

**HX8001 - PROFESSIONAL READINESS FOR INNOVATION,
EMPLOYABILITY AND ENTREPRENEURSHIP**

CUSTOMER CARE REGISTRY

IBM NAALAIYA THIRAN

(TEAM ID: PNT2022TMID50123)

A PROJECT REPORT

Submitted by

AUGUSTA S (951319104008)

FATIMA STANY J (951319104020)

YOGESWARI A (951319104054)

DIVYANANDHI A (951319104016)

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING

JAYARAJ ANNAPACKIAM CSI COLLEGE OF ENGINEERING

NAZARETH – TUTICORIN



NOVEMBER- 2022



ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**CUSTOMER CARE REGISTRY**” is the bonafide work of **AUGUSTA S (951319104008)**, **FATIMA STANY J(951319104020)**, **YOGESWARI A(951319104054)**, **DIVYANANDHI A(951319104016)** who carried out the **IBM NAALAIYA THIRAN** project work under our supervision.

Industry Mentor

(Mr.vasudev Hanush ,IBM)

Faculty Mentor

(Mrs.Suganya S/AP)

Faculty Evaluator

(Mr.Dr.P Edwin Dhass)

Head of the Department

(Mrs.Dr.G.Jemilda)

ACKNOWLEDGEMENT

First and foremost we thank the almighty for helping us in all situations for bringing out this project successfully.

We express our sincere heartfelt gratitude to **Mr.Dr.S.Jeyakumar Ruban, correspondent** of Jayaraj annapackiam CSI College of Engineering, Nazareth.

We would like to express our thanks to our principal **Dr.S.Jeyakumar** For his kind consent, inspiration and constant encouragement towards this project work.

We profusely thank **Mrs.Dr.G.Jemilda** Head of the department, Computer Science and Engineering for the help and support, without which our project would have been sculpted successfully.

We express our heart full thanks to our Industry Mentor **Mr.vasudev Hanush, IBM** Faculty Mentor **Mrs.Suganya S,Assistant Professor**, Faculty Evaluator **Mr.Dr.P Edwin Dhass** For invaluable support, guidance, utmost patience, inspirational coordination and constant encouragements in completing this project successfully.

Also we would like to thank all the faculty members and non teaching staff of the computer science and Engineering department for the kind advice and encouragement

TABLE OF CONTENT

CHAPTER	CONTENTS	PAGE NO
1	INTRODUCTION	6
	1.1PROJECT OVERVIEW	
	1.2 PURPOSE	
2	LITRATURE SURVEY	12
	2.1 EXISTING PROBLEM	
	2.2 REFERENCES	
	2.3 PROBLEM STATEMENT DEFINITION	
3	IDEATION&PROPOSED SOLUTION	18
	3.1 EMPATHY MAP CANVAS	
	3.2 IDEATION & BRAININSTROMING	
	3.3 PROPOSED SOLUTION	
	3.4 PROBLEM SOLUTION FIT	
4	REQUIREMENT ANALYSIS	26
	4.1 FUNCTIONAL REQUIREMENTS	
	4.2 NON-FUNCTIONAL REQUIREMENTS	
5	PROJECT DESIGN	37
	5.1 DATA FLOW DIAGRAMS	
	5.2 SOLUTION&TECHNICAL ARCHITECTURE	
	5.3 USER STORIES	

6	PROJECT PLANNING & SCHEDULEING	43
	6.1 SPRINT PLANNING & ESTIMATION	
	6.2 SPRINT DELIVERY SCHEDULE	
	6.3 REPORTS FROM JIRA	
7	CODING & SOLUTIONING	55
	7.1 FEATURE 1	
	7.2 FEATURE 2	
	7.3 DATABASE SCHEMA (IF APPLICATION)	
8	TESTING	62
	8.1 TEST CASES	
	8.2 USER ACCEPTANCE TESTING	
9	RESULT	70
	9.1 PERFORMANCE METRICS	
10	ADVANTAGES & DISADVANTAGES	67
11	CONCLUSION	69
12	FUTURE SCOPE	70
13	APPENDIX	72
	SOURCE CODE	
	GITHUB & PROJECT DEMO LINK	

