TITLE

CUSTOMER CARE REGISTRY

DOMAIN

CLOUD APPLICATION DEVELOPMENT

Team ID : PNT2022TMID40350

Team Leader : Deebatharani N

Team Member : Kiruthika Y

Team Member : Bhuvaneshwari T

Team Member : Swathi S

Github Link: https://github.com/IBM-EPBL/IBM-Project-48298-1660806417

Project Demo Link: https://youtu.be/x9jcTL7nJ0M

Application Link: http://169.51.204.215:30106/signinpage

1. INTRODUCTION

a. Project Overview

This Application has been developed to help the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to a customer they will be notified with an email alert. Customers can view the status of the ticket till the service is provided.

b. Purpose

The purpose is to deliver customer friendly customer care service

2. LITERATURE SURVEY

a. Existing problem

Customer reaching and asking queries in person is a hard task.

b. References

[1] "Models of consumer satisfaction formation: An extension" by D. K. Tse and P. C. Wilton.

[2] Customer Satisfaction- Aware Profit Optimization Model to Find the Numeric Optimal Cloud Configuration for Cloud Service Providers by Ponnuru Aruna , J.Raghunath

[3] An intelligent cloud-based customer relationship management system to determine flexible pricing for customer retention by H.Y.Choy, W.Y.Stephen and K.L. Cheng

c. Problem Statement Definition

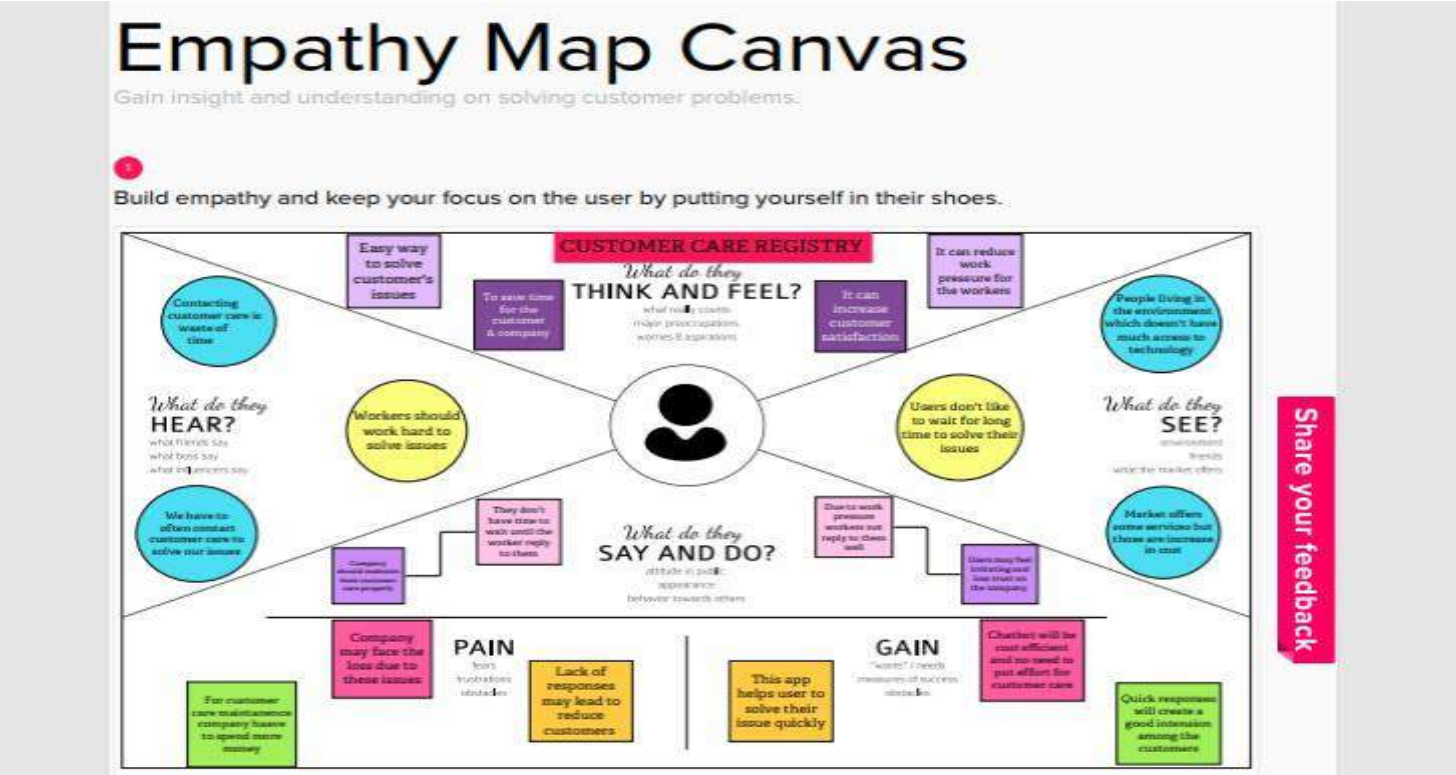
The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to a customer they will be notified with an email alert. Customers can view the status of the ticket till the service is provided.

ADMIN : The main role and responsibility of the admin are to take care of the whole process. Starting from Admin login followed by the agent creation and assigning the customer's complaints. Finally, He will be able to track the work assigned to the agent and a notification will be sent to the customer.

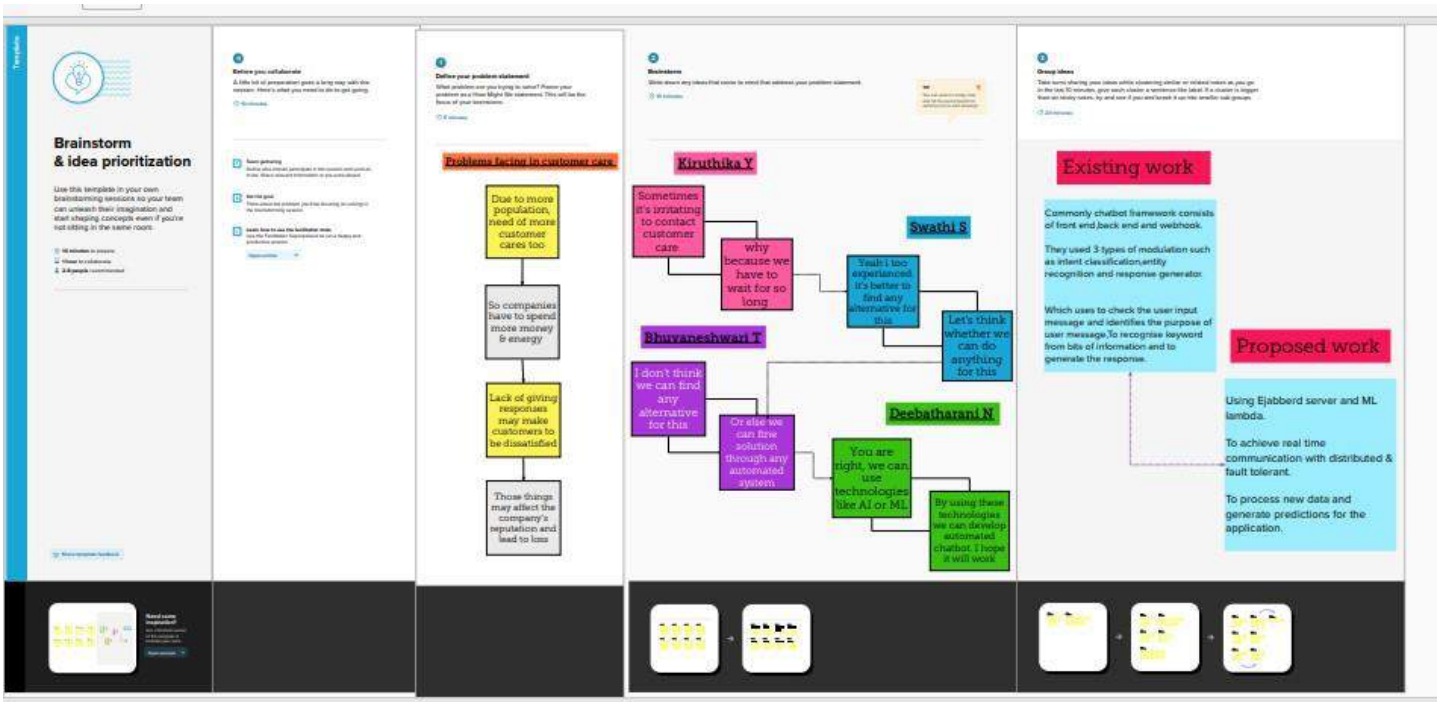
USER : They can register for an account. After the login, they can create the complaint with a description of the problem they are facing. Each user will be assigned with an agent. They can view the status of their complaint.

3. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas



b. Ideation & Brainstorming



c. Proposed Solution

Proposed solution :

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	• If a customer has any problem in a product they bought,they will approach the company's customer care.Customer care can contact with 1 or 2 customer at a time.But if there are more customers approaching at the same time,sometimes it is difficult to understand and rectify their issues. • It may make customer irritating or lack of good interaction to that company. • At the same time,Making more number of customer service sometimes may lead to loss and need more maintanance.
2.	Idea / Solution description	• Introduce a Automated chatbot will be a good solution for the customer care issues • It is Automated techmology,so no needed to make more customer services for a company. • It can interact with many people at a time. • It will reduce cost spending for customer service and also reduce works.
3.	Novelty / Uniqueness	• It is different from other normal chatbots.Informations won't disappear,if network issue occurs. • This Chatbot will be loaded with so many information about products. • It will be interactive and interesting.
4.	Society Impact / Customer Satisfaction	• If a problem is rectified quickly,customers will be satisfied. It may also increases trust on that company. • Customers don't have to wait or spend so much time for this.
5.	Business Model (Revenue Model)	• It's a chatbot which can directly interact with customers and help them rectify their issues. • Due to this automated chatbot, company's income will increase due to the less care about the customers service and customer will be increasing due to the good interaction with them.

Problem solution fit:

Proposed solution :

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	• If a customer has any problem in a product they bought,they will approach the company's customer care.Customer care can contact with 1 or 2 customer at a time.But if there are more customers approaching at the same time,sometimes it is difficult to understand and rectify their issues. • It may make customer irritating or lack of good interaction to that company. • At the same time,Making more number of customer service sometimes may lead to loss and need more maintanance.
2.	Idea / Solution description	• Introduce a Automated chatbot will be a good solution for the customer care issues • It is Automated technology,so no needed to make more customer services for a company. • It can interact with many people at a time. • It will reduce cost spending for customer service and also reduce works.
3.	Novelty / Uniqueness	• It is different from other normal chatbots.Informations won't disappear,if network issue occurs. • This Chatbot will be loaded with so many information about products. • It will be interactive and interesting.
4.	Society Impact / Customer Satisfaction	• If a problem is rectified quickly,customers will be satisfied. It may also increases trust on that company. • Customers don't have to wait or spend so much time for this.
5.	Business Model (Revenue Model)	• It's a chatbot which can directly interact with customers and help them rectify their issues. • Due to this automated chatbot, company's income will increase due to the less care about the customers service and customer will be increasing due to the good interaction with them.

Requirement analysis:

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 is done through the mobile verification and gmail code verification
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP Confirmation via receiving the SMS
FR-3	User Login	User can enter their login credential like username and password
FR-4	Enter your expense	Collects the users expense data with the date and time included
FR-5	Expense Report is generated	Users data can be represented in the pdf format and graphical manner to understand the report
FR-6	Categories and type of expenses	This app can add more features regarding the various expenses and better USER INTERFACE to attract the users

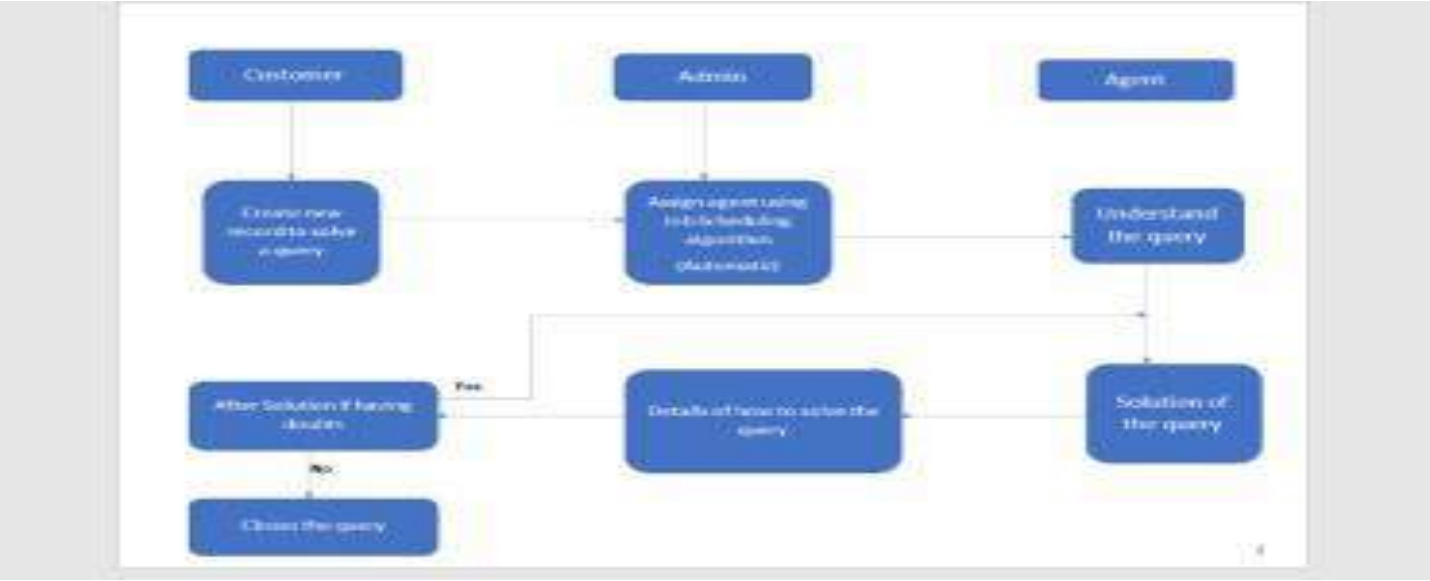
Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	This app is user friendly and the user interface is more attractive and track all the expense in day-today
NFR-2	**Security**	This app provides the security because of cloud storage and protected by registering by single username and password and verifying by the mobile numbers
NFR-3	**Reliability**	This app can access by anytime and anywhere
NFR-4	**Performance**	Greater efficiency and performance is high and the data using this app is very less
NFR-5	**Availability**	This application can be accessed at any time
NFR-6	**Scalability**	We can attract the user by the attractive UI and the storage of the data is high

Project Design:

Data flow diagram:



Solution & Technical architecture:

Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.

Its goals are to:

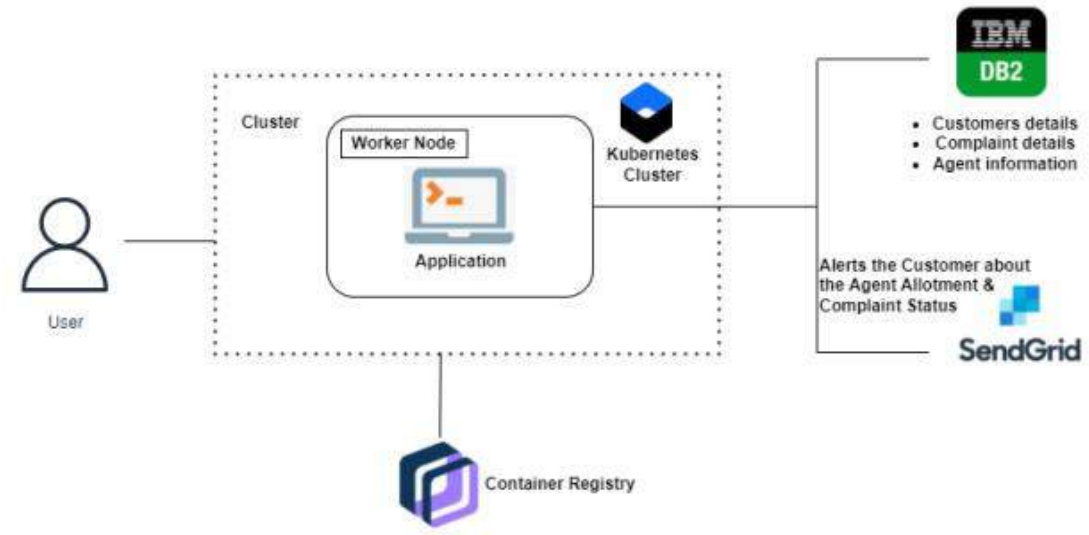
• Find the best tech solution to solve existing business problems.

• Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.

• Define features, development phases, and solution requirements.

• Provide specifications according to which the solution is defined, managed, and delivered.

