CUSTOMER CARE REGISTRY

USING CLOUD APPLICATION

A Project report submitted in partial fulfilment of 7th semester in degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Team ID: PNT2022TMID43979

Aadharsh S M	723719104001
Bala Logesh M	723719104013
Bathresh Bas B	723719104014
Dinesh M	723719104022
Hariharan K	723719104029
Karthik M	723719104035

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VSB COLLEGE OF ENGINEERING TECHNICAL CAMPUS, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600025

VSB COLLEGE OF ENGINEERING TECHNICAL CAMPUS, COIMBATORE

(A Constituent College of Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report "**CUSTOMER CARE REGISTRY**" is the bonafide record work done by **MR. S M AADHARSH (723719104001), MR. M BALA LOGESH (723719104013), MR. B BATHRESH BAS (723719104014), MR. M DINESH (723719104022), MR. K HARIHARAN (723719104029), MR. M KARTHIK (723719104035)** for **IBM-NALAYATHIRAN** in VII semester of B.E., degree course in Computer science and Engineering branch during the academic year of 2022-2023.

Staff-In charge

V.Radha

Evaluator

Mr.B.MariKumar

Head Of The Department

Mr.P.Dinesh Kumar

ACKNOWLEDGEMENT

We express our breathless thanks to our **Dr.V.VELMURUGAN, M.E, Ph.D** the Principal Of VSB College Of Engineering Technical Campus,Coimbatore for giving constant motivation in succeeding in our goal.

We acknowledge our sincere thanks to Head of the Department (i/c) **MR.P.DINESH KUMAR** for giving us valuable suggestion and help towards us throughout thisProject.

We are highly grateful to thank our Project coordinator **MRS.V.RADHA** and our Project Evaluator **MR.B.MARIKUMAR** Department of Computer Science and Engineering, VSB college of Engineering technical campus, Coimbatore, for coordinating us throughout this Project.

We are very much indebted to thank all the faculty members of Department of Computer science and Engineering in our Institute, for their excellent moral support and suggestions to complete our Project work successfully.

Finally our acknowledgment does our parents, sisters and friends those who had extended their excellent support and ideas to make our Project a pledge one.

Team Members:

Aadharsh S.M

Bala Logesh M

Bathresh Bas B

Dinesh M

Hariharan K

Karthik M

ABSTRACT

This Application has been developed to help the customer in processing their complaints. The customerscan raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer tosolve 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.

TABLE OF FIGURES

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code GitHub & Project Demo Link

Customer Care Registry

1. INTRODUCTION

An online comprehensive Customer Care Registry is to manage customer interaction and complaints with service provider over phone or through an e-mail. The system should have capability to integrate with any service provider from any domain or industry like Banking, Telecom(Online shopping), Insurance, etc.. Customer Care is also known as Client Service is the provision of service to customers its significance varies by product, industry and domain. In many cases customer care services is more important if the purchase relates to a service as opposed to a product. Customer care Registry may be provided by a Person or Sales & Services Representatives Customer Care Registry is normally an integral part of a company's customer value proposition.

1.1 Project overview

Online customer care and service center is a web-based application Developed using python programming language. With a platform of a typical “service center”, this system provides online technical services to its customers on a 24×7 basis. The whole process involves writing large volume of data in registers and preparing several reports daily.the basic services include hardware and software of a computer. It also maintains database of their employ details of their customers, and many more. Online customer care and service center application is developed to automate all the office activities of a typical service center.The main objective of this Online Customer Care and Service Center software is to develop an information system to store, maintain, update and process data relating to the shop. It will prepare various reports to aid in smooth and speedy functioning of ‘Service Center’ activities.

1.2 Purpose

As we live in the modern world, The people using the internet are increasing daily. So there are many problems occur in each and every field. So the need of Customer care is wanted for the people and most of them are not aware of this. The purpose of this purpose is to ensure all the people problems are solved by using this application developed using IBM cloud and python. By using chatbots, people feel convienent to open up and ready to solve their issues.