CHAPTER 1

INTRODUCTION

Customer service is the support and assistance businesses offer before, during, and after purchasing the products/services. Quality customer service adds an immense amount of value to a product and helps build long-lasting relationships with customers. Today's customer service is much more than traditional phone support. Rapid tech advancements reshaped the way businesses interact with customers and created the proliferation of digital service channels. Receiving fast, efficient, personalized support and a seamless experience is what consumers generally expect from brands these days.

1.1 PROJECT OVERVIEW

Web based project Customer service also known as client service is the provision of service to customers. Provided by a service representatives customer service is normally an integral part of company's customer value proposition.

An online comprehensive customer care solution is to manage customer interaction and complaints with service providers over phone or through and e-mail. The system should have capability to integrate with any service provider from any domain or industry like Banking, Telecom, Insurance .

1.2 PURPOSE

Customer care is when companies treat their customers with respect and kindness and build an emotional connection with them. It's something that can—and should—be handled by everyone on the team, not just a customer service representative or a customer success manager. Customer care is more than just delivering the services that consumers expect from the business or providing the right technical support. It's about meeting their emotional needs and fostering relationships. To do so, you must treat customers how they want to be treated. You need to listen to each individual's needs and find the best solution.

CHAPTER 2

LITERATURE SURVEY

In this paper they described the customer care concept with the help of CEM. Customer experience management (CEM) is the collection of processes a company uses to track, oversee and organize every interaction between a customer and the organization throughout the customer lifecycle. The goal of CEM is to optimize interactions from the customer's point of view and, as a result, promote customer loyalty. Customer experience management (CEM) is defined as “the discipline of managing and treating customer relationships as assets with the goal of transforming satisfied customers into loyal customers, and loyal customers into advocates of your brand.” A customer experience is an interaction between an organization and a customer as perceived through a customer's conscious and subconscious mind. It is a blend of an organization's rational performance, the

senses stimulated and the emotions evoked and intuitively measured against customer expectations across all moments of contact. This paper tells as, having access to online shopping has truly revolutionized and influenced our society as a whole. This use of technology has opened new doors and opportunities that enable for a more convenient lifestyle today. Variety, quick service and reduced prices were three significant ways in which online shopping influenced people from all over the world. However, this concept of online shopping led to the possibilities of fraud and privacy conflicts. Unfortunately, it has shown that it is possible for criminals to manipulate the system and access personal information. Luckily, today with the latest features of technology, measures are being taken in order to stop hackers and criminals from inappropriately accessing private databases. Through privacy and security policies, website designers are doing their best to put an end to this unethical practice.

In this paper they described the customer care concept with the help of CEM. Customer experience management (CEM) is the collection of processes a company uses to track, oversee and organize every interaction between a customer and the organization throughout the customer lifecycle. The goal of CEM is to optimize interactions from the customer's point of view and, as a result, promote customer loyalty. Customer experience management (CEM) is defined as “the discipline of managing and treating customer relationships as assets with the goal of transforming satisfied customers into loyal customers, and loyal customers into advocates of your brand.” A customer experience is an interaction between an organization and a customer as perceived through a customer’s conscious and subconscious mind. It is a blend of an organization’s rational performance, the senses stimulated and the emotions evoked and intuitively measured against customer expectations across all moments of contact. The output of the research

proposed in this paper would lead to effective measurement scales for the e-marketer to use in the identification of relevant inputs and outputs of an effective OCE for retail websites.

2.1 EXISTING PROBLEM

The existing system is a semi-automated at where the information is stored in the form of excel sheets in disk drives.