Example - Solution Architecture Diagram:



Reference :

https://www.researchgate.net/publication/257344357_Information_technology_help_desk_survey_To_identify_the_classification_of_simple_and_routine_enquiries

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 (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through any social medias	I can register & access the dashboard with social media login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can access through the gmail account login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	Can access this application	High	Sprint-1
	Dashboard	USN-6	As a user they can seen their daily expenses	Monitor their money where and when they spend	High	Sprint-1
Customer (Web user)		USN-7	Web user can access this application by their login credentials	Valid login Credentials can be accepted	High	Sprint-1
Customer Care Executive		USN-8	Any feedback and reports received from the customer can be solved within a week	We can provide the toll free number for the user 24x7	High	Sprint-1
Administrator		USN-9	We can update this application every month and introduce more features	We can fix the bug in this application	Medium	Sprint-1

6. Project planning & Scheduling:

Sprint planning & Estimation:

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

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Deebatharani.N Bhuvaneshwari T
Sprint-1	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Kiruthika Y Swathi S
Sprint-2	Registration	USN-3	As a user, I can register for the application through the link	2	Low	Deebatharani N Kiruthika Y

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Registration	USN-4	As a user, I can register for the application through Gmail or google account	2	Medium	Bhuvaneshwari T Swathi S
Sprint-1	Registration	USN-5	As a user, I can log into the application by entering email & password	1	High	Deebatharani N Swathi S
Sprint-1	Login	USN-6	As a user, I can text my problem to the applicatiom	1	High	Bhuvaneshwari T Kiruthika Y
Sprint-3	Dashboard	USN-7	As a user, I can provide necessary details and answer the questions in the web application	2	Low	Kiruthika Y Swathi S
Sprint-3	Dashboard	USN-8	As a user, I can able to find solutions for my issues regards the products	2	High	Deebatharani N Bhuvaneshwari T Kiruthika Y
Sprint-3	Dashboard	USN-9	As a Administer, I will update our web application with the information regards products	2	High	Kiruthika Y Swathi S

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Dashboard	USN-10	As a User, I will maintain user's details carefully	1	Low	Deebatharani N Bhuvaneshwari T
Sprint-4	Management	USN-11	As a administrator, I will collect the datas ans keep the application well interactive	2	Low	Kiruthika Y Swathi s
Sprint-4	Management	USN-12	As a Administrator, I can maintain the user's privacy safely.	2	High	Deebatharani N Bhuvaneshwari T

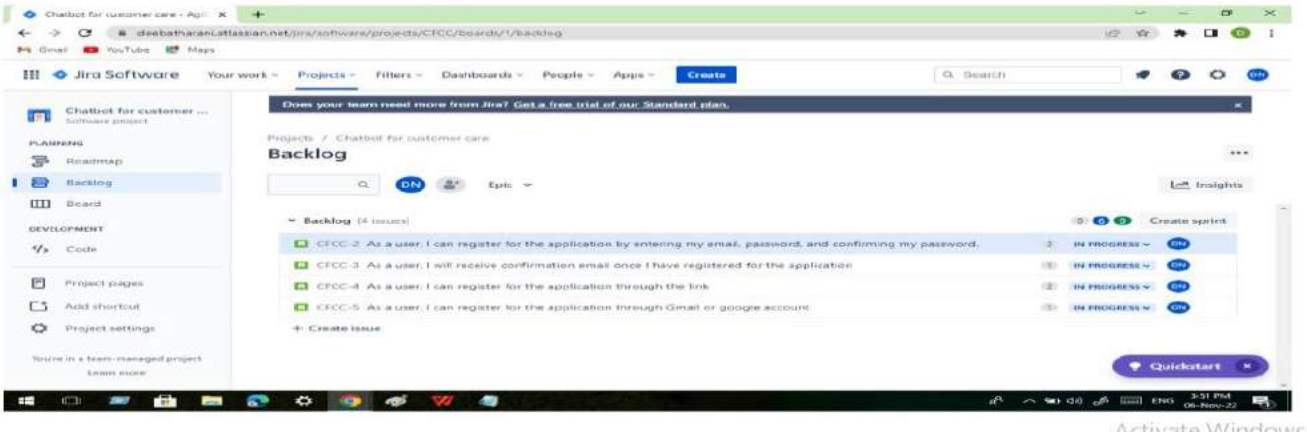
Sprint Delivary schedule:

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	6	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	6	6 Days	31 Oct 2022	05 Nov 2022	6	05 Nov 2022
Sprint-3	6	6 Days	07 Nov 2022	12 Nov 2022	6	12 Nov 2022
Sprint-4	6	6 Days	14 Nov 2022	19 Nov 2022	6	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)

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

Reports from jira:



Coding & Solutioning :

a. User Dashboard

By using user dashboard user can view thier status of tickets and can create new ticket or query.

code:

```
{% extends 'base.html' %}
 {% block head %}
<title>
 Dashboard
</title>
{% endblock %}
 {% block body %}
<br>
<!-- <br>
{% for i in range(11) %}
  {{ i }}
{% endfor %}
 <br>
{% for i in complaints %}
{{ i['USERNAME'] }}
<br>
```

```
{% for j in i.values() %}
   {{ j }}
{% endfor %}
<br>
{% endfor %} -->


<div class="fordashboardtop">
   <div class="fordashboardtopelements1">
      Welcome {{ name }},
   </div>
   <div class="fordashboardtopelements2">
      <a href="/login"><button class="forbutton">Sign out</button></a>
   </div>
 </div>
<br>
<div class="outerofdashdetails">
   <div class="fordashboarddetails">
      <br>
      <!-- table of customers complaints -->
      <table class="fortable">
         <thead>
            <th>Complaint ID</th>
            <th class="pad">Complaint Detail</th>
            <th>Assigned Agent</th>
            <th>Status</th>
            <th>Solution</th>
         </thead>
         <tbody>
            {% for i in complaints %}
            <tr>
               <td>
                  {{ i['C_ID'] }}
```

```
            </td>
            <td class="pad">
                {{ i['TITLE'] }}
            </td>
            <td>
                {{ i['ASSIGNED_AGENT'] }}
            </td>
            <td>
                {% if i['STATUS'] == 1 %}
                Completed
                {% elif i['STATUS'] == 0 %}
                Not completed
                {% else %}
                In progress
                {% endif %}
            </td>
            <td>
                {{ i['SOLUTION'] }}
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
 <br>
    <center>
    <div class="fordashboarddetails">
     <button type="button" class="collapsible">Add new complaint ✚</button>
        <div class="content">
            <br>
            <form action="/addnew" method="post">
                <div class="forform">
```

```
                    <div class="textinformleft">
                        Title
                    </div>
                    <div class="textinformright">
                        <input type="name" name="title">
                    </div>
                </div>
                <div class="forform">
                    <div class="textinformleft">
                        Complaint
                    </div>
                    <div class="textinformright">
                        <textarea name="des" style="border-radius: 1rem;width: 90%;height: 150%;background-color: black;color: white;"></textarea>
                    </div>
                </div>
                <br>
                <br>
                <div>
                    <button class="forbutton" type="submit"> Submit </button>
                </div>
            </form>
            <br>
        </div>
    </div>
    </center>
  </div>
</div>
 {% endblock %}
```

b. Agent Dashboard

By using agent dashboard agent can view thier status of tickets and can solve new ticket or query.

Code:

```
{% extends 'base.html' %}
{% block head %}
<title>
   Agent Dashboard
</title>
{% endblock %}
{% block body %}
 <br>
 <div class="fordashboardtop">
   <div class="fordashboardtopelements1">
      Welcome {{ name }},
   </div>
   <div class="fordashboardtopelements2">
      <a href="/login"><button class="forbutton">Sign out</button></a>
   </div>
 </div>
<br>
<div class="outerofdashdetails">
  <div class="fordashboarddetails">
      <br>
      <!-- table of customers complaints -->
      <table class="fortable">
         <thead>
            <th>Complaint ID</th>
            <th class="pad">Username</th>
            <th>Title</th>
            <th>Complaint</th>
            <th>Solution</th>
```

```
        <th>Status</th>
    </thead>
    <tbody>
        {% for i in complaints %}
        <tr>
            <td class="pad">
                {{ i['C_ID'] }}
            </td>
            <td class="pad">
                {{ i['USERNAME'] }}
            </td>
            <td>
                {{ i['TITLE'] }}
            </td>
            <td>
                {{ i['COMPLAINT'] }}
            </td>
            <td>
                {{ i['SOLUTION'] }}
            </td>
            <td>
                {% if i['STATUS'] == 1 %}
                Completed
                {% else %}
                Not Completed
                {% endif %}
            </td>
        </tr>
        {% endfor %}
    </tbody>

</table>
```

```
<br>
   <center>
  <div class="fordashboarddetails">
   <button type="button" class="collapsible">Solve an Issue ⚡</button>
        <div class="content">
           <br>
           <form action="/updatecomplaint" method="post">
              <div class="forform">
                 <div class="textinformleft">
                    Complaint ID
                 </div>
                 <div class="textinformright">
                    <input type="name" name="cid">
                 </div>
              </div>
              <div class="forform">
                 <div class="textinformleft">
                    Solution
                 </div>
                 <div class="textinformright">
                    <input type="text" name="solution">
                 </div>
              </div>
              <br>
              <br>
              <div>
                 <button class="forbutton" type="submit"> Submit </button>
              </div>
           </form>
           <br>
        </div>
```

```
        </div>
      </center>
   </div>
</div>
 {% endblock %}
```

c. Admin Dashboard

Admin can add new agent and can assign an agent for customer's query.

