Customer Care Registry

Project Report

Academic Year:2022-2023

Team ID: PNT2022TMID18048

Team Members

Prakash R(TL)

Selva Yogiraam C

Jetson Cyrus J

Mari Saravanan P

Customer Care Registry

Team ID: PNT2022TMID18048

App link: http://159.122.178.36:30009/

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

A customer care registry is built as an cloud application for any kind of software to solve the basic queries of the customers using the software. This prevents the customer to get an bad impression while using the software, Because if the customer has some queries while using the software but if it is not clarified properly the customer may start losing interest in using that particular software

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.

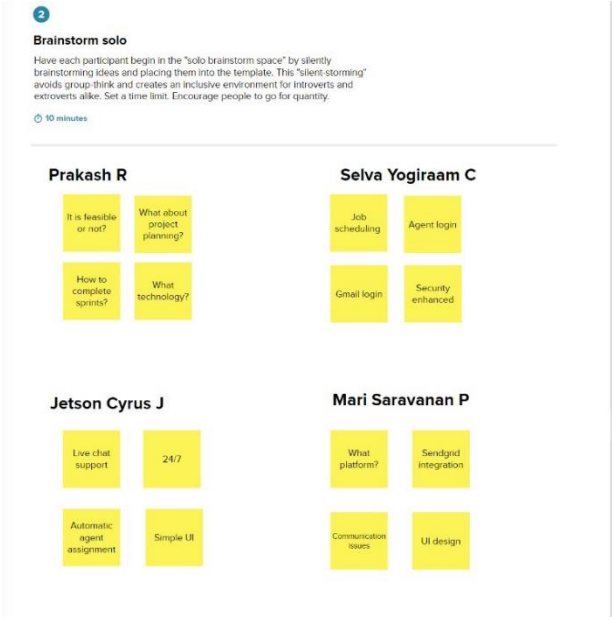
ADMIN : The main role and responsibility of the admin are to take care of the whole process. Starting from Admin login followed by the agent creation and assigning the customer's complaints. Finally, He will be able to track the work assigned to the agent and a notification will be sent to the customer.

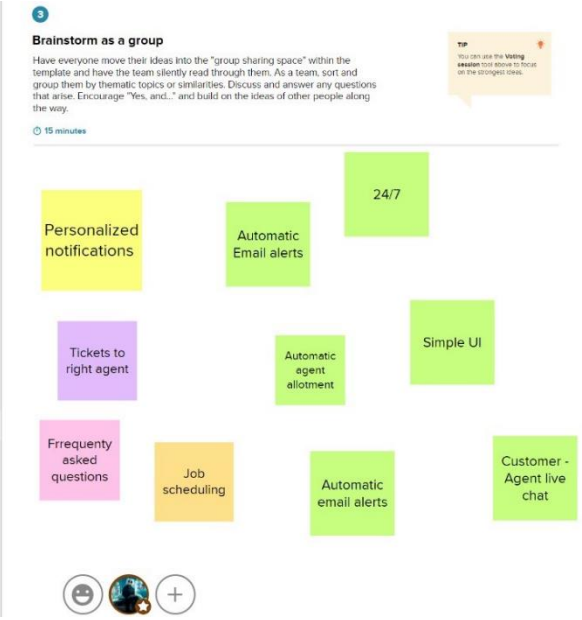
USER : They can register for an account. After the login, they can create the complaint with a description of the problem they are facing. Each user will be assigned with an agent. They can view the status of their complaint.

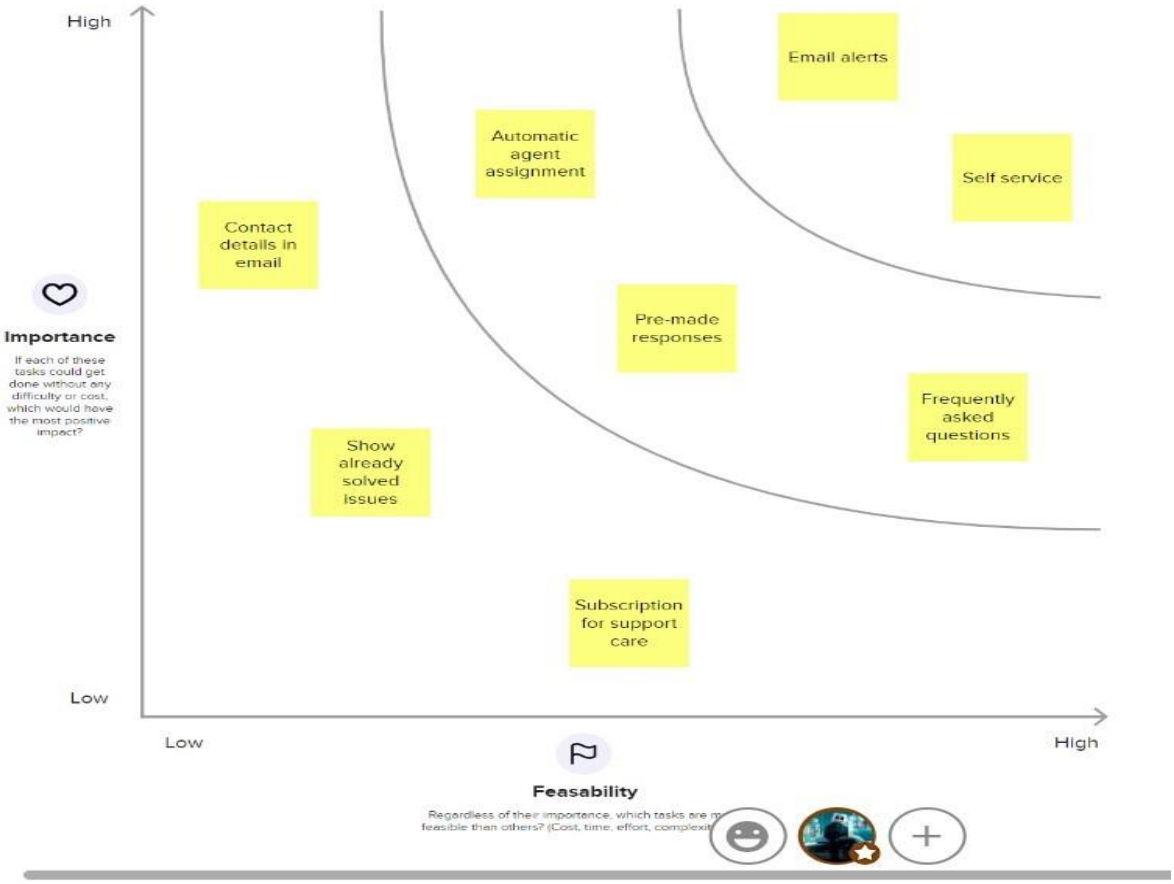
3. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas

b. Ideation & Brainstorming







c. Proposed Solution

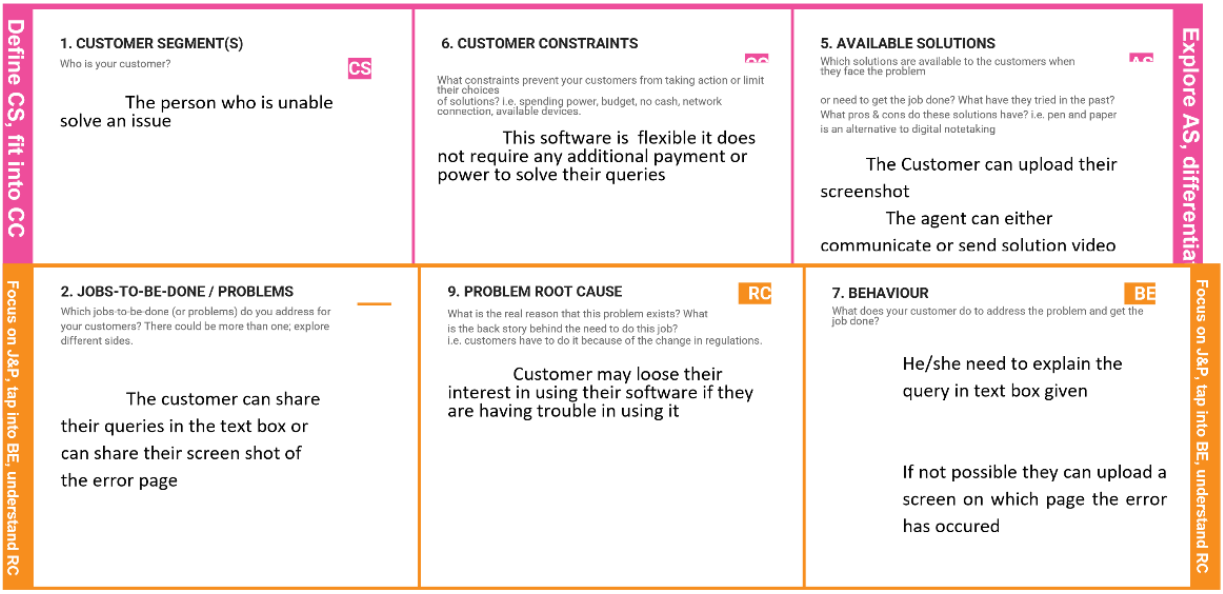
1	Problem statement(Problem to be solved)	To solve the problem of the complaints from the customer with the help of cloud application.
2	Idea/solution description	The solution to the problem is by assigning an agent based on the complaint , the agent after solving customer problem can update the status in the email of customer, same in the database, feedback is asked to customer for service satisfaction.
3	Novelty/uniqueness	Chatbot for feedback and friendly service, automatic email and messaging service for customer notifications.

4	Social impact/Customer satisfaction	Customers can get their problems solved easily with the help of agents solutions, efficient solution,trustable,reachable.
5	Business Model(Revenue model)	1. Customer care maintenance system. 2. Key resources,support engineers. 3. Knowledge based channel,customer service 24/7. 4. Cost structures expresses cloud platform.
6	Scalability of the solution	The main scalability of the solution is introduce an chatbot for customer friendly service, response time for problem solving by agents must be efficient and quick.

d. Problem Solution fit

3. TRIGGERS TR What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news. When customers think th ey cannot solve their queries of their own	**10. YOUR SOLUTION** SL If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. 1. Develop a flask app that solve their query	**8. CHANNELS of BEHAVIOUR** CH **8.1 ONLINE** All their data are secured and updated to cloud storage **8.2 OFFLINE**
4. EMOTIONS: BEFORE / AFTER EM How do customers feel when they face a problem or a job and afterwards?		