2.2 REFERENCES

- Mona N. Shah, Vineet Raitani, Aditya Oza, Kunal Gupta, Customer Satisfaction Study of the Mumbai Metro Service.
- Susan Rose, Neil Hair, Moira Clark, Online Customer Experience: A Review of the Business-to-Consumer Online Purchase Context, 2011.
- Ebenezer Paul Rajan, Customer Experience Management in Online Retailing: A Literature Review, Karpagam Academy of Higher Education, 2015.
- Shenbhaga vadivu Thangavel, A Study on customer Satisfaction towards Online Shopping, Sri Krishna College of Arts and Science, 2015.

2.3 PROBLEM STATEMENT DEFINITION

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service

CHAPTER 3

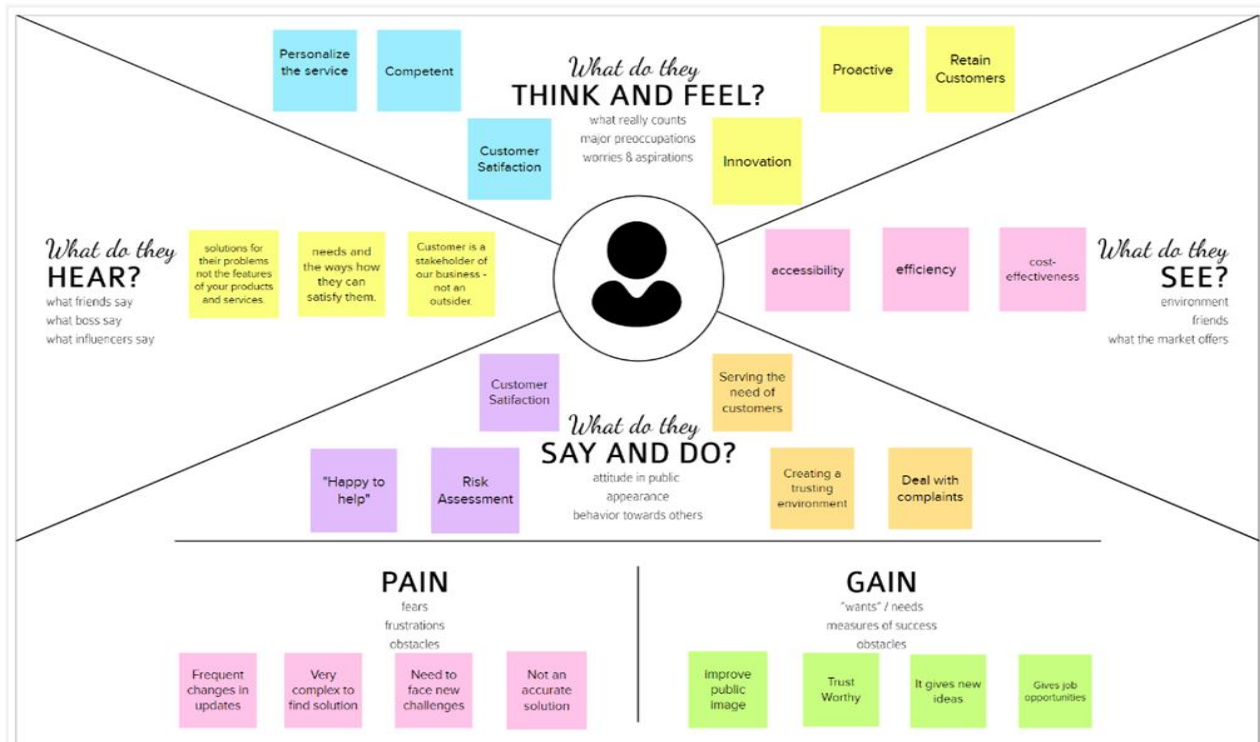
IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Reference: [Customer Care Registry-Empathy Map • Customer Care Registry \(mural.co\)](#)



3.2 IDEATION & BRAINSTORMING


Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Reference: <https://app.mural.co/t/customercareregistry2318/m/customercareregistry2318/1665330324388/31b08b9b843c7d627f10bd90997127e8308e2bba?sender=u3df3abdd0a74bf81ce394361>




Step-1: Team Gathering, Collaboration and Select the Problem Statement


Template



Brainstorm & idea prioritization


Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

 10 minutes to prepare
 1 hour to collaborate
 2-8 people recommended



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

 10 minutes

A


Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.


