CUSTOMER CARE REGISTRY

PROJECT REPORT

Submitted by:

TEAM ID:PNT2022TMID49560

D.Asha Mol(950019104005)

S.Mahesh(950019104024)

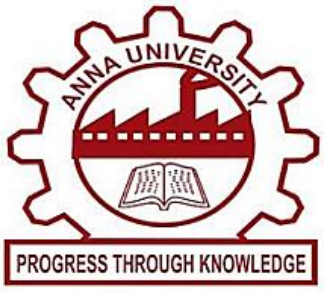
S.Nikitha(950019104033)

T.Sugunadevi(950019104048)

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. **TEST CASE**
   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:

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/

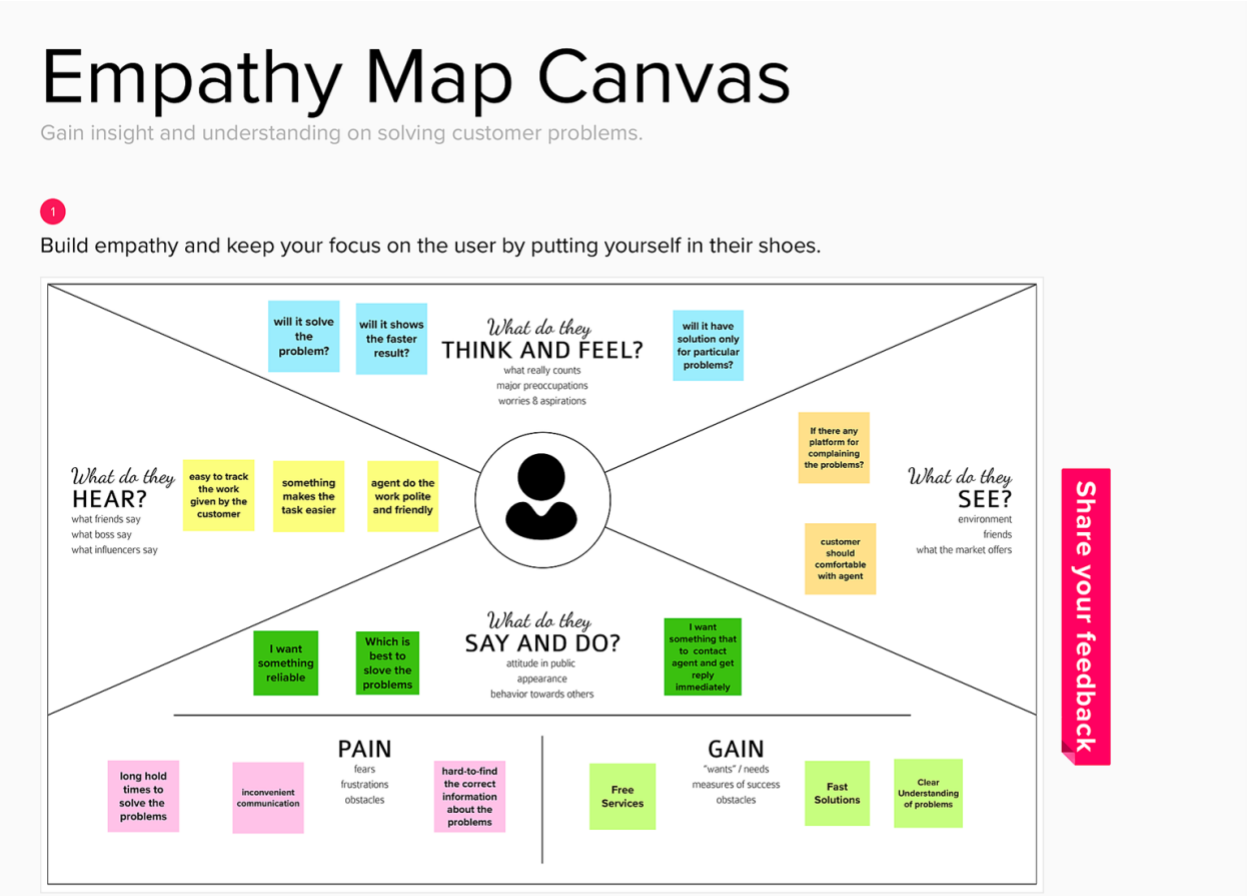
2.3 Problem Statement Definition:

The user is a customer who need a problem solving system because they need a solution for the problem that they are faced in their real time and also get a best services, So the customer care registry system must be developed.

Who does the problem affects?	Customers, Agents
What are the boundaries of the problem?	IT Sectors, Ticket reservations, Online exams, e-Commerce websites
What is the issue?	Understanding customer expectations, Customer demands something you cannot do, Handling angry customers, Inconsistence complaint handling, Customer request a feature you want to build.
When does the issue occurring?	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 issues occurring in IT sectors, e-commerce websites.
Why is it important that we fix the problem?	By solving the issue, listening to the customers queries and shown genuine empathy.

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map:



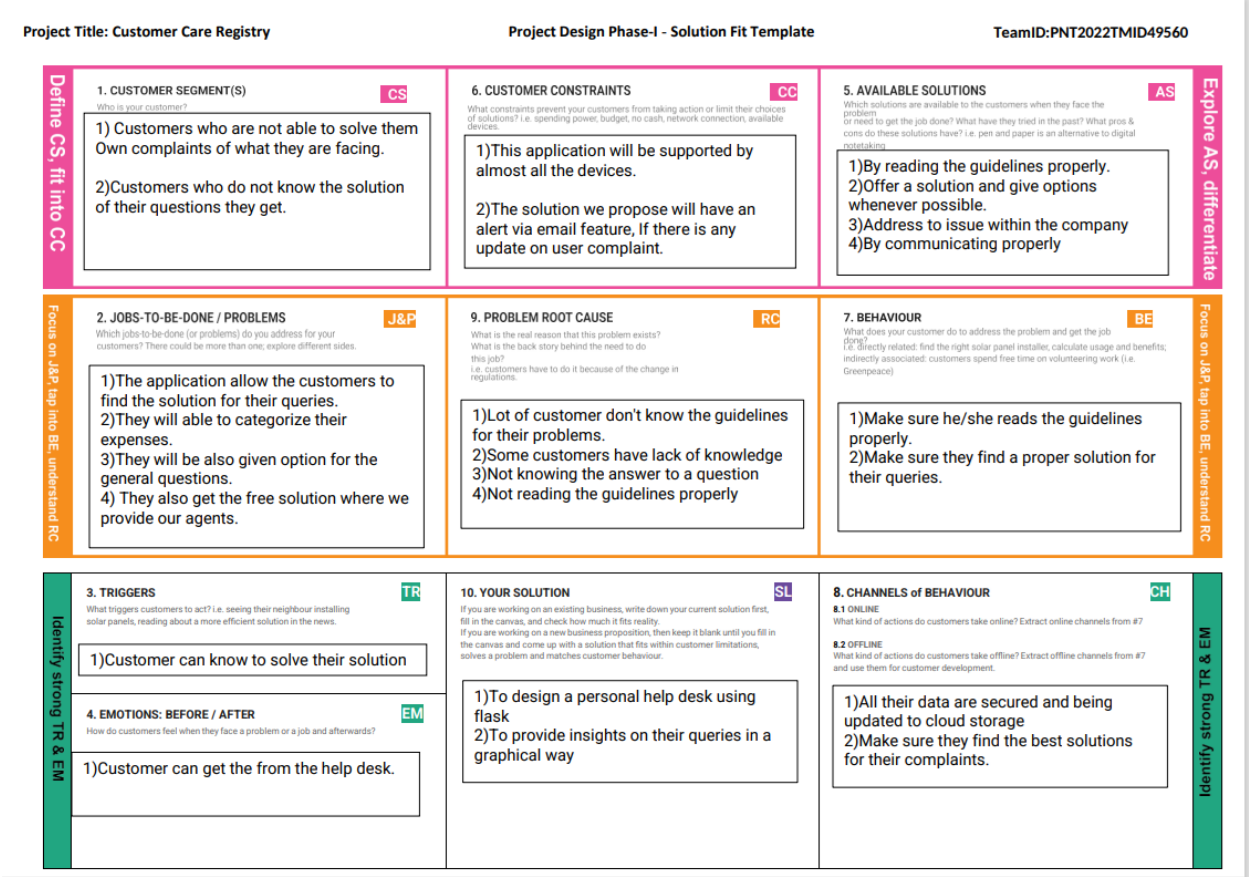
3.2 Ideation & Brainstorming:

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:



4.REQUIREMENT ANALYSIS

4.1 Functional & Non-Functional requirements:

Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	15 October 2022
Team ID	PNT2022TMID49560
Project Name	Customer Care Registry
Maximum Marks	4 Marks

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	Users Complaint	Enter the complaints in the complaint form
FR-5	Assign Agent	Assign individual agents for every user
FR-6	Complaint status	Status of the complaint send via e-mail

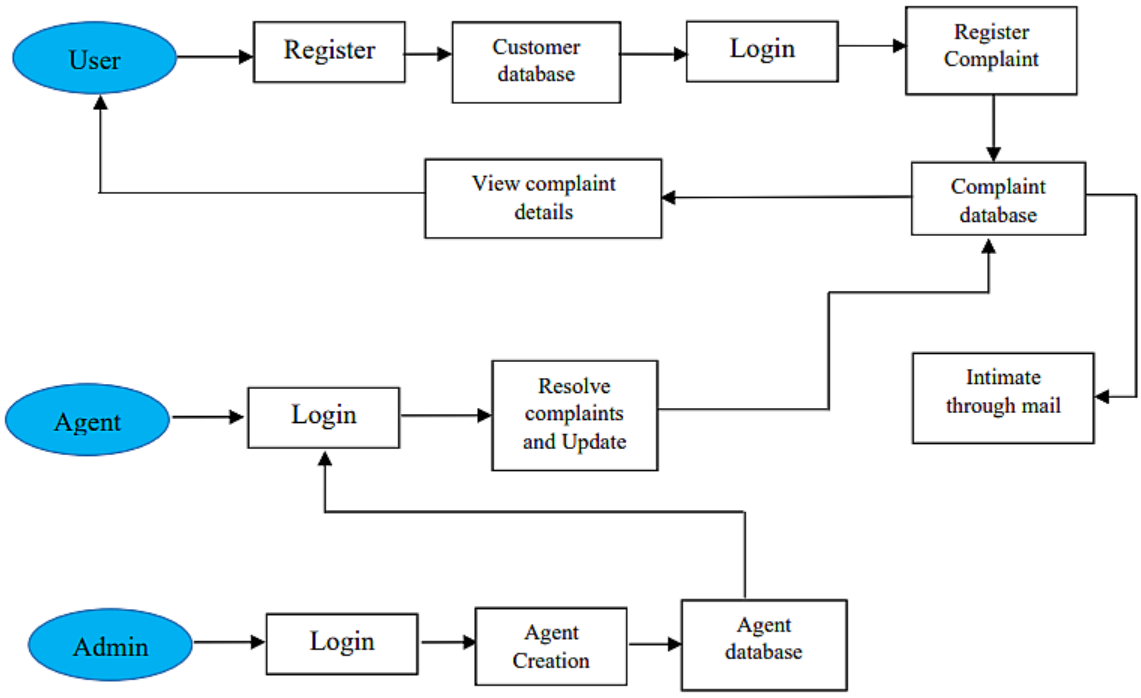
Non-functional Requirements:

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.

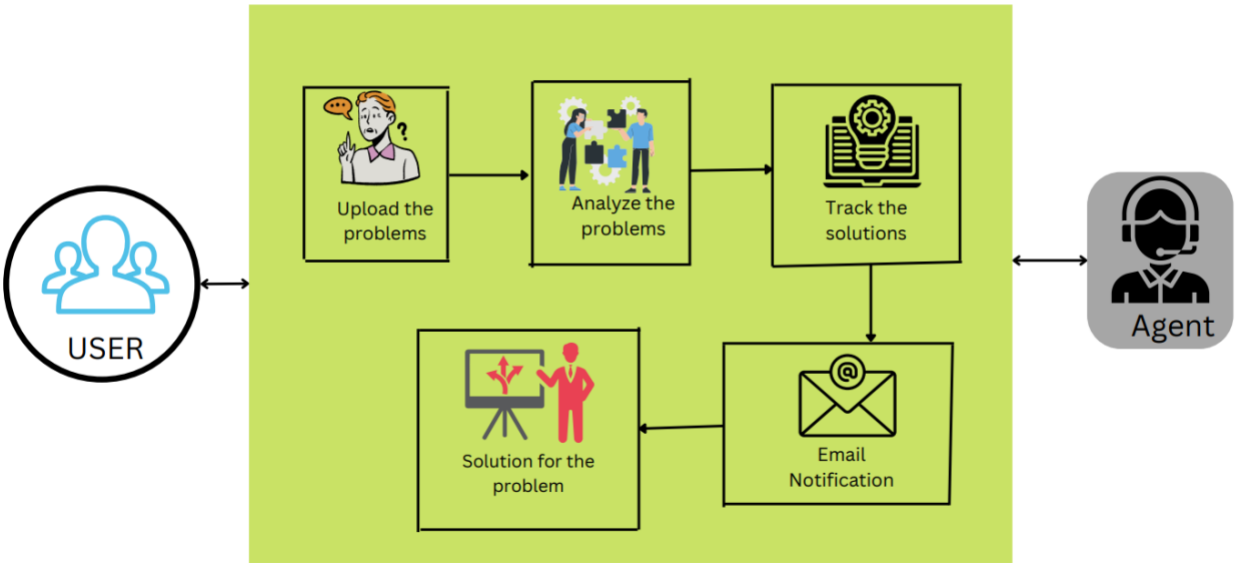
5.PROJECT DESIGN:

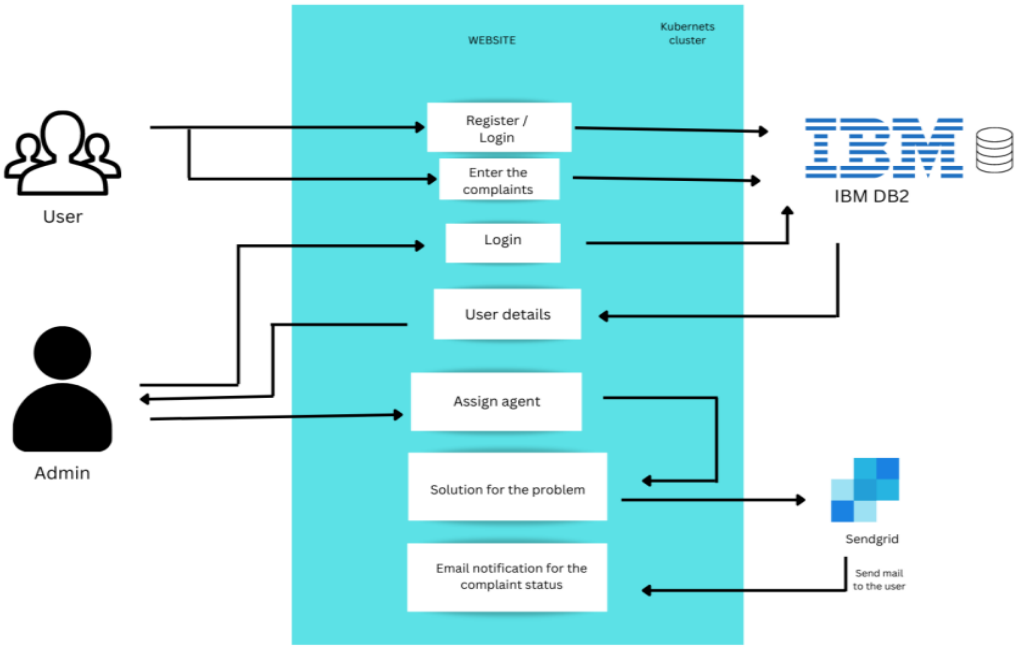
5.1 Data Flow Diagram:



5.2 Solution & Technical Architecture:

SOLUTION ARCHITECTURE





5.3 User Stories:

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 (Client User)	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 (Client User)	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 (Admin User)	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(Website 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	Asha mol,Nikitha
Sprint-1		Login	USN-2	As a customer, I can login to the application by entering correct email and password	1	High	Mahesh
Sprint-1		Dashboard	USN-3	As a customer, I can see all the tickets raised by me and lot more	3	High	Sugunadevi
Sprint-2		Complaint raise	USN-4	As a customer, I can create a new complaint with the detailed description of my query	2	High	Asha mol
Sprint-3		Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	3	High	Mahesh,Sugunadevi
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	Nikitha,Mahesh
Sprint-4		Complaint details	USN-7	As a customer, I can see the current status of my complaint	2	Medium	Asha mol,Sugunadevi

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	Asha mol
Sprint-3		Dashboard	USN-2	As an agent, I can see all the complaints assigned to me by the admin	3	High	Mahesh
Sprint-3		Address Column	USN-3	As an agent, I get to have conversations with the customer and clear his/her queries	3	High	Nikitha,Sugunadevi
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	Mahesh,Asha mol
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	Sugunadevi,Mahesh
Sprint-1		Dashboard	USN-2	As an admin, I can see all the complaints raised in the entire system and lot more	3	High	Asha mol
Sprint-2		Agent creation	USN-3	As an admin, I can create an agent for clarifying the customer's queries	2	High	Nikitha
Sprint-2		Assigning agent	USN-4	As an admin, I can assign an agent for each complaint created by the customer	3	High	Nikitha,Sugunadevi
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	Asha mol,Mahesh

6.2 Sprint Delivery Schedule:

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

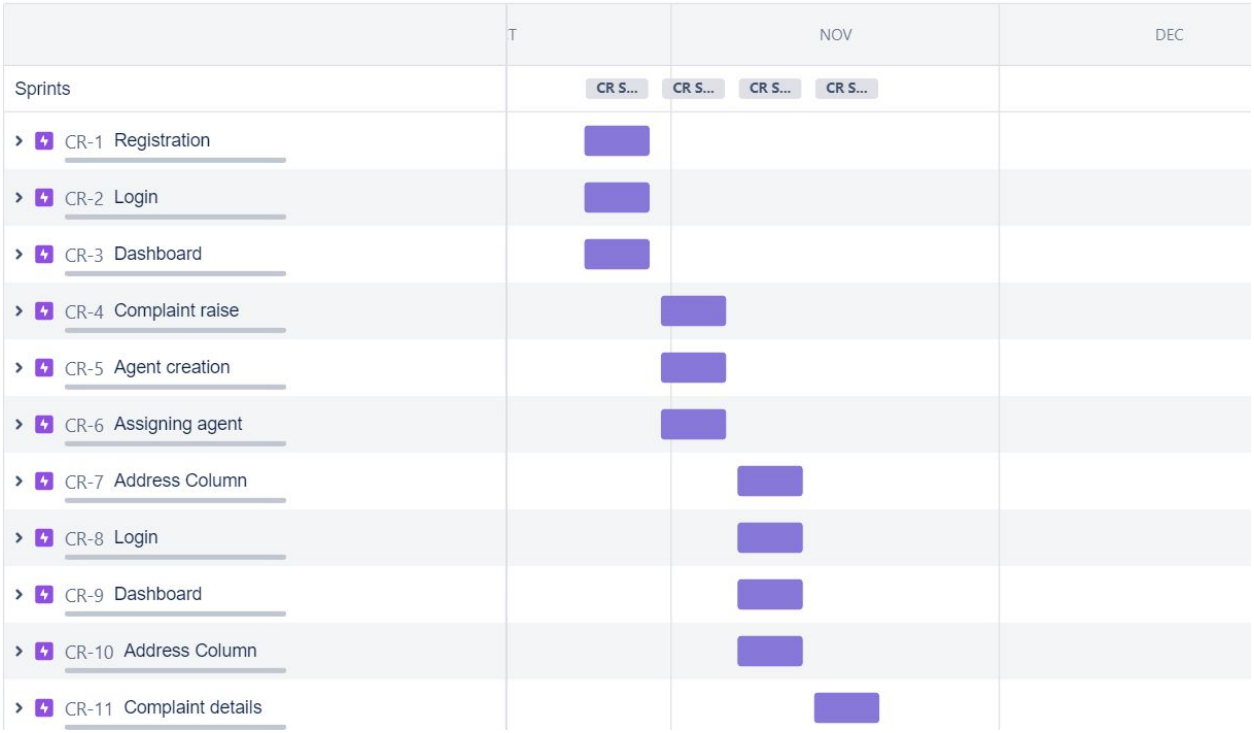
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022
Sprint-2	7	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022
Sprint-3	11	6 Days	07 Nov 2022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

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)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

6.3 Reports from JIRA:



7. CODING & SOLUTIONING:

7.1 Feature 1:

User register and login:

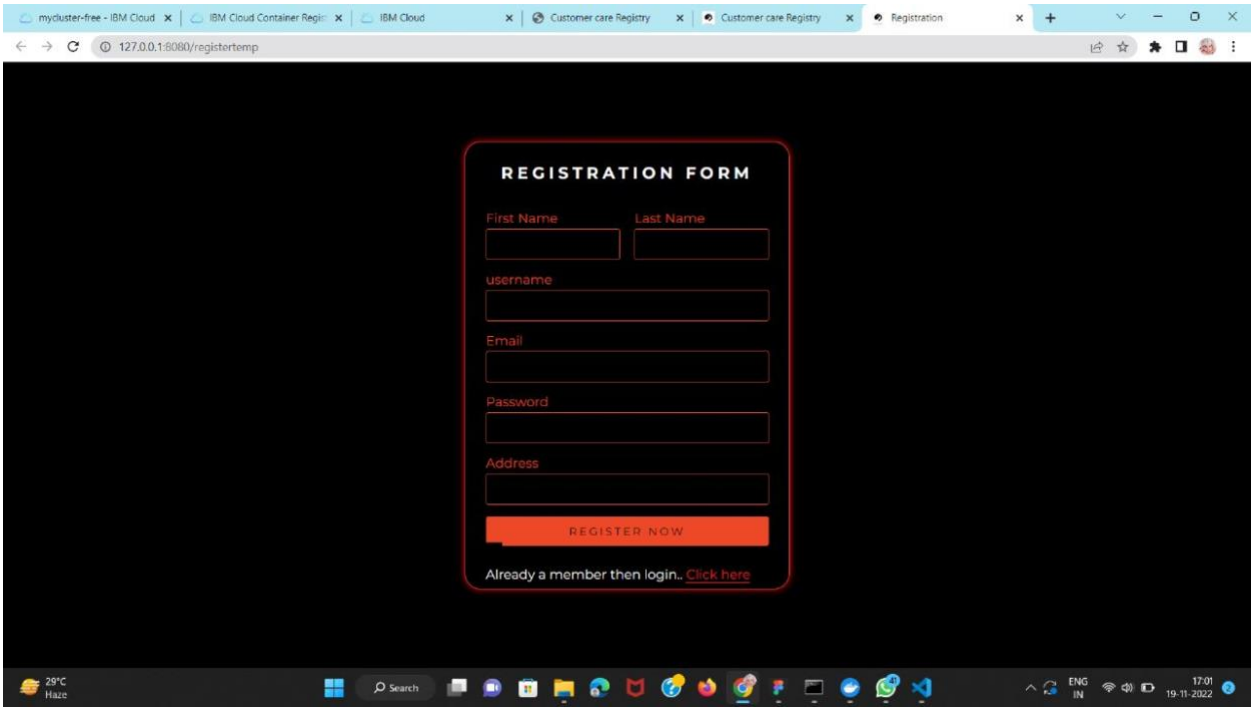
User register and login through this web application.

Source code:

register.html:

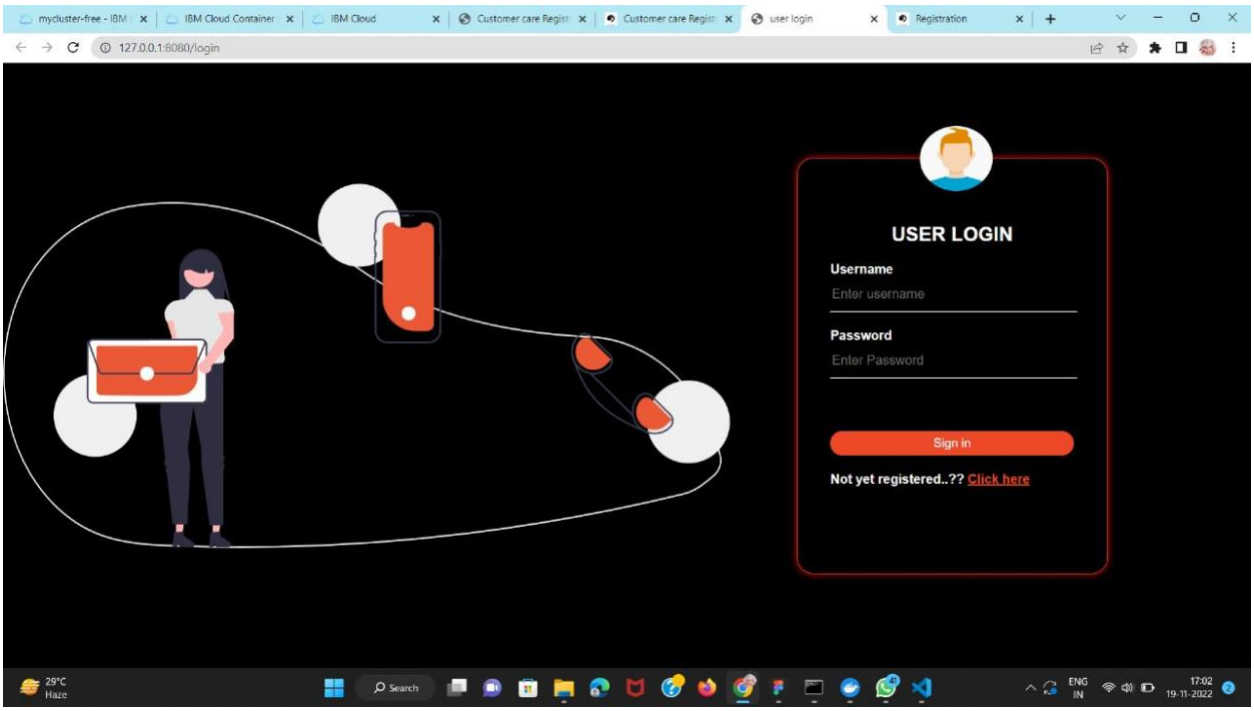
```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Registration</title>
   <link rel="shortcut icon" type="image/jpg" href="../static/img3.jpg">
   <link rel="stylesheet" href="../static/register.css">
</head>
<body>
   <div class="wrapper">
      <img src= "{{url_for('static',filename='img6.jpg')}}" alt="" class="hello" width="100%"
height="100%" style="filter: blur(2px);">
      <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 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>
```



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>
		<img src= "{{url_for('static',filename='img5.jpg')}}" alt="" class="hello"
width="100%" height="100%" style="filter: blur(2px);">

	<div class="contact-form">
		<img  src="{{url_for('static',filename='a1.jpg')}}" alt="" class="avatar">
		<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:

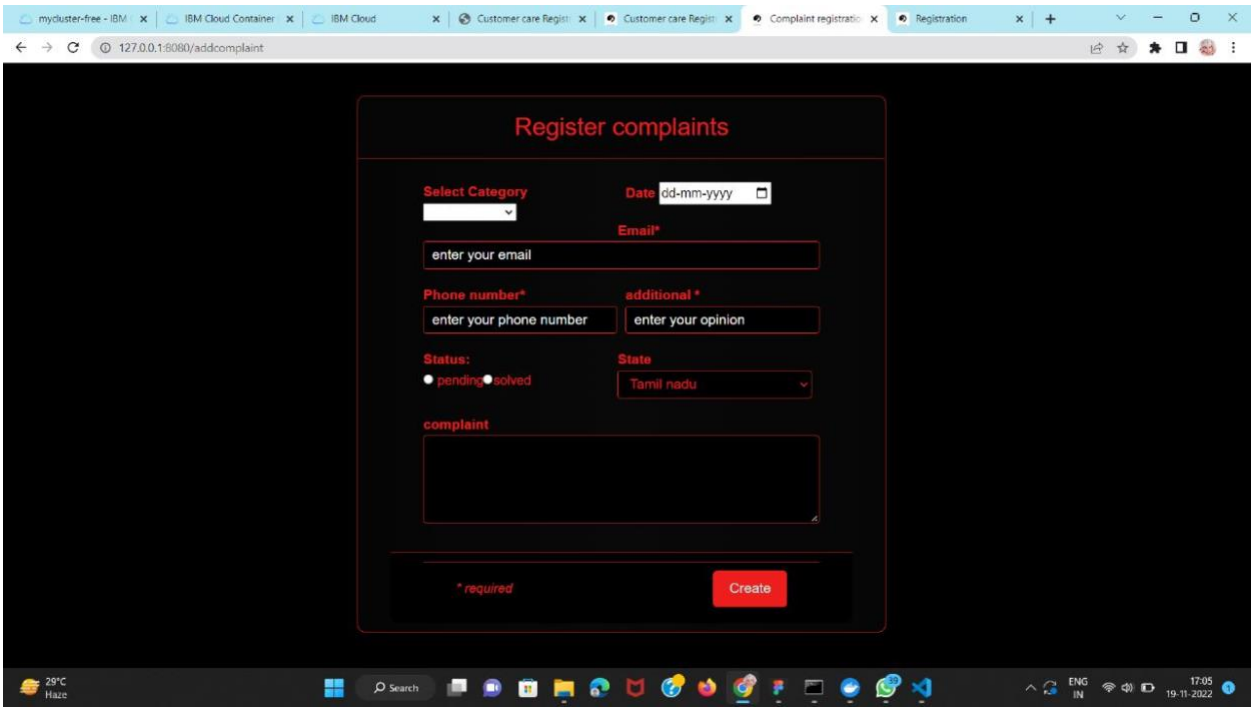
Userlodgecomp.html:

{% extends '_userdashboard.html' %}

{% block body %}

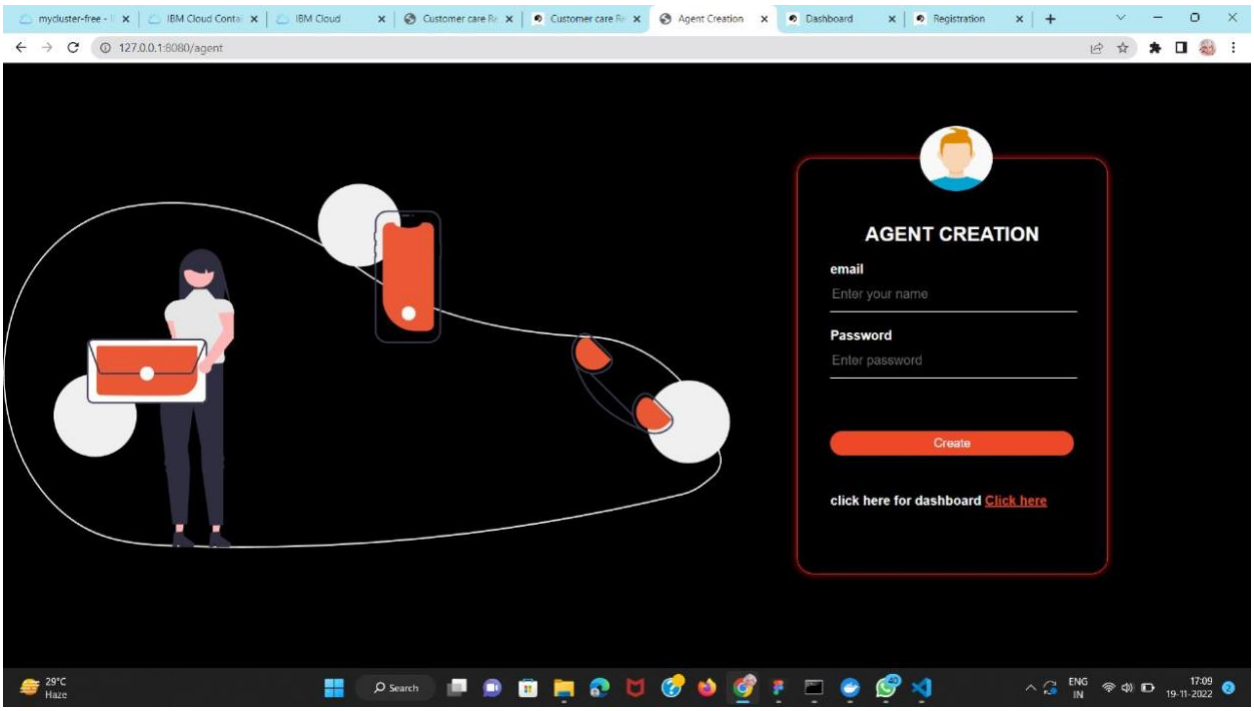
```
 <div class="title">Lodge Complaint</div>
<form action="/complaint" method="POST">
    <span class="text"style="color: green;font-weight:bold;">{{a}}</span>
    <div class="user-details">
        <div class="input-box" >
          <label for="select-category"  >Select Category</label>
          <select name="selectcategory" required>
            <option  value="category"></option>
            <option  value="E-commerse">E-commerse</option>
            <option  value="general">General</option>
          </select>
        </div>
        <div class="input-box">
        <label for="select-subcategory" >Select-sub Category</label>
        <select name="selectsubcategory" required>
          <option  value="select-sub category"></option>
          <option  value="Online-shopping">Online Shopping</option>
          <option  value="E-wallet">E-wallet</option>

        </select>
        </div>
        <div class="input-box">
         <label for="complaint-type" >Complaint-Type</label>
          <select name="type" required>
            <option  value="complaint-type"></option>
            <option  value="general">General</option>
            <option  value="Query">Query</option>
           </select>
        </div>
        <div class="input-box">
```

```
            <label for="State">State</label>
             <select name="state" required>
                <option  value="Tirunelveli">Tirunelveli</option>
                <option  value="Tamil Nadu">Tamil Nadu</option>
                <option  value="others">Others</option>
             </select>
           </div>
      </div>
          <div class="input-box">
            <label for="complaint" id="complaint">Complaint Details</label>
            <textarea  rows="15" cols="60" placeholder="Enter the complaint details with
date" name="complaint" required></textarea>
          </div>
          <div class="button">
            <input type="date" name="date">
          <input  type="submit" value="Register">
          </div>
   </form>
</div>
{% endblock %}
```



adminagent.html:

```
{% extends '_admindashboard.html' %}
{% block 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" required>
      <p>Password</p><input placeholder="Enter password" type="password"
name="password" required>
       <input type="submit" value="create">
       {{a}}
   </form>
</div>
{% endblock %}
```



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>
	<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 sendemail import usermail,agentmail

app = Flask(_name_)


app.secret_key='a'
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;Securi
ty=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=fqt70379;PWD=KdUUJ3Rn
MlMqAimg;",'','')