i.e. lost, insecure > confident, in control - use it in your communication strategy & design. Before:- When they can`t solve the problem on their own they feel uncomfortable to use the app/software After:- They feel the software is very flexible		2. Share their problem in Screenshot		Make sure they find best solution for their Complaints



4. REQUIREMENT ANALYSIS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	User Registration	Registration through Form
2	User Confirmation	Confirmation via Email
3	User Login	Login via google with email and password
4	Admin Login	Login via google with email and password
5	Query Form	Description of issues or share screen shot
6	Feedback	Through the customer feedback page

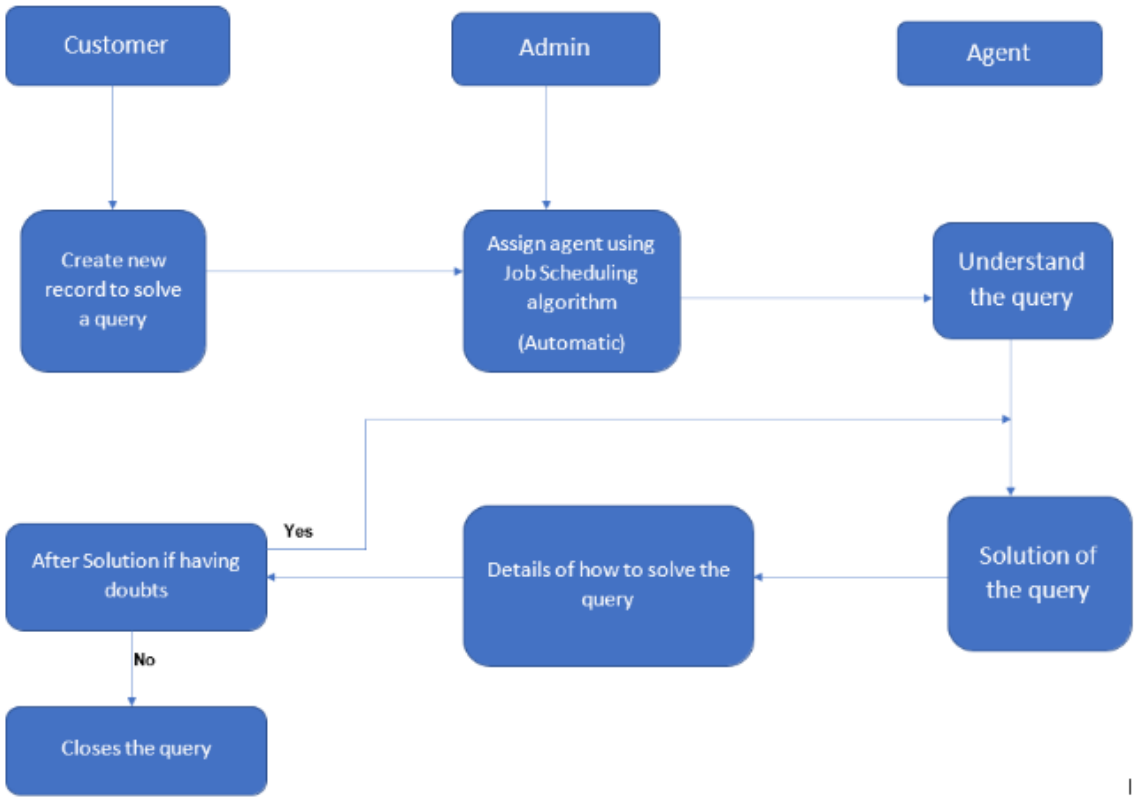
Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
1	**Usability**	To provide solution to their query
2	**Security**	Track Login authentication
3	**Reliability**	Tracking the status through the email
4	**Performance**	Great user interface through the web application
5	**Availability**	Always (Any day and any time)
6	**Scalability**	By the number of customers attended by agent

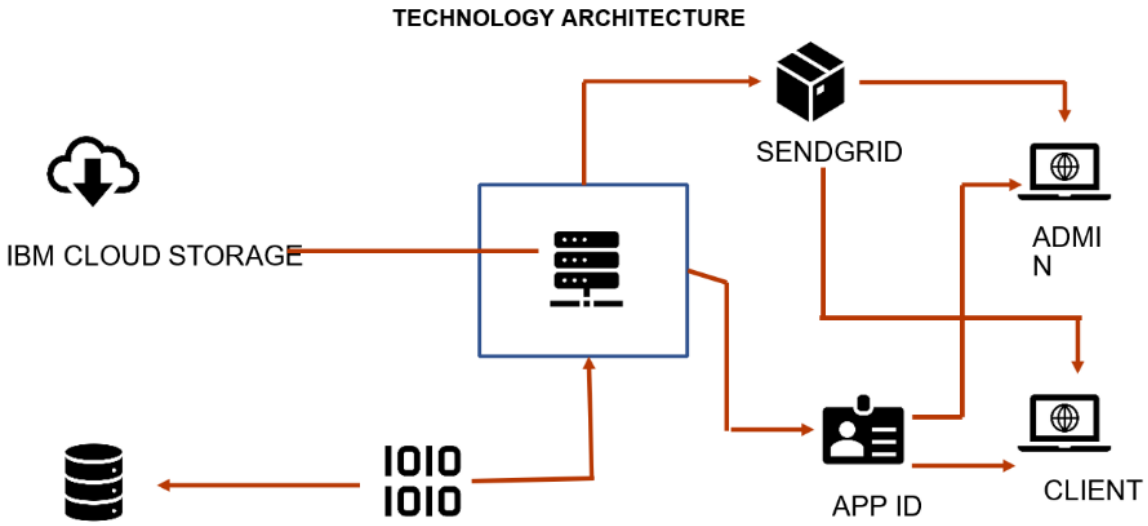
5. PROJECT DESIGN

a. Data Flow Diagrams



b. Solution & Technical Architecture

Technical Architecture:



Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Python flask
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. Encryptions, IAM Controls, Firewalls.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	supports higher workloads without any fundamental changes to it.
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	High availability enables your IT infrastructure to continue functioning even when some of its components fail.
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Performance technology, therefore, is a field of practice that uses various tools, processes, and ideas in a scientific, systematic manner to improve the desired outcomes of individuals and organizations.

c. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a Customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a Customer, I can login once they have registered	I can access my account / dashboard	High	Sprint-1
	Dashboard	USN-3	As a Customer, I can see my history of queries	I can see the Solution I got	Low	Sprint-2
	Request Help	USN-4	As a Customer I can request a help through Customer service	I can Ask my query	Medium	Sprint-1

	Share Screenshot	USN-5	As a Customer I can tag my screen shot to dashboard	I can share my screen shot of error to the agent	High	Sprint-3
	Request details	USN-6	Status of the current query requested	I can view the current status of my query	Medium	Sprint-4
Agent(Web User)	Login	USN-1	As an Agent, I can login to dashboard	View the Customer who are alloted	High	Sprint-2
	Dashboard	USN-2	View the list of details of customer query	List of queries with appropriate screen shots	High	Sprint-3
	Communication	USN-3	As an Agent I can Communicate to user in Communication area	I can communicate with the customer	Medium	Sprint-4
Admin	Login	USN-1	As an Admin I can Login into my account entering the correct	Access my account/Dash board	Low	Sprint-4
	Add Agent	USN-2	Create an agent and enlarge the service	Add new Agents	Medium	Sprint-1

6. PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and go through the services available on the webpage	20	High	Selva Yogiraam C R Prakash Jetson Cyrus J
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the availability and have a track of all the things that the users are going to service	20	High	Selva Yogiraam C
Sprint-3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding the services. Get	20	High	R Prakash Jetson Cyrus J

			the recommendations based on information provided by the user			
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application	20	High	Jetson Cyrus J Mari Saravanan P

b. Sprint Delivery 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	20	6 Days	24 Oct 2022	29 Oct 2022		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		19 Nov 2022

