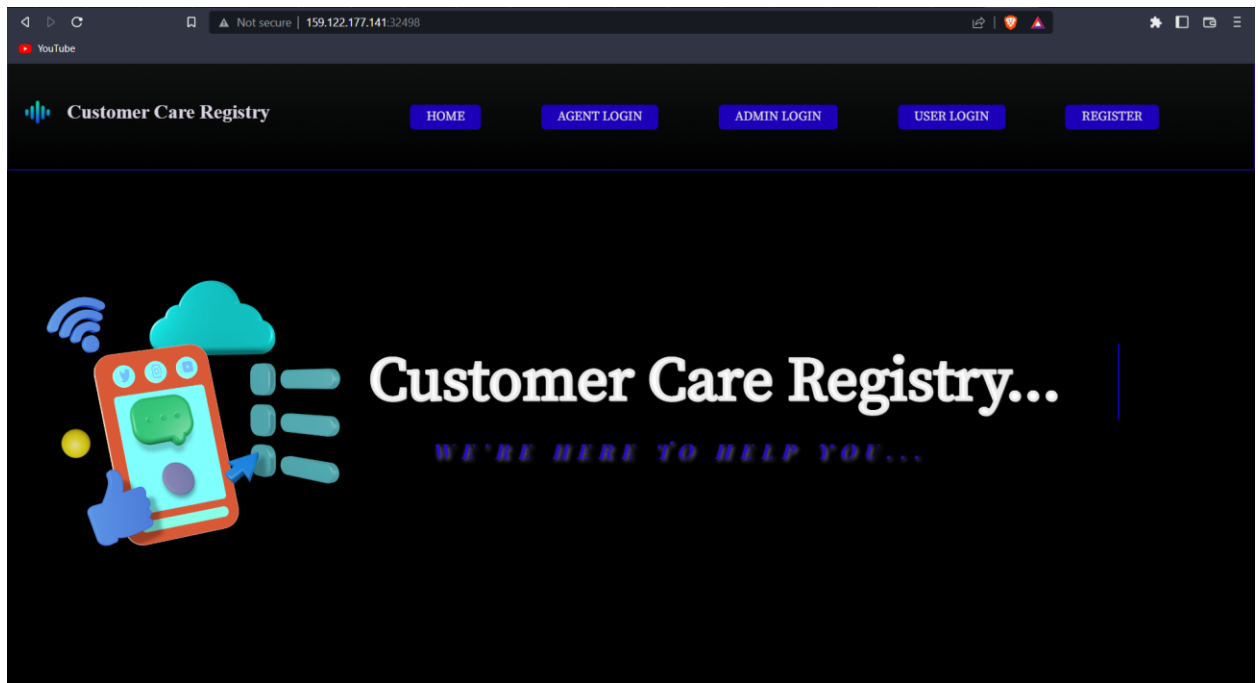
```
    mail.send(TEXT)
```

```
    return render_template('solve.html', a="Complaint Resolved" )
```

```
if __name__ == '__main__':
```

```
    app.debug=True
```

```
    app.run(host='0.0.0.0',port=8080)
```