2. LITERATURE SURVEY

2.1 Existing problem

The present computer service centers generally keep the details of the customers and products in word documents, spreadsheets or paper register, and the management of all records is illegal to some extent.There are problems relating redundancy of data like customer name and address, details of their account and also allocation of duties to the employees. When a customer takes some kind of services, the charge is calculated manually, and this process is time consuming. Also, regular and overtime duties are not maintained properly. This leads to improper calculation of employees becomes quite complicated for every employee.Another problem usually faced by the organization which has been solved in the proposed Online Customer Care and Service Center Project is the frequent complaints by the customers for not getting timely services.

2.2 Literature Survey

Literature Survey was taken from these websites.

https://www.tandfonline.com/doi/full/10.1080/14783363.2022.2038558

By Daniel Gyllenhammar, Department of Technology Management and Economics, Chalmers University of Technology, Gothenburg, Sweden

https://www.researchgate.net/publication/224053675_Improving_Customer_Service_in_Healthcare_with_CRM_20

By Muhammad Anshari, Doctor Of Philisophy,University Brunei Darussalam,School Of Business and Economics, Brunei

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0224-1

By Sachin Kumar, Department of System Programming, South Ural State University, Chelyabinsk, Russian Federation. And Mikhail Zymbler, Department of System Programming, South Ural State University, Chelyabinsk, Russian Federation.

https://www.academia.edu/40485359/Cybercrime_Case_As_Impact_Development_Of_Communication_Technology_That_Troubling_Society\

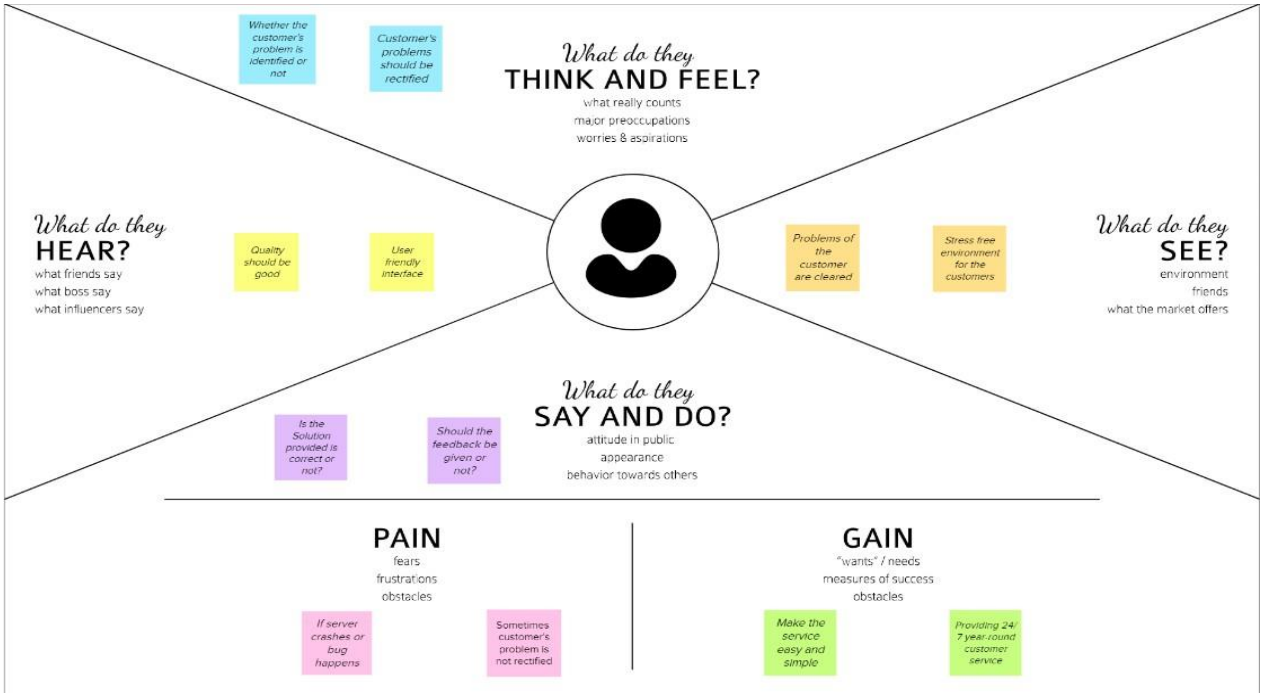
By M.Chairul Umanailo,Faculty Member, Agriculture and Forestry,University of Iqra Buru, Indonesia.