C

Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) 


1


Define your problem statement
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.


 5 minutes


PROBLEM


How might we can solve the issue given by the customer?


**Key rules of brainstorming**
To run an smooth and productive session


 Stay in topic.

 Encourage wild ideas.

 Defer judgment.

 Listen to others.

 Go for volume.

 If possible, be visual.

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

Augusta S

solution to customer	Customer privacy	Asking for rating
Live chat	Deals the problem	Providing service details
Allocating agent	Providing chatbot	Security
Customer satisfaction	Problem must faster	Email alert
Tracking of services	Notification alert	Customer needs
Notify Customer	Providing services	Agent details

Divyanandhi A

Fatima Stany J

Solution For Issues	Customer queries	Faster Services
Providing Tokens	Token Tracking	Solving Of Issues
Agent Provisioning	Service Provider	Fixing Bugs
Reduce customer churn	Cultivate customer loyalty	Acquire more customers
Customer Journey	Active listening	Create FAQ page
Send a token of care	Go the extra mile	Communicate next step

Yogeswari A

3

Group Ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

Customer

Solution for customer needs

Email alert

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Feedbacks

Customer Satisfaction

User feedback

Asking for rating

Services

Allocating agents

services provides on time

Tracking of complaints

Security

Customer privacy

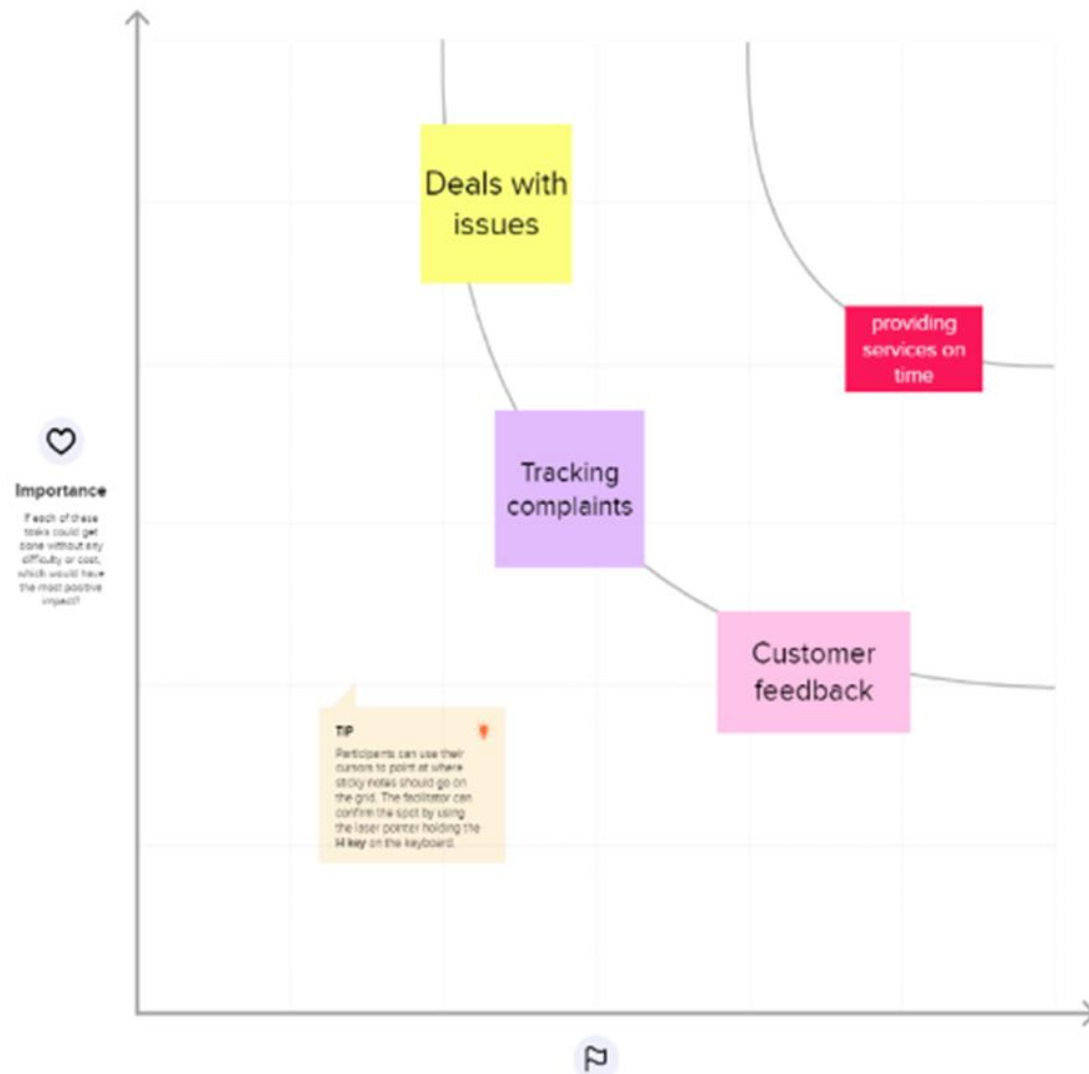
Security

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



3.3 PROPOSED SOLUTION

project team shall fill the following information in proposed solution template.

S.NO.	Parameter	Description
1	Problem Statement (Problem to be solved)	To solve the issues facing by the customer.
2	Idea / Solution description	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.
3	Novelty / Uniqueness	Each user will be assigned with an agent. They can view the status of their complaint.
4	Social impact /Customer Satisfaction	Customer can track their status with agent communication.