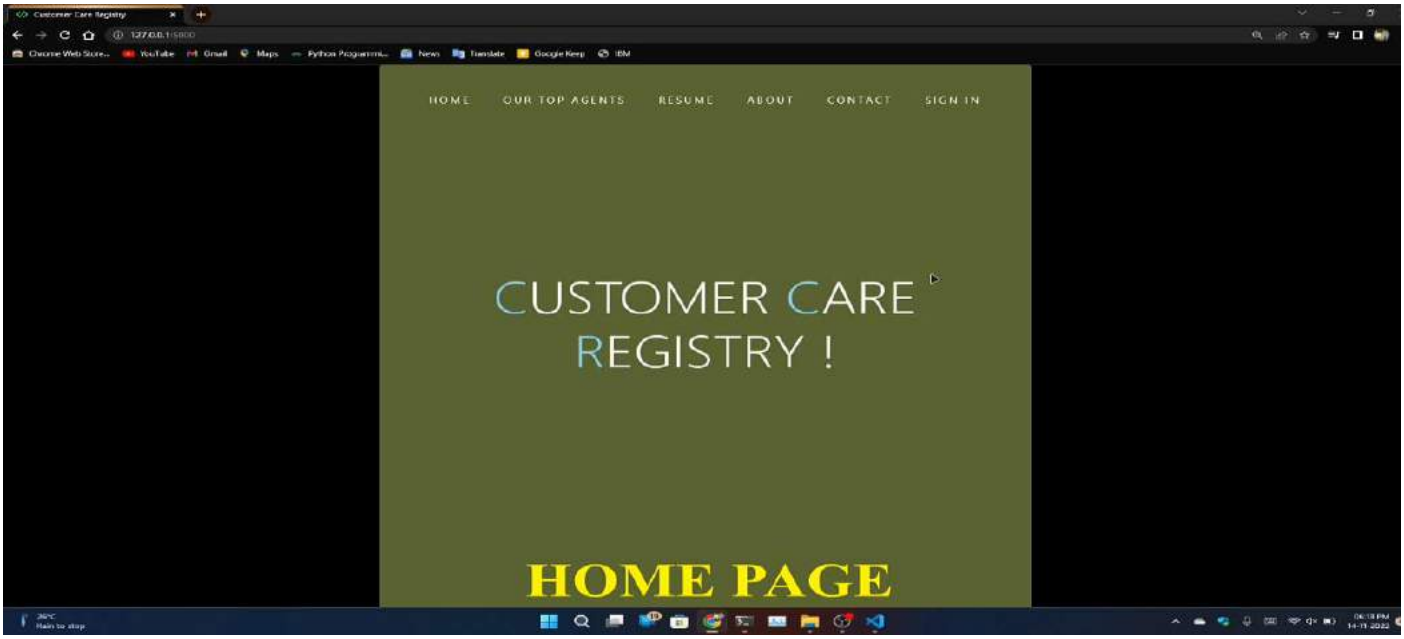
8. TESTING

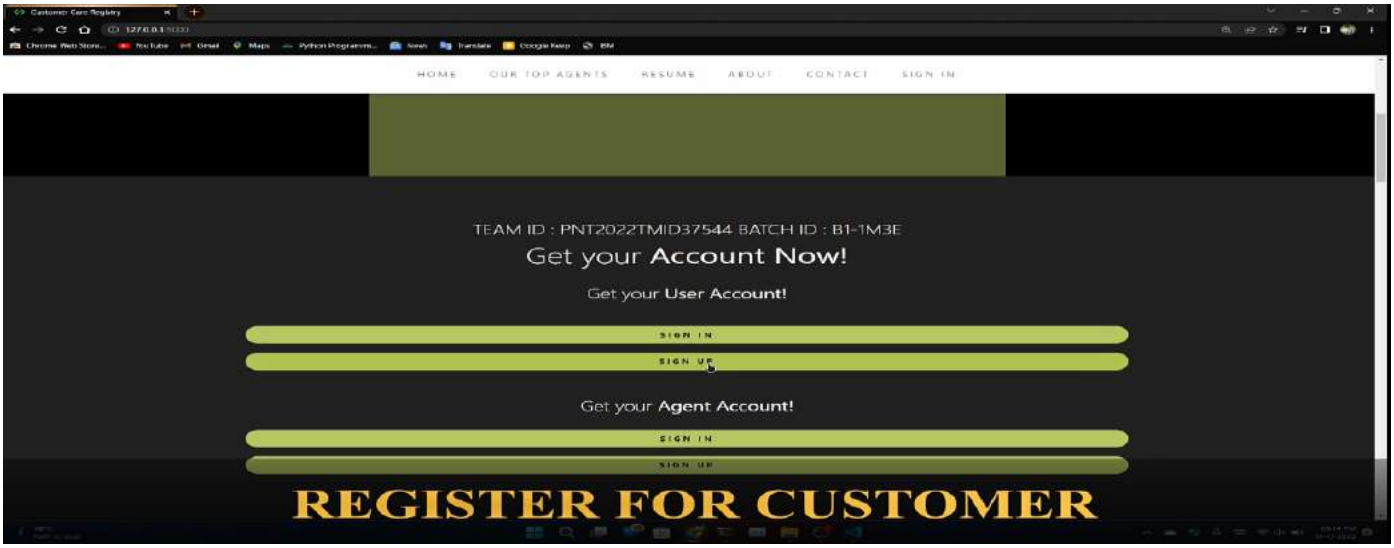
Code:

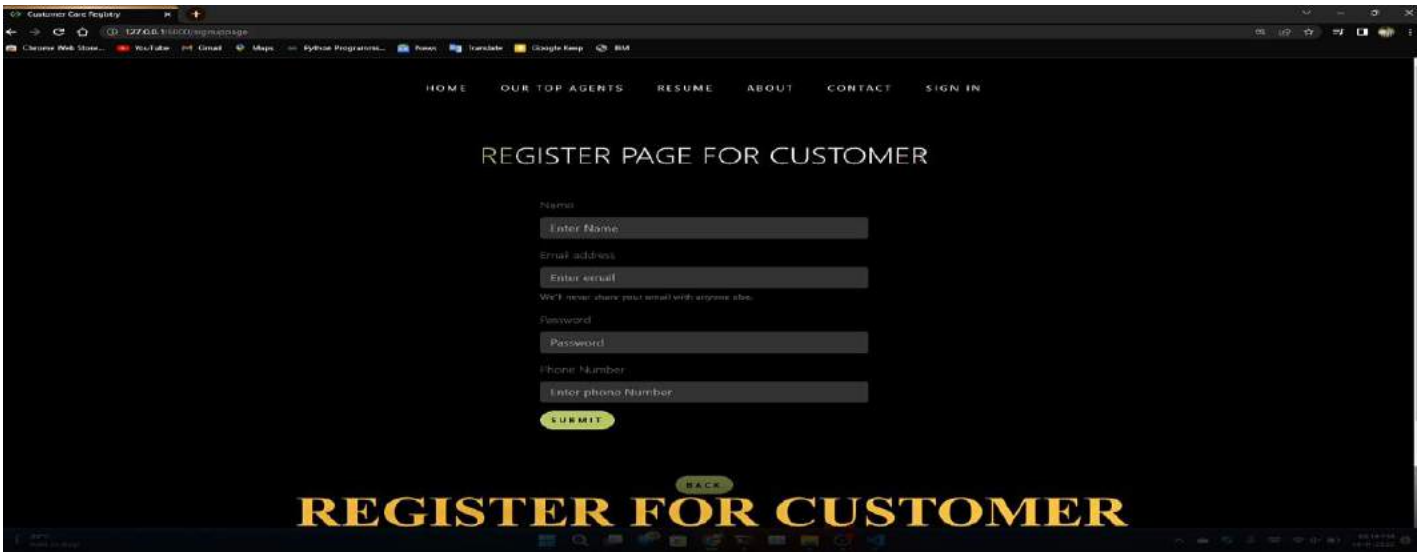
```
{% extends 'base.html' %}
 {% block head %}
<title>
   Admin Dashboard
</title>
{% endblock %}
{% block body %}
 <div class="fordashboardtop">
   <div class="fordashboardtopelements1">
      Welcome Admin,
   </div>
   <div class="fordashboardtopelements2">
      <a href="/login"><button class="forbutton">Sign out</button></a>
   </div>
 </div>
<br>
<div class="outerofdashdetails">
  <div class="fordashboarddetails">
      <br>
      <!-- table of customers complaints -->
```