2.3 Problem Statement Definition

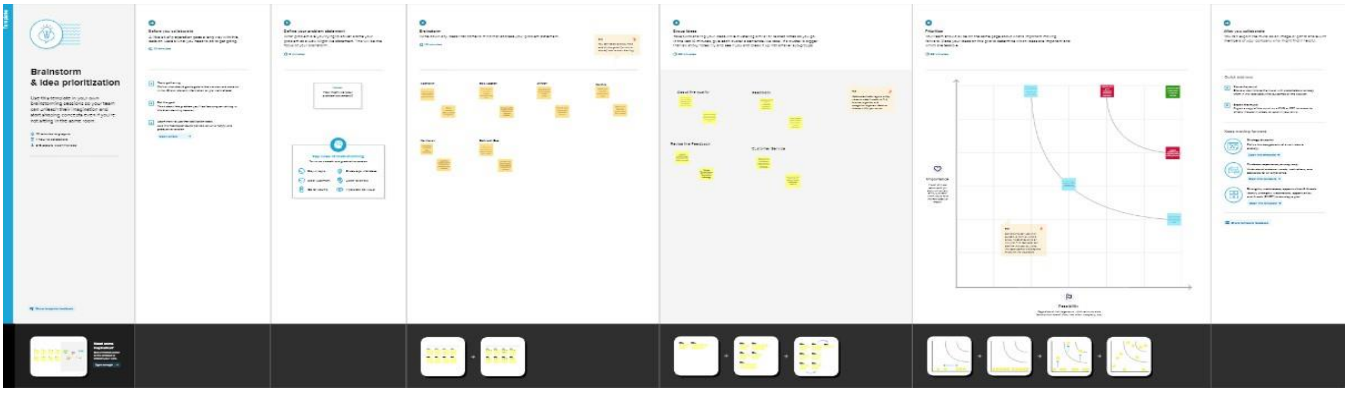
The Customer can have any type of issue that cannot be solved by them. Then they will reach for the customer care services. How they will resolve their issue? By raising a complaint through the email support to the respected organization or contacting through mobile phone. This application provides the customer to raise a ticket for each customer uniquely and update every information about the issue to the customer. This also enchances the customer to interact with online chatbot which helps them to resolve the issue on their own or connecting with the live agent. So the issue will be solved for the customer.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



3.3 Proposed Solution

S.No	Parameter	Description
1.	Problem to be solved or problem statement	To solve the customer issues using cloud application development using chatbots.
2.	Solution Description	Assigned Agent routing can be solved by directly routing to the specific agent about the issue using the specific email. Automated Ticket closure by using daily sync of the daily database. Status Shown to the customer can display the status of the ticket to the customer. Regular data retrieval in the form of retrieving lost data.
3.	Solution provided should be unique and true	Assigned agent routing,Automated ticket closure, status shown to the customer, backup data for the later use in case of any failures
4.	Customer Satisfication	Customers should track their complaint in the form of some ticket or some tracking id for easy and better communication. Make sure the customer satisfied with the solution provided.
5.	Business Model	Key Partners are Third-party applications, agents, and customers. Activities held as Customer Service, System Maintenance. Key Resources support Engineers, Multichannel. Customer Relationship have 24/7 Email Support, Knowledge-based channel. Cost Structure expresses Cloud Platform, Offices
6.	Scalability and reliability of the solution	The real goal of scaling customer service is providing an environment that will allow your customer service specialists or agents to be as efficient as possible. An environment where they will be able to spend less time on grunt work and more time on actually resolving critical customer issues.