5	Business Model (Revenue Model)	Customer relationship have 24/7 email support. -Key partners are Third party applications,agents and customers.
6	Scalability of the Solution	User-generated content is often

		<p>underrated in customer support. When you have customers who'd like to share their experience using your product/service and also help other users with their questions, building a community that enables them to do so goes a long way in ensuring queries are answered on time.</p>
--	--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3.4 PROBLEM SOLUTION FIT

Project Title: Customer Care Registry			Project Design Phase-1- Problem Solution Fit			Team ID: PNT2022TMID50123		
Define CS, fit into	1. CUSTOMER SEGMENT(S) CS	6. CUSTOMER CC	5. AVAILABLE SOLUTIONS AS			Explore AS,		
	<ul style="list-style-type: none">Not able to solve their issues.Doesn't know the solution of their issues.	<ul style="list-style-type: none">Alert via email.Solution also provides insights in a graphical way.	<ul style="list-style-type: none">Reading guidelines carefully.Complaint the issues to the company.					
Focus on J&P, tap into BE, understand	2. JOBS-TO-BE-DONE / PROBLEMS J&P	9. PROBLEM ROOT CAUSE RC	7. BEHAVIOUR BE			Focus on J&P, tap into BE, understand		
	<ul style="list-style-type: none">Long wait times for customer service responses.Giving the necessary information for particular thing which needs for customer	<ul style="list-style-type: none">Unaware of the object.Customer service is ineffective.	<ul style="list-style-type: none">When the user doesn't have the knowledge about particular thing this kind of situation occurs.					
Identify strong TR & EM	3. TRIGGERS TR	10. YOUR SOLUTION SL	8. CHANNELS of BEHAVIOUR CH			Extract online & offline CH of BE		
	<ul style="list-style-type: none">Customer should know to solve their issues.	<p>To design a personal help desk.</p>	<p>8.1 ONLINE</p> <ul style="list-style-type: none">Data must be secured and updated in cloud storage. <p>8.2 OFFLINE</p> <ul style="list-style-type: none">Make sure that they find solutions.					
	4. EMOTIONS: BEFORE / AFTER EM							
	<p>Before: unease about something with an uncertain outcome (showing worry) After: pleasure of blessedness and brightness in face</p>							