@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)
        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(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 = """\<!DOCTYPE html>
            <html>
            <body>
               <div class="containter" style="display: block;">
                  <h3 style="font-size: 24px; font-family:serif"> Dear """+acc[1]+"
"+acc[2]+""", </h3>
                  <div class="side" style="width: 400px; height: 150px; padding:30px;
border-radius:10px; position:relative; left:100px;" >
                     <div class="details"style="position:relative;left:60px; font-
size:20px;text-align:left;">
                        <p style=" font-weight:bold;">Your complaint has been succesfully
registered...!!</p>
                        <p style=" font-weight:bold;">One of our agent is assigned , will
surely resolve your complaint as soon as possible</p>
                        <p style=" font-weight:bold;">Please check the complaint history tab
for status of complaint.</p>
                     </div>
                  </div>
               </div>
            </body>
            </html>"""
```

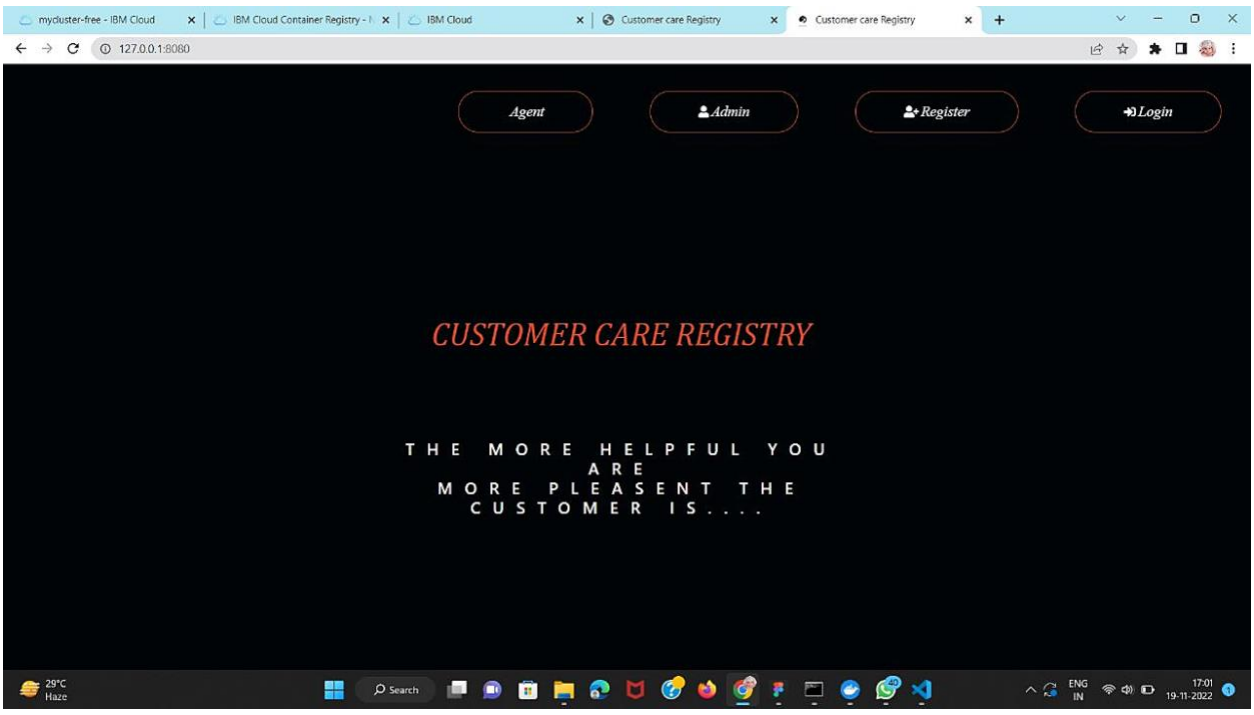
```
        TEXT = """\<!DOCTYPE html>
                <html>
                <body>
                   <div class="containter" style="display: block;">
                      <h3 style="font-size: 24px; font-family:serif"> New Complaint from
"""+acc[1]+" "+acc[2]+""" </h3>
                      <div class="side" style="width: 400px; height: 150px; padding:30px;
border-radius:10px; position:relative; left:100px;" >
                         <div class="details"style="position:relative; top:20px; left:60px; font-
size:20px;text-align:left;">
                            <p style=" font-weight:bold;"> Complaint Description :</p>
                            <p>"""+complaint+"""</p>
                         </div>
                      </div>
                   </div>
                </body>
                </html>"""
        se = ibm_db.exec_immediate(conn, 'SELECT * FROM agentinfo ORDER BY
RAND() LIMIT 1')
        agmail = ibm_db.fetch_tuple(se)
        agentemail = agmail[1]
        usermail(TEXT1,email)
        agentmail(TEXT,agentemail)
        return render_template('userlodgecomp.html',a = msg)
    return render_template('userlodgecomp.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():
        stmt = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails')
        ibm_db.execute(stmt)
        comp = ibm_db.fetch_tuple(stmt)
        return render_template('adminhistory.html',row = comp)
```

```
@app.route('/agcomplainthist',methods=['POST','GET'])
def agcomplainthist():
        stmt = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails')
        ibm_db.execute(stmt)
        comp = ibm_db.fetch_tuple(stmt)
        return render_template('agentcomphist.html',row  = comp)

@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('agentlogin.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("agentlogin.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)
    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

8.1TEST CASE

Date	17-Nov-22
Team ID	PNT2022PMID49597
Project Name	Project - PLASMA DONOR APPLICATION
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
IndexPage_TC_OO1	Functional	Home Page	Both the donor and receipient are verified with their choosing process.		1.Enter URL and click go 2.Click on Donor/recipient to login or register. 3.Verify Login/Register is displayed or not	https://shopenzer.com/	Login/Register page of both Donor and Recipient should display	Working as expected	Pass				
LoginPage_TC_OO2	UI	Login Page of Donor	Verify the UI elements in Login page		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	https://shopenzer.com/	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			BUG-1234	
LoginPage_TC_OO3	Functional	Login page of Donor	Credentials of the donor is verified, If not can be registered		1.Enter URL and click go 2.Enter Valid email in Email text box 3.Enter valid password in password text box 4.Click on login button	email: yoganraja.126@gmail.com password:abc@987	User should get verification upon successful login of donor.	Working as expected	Pass				
LoginPage_TC_OO4	UI	Login page Of recipient	Verify user is able to log into application with Valid credentials		1.Enter URL and click go 2.Enter Valid email in Email text box 3.Enter valid password in password text box	email: yoganraja.126@gmail.com password:abc@987	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				
LoginPage_TC_OO5	Functional	Login page of Recipient	Credentials of the recipient is verified, If not can be registered		1.Enter URL and click go 2.Enter Valid email in Email text box 3.Enter valid password in password text box 4.Click on login button	email: yoganraja.126@gmail.com password:abc@987	User should get verification upon successful login of recipient.	Working as expected	Pass				
RegisterPage_TC_006	UI	Register page of Donor	Verify user is able to register into application with Valid credentials		1.Enter URL and click go 2.Verify register with below UI elements: a.fullname text box b.Blood Group text box c.email text box d.password text box e.phone number text box f.gender text box g.age text box h.address text box i.district text box j.State text box k.do you have any illness text box.	URL: a.fullname:RajaCheraKesaree b.Blood Group: B c.email:yoganraja.126@gmail.com d.password:abc@987 e.phone number:9600000010 f.gender:Male g.age: 20 h.address:kamaraj st i.district:Tirunelveli j.State:TamilNadu k.do you have any illness:No	Application should show below UI elements: 1.Enter URL and click go a.fullname text box b.Blood Group text box c.email text box d.password text box e.phone number text box f.gender text box g.age text box h.address text box i.district text box j.State text box	Working as expected	Pass				
RegisterPage_TC_007	UI	Register page of Recipient	Verify user is able to register into application with Valid credentials		1.Enter URL and click go 2.Verify register with below UI elements: a.Admin Organisation Name text box b.Admin Email text box c.Patient's name text box d.Patient's Blood group text box e.Patient's Age text box f.Type of request text box g.phonenumber text box h.address text box i.Submit button	URL: a.Admin Organisation Name : SuswinSarav Hospital b.Admin Email:SShos22@gmail.com c.Patient's name:Remy d.Patient's Blood group :O e.Patient's Age:21 f.Type of request:EMERGENCY g.phonenumber:9600000010 h.address:Anna nagar,Tvl i.Submit button	Application should show below UI elements: 1.Enter URL and click go a.Admin Organisation Name text box b.Admin Email text box c.Patient's name text box d.Patient's Blood group text box e.Patient's Age text box f.Type of request text box g.phonenumber text box	Working as expected	Pass				
RequestPage_TC_008	UI	Request page	Verify the UI elements of Request Page		1.verify the page with below UI elements: a.Type of Blood Group needed textbox	a.Type of Blood Group needed :O	a.Type of Blood Group needed textbox	Working as expected	Pass				
RequestPage_TC_009	Functional	Request page	verify the user able to enter and validate the request		1.Enter the type of bloodgroup needed. 2.Click the submit Button	a.Type of Blood Group needed :O	Upon entering type of bloodgroup,user able to see the list.	Working as expected	Pass				
DonorDetailsPage_TC_010	UI	donor details page	Verify the UI elements of Donor details page		A.verify the page with below UI elements: 1.user able to view the list of Donors Available 2.user can send request 3.Send Request button	1.Recipient able to view the list of Donors Available : Names Jebin ,Saravanan,Raja 2.user can send request : yes 3.Send Request button	1.user able to view the list of Donors Available 2.Recipient can send request 3.Send Request button	Working as expected	Pass				
DonorDetailsPage_TC_011	Functional	donor details page	Verify the User able to view the Donor Details		1.user able to view the list of Donors Available 2.user can send request 3.Send Request button	1.Recipient able to view the list of Donors Available : Names Jebin,Saravanan,Raja 2.user can send request : yes 3.Send Request button	Recipient can able to see list of donors and can send request.	Working as expected	Pass				
NotificationPage_TC_012	UI	notification page	Verify the UI elements in Notification page		A.verify the page with below UI elements: 1.Email sent successfully	email sent successfully	upon send request ,user can able to see confirmation notification.	Working as expected	Pass				
DonorNotification_TC_013	UI	Donor Notification mail	Verify the UI elements in Notification page		1.verify the page with below UI elements: 2.Donor receive request from recipient	t :Here the requested blood	Donor recieve email from the recipient	Working as expected	Pass				