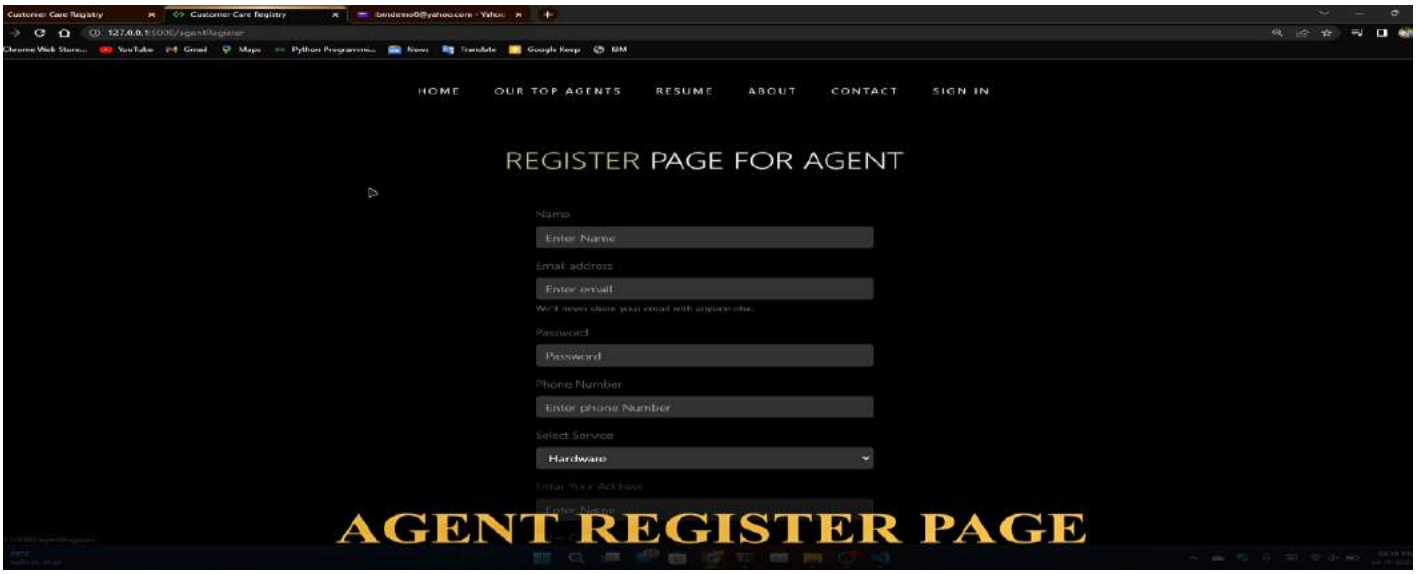
```
        <table class="fortable">
            <thead>
            </thead>
            <tbody>
                <tr>
                    <td class="pad">
                        <a href="/agents">Agent Details</a>
                    </td>
                    <td class="pad">
                        <a href="/tickets">Customer Ticket Details</a>
                    </td>
                </tr>
            </tbody>
        </table>
    <br>
    </div>
</div>
{% endblock %}
```

9. Results

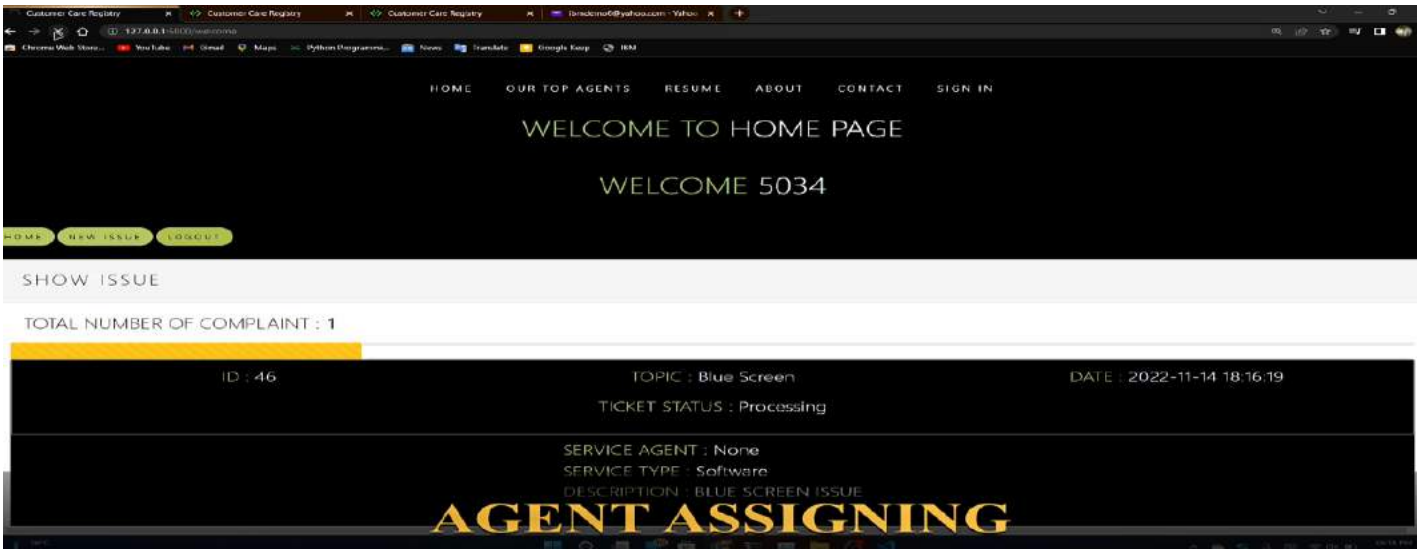


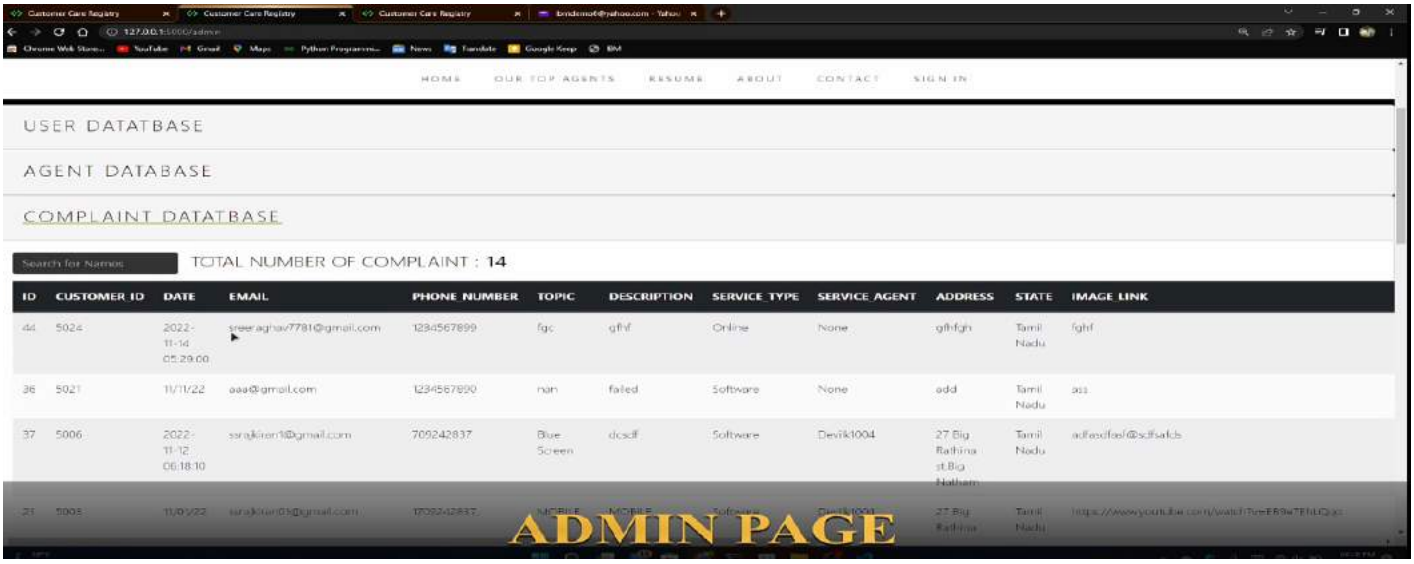

HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
TEAM ID : PNT2022TMID37544 BATCH ID : B1-1M3E
Get your Account Now!
Get your User Account!
SIGN IN
SIGN UP
Get your Agent Account!
SIGN IN
SIGN UP
REGISTER FOR CUSTOMER


HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
REGISTER PAGE FOR CUSTOMER
Name
Enter Name
Email address
Enter email
Password
Password
Phone Number
Enter phone Number
SUBMIT
BACK
REGISTER FOR CUSTOMER


HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
REGISTER PAGE FOR AGENT
Name
Enter Name
Email address
Enter email
Password
Password
Phone Number
Enter phone Number
Select Service
Hardware
AGENT REGISTER PAGE


HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
COMPLAINT PAGE FOR CUSTOMER
CUSTOMER ID
5034
Email address
Enter email
Phone Number
Enter phone Number
Select Service
Select one of these...
Enter topic
Enter Name
COMPLAINT REGISTER


HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
WELCOME TO HOME PAGE
WELCOME 5034
HOME
NEW ISSUE
LOGOUT
SHOW ISSUE
TOTAL NUMBER OF COMPLAINT : 1
ID : 46
TOPIC : Blue Screen
DATE : 2022-11-14 18:16:19
TICKET STATUS : Processing
SERVICE AGENT : None
SERVICE TYPE : Software
DESCRIPTION : BLUE SCREEN ISSUE
AGENT ASSIGNING


HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
USER DATATBASE
AGENT DATABASE
COMPLAINT DATATBASE
Search for Names
TOTAL NUMBER OF COMPLAINT : 14
ID
CUSTOMER_ID
DATE
EMAIL
PHONE_NUMBER
TOPIC
DESCRIPTION
SERVICE_TYPE
SERVICE_AGENT
ADDRESS
STATE
IMAGE_LINK
ADMIN PAGE

Advantages and Disadvantages :

Advantages:

By using this application user can query using online.

Disadvantages:

Sometimes application may not work due to internet problems.

11. CONCLUSION

Hence, customer care registry as a cloud app will definitely highly useful those who can't visit the shop for their queries.

12. FUTURE SCOPE

This app can be further optimized to solve quries much better.

13. GitHub & Project Demo Link

a. Github Link: https://github.com/IBM-EPBL/IBM-Project-48298-1660806417

b. Project Demo Link: https://youtu.be/x9jcTL7nJ0M

c. Application Link: http://169.51.204.215:30106/signinpage

Source Code

app.py:

```
 from flask import Flask, render_template, request, redirect, session, url_for
import ibm_db_sa
import ibm_db
import re
 app = Flask(__name__)
 # for connection
# conn= ""
```

```
app.secret_key = 'a'

print("Trying to connect...")

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30119;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=qvk70423;PWD=saDlGasU4iQy1yvk;", '', '')

print("connected..")

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

def signup():
    global userid
    msg = ''
    if request.method == 'POST':
        username = request.form['username']
        name = request.form['name']
        email = request.form['email']
        phn = request.form['phn']
        password = request.form['pass']
        repass = request.form['repass']
        print("inside checking")
        print(name)
        if len(username) == 0 or len(name) == 0 or len(email) == 0 or len(phn) == 0 or len(password) == 0 or len(repass) == 0:
            msg = "Form is not filled completely!!"
            print(msg)
            return render_template('signup.html', msg=msg)
        elif password != repass:
            msg = "Password is not matched"
            print(msg)
            return render_template('signup.html', msg=msg)
        elif not re.match(r'[a-z]+', username):
            msg = 'Username can contain only small letters and numbers'
            print(msg)
            return render_template('signup.html', msg=msg)
```

```
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
    msg = 'Invalid email'
    print(msg)
    return render_template('signup.html', msg=msg)
elif not re.match(r'[A-Za-z]+', name):
    msg = "Enter valid name"
    print(msg)
    return render_template('signup.html', msg=msg)
elif not re.match(r'[0-9]+', phn):
    msg = "Enter valid phone number"
    print(msg)
    return render_template('signup.html', msg=msg)
 sql = "select * from users where username = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
print(account)
if account:
    msg = 'Acccount already exists'
else:
    userid = username
    insert_sql = "insert into users values(?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, username)
    ibm_db.bind_param(prep_stmt, 2, name)
    ibm_db.bind_param(prep_stmt, 3, email)
    ibm_db.bind_param(prep_stmt, 4, phn)
    ibm_db.bind_param(prep_stmt, 5, password)
    ibm_db.execute(prep_stmt)
    print("successs")
    msg = "succesfully signed up"
```

```
        return render_template('dashboard.html', msg=msg, name=name)
    else:
        return render_template('signup.html')
@app.route('/dashboard')
def dashboard():
    return render_template('dashboard.html')
@app.route('/')
def base():
    return redirect(url_for('login'))
@app.route('/login', methods=["GET", "POST"])
def login():
    global userid
    msg = ''
    if request.method == 'POST':
        username = request.form['username']
        userid = username
        password = request.form['pass']
        if userid == 'admin' and password == 'admin':
            print("its admin")
            return render_template('admin.html')
        else:
            sql = "select * from agents where username = ? and password = ?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, password)
            ibm_db.execute(stmt)
            account = ibm_db.fetch_assoc(stmt)
            print(account)
            if account:
                session['Loggedin'] = True
                session['id'] = account['USERNAME']
                userid = account['USERNAME']
```

```
        session['username'] = account['USERNAME']
        msg = 'logged in successfully'
        # for getting complaints details
        sql = "select * from complaints where assigned_agent = ?"
        complaints = []
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        dictionary = ibm_db.fetch_assoc(stmt)
        while dictionary != False:
            complaints.append(dictionary)
            dictionary = ibm_db.fetch_assoc(stmt)
        print(complaints)
        return render_template('agentdash.html', name=account['USERNAME'], complaints=complaints)
sql = "select * from users where username = ? and password = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, password)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
print(account)
if account:
    session['Loggedin'] = True
    session['id'] = account['USERNAME']
    userid = account['USERNAME']
    session['username'] = account['USERNAME']
    msg = 'logged in successfully'
    # for getting complaints details
    sql = "select * from complaints where username = ?"
    complaints = []
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
```

```
            ibm_db.execute(stmt)
            dictionary = ibm_db.fetch_assoc(stmt)
            while dictionary != False:
                # print "The ID is : ",  dictionary["EMPNO"]
                # print "The Name is : ", dictionary[1]
                complaints.append(dictionary)
                dictionary = ibm_db.fetch_assoc(stmt)
            print(complaints)
            return render_template('dashboard.html', name=account['USERNAME'], complaints=complaints)
        else:
            msg = 'Incorrect user credentials'
            return render_template('dashboard.html', msg=msg)
    else:
        return render_template('login.html')
@app.route('/addnew', methods=["GET", "POST"])
def add():
    if request.method == 'POST':
        title = request.form['title']
        des = request.form['des']
        try:
            sql = "insert into complaints(username,title,complaint) values(?,?,?)"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, userid)
            ibm_db.bind_param(stmt, 2, title)
            ibm_db.bind_param(stmt, 3, des)
            ibm_db.execute(stmt)
        except:
            print(userid)
            print(title)
            print(des)
            print("cant insert")
        sql = "select * from complaints where username = ?"
```

```
        complaints = []
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, userid)
        ibm_db.execute(stmt)
        dictionary = ibm_db.fetch_assoc(stmt)
        while dictionary != False:
            # print "The ID is : ",  dictionary["EMPNO"]
            # print "The Name is : ", dictionary[1]
            complaints.append(dictionary)
            dictionary = ibm_db.fetch_assoc(stmt)
        print(complaints)
        return render_template('dashboard.html', name=userid, complaints=complaints)
 @app.route('/agents')
def agents():
    sql = "select * from agents"
    agents = []
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    while dictionary != False:
        agents.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
    return render_template('agents.html', agents=agents)
 @app.route('/addnewagent', methods=["GET", "POST"])
def addagent():
    if request.method == 'POST':
        username = request.form['username']
        name = request.form['name']
        email = request.form['email']
        phone = request.form['phone']
        domain = request.form['domain']
        password = request.form['password']
```

```
        try:
            sql = "insert into agents values(?,?,?,?,?,?,2)"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, name)
            ibm_db.bind_param(stmt, 3, email)
            ibm_db.bind_param(stmt, 4, phone)
            ibm_db.bind_param(stmt, 5, password)
            ibm_db.bind_param(stmt, 6, domain)
            ibm_db.execute(stmt)
        except:
            print("cant insert")
        sql = "select * from agents"
        agents = []
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.execute(stmt)
        dictionary = ibm_db.fetch_assoc(stmt)
        while dictionary != False:
            agents.append(dictionary)
            dictionary = ibm_db.fetch_assoc(stmt)
        return render_template('agents.html', agents=agents)
 @app.route('/updatecomplaint', methods=["GET", "POST"])
def updatecomplaint():
    if request.method == 'POST':
        cid = request.form['cid']
        solution = request.form['solution']
        try:
            sql = "update complaints set solution =?,status=1 where c_id = ? and assigned_agent=?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, solution)
            ibm_db.bind_param(stmt, 2, cid)
            ibm_db.bind_param(stmt, 3, userid)
```

```
            ibm_db.execute(stmt)
            sql = "update agents set status =3 where username=?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, userid)
            ibm_db.execute(stmt)
        except:
            print("cant insert")
        sql = "select * from complaints where assigned_agent = ?"
        complaints = []
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, userid)
        ibm_db.execute(stmt)
        dictionary = ibm_db.fetch_assoc(stmt)
        while dictionary != False:
            complaints.append(dictionary)
            dictionary = ibm_db.fetch_assoc(stmt)
        # print(complaints)
        return render_template('agentdash.html', name=userid, complaints=complaints)
 @app.route('/tickets')
def tickets():
    sql = "select * from complaints"
    complaints = []
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    while dictionary != False:
        complaints.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
   sql = "select username from agents where status <> 1"
    freeagents = []
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
```

```
    dictionary = ibm_db.fetch_assoc(stmt)
    while dictionary != False:
        freeagents.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
    print(freeagents)
    return render_template('tickets.html', complaints=complaints, freeagents=freeagents)
 @app.route('/assignagent', methods=['GET', 'POST'])
def assignagent():
    if request.method == "POST":
        ccid = request.form['ccid']
        agent = request.form['agent']
        print(ccid)
        print(agent)
        try:
            sql = "update complaints set assigned_agent =? where c_id = ?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, agent)
            ibm_db.bind_param(stmt, 2, ccid)
            ibm_db.execute(stmt)
            sql = "update agents set status =1 where username = ?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, userid)
            ibm_db.execute(stmt)
        except:
            print("cant update")
        return redirect(url_for('tickets'))
 if __name__ == "__main__":
    app.run(debug=False)
```