3.4 Problem Solution fit

1. Customer Segments :

Customers who are not able to solve their problems and who do not know the solution for the problem.

For this problem , the solution is That the solution we propose will have provide the customers to provide solutions to their problems via email support or by connecting to a agent which will be very comfortable for better interactions and better results.

2. Problem Root Cause :

The customer don't know the root cause for the problem and then what happens when a problem arises.

We propose that our solution makes clear to them and next time make sure them to read the guidelines and instructions carefully before doing anything. With our chatbots and Connecting to an agent function, it can be solved.

3. Data Storage Issues :

Some of the customers may forget their login credentials, and they will raise a complaint for that.

We propose that the data which is the login credentials, are stored in the cloud storages in a secure manner. Then if they lost their credentials means, with the help desk , we can reach the customer for more support.

4. Availability Issues :

Some of the customer care service organisations does not work late in night shifts. It make the availability issues for the customers.

We propose that our online customer care service included with the chat bot service , The application will provide the 24/7 service top the customers and at anytime the issue will be solved.

5. Response Time Too Long :

Customers today expect communication with service departments to be instant. In fact, they want immediate resolution of their concerns too. This is, indisputably, the first in the long list of the common problem with customer service that needs to be addressed by businesses.

We propose , Ensure that your agents are aware of their roles and responsibilities along with who they are accountable to if and when there are lapses in service Allow your customers to reach you via multiple channels including email, website chat, phone, social, text message and allocate resources accordingly

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Any Requirement Which Specifies What The System Should Do is called the Functional Requirements. Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form, Email, Websites
FR-2	User Confirmation	Confirmation via Email OTP from mobile number.
FR-3	User Login	Login via user ID and password or through google account
FR-4	Admin login	Login via Admin ID and Password
FR-5	Query or issue form	Issues described through message and their contact information like Email ID
FR-6	Email	For the confirmation of the correct login and verify it
FR-7	Feedback	User feedback collected through the forms

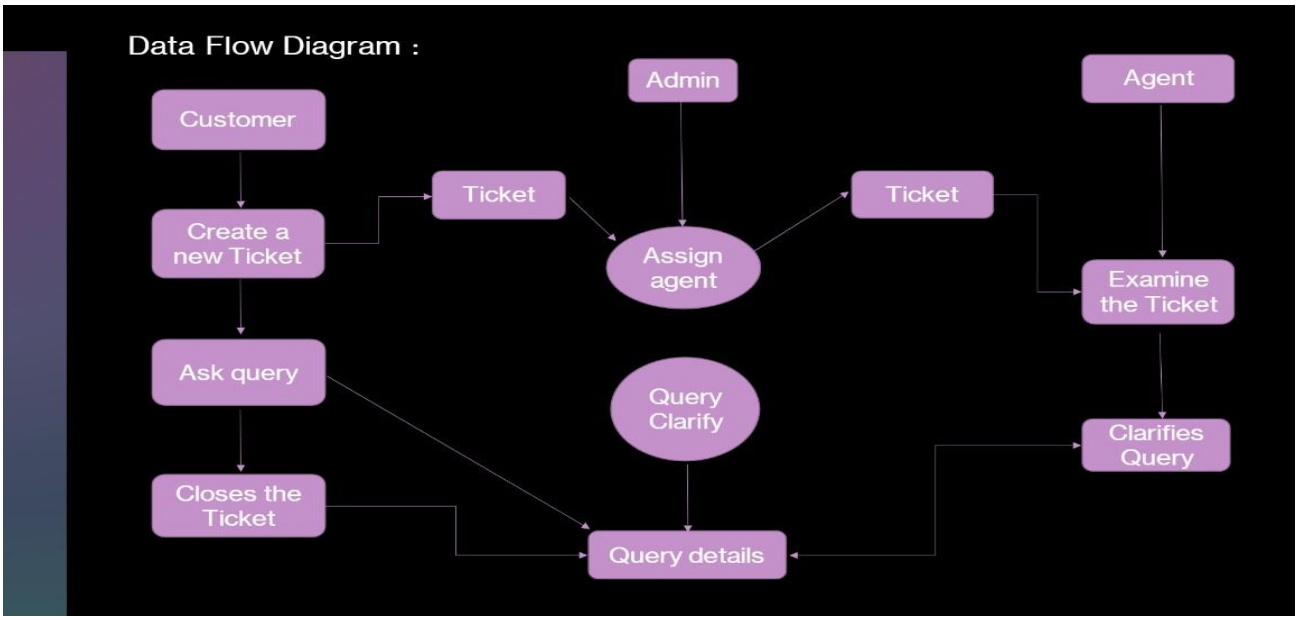
4.2 Non-functional Requirements:

Any Requirement That Specifies How The System Performs A Certain Function is called the non-functional requirements.Following are the non-functional requirements of the proposed solution.

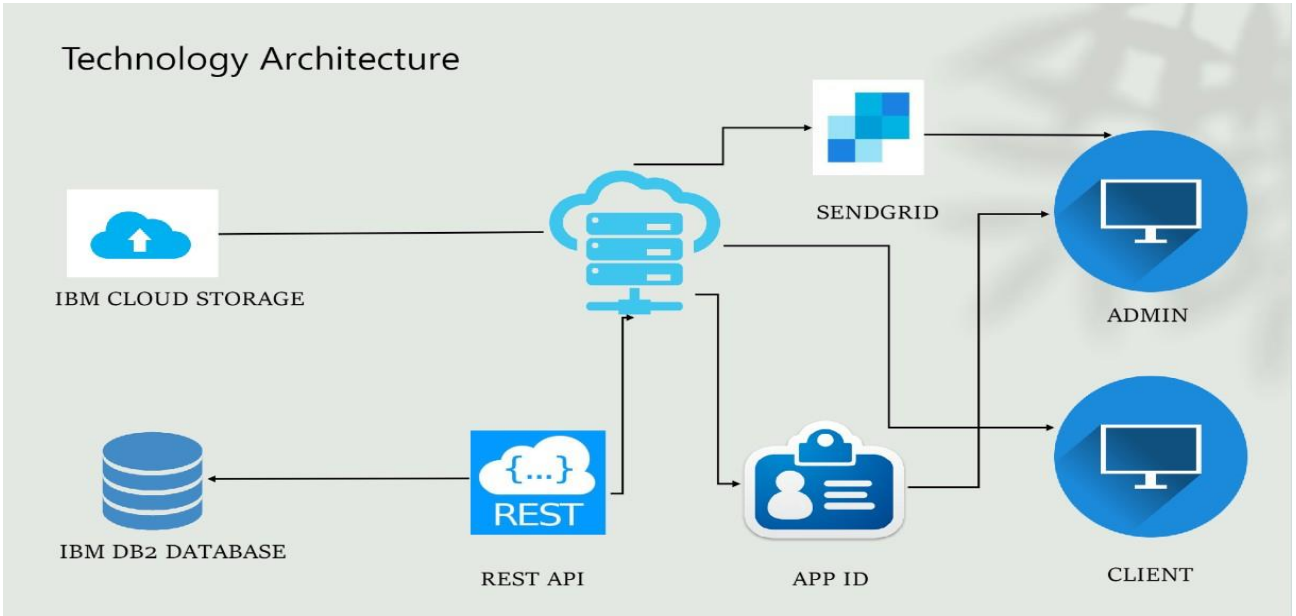
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Providing the exact solution to the issue arised by the customer.
NFR-2	**Security**	To secure the login credentials of the user
NFR-3	**Reliability**	Tracking of the status
NFR-4	**Performance**	Effective development of web application
NFR-5	**Availability**	24/7 support will be provided
NFR-6	**Scalability**	Agents are assigned to the limited amount of customers
NFR-7	**Speed**	Speed determines how fast an application responds to commands. In this , How an agent responses quickly to the customer.
NFR-8	**Portability**	This application is portable in all platforms like mobile, system,ios,etc