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

FR NO.	Functional Requirement (Epic)	Sub Recruitment-(story-sub-task)
FR-1	User Registration	Registration through form Registration through Gmail Registration through Google
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login via Google with Email id and Password
FR-4	Admin Login	Login via Google with Email id and Password
FR-5	Query Form	Description of the issues contact information
FR-6	E-MAIL	Login alertness
FR-7	Feedback	Customer feedback

4.2 NON –FUNCTIONAL REQUIREMENTS

Following are the non- functional requirements of the proposed solution

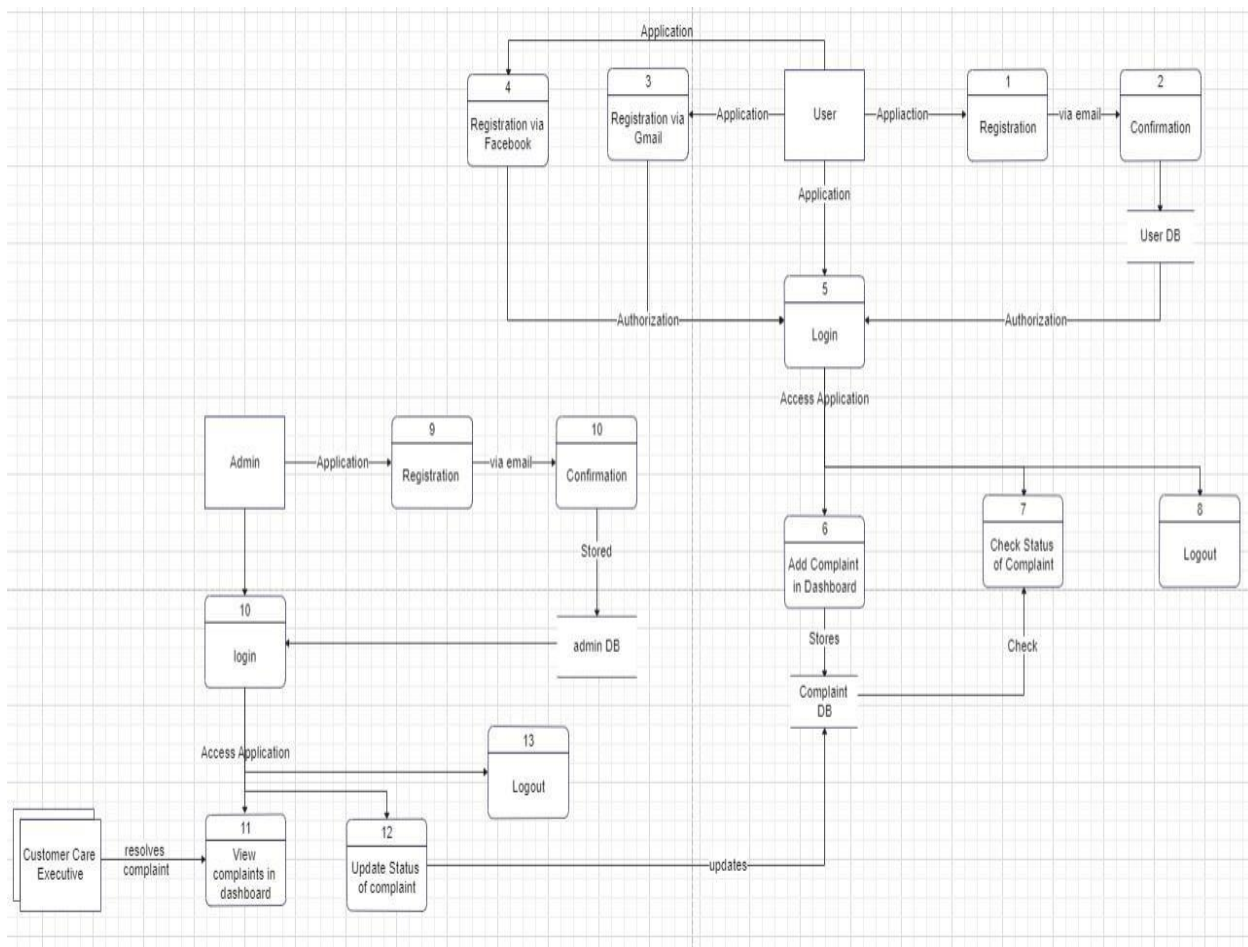
FR No.	Non-Functional Requirements	Description
NFR- 1	Usability	To provide the solution to the problem
NFR- 2	Security	Tracking the login Authentication
NFR-3	Reliability	Tracking of decade status through email
NFR-4	Performance	Effective development of web application
NFR-5	Availability	24/7 service
NFR-6	Scalability	Agents scalability as per the number of customers