7. 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>
```


{% endblock %}

8. Testcases

a. User Acceptance Testing

Defect Analysis

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

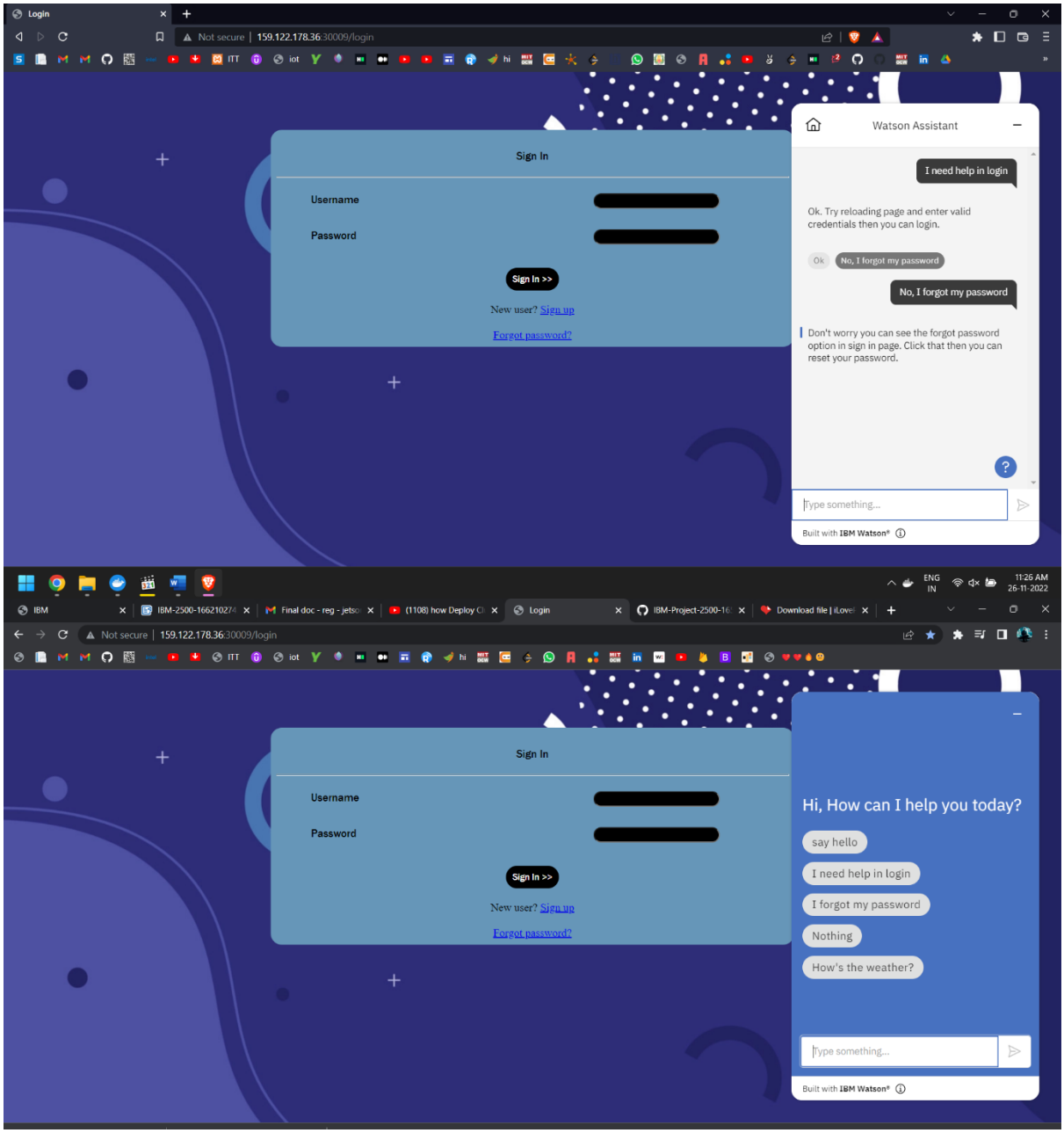
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	1	5	5	24
Duplicate	2	0	2	0	4
External	2	2	1	5	10
Fixed	15	3	5	10	35
Not Reproduced	0	0	0	0	0
Skipped	0	0	1	1	2
Won't Fix	0	0	1	0	1
Totals	32	6	16	17	77

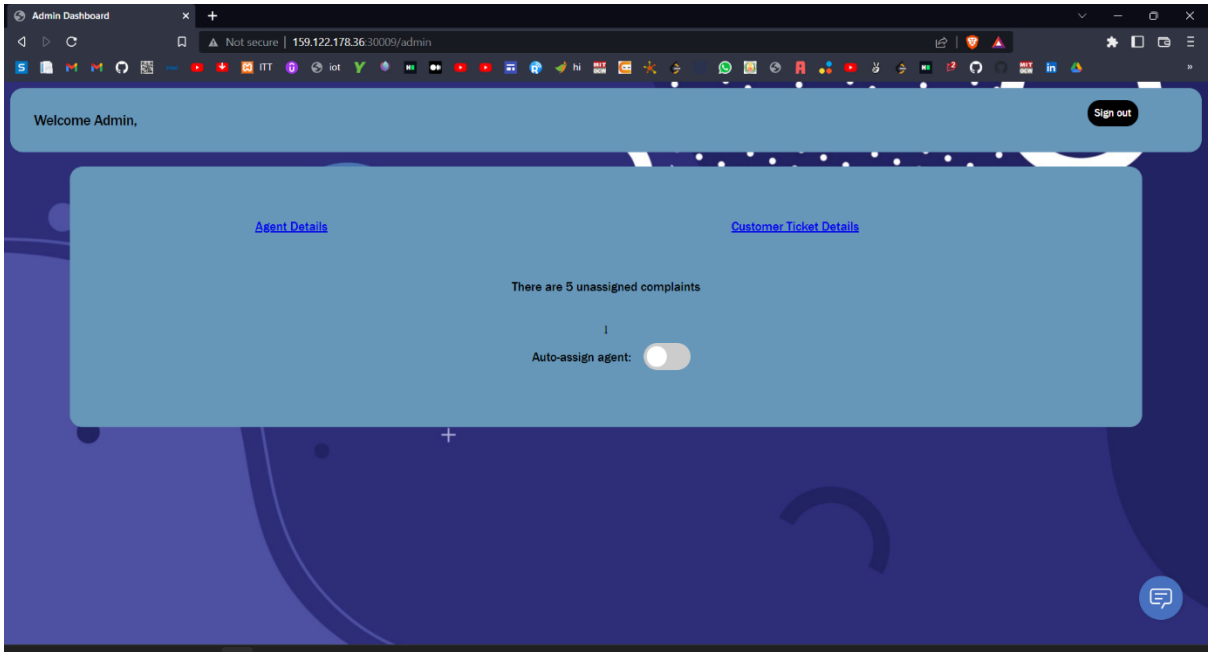
Test Case Analysis

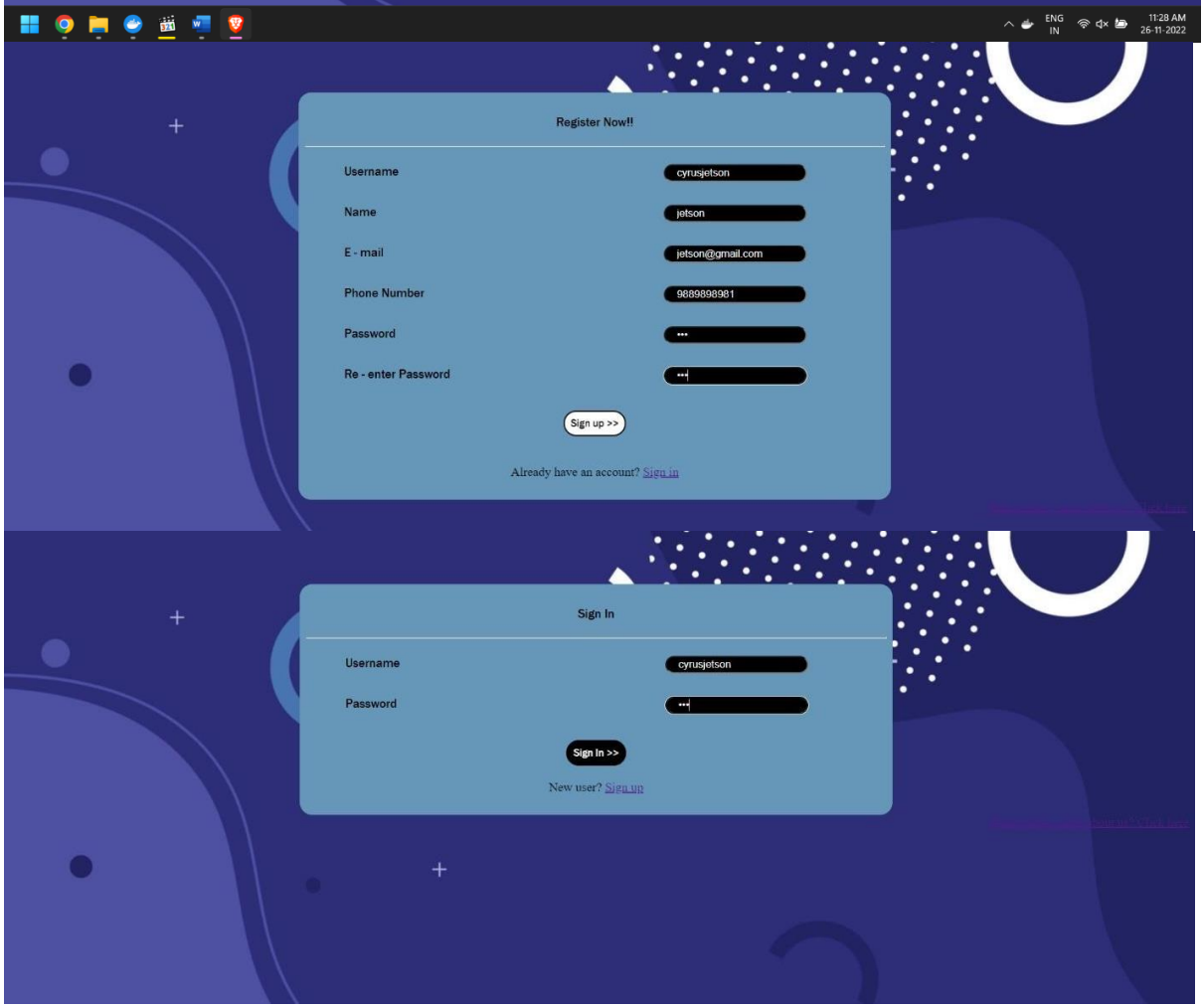
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Client Application	40	0	0	40
Exception Reporting	10	0	0	10
Version Control	2	0	0	2

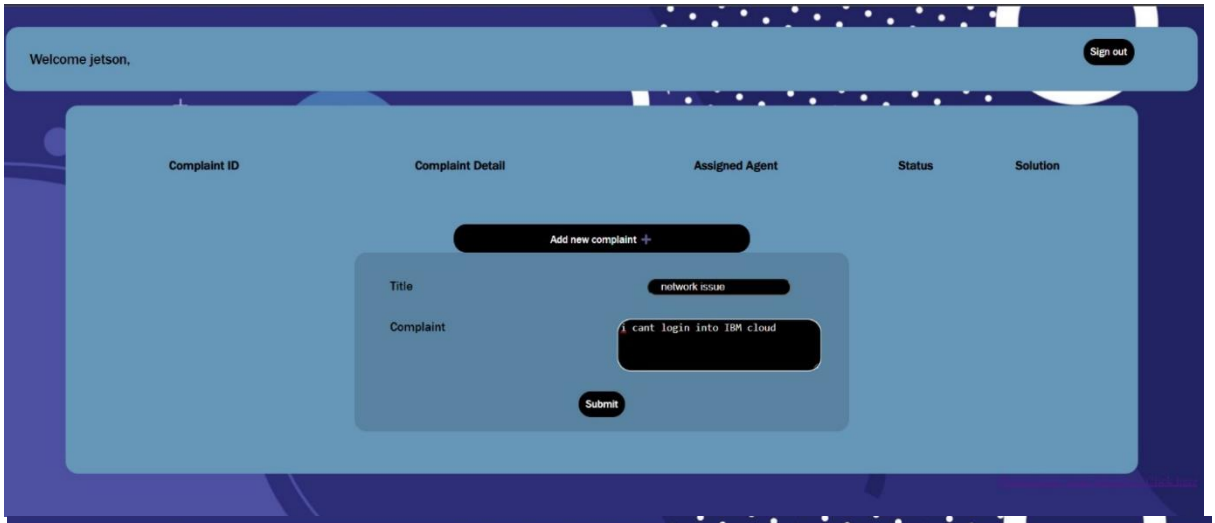
9. RESULTS

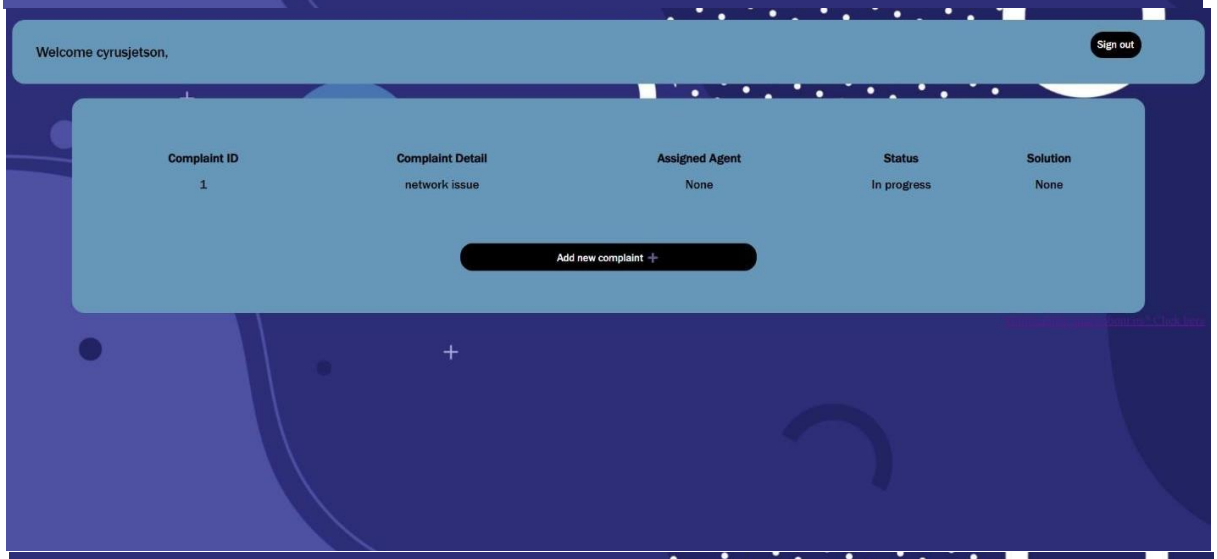

Login
Not secure | 159.122.178.36:30009/login
Sign In
Username
Password
Sign In >>
New user? Sign up
Forgot password?
Watson Assistant
I need help in login
Ok. Try reloading page and enter valid credentials then you can login.
Ok