8. Testing

8.1 Test case

1

2

3

4

5

6

7

8

9

10

				Date	17-Nov-22					
				Team ID	PNT2022TMD49581					
				Project Name	Project - Customer care registry					
				Maximum Marks	4 marks					
	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
	IndexPage_TC_OO1	Functional	Index Page	Check whether the user can view the required field .		1.Enter URL and click go 2.Verify Login/Register is displayed or not	http://159.122.177.141:3249/8/	Login/Register page of both agent and customer should display	Working as expected	Pass
	LoginPage_TC_OO2	UI	Home page	Verify the UI elements in Home/login/register buttons		1.Enter URL and click go 2.Verify login with below UI elements: a.email text box b.password text box c.Login button d.Don't have already an account? Or Signup	http://159.122.177.141:3249/8/	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.Don't have already an account? Or Signup	Working as expected	Pass
	LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with valid credentials		1.Enter URL and click go 2.Enter Valid username in text box 3.Enter valid password in password text box 4.Click on login button	username:raja@123 password:12345	User should get verification upon successful login of donor.	Working as expected	Pass
	LoginPage_TC_OO4	UI	Login page	Verify user is able to log into application with Valid credentials		1.Enter URL and click go 2.Enter Valid username in text box 3.Enter valid password in password text box	username:raja@123 password:12346	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
	LoginPage_TC_OO5	Functional	Dashboard	Verified user can access user profile, register complaints, Complaint history		1.Enter URL and click go 2.Enter Valid username in text box 3.Enter valid password in password text box 4.Click on login button	username:raja@123 password:12345	User should get verification upon successful login of recipient.	Working as expected	Pass
						1.Enter URL and click go	1.Enter URL and click go	Application should show		

Sheet1

Ready Accessibility: Investigate

100%

[illegible]

8.2 User Acceptance Testing

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By design	10	4	2	3	18
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	18	35
Not reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	0	1	8
Totals	22	14	11	24	74

1.Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2.Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

8.3 Test case analysis

Section	Total cases	Not tested	Fail	Pass
Home page	5	0	0	5
Login	10	0	0	10
Register	3	0	0	3
Email verification	5	0	0	5
Agent creation	5	0	0	5
User details	5	0	0	5
Complaint registered	10	0	0	10

9.RESULTS

9.1 Performance Metrics:

NFT - Risk Assessment									
S.No	Project Name	Scope/feature	Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Customer Care Registry Application	New	Low	No Changes	Moderate	Yes, 2hrs	>10 to 30%	GREEN	Added some additional features , so there is a software change and have no risk
NFT - Detailed Test Plan									
S.No	Project Overview	NFT Test approach		Assumptions/Dependencies/Risks		Approvals/SignOff			
1	User Registration	1) Open the Customer Care Registry Application 2) Register with user Credentials		No Risks		N/A			
2	User Login	1) Open the Customer Care Registry Application 2) Login with the user credentials		No Risks		N/A			
3	Users Complaints	1) Log in to Customer Care Registry Application 2) The user register their complaints.		No Risks		N/A			
4	Assign Agent	1) The admin create the agent and send the email notification to the agent. 2) Then individual agent are assigned to individual user.		No Risks		N/A			
5	Complaint status	1) Log in to Customer Care Registry Application 2)The user view the status of their complaints.		No Risks		N/A			
End Of Test Report									
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff	
	Customer Care	1) Log in to Customer Care Registry Application 2) Test for all test cases. 3) Log out the Customer Care Registry							

10. 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 solutions are 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.

11. CONCLUSION

The proposed web application is an customer care registry. In this application , the user can register into the application and then login into the application. If the user is already a member then they login directly into the application. In this application the user can see their profile and they able to register the complaints. After registering there complaints the Email notifications will be send to the user registered Email ID. Then the admin create the individual agent for the individual user. After creating the agent , the message will send to agent through email. The user track their problem to know the completion status of their problems.

12. FUTURE SCOPE

As part of our future enhancements, we aim to user directly contact the agent. Also we aim to provide the livechat in this application and solve the problems of all resources.

13. 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">
          
          <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">
                

```



```

        <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:

```

<!DOCTYPE html>
<html lang="en" dir="ltr">