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional requirements	User Story Number	User Story	Acceptance Criteria	Priority	Release
Customer	Registration	USN-1	I can sign up for the application as a customer by providing my email address, password, and password confirmation.	I may access the application as a customer by entering my accurate email address and password.	High-1	Sprint -1
	Login	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
	Dashboard	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
	Order creation	USN-4	As a user, I can register for the application through Gmail	I can access the dashboard through Gmail	Medium	Sprint -1
	Address column	USN-5	As a user, I can log into the application by entering email & password	I can access the dashboard by Email and password	High	Sprint-1
	Forgot password	USN-6	As a user I can reset my password	I get to access my account again	Medium	Sprint-4
	Order Details	USN-7	I can view the most recent order statistics as a customer	I am more able to grasp the idea	Medium	Sprint-4

User Type	Functional Requirements	User Story Number	User Story	Acceptance Criteria	Priority	Sprint
Customer	Login	USN-1	I may access the application as an agent by entering the proper email address and password	I can log in to my account	High	Sprint-3
	Dashboard	USN-2	I may view the order specifics that admin assigned to me as an agent	I can see the tickets that need responses	High	Sprint-3
	Address column	USN-3	I get to converse with the consumer as an agent and address any concerns they may have	I can explain the problems	High	Sprint-4
	Forgot password	USN-4	If I forget my old password as an agent, I can reset it using this option	I can access my account once more	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	26 September 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	22 September 2022
Ideation BrainStorming	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility &importance.	23 September 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	21 September 2022
Problem Solution Fit	Prepare problem solution fit document	28 September 2022
Solution Architecture	Prepare solution architecture document.	26 September 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	24 October 2022
Solution Requirement	Prepare the functional requirement document	12 October 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	13 October 2022
Technology Architecture	Prepare the technology architecture diagram.	14 October 2022
Prepare Milestone & Activity List	Prepare the milestones &activity list of the project.	26 October 2022

6.2 Sprint Delivery Schedule

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

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Customer Pane	USN-1	As a Customer, I can register for the application by entering my email, password, and confirming my password and I will be able to Access my dashboard for creating a Query Order	2	High	Aadharsh Bala Logesh Bathresh Bas Dineh Hariharan Karthik
Sprint-2	Admin Panel	USN-2	As an admin, I can Login to the Application by entering correct login credentials and I will be able to Access My dashboard to create Agents and Assign an Agent to a Query Order	2	High	Aadharsh Bala Logesh Bathresh Bas Dineh Hariharan Karthik
Sprint-3	Agent Panel	USN-3	As an agent, I can Login to the Application by entering correct login credentials and I will be able to Access my Dashboard to check the Query Order and I can Clarify the Issues	2	High	Aadharsh Bala Logesh Bathresh Bas Dineh Hariharan Karthik
Sprint-4	Final Delivery	USN-5	Container of applications using docker kubernetes and deployment the application .Create the documentation and final submit the application	2	High	Aadharsh Bala Logesh Bathresh Bas Dineh Hariharan Karthik

---

7. CODING & SOLUTIONING

Forgot password-

Forpass.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
    <meta name="generator" content="Hugo 0.84.0">
    <title>Sign In</title>
```

```
    <link rel="canonical" href="https://getbootstrap.com/docs/5.0/examples/sign-in/">

    <link href="https://getbootstrap.com/docs/5.0/assets/css/docs.css" rel="stylesheet">

    <!-- Bootstrap core CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT"
crossorigin="anonymous">

    <!-- Favicons -->

<link rel="apple-touch-icon" href="/docs/5.0/assets/img/favicons/apple-touch-icon.png"
sizes="180x180">

<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-32x32.png" sizes="32x32"
type="image/png">

<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-16x16.png" sizes="16x16"
type="image/png">

<link rel="manifest" href="/docs/5.0/assets/img/favicons/manifest.json">

<link rel="mask-icon" href="/docs/5.0/assets/img/favicons/safari-pinned-tab.svg"
color="#7952b3">