No, I forgot my password
No, I forgot my password
Don't worry you can see the forgot password option in sign in page. Click that then you can reset your password.
Type something...
Built with IBM Watson®
Hi, How can I help you today?
say hello
I need help in login
I forgot my password
Nothing
How's the weather?

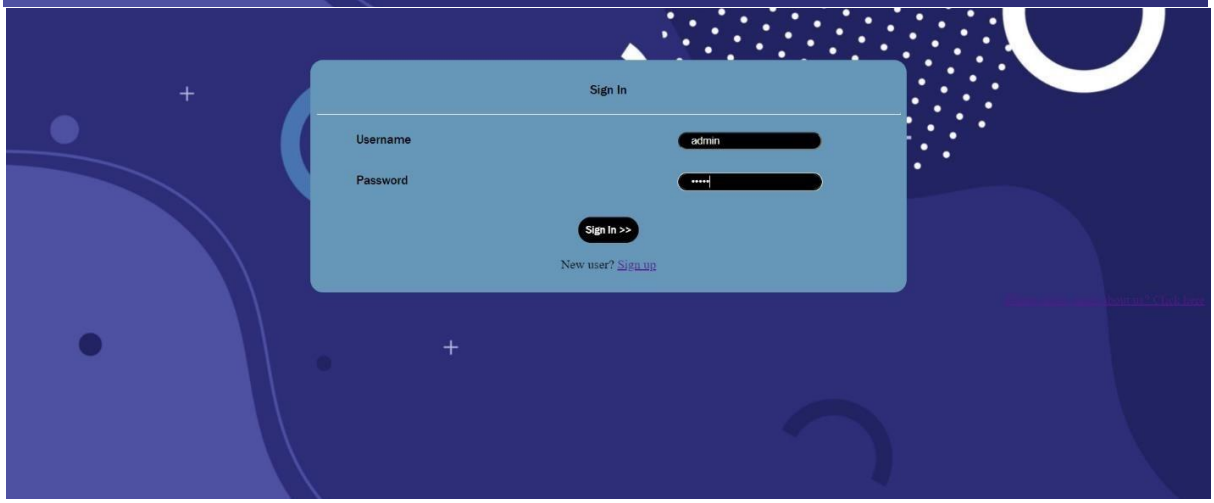

Admin Dashboard
Not secure | 159.122.178.36:30009/admin
Welcome Admin,
Sign out
Agent Details
Customer Ticket Details
There are 5 unassigned complaints
Auto-assign agent:

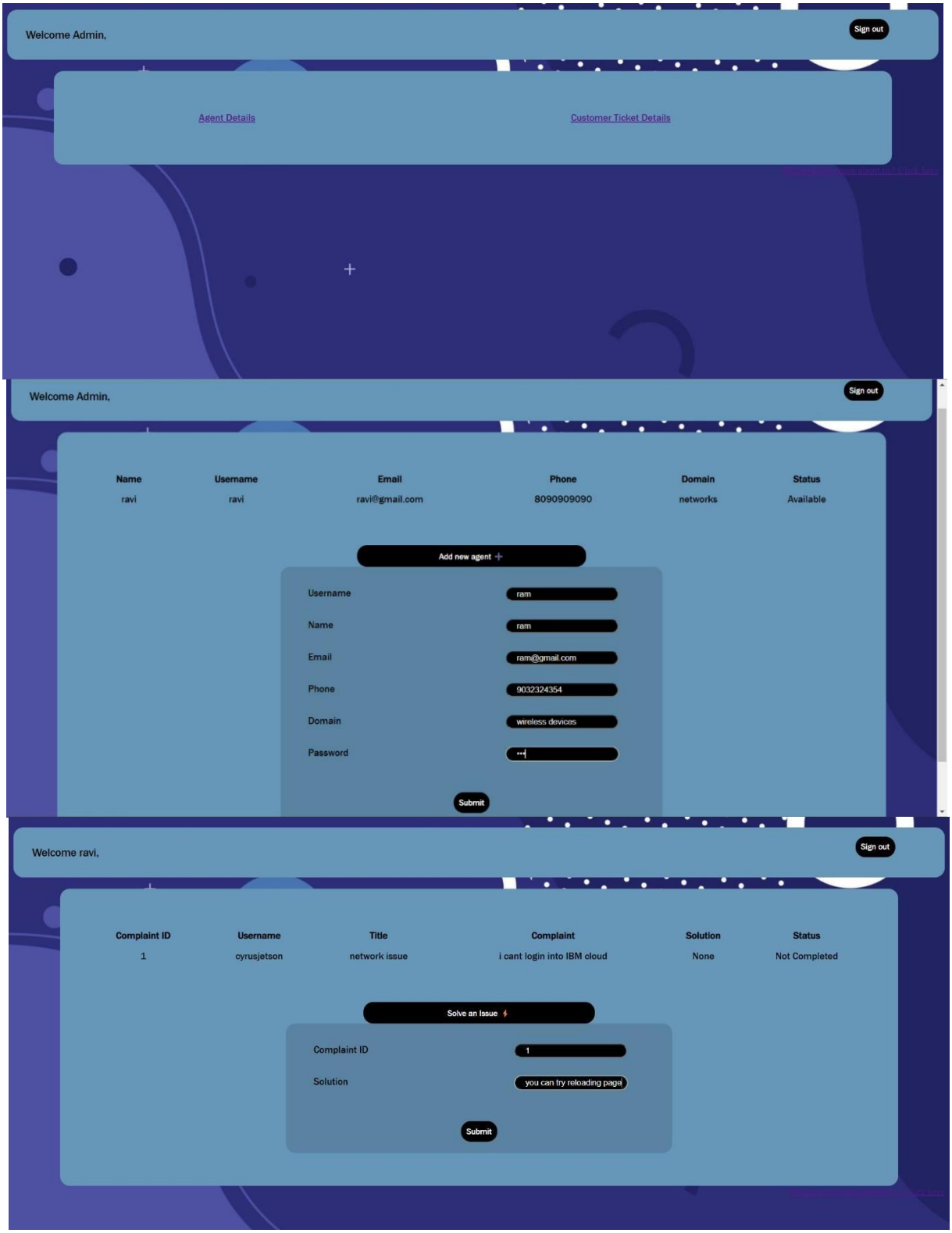

ENG
IN
11:28 AM
26-11-2022
Register Now!!
Username
cyrusjetson
Name
jetson
E - mail
jetson@gmail.com
Phone Number
9889898981
Password
Re - enter Password
Sign up >>
Already have an account? Sign in

Sign In
Username
cyrusjetson
Password
Sign In >>
New user? Sign up


Welcome jetson,
Sign out
Complaint ID
Complaint Detail
Assigned Agent
Status
Solution
Add new complaint +
Title
network issue
Complaint
1 cant login into IBM cloud
Submit