```

```
<head>
  <meta charset="utf-8">
  <title>Dashboard</title>
  <link rel="stylesheet" href="../static/userd.css">
  <link rel="shortcut icon" type="image/jpg" href="../static/img3.jpg">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"/>
</head>
<body>
  <input type="checkbox" id="check">
  <label for="check">
    <i class="fas fa-bars" id="btn"></i>
    <i class="fas fa-times" id="cancel"></i>
  <div class="top"></div>
</label>
<div class="sidebar">
  <header>Dashboard</header>
  <ul>
    <li><a href="/home"><i class="fas fa-qrcode"></i>Dashboard</a></li>
    <li><a href="/profile"><i class="fas fa-link"></i>Profile</a></li>
    <li><a href="/addcomplaint"><i class="fas fa-stream"></i>Lodge complaint</a></li>
    <li><a href="/comphistory"><i class="fas fa-calendar-week"></i>Complaint
history</a></li>
    <li><a href="/logout "><i></i>Logout</a></li>
  </ul>
</div>
<section>
  <div class="dash">
    <h1>Dear {{ name }}</h1>
    <p>welcome to our portal...</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>

<ion-icon name="home-outline"></ion-icon>

Dashboard

<ion-icon name="person-circle-outline"></ion-icon>

Profile

<ion-icon name="document-text-outline"></ion-icon>

Lodge Complaint

<ion-icon name="chatbubbles-outline"></ion-icon>

Complaint History


```

        <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">
    
</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">
  
</div>
<div class="vj1">
  
</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:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Admin</title>
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600;700;900
&display=swap" rel="stylesheet">
  <link href="../static/admin.css" rel="stylesheet">
  <link rel="shortcut icon" type="image/jpg" href="../static/img3.jpg">
</head>
<body>
  <div class="contact-form">
    
    <h2>Admin login</h2>
    <form action="/adminlog" method="POST">
      <p>Email</p><input placeholder="Enter Email" type="email"
name="email" required>
      <p>Password</p><input placeholder="Enter Password"
type="password" name="password" required>
      <p><input type="checkbox">Remember Me</p>
```

```
        <input type="submit" value="Sign in">
        {{a}}
    </form>
</div>
</body>
</html>
```

admin dashboard.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>admin dashboard</title>
    <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>
    <input type="checkbox" id="check">
    <label for="check">
      <i class="fas fa-bars" id="btn"></i>
      <i class="fas fa-times" id="cancel"></i>
    </label>
    <div class="sidebar">
      <header>Dashboard</header>
      <ul>
        <li><a href="#"><i class="fas fa-qrcode"></i>Dashboard</a></li>
```

```

        <li><a href="/agent"><i class="fas fa-link"></i>Agent Creation</a></li>
        <li><a href="/adcomplainthist"><i class="fas fa-calendar-week"></i>Complaint history</a></li>
        <li><a href="/logout"><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://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.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="/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="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></td>
    </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>
        </ul>
    </div>

```



```

    </a></li>
    <li><a href="/adcomplainthis">
      <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="vj">
  
</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>
    </form>
  </div>
</div>

```

```
<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:

```
<!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>
    <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>
</body>
</html>

```

app.py:

```

from flask import Flask, render_template, request, redirect, url_for, session
import 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'] = 'dutexqvejaxonmmew'
```

```
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.c1logj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;Security=SSL;SSLServe  
rCertificate=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(prepare_stmt, 1, firstname)
    ibm_db.bind_param(prepare_stmt, 2, lastname)
    ibm_db.bind_param(prepare_stmt, 3, username)
    ibm_db.bind_param(prepare_stmt, 4, email)
    ibm_db.bind_param(prepare_stmt, 5, password)
    ibm_db.bind_param(prepare_stmt, 6, address)
    ibm_db.execute(prepare_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], address=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(prepare_stmt, 1, uid)
```

```
            ibm_db.bind_param(prepare_stmt, 2, selectcategory)
```

```
            ibm_db.bind_param(prepare_stmt, 3, date)
```

```
            ibm_db.bind_param(prepare_stmt, 4, phone)
```

```
            ibm_db.bind_param(prepare_stmt, 5, state)
```

```
            ibm_db.bind_param(prepare_stmt, 6, email)
```

```
            ibm_db.bind_param(prepare_stmt, 7, complaint)
```

```
            ibm_db.bind_param(prepare_stmt, 8, 'pending')
```



```

ibm_db.execute(prepare_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 successfully 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('/logout')
def logout():
    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>

<https://www.youtube.com/embed/JU6GyGkjqm0>