<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon.ico">

<meta name="theme-color" content="#7952b3">

    <style>

      .bd-placeholder-img {

        font-size:  1.125rem;

        text-anchor: middle;

        -webkit-user-select: none;

        -moz-user-select: none;

        user-select: none;

      }
```

```
    @media (min-width: 768px) {
      .bd-placeholder-img-lg {
        font-size: 3.5rem;
      }
    }
  </style>
  <!-- Custom styles for this template -->
  <link href="static/sheets/signin.css" rel="stylesheet">
  <link href="static/sheets/colors.css" rel="stylesheet">
</head>
<body class="text-center">
    <nav class="navbar navbar-dark bg-dark fixed-top">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">News Tracker</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data-bs-target="#offcanvasDarkNavbar" aria-controls="offcanvasDarkNavbar">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="offcanvas offcanvas-end text-bg-dark" tabindex="-1" id="offcanvasDarkNavbar" aria-labelledby="offcanvasDarkNavbarLabel">
          <div class="offcanvas-header">
            <h5 class="offcanvas-title" id="offcanvasDarkNavbarLabel">Profile</h5>
            <button type="button" class="btn-close btn-close-white" data-bs-dismiss="offcanvas" aria-label="Close"></button>
          </div>
          <div class="offcanvas-body">
            <ul class="navbar-nav justify-content-end flex-grow-1 pe-3">
              <li class="nav-item">
                <a class="nav-link active" aria-current="page" href="home.html">Home</a>
```

```
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="base.html">Fetch News</a>
                  </li>
                <li class="nav-item">
                  <a class="nav-link" href="about.html">About Us</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="signin.html">Sign In</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="signup.html">Sign Up</a>
                </li>
              </div>
            </div>
          </div>
        </nav>
<main class="form-signin  ">
  <form action = "{{ url_for("getUser")}}" method="POST">
   <img class="mb-4" src="static/Images/house-user-solid.svg" alt="" width="72" height="57">
   <h1 class="h3 mb-3 fw-normal white">Change your password</h1>
   <div class="form-floating">
    <input type="email" class="form-control" id="floatingInput" name = "uname" placeholder="name@example.com">
    <label for="floatingInput">Email address</label>
   </div>
   <br>
   <button class="w-100 btn btn-lg btn-warning btn-outline-warning" type="submit"><a href="static/templates/changepass.html">Change Password</a></button><br><br>
```

```
  </form>
</main>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-u1OknCvxWvY5kfmNBILK2hRnQC3Pr17a+RTT6rIHI7NnikvbZlHgTPOOmMi466C8" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@docsearch/js@3"></script>
<script src="https://cdn.jsdelivr.net/npm/@stackblitz/sdk@1/bundles/sdk.umd.js"></script>
<script src="/docs/5.2/assets/js/docs.min.js"></script>
<script>
  document.querySelectorAll('.btn-edit').forEach(btn => {
    btn.addEventListener('click', event => {
      const htmlSnippet = event.target.closest('.bd-code-snippet').querySelector('.bd-example').innerHTML
      const classes = Array.from(event.target.closest('.bd-code-snippet').querySelector('.bd-example').classList).join(' ')
      const jsSnippet = event.target.closest('.bd-code-snippet').querySelector('.btn-edit').getAttribute('data-sb-js-snippet')
      StackBlitzSDK.openBootstrapSnippet(htmlSnippet, jsSnippet, classes)
    })
  })
  StackBlitzSDK.openBootstrapSnippet = (htmlSnippet, jsSnippet, classes) => {
    const markup = `<!doctype html>
<html lang="en">
  <head>
```

```
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link     href="https:\/\/cdn.jsdelivr.net\/npm\/bootstrap@5.2.1\/dist\/css\/bootstrap.min.css"
rel="stylesheet">

    <link href="https://getbootstrap.com/docs/5.2/assets/css/docs.css" rel="stylesheet">