Welcome cyrusjetson,
Sign out
Complaint ID
Complaint Detail
Assigned Agent
Status
Solution
1
network issue
None
In progress
None
Add new complaint +


Sign In
Username
admin
Password
Sign In >>
New user? Sign up



10. ADVANTAGES & DISADVANTAGES

Advantages:

- The problem of optimal multi-server configuration for profit maximization in a cloud computing environment is studied

- Expectation and subjective disconfirmation seem to be the best conceptualizations in capturing satisfaction formation. The results suggest multiple comparison processes in satisfaction formation

• Provide new theoretical insights into power management and performance optimization in data centres

• Adopts the thought in Business Administration, and firstly defines the customer satisfaction level of cloud computing.

• Based on the definition of customer satisfaction, we build a profit maximization model in which the effect of customer satisfaction on quality of service (QoS) and price of service (PoS) is considered.

. Auto-assign agent can be turned off or can be turned on.

Disadvantages:

• Though this project allots the agent to the customer there will not be a brief communication about the problem to the agent

• There maybe network issue in communicating with agent.

11. CONCLUSION

By the above description there some advantages as well as some disadvantages with the project Customer care registry which has been developed. To overcome those issues some advanced technologies can be used so it becomes more user friendly and efficient to use by both the agents and the customers

This project has been successfully completed with the help of IBM team we thank for everyone who guided with the knowledge session. We show our thankfullness for this opportunity to get knowledge on cloud application development

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-2500-1658472881

b. Project Demo Link: https://www.youtube.com/watch?v=Yi_fo4sqhQA

14.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 %}
```