9.RESULTS

9.1 Performance Metrics:

			NFT - Risk Assessment						
o	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:

- 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.
- The problems of the user are understand clearly and the appropriate solutions are given to them.
- Builds customer trust and loyalty.
- The customers are able to track their problems.
- The agents makes the customers very comfortable by solving their problems.
- Every individual customers has individual agent.

Disadvantages:

- Sometimes this application leads to inconvenient communication.
- 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">
</head>
<body>
    <div class="wrapper">
        <img src= "{{url_for('static',filename='img6.jpg')}}" alt="" class="hello" width="100%"
height="100%" style="filter: blur(2px);">
        <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 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>
```

login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>user login</title>

   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="../static/login.css">
</head>
<body>
        <div class="container" >
                <div class="row">
                        <div class="column1">



                        </div>
                        <div class="column2">
                                <div class="contact-form">
                                        <img    src="{{url_for('static',filename='a1.jpg')}}"  alt=""
class="avatar">
                                        <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>

                            </div>
                        </div>
                </div>
</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:

```
{% extends '_userdashboard.html' %}
{% block body %}
<div class="box">
   <div class="profile">
      <label>fullname:</label>
      <span><p>{{fullname}}</p></span>
```

```
    </div>
    <div class="profile">
        <label>Username :</label>
        <span><p>{{ username }}</p></span>
    </div>
    <div class="profile">
        <label>Email :</label>
        <span><p>{{ email }}</p></span>
    </div>
    <div class="profile">
        <label>Address:</label>
        <span><p>{{ address }}</p></span>
    </div>
</div>
{% endblock %}
```

Userlodgecomplaints.html:

```
{% extends '_userdashboard.html' %}
{% block body %}
<div class="container">
    <div class="title">Lodge Complaint</div>
   <form action="/complaint" method="POST">
        <span class="text"style="color: green;font-weight:bold;">{{a}}</span>
        <div class="user-details">
              <div class="input-box" >
                 <label for="select-category"  >Select Category</label>
                 <select name="selectcategory" required>
                    <option  value="category"></option>
                    <option  value="E-commerse">E-commerse</option>
```

```
            <option  value="general">General</option>
          </select>
        </div>
        <div class="input-box">
        <label for="select-subcategory" >Select-sub Category</label>
        <select name="selectsubcategory" required>
          <option  value="select-sub category"></option>
          <option  value="Online-shopping">Online Shopping</option>
          <option  value="E-wallet">E-wallet</option>

        </select>
        </div>
        <div class="input-box">
         <label for="complaint-type" >Complaint-Type</label>
          <select name="type" required>
            <option  value="complaint-type"></option>
            <option  value="general">General</option>
            <option  value="Query">Query</option>
           </select>
        </div>
        <div class="input-box">
          <label for="State">State</label>
           <select name="state" required>
             <option  value="Tirunelveli">Tirunelveli</option>
             <option  value="Tamil Nadu">Tamil Nadu</option>
             <option  value="others">Others</option>
           </select>
         </div>
    </div>
        <div class="input-box">
```

```
            <label for="complaint" id="complaint">Complaint Details</label>
            <textarea  rows="15" cols="60" placeholder="Enter the complaint details with
date" name="complaint" required></textarea>
        </div>
        <div class="button">
            <input type="date" name="date">
        <input  type="submit" value="Register">
        </div>
   </form>
</div>
{% endblock %}
```

usercomphist.html:

```
{% extends '_userdashboard.html' %}
{% block body %}
   <div class="history" style="position: relative;
   top: 10px;
   left: 280px;">
     <table>
        <thead>
            <th> ComplaintNumber</th>
            <th> Category</th>
            <th> Subcategory</th>
            <th> Complaintyype</th>
            <th> State</th>
            <th> RegisteredDate</th>
            <th> Status</th>
```