    <title>Bootstrap Example</title>

    <${'script'}
src="https:\/\/cdn\.jsdelivr\.net\/npm\/bootstrap@5\.2\.1\/dist\/js\/bootstrap\.bundle\.min\.js">

</${'script'}>

  </head>

  <body class="p-3 m-0 border-0 ${classes}">

    <!-- Example Code -->

${htmlSnippet.replace(/^/gm, '    ')}

    <!-- End Example Code -->

  </body>

</html>`
```

8. TESTING

8.1 Test Cases

Test case Id	Test Case description	Test steps	Test data	Expected results	Actual result	Pass/fail
TC-1	Customer registration with invalid password	Go to the application and fill the credentials and then register	First name= Kumar last name=L email=xyz@gmail.com password=123456 confirm password = 1234567	The alert shows it is password does not match	As expected	Pass
TC-2	Customer registration with invalid email	Go to the application and fill the credentials and then register	First name=Kumar last name=L email=xyzgmailcom password=123456 confirm password = 1234567	The alert shows it is invalid email	As expected	Pass
TC-3	Customer login with invalid user data	Go to the application and login	email=xyz@gmail.com password=123456	The alert shows it is user does not exist	As expected	Pass
TC-4	Customer login with invalid password	Go to the application and login	email=xyz@gmail.com password=12345678	The alert shows it is invalid password	As expected	Pass
TC-5	Admin login with invalid data	Go to the application and login	Admin id=abc123 password=0212	The alert shows it is user does not exist	As expected	Pass
TC-6	Admin login with invalid user data	Go to the application and login	Admin id=abc123 password=02123	The alert shows it is invalid password	As expected	Pass

8.2 User Acceptance Testing

This shows that the number of bugs are found in the UAT testing.

Resolution	Severity 1	Severity 2	Severity 3	Severity 5	Severity 6
By design	10	4	5	5	24
Duplicate	2	0	2	0	4
External	5	3	2	1	11
Fixed	15	5	5	10	35
Not reproduced	0	0	0	0	0
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Total	32	17	17	18	84

Test Case Analysis:

This test cases are analysed and the results are below.

Section	Total cases	Not tested	Pass	Fail
Print Engine	10	0	10	0
Client Application	40	0	40	0
Security	5	0	5	0
Outsource Shipping	3	0	3	0
Exception reporting	10	0	10	0
Final report output	4	0	4	0
Version control	4	0	4	0

Result:

Customer Care Registry application using cloud is developed and executed at the level ofcompleted progress .

10. ADVANTAGES & DISADVANTAGES

Advantages :

- Easier Communication with the customers.
- Increase Conversation and sales
- Provide 24/7 support
- Automatically route with the agents
- Easily Track and view the tickets
- Lower Operational Costs
- Improve the customer support performance

Disadvantages :

- Special Technical persons are needed to operate
- Management is hard
- By providing 24/7 support, it is hard for the service providers to provide service.

11. CONCLUSION

By using this cloud application on customer care registry, Each and every customer can solve their issue with interacting with the chatbots. If the chatbot does not provide the solution means it will assign an live agent to the customer.Meanwhile,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. Through the email alerts,the customer can know about the status of their complaint and review about their remarks for the complaint. Finally, from every customer , Feedback will be provided from each customer in order to maintain a good service and bad service and can learn from the feedbacks.

12. FUTURE SCOPE

In the future we will further explore the design of adaptive interfaces, in order tobe in a position to demonstrate a complete adaptive framework for the customer care registry.

13. APPENDIX:

Source Code :

App.py :

```
from flask import Flask, render_template, request, redirect, session, url_for
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;S
SLServerCertificate=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=True)
```

Docker File :

```
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"]
```

Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-node-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flasknode
  template:
    metadata:
      labels:
        app: flasknode
    spec:
      containers:
      - name: flasknode
        image: au.icr.io/customer-care-ibm/customer-care-ibm
        ports:
        - containerPort: 5000
```

Admin.html

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

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 %}
<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 %}
<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 %}
```

Github Link :

https://github.com/IBM-EPBL/IBM-Project-32863-1660212731