Dockerfile

```
FROM python:3.6
WORKDIR /app
ADD . /app
COPY requirements.txt /app
RUN python3 -m pip install -r requirements.txt
RUN python3 -m pip install ibm_db
EXPOSE 5000
CMD ["python","app.py"]
```

flaskapp.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: flaskapp
spec:
  selector:
      app: flaskapp
  ports:
  - port: 5000
  type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flaskapp
  labels:
    app: flaskapp
spec:
  selector:
    matchLabels:
      app: flaskapp
```

```
  replicas: 1
  template:
    metadata:
      labels:
        app: flaskapp
    spec:
      containers:
      - name: flaskapp
        image: au.icr.io/customer-care-ibm/customer-care
        ports:
        - containerPort: 5000
        env:
        - name: DISABLE_WEB_APP
          value: "false"
```

requirements.txt:

```
Flask
ibm_db
```

admin.html:

```
{% extends 'base.html' %}
 {% block head %}
 <title>
    Admin Dashboard
</title>



{% endblock %}
```

```
{% block body %}



<div class="fordashboardtop">
    <div class="fordashboardtopelements1">
        Welcome Admin,
    </div>
    <div class="fordashboardtopelements2">
        <a href="/login"><button class="forbutton">Sign out</button></a>
    </div>

</div>
<br>
<div class="outerofdashdetails">

    <div class="fordashboarddetails">
        <br>
        <!-- table of customers complaints -->
        <table class="fortable">
            <thead>
            </thead>
            <tbody>
                <tr>
                    <td class="pad">
                        <a href="/agents">Agent Details</a>
                    </td>
                    <td class="pad">
                        <a href="/tickets">Customer Ticket Details</a>
                    </td>
                </tr>
```

```
            </tbody>

        </table>

        <br>

    </div>

</div>

{% endblock %}
```

agentdash.html:

```
{% extends 'base.html' %}



{% block head %}

<title>
    Agent Dashboard
</title>



{% endblock %}



{% block body %}
```

```
<br>



<div class="fordashboardtop">
    <div class="fordashboardtopelements1">
        Welcome {{ name }},
    </div>
    <div class="fordashboardtopelements2">
        <a href="/login"><button class="forbutton">Sign out</button></a>
    </div>
 </div>
<br>
<div class="outerofdashdetails">
     <div class="fordashboarddetails">
       <br>
        <!-- table of customers complaints -->
        <table class="fortable">
            <thead>
                <th>Complaint ID</th>
                <th class="pad">Username</th>
                <th>Title</th>
                <th>Complaint</th>
                <th>Solution</th>
                <th>Status</th>
            </thead>
            <tbody>
                {% for i in complaints %}
                <tr>
                    <td class="pad">
                        {{ i['C_ID'] }}
                    </td>
                    <td class="pad">
```

```
            {{ i['USERNAME'] }}
          </td>
          <td>
            {{ i['TITLE'] }}
          </td>
          <td>
            {{ i['COMPLAINT'] }}
          </td>
          <td>
            {{ i['SOLUTION'] }}
          </td>
          <td>
            {% if i['STATUS'] == 1 %}
            Completed
            {% else %}
            Not Completed
            {% endif %}
          </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
<br>
<center>
  <div class="fordashboarddetails">
       <button type="button" class="collapsible">Solve an Issue ⚡</button>
      <div class="content">
        <br>
        <form action="/updatecomplaint" method="post">
          <div class="forform">
            <div class="textinformleft">
```

```
                    Complaint ID
                </div>
                <div class="textinformright">
                    <input type="name" name="cid">
                </div>
            </div>
            <div class="forform">
                <div class="textinformleft">
                    Solution
                </div>
                <div class="textinformright">
                    <input type="text" name="solution">
                </div>
            </div>
            <br>
            <br>
            <div>
                <button class="forbutton" type="submit"> Submit </button>
            </div>
        </form>
        <br>
    </div>
  </div>
 </center>
</div>
</div>
{% endblock %}
```

agents.html:

```
{% extends 'base.html' %}
{% block head %}
```

```
 <title>
    Dashboard
</title>



 {% endblock %}
 {% block body %}
 <!-- things
   div 1
welcome jetson,   sign out
    div 2
your complaints status
 add new complaint -->
<br>
<!-- <br>
{% for i in range(11) %}
  {{ i }}
{% endfor %}
 <br>
{% for i in complaints %}
{{ i['USERNAME'] }}
<br>
{% for j in i.values() %}
    {{ j }}
{% endfor %}
<br>
{% endfor %} -->
 <div class="fordashboardtop">
    <div class="fordashboardtopelements1">
        Welcome Admin,
    </div>
    <div class="fordashboardtopelements2">
```

```
        <a href="/login"><button class="forbutton">Sign out</button></a>
    </div>
 </div>
<br>
<div class="outerofdashdetails">
    <div class="fordashboarddetails">
        <br>
        <!-- table of customers complaints -->
        <table class="fortable">
            <thead>
                <th class="pad">Name</th>
                <th>Username</th>
                <th>Email</th>
                <th>Phone</th>
                <th>Domain</th>
                <th>Status</th>
            </thead>
            <tbody>
                {% for i in agents %}
                <tr>
                    <td class="pad">
                        {{ i['NAME'] }}
                    </td>
                    <td class="pad">
                        {{ i['USERNAME'] }}
                    </td>
                    <td>
                        {{ i['EMAIL'] }}
                    </td>
                    <td>
                        {{ i['PHN'] }}
                    </td>
```

```
            <td>
                {{ i['DOMAIN'] }}
            </td>
            <td>
                {% if i['STATUS'] == 1 %}
                Assigned to job
                {% elif i['STATUS'] == 0 %}
                not Available
                {% else %}
                Available
                {% endif %}
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
<br>
<center>
    <div class="fordashboarddetails">
        <button type="button" class="collapsible">Add new agent ✚</button>
        <div class="content">
            <br>
            <form action="/addnewagent" method="post">
                <div class="forform">
                    <div class="textinformleft">
                        Username
                    </div>
                    <div class="textinformright">
                        <input type="name" name="username">
                    </div>
                </div>
```

```
<div class="forform">
    <div class="textinformleft">
        Name
    </div>
    <div class="textinformright">
        <input type="name" name="name">
    </div>
</div>
<div class="forform">
    <div class="textinformleft">
        Email
    </div>
    <div class="textinformright">
        <input type="name" name="email">
    </div>
</div>
<div class="forform">
    <div class="textinformleft">
        Phone
    </div>
    <div class="textinformright">
        <input type="name" name="phone">
    </div>
</div>
<div class="forform">
    <div class="textinformleft">
        Domain
    </div>
    <div class="textinformright">
        <input type="name" name="domain">
    </div>
</div>
```

```
                <div class="forform">
                    <div class="textinformleft">
                        Password
                    </div>
                    <div class="textinformright">
                        <input type="password" name="password">
                    </div>
                </div>
                <br>
                <br>
                <div>
                    <button class="forbutton" type="submit"> Submit </button>
                </div>
            </form>
            <br>
        </div>
    </div>
  </center>
 </div>
</div>
{% endblock %}
```

base.html:

```
<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="static/css/main.css"/>
  {% block head %}
  {% endblock %}
</head>
<body>
  {% block body %}
```

```
    {% endblock %}
    <script>
        var coll = document.getElementsByClassName("collapsible");
        var i;
        for (i = 0; i < coll.length; i++) {
            coll[i].addEventListener("click", function () {
                this.classList.toggle("active");
                var content = this.nextElementSibling;
                if (content.style.display === "block") {
                    content.style.display = "none";
                } else {
                    content.style.display = "block";
                }
            });
        }
    </script>
    <footer style="text-align: right ;">
        <a href="/about">Wanna know more about us? Click here</a>
    </footer>
</body>
 </html>
```

dashboard.html:

```
{% extends 'base.html' %}
{% block head %}
 <title>
    Dashboard
</title>
 {% endblock %}
 {% block body %}
 <!-- things
```

```
   div 1
welcome jetson,   sign out


   div 2
your complaints status
 add new complaint -->
<br>
<!-- <br>
{% for i in range(11) %}
  {{ i }}
{% endfor %}
 <br>
{% for i in complaints %}
{{ i['USERNAME'] }}
<br>
{% for j in i.values() %}
   {{ j }}
{% endfor %}
<br>
{% endfor %} -->
 <div class="fordashboardtop">
   <div class="fordashboardtopelements1">
      Welcome {{ name }},
   </div>
   <div class="fordashboardtopelements2">
      <a href="/login"><button class="forbutton">Sign out</button></a>
   </div>
 </div>
<br>
<div class="outerofdashdetails">
   <div class="fordashboarddetails">
      <br>
```

```
<!-- table of customers complaints -->
<table class="fortable">
    <thead>
        <th>Complaint ID</th>
        <th class="pad">Complaint Detail</th>
        <th>Assigned Agent</th>
        <th>Status</th>
        <th>Solution</th>
    </thead>
    <tbody>
        {% for i in complaints %}
        <tr>
            <td>
                {{ i['C_ID'] }}
            </td>
            <td class="pad">
                {{ i['TITLE'] }}
            </td>
            <td>
                {{ i['ASSIGNED_AGENT'] }}
            </td>
            <td>
                {% if i['STATUS'] == 1 %}
                Completed
                {% elif i['STATUS'] == 0 %}
                Not completed
                {% else %}
                In progress
                {% endif %}
            </td>
            <td>
                {{ i['SOLUTION'] }}
```

```
            </td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
      <br>
      <center>
        <div class="fordashboarddetails">
          <button type="button" class="collapsible">Add new complaint ✚</button>
          <div class="content">
            <br>
            <form action="/addnew" method="post">
              <div class="forform">
                <div class="textinformleft">
                  Title
                </div>
                <div class="textinformright">
                  <input type="name" name="title">
                </div>
              </div>
              <div class="forform">
                <div class="textinformleft">
                  Complaint
                </div>
                <div class="textinformright">
                  <textarea name="des" style="border-radius: 1rem;width: 90%;height: 150%;background-color: black;color: white;"></textarea>
                </div>
              </div>
              <br>
              <br>
              <div>
```

```
                    <button class="forbutton" type="submit"> Submit </button>
                </div>
            </form>
            <br>
        </div>
    </div>
</center>
</div>
</div>
{% endblock %}
```

login.html:

```
{% extends 'base.html' %}
{% block head %}
<title>
  Login
</title>
{% endblock %}
{% block body %}
<div class="forpadding">
  <!-- for box of the signup form -->
  <div class="sign">
    <div>
      <p class="fortitle">
        Sign In
      </p>
      <hr>
      <form action="/login" method="post">
        <div class="forform">
          <div class="textinformleft">
            Username
```

```
            </div>
            <div class="textinformright">
                <input type="name" name="username">
            </div>
        </div>

        <div class="forform">
            <div class="textinformleft">
                Password
            </div>
            <div class="textinformright">
                <input type="password" name="pass">
            </div>
        </div>
        <br>
        <div>
            <button class="forbutton" type="submit"> Sign In >></button>
        </div>
    </form>
    <br>
    <div>
        New user? <a href="/signup">Sign up</a>
    </div>
    <br>
  </div>
 </div>
</div>
 {% endblock %}
```

signup.html:

```
{% extends 'base.html' %}
```

```
 {% block head %}
 <title>
   Sign Up
</title>
{% endblock %}
 {% block body %}

<div class="forpadding">
   <!-- for box of the signup form -->
   <div class="sign">
      <div>
         <p class="fortitle">
            Register Now!!
         </p>
         <hr>
         <form action="/signup" method="post">
            <div class="forform">
               <div class="textinformleft">
                  Username
               </div>
               <div class="textinformright">
                  <input type="name" name="username">
               </div>
            </div>
            <div class="forform">
               <div class="textinformleft">
                  Name
               </div>
               <div class="textinformright">
                  <input type="name" name="name">
               </div>
            </div>
```

```
<div class="forform">
    <div class="textinformleft">
        E - mail
    </div>
    <div class="textinformright">
        <input type="name" name="email">
    </div>
</div>
<div class="forform">
    <div class="textinformleft">
        Phone Number
    </div>
    <div class="textinformright">
        <input type="name" name="phn">
    </div>
</div>
<div class="forform">
    <div class="textinformleft">
        Password
    </div>
    <div class="textinformright">
        <input type="password" name="pass">
    </div>
</div>
<div class="forform">
    <div class="textinformleft">
        Re - enter Password
    </div>
    <div class="textinformright">
        <input type="password" name="repass">
    </div>
</div>
```

```
            <br>
            <div>
                <button class="forbutton" type="submit"> Sign up >></button>
            </div>
        </form>
        <br>
        <div>
            {{msg}}
        </div>
        <br>
        <div>
            Already have an account? <a href="/login">Sign in</a>
        </div>
        <br>
    </div>
  </div>
</div>
{% endblock %}
```

tickets.html

```
{% extends 'base.html' %}
 {% block head %}
<title>
   Agent Dashboard
</title>
{% endblock %}
{% block body %}
 <!-- things
   div 1
welcome jetson,   sign out
   div 2
```

```
your complaints status
 add new complaint -->
<br>
<!-- <br>
{% for i in range(11) %}
  {{ i }}
{% endfor %}
 <br>
{% for i in complaints %}
{{ i['USERNAME'] }}
<br>
{% for j in i.values() %}
    {{ j }}
{% endfor %}
<br>
{% endfor %} -->
 <div class="fordashboardtop">
    <div class="fordashboardtopelements1">
        Welcome Admin,
    </div>
    <div class="fordashboardtopelements2">
        <a href="/login"><button class="forbutton">Sign out</button></a>
    </div>
 </div>
<br>
<div class="outerofdashdetails">
    <div class="fordashboarddetails">
        <br>
        <!-- table of customers complaints -->
        <table class="fortable">
            <thead>
                <th>Complaint ID</th>
```

```
    <th class="pad">Username</th>
    <th>Title</th>
    <th>Complaint</th>
    <th>Solution</th>
    <th>Status</th>
</thead>
<tbody>
    {% for i in complaints %}
    <tr>
        <td>{{ i['C_ID'] }}</td>
        <td class="pad">
            {{ i['USERNAME'] }}
        </td>
        <td>
            {{ i['TITLE'] }}
        </td>
        <td>
            {{ i['COMPLAINT'] }}
        </td>
        <td>
            {{ i['SOLUTION'] }}
        </td>
        <td>
            {% if i['STATUS'] == 1 %}
            Completed
            {% else %}
            Not Completed
            {% endif %}
        </td>
    </tr>
    {% endfor %}
</tbody>
```

```
</table>
<br>
<center>
    <div class="fordashboarddetails">
        <button type="button" class="collapsible">Assign an agent ⚡</button>
        <div class="content">
            <br>
            <form action="/assignagent" method="post">
                <div class="forform">
                    <div class="textinformleft">
                        Complaint ID
                    </div>
                    <div class="textinformright">
                        <input type="name" name="ccid">
                    </div>
                </div>
                <div class="forform">
                    <div class="textinformleft">
                        <label for="agent">Choose an agent:</label>
                    </div>
                    <div class="textinformright">
                        <select name="agent" id="agent">
                            {% for i in freeagents %}
                            <option value={{ i['USERNAME'] }}>{{ i['USERNAME'] }}</option>
                            {% endfor %}
                        </select>
                    </div>
                </div>
                <br>
                <br>
```

```
                    <div>
                        <button class="forbutton" type="submit"> Submit </button>
                    </div>
                </form>
                <br>
            </div>
        </div>
    </center>
  </div>
</div>
{% endblock %}
```