```
        </thead>
        <tbody>
          {% for row in complaints: %}
           <tr>
              <td>{{ row.0 }}</td>
              <td>{{ row.2 }}</td>
              <td>{{ row.3 }}</td>
              <td>{{ row.4 }}</td>
              <td>{{ row.5 }}</td>
              <td>{{ row.7 }}</td>
              <td>{{ row.8 }}</td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
   </div>
{% endblock %}
```

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;7
00;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">
                <img alt="" class="avatar"
src="https://i.postimg.cc/zDyt7KCv/a1.jpg">
                <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="/loggout"><i></i>Logout</a></li>
            </ul>
        </div>
    </body>
</html>
```

admincomphist.html:

```
{% extends '_admindashboard.html' %}
{% block body %}
   <div class="history">
         <span class="tab"  style="position: relative;
         top: -250px;
         left: -400px;
         padding: 10px; width: 940px;
         font-weight: bold;">{{a}}</span>
         {% for row in complaints: %}
          <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.1 }}</span>
            <span style="margin-left: 20px;
            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: 30px;
            text-align: left;">{{ row.5 }}</span>
            <span style="margin-left: 20px;
            text-align: left;">{{ row.7 }}</span>
            <a href="/solved/{{ row.0 }} "><span style="margin-left: 20px;
```

```
            text-align: left;">{{ row.8 }}</span></a>
        </div>
        {% endfor %}
    </div>
{% endblock %}

<!--<table>
        <thead>
            <th> ComplaintNumber</th>
            <th> Category</th>
            <th> Subcategory</th>
            <th> Complaintyype</th>
            <th> State</th>
            <th> RegisteredDate</th>
            <th> Status</th>
        </thead>
        anthakanna m chylem ga iga
        mail etlaaag

        <tbody>
            division chylema line tho mari m ardam kakunnda untad ani
        </tbody>
      </table>-->
```

adminagent.html:

```
{% extends '_admindashboard.html' %}
{% block 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" required>
    <p>Password</p><input placeholder="Enter password" type="password" name="password" required>
     <input type="submit" value="create">
     {{a}}
  </form>
</div>
{% endblock %}
```

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=swap" 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 sendemail import usermail,agentmail
app = Flask(__name__)

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=jll83843;PWD=miOgJr4TYkdKBff6;","","")

ibm_db.autocommit(conn, ibm_db.SQL_AUTOCOMMIT_OFF)

@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)
    return render_template('register.html',a = msg,indicator="success")

@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_assoc(stmt)
    if account:
      session['loggedin'] = True
      session['username'] = account['USERNAME']
      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:
        username = session['username']
        return render_template('userdashboard.html',name=username)
    return render_template("userdashboard.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_assoc(stmt)
        return
render_template('userprofile.html',fullname=acc[2]+acc[3],username=acc[4],email=acc[
5],address=acc[7])

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

@app.route('/complaint',methods=["POST","GET"])
def complaint():
    if request.method == "POST":
        if 'id' in session:
            msg = ''
            uid=session['id']
```

```
        selectcategory = request.form['selectcategory']
        selectsubcategory = request.form['selectsubcategory']
        complainttype = request.form['type']
        state = request.form['state']
        complaint = request.form.get('complaint')
        date = request.form['date']
        stmt = ibm_db.prepare(conn,"SELECT * FROM users WHERE id = % s",(uid,))
        ibm_db.bind_param(stmt, 1, uid)
        ibm_db.bind_param(stmt, 2, selectcategory)
        ibm_db.bind_param(stmt, 3, selectsubcategory)
        ibm_db.bind_param(stmt, 4, complainttype)
        ibm_db.bind_param(stmt, 5, state)
        ibm_db.bind_param(stmt, 6, complaint)
        ibm_db.bind_param(stmt, 7, date)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc()
        email = acc[4]
        ibm_db.prepare(conn,'INSERT INTO complaintdetails VALUES (NULL, ?, ?, ?, ?, ?, ?,
?,
?)',(uid,selectcategory,selectsubcategory,complainttype,state,complaint,date,'pending'))
        msg = 'You have successfully registered your complaint'
        TEXT1 = """\<!DOCTYPE html>
            <html>
            <body>
               <div class="containter" style="display: block;">
                  <h3 style="font-size: 24px; font-family:serif"> Dear """+acc[1]+"
"+acc[2]+""", </h3>
                  <div class="side" style="width: 400px; height: 150px; padding:30px;
border-radius:10px; position:relative; left:100px;" >
```

```
                    <div class="details"style="position:relative;left:60px; font-size:20px;text-align:left;">
                      <p style=" font-weight:bold;">Your complaint has been succesfully registered...!!</p>
                      <p style=" font-weight:bold;">One of our agent is assigned , will surely resolve your complaint as soon as possible</p>
                      <p style=" font-weight:bold;">Please check the complaint history tab for status of complaint.</p>
                    </div>
                  </div>
                </div>
              </body>
              </html>"""
        TEXT = """\<!DOCTYPE html>
              <html>
              <body>
                <div class="containter" style="display: block;">
                  <h3 style="font-size: 24px; font-family:serif"> New Complaint from """+acc[1]+" "+acc[2]+""" </h3>
                  <div class="side" style="width: 400px; height: 150px; padding:30px; border-radius:10px; position:relative; left:100px;" >
                    <div class="details"style="position:relative; top:20px; left:60px; font-size:20px;text-align:left;">
                      <p style=" font-weight:bold;"> Complaint Description :</p>
                      <p>"""+complaint+"""</p>
                    </div>
                  </div>
                </div>
              </body>
              </html>"""
```

```
        agentemail = 'agentcustomercareregistry@gmail.com'
        usermail(TEXT1,email)
        agentmail(TEXT,agentemail)
        return render_template('userlodgecomp.html',a = msg)


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

@app.route('/comphistory',methods=['POST','GET'])
def compview():
    if 'id' in session:
        uid=session['id']
        ibm_db.prepare(conn,'SELECT * FROM complaintdetails WHERE userid = % s',(uid,))
        ibm_db.commit(conn)
        comp = ibm_db.fetch_assoc()
        return render_template('usercomphist.html',complaints = comp)

@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():

        stmt = ibm_db.prepare(conn, 'SELECT * FROM complaintdetails')
        ibm_db.execute(stmt)
        comp = ibm_db.fetch_assoc(stmt)
        return render_template('admincomphist.html',complaints = comp)

@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():
    if 'id ' in session:
        session.pop('id',None)
        session.pop('name',None)
        session.pop('username',None)
    return redirect(url_for('home'))

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


@app.route('/solved/<no>')
def solved(no):
    i = no
```

```
    stmt = ibm_db.prepare(conn,"UPDATE complaintdetails SET status = ? WHERE id =
?",("Solved",i))
    ibm_db.bind_param(stmt, 1, "Solved")
    ibm_db.execute(stmt)
    return render_template('admincomphist.html',a="Complaint Resolved")

if __name__ == '__main__':
    app.debug=True
    app.run(host='0.0.0.0',port=8080)
```

sendemail.py:

```
import smtplib
from email.message import EmailMessage
def usermail(TEXT,tomail):
    msg = EmailMessage()
    msg['Subject'] = 'Complaint Registered | Customer Care Registry'
    msg['From'] = 'selva.nellai72@gmail.com'
    msg['To'] = tomail
    msg.set_content('')
    msg.add_alternative(TEXT,subtype='html')


    with smtplib.SMTP('smtp.gmail.com',587,'localhost',10) as smtp:
        smtp.ehlo()
        smtp.starttls()
        smtp.ehlo()
        smtp.login('selva.nellai72@gmail.com','ydpcwysimzdtzkyc')
        smtp.send_message(msg)
```

```
def agentmail(TEXT,tomail):
    msg = EmailMessage()
    msg['Subject'] = 'New Complaint Registered | Customer Care Registry'
    msg['From'] = 'selva.nellai72@gmail.com'
    msg['To'] = tomail
    msg.set_content('')
    msg.add_alternative(TEXT,subtype='html')


    with smtplib.SMTP('smtp.gmail.com',587) as smtp:
        smtp.ehlo()
        smtp.starttls()
        smtp.ehlo()
        smtp.login('selva.nellai72@gmail.com','ydpcwysimzdtzkyc')
        smtp.send_message(msg)
```