CHAPTER 5

PROJECT DESIGN

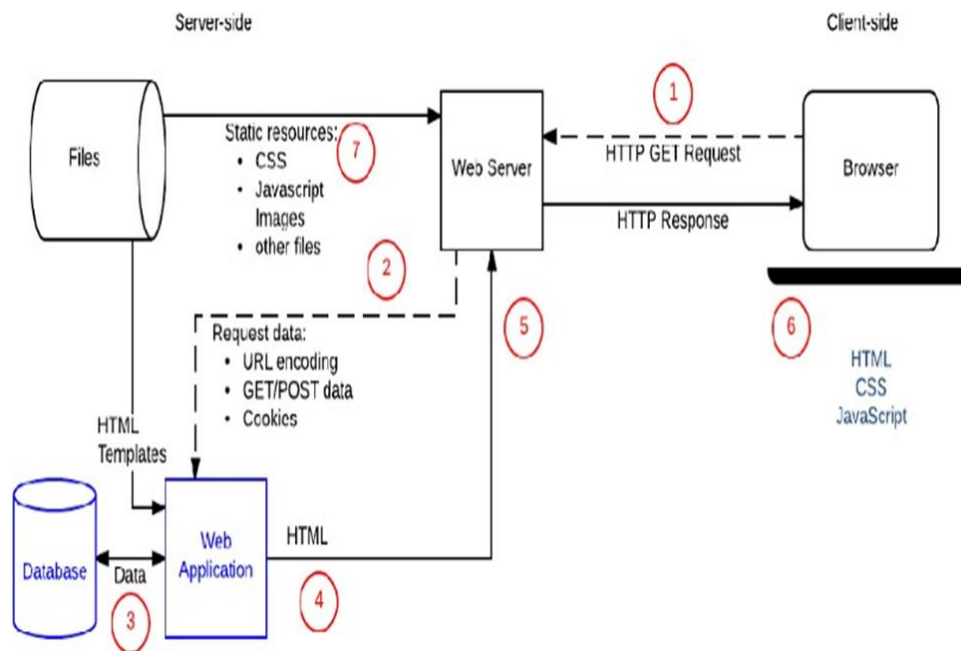
5.1 DATA FLOW DIAGRAMS

A DATA FLOW DIAGRAM (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



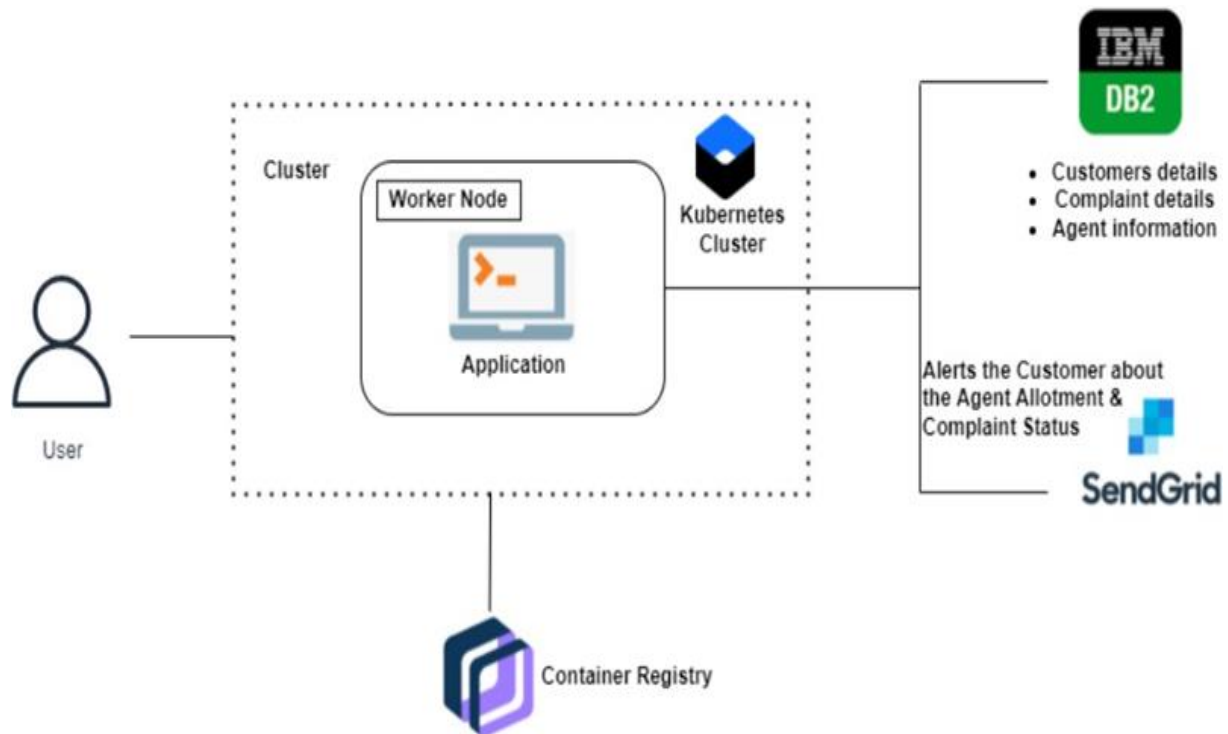
5.2 SOLUTION ARCHITECTURE

Solution architecture is a practice to provide ground for software development projects by tailoring IT solutions to specific business needs and defining their functional requirements and stages of implementation. It is comprised of many sub processes that draw guidance from various enterprise architecture viewpoints.



5.2 TECHNICAL ARCHITECTURE

Technology architecture associates application components from application architecture with technology components representing software and hardware components. Its components are generally acquired in the marketplace and can be assembled and configured to constitute the enterprise's technological infrastructure. Technology architecture provides a more concrete view of the way in which application components will be realized and deployed. It enables the migration problems that can arise between the different steps of the IS evolution path to be studied earlier. It provides a more precise means of evaluating responses to constraints (nonfunctional requirements) concerning the IS, notably by estimating hardware and network sizing needs or by setting up server or storage redundancy. Technology architecture concentrates on logistical and location problems related to hardware location, IS management capabilities, and the sites where the different parts of the IS are used. Technology architecture also ensures the delivered application components work together, confirming that the required business integration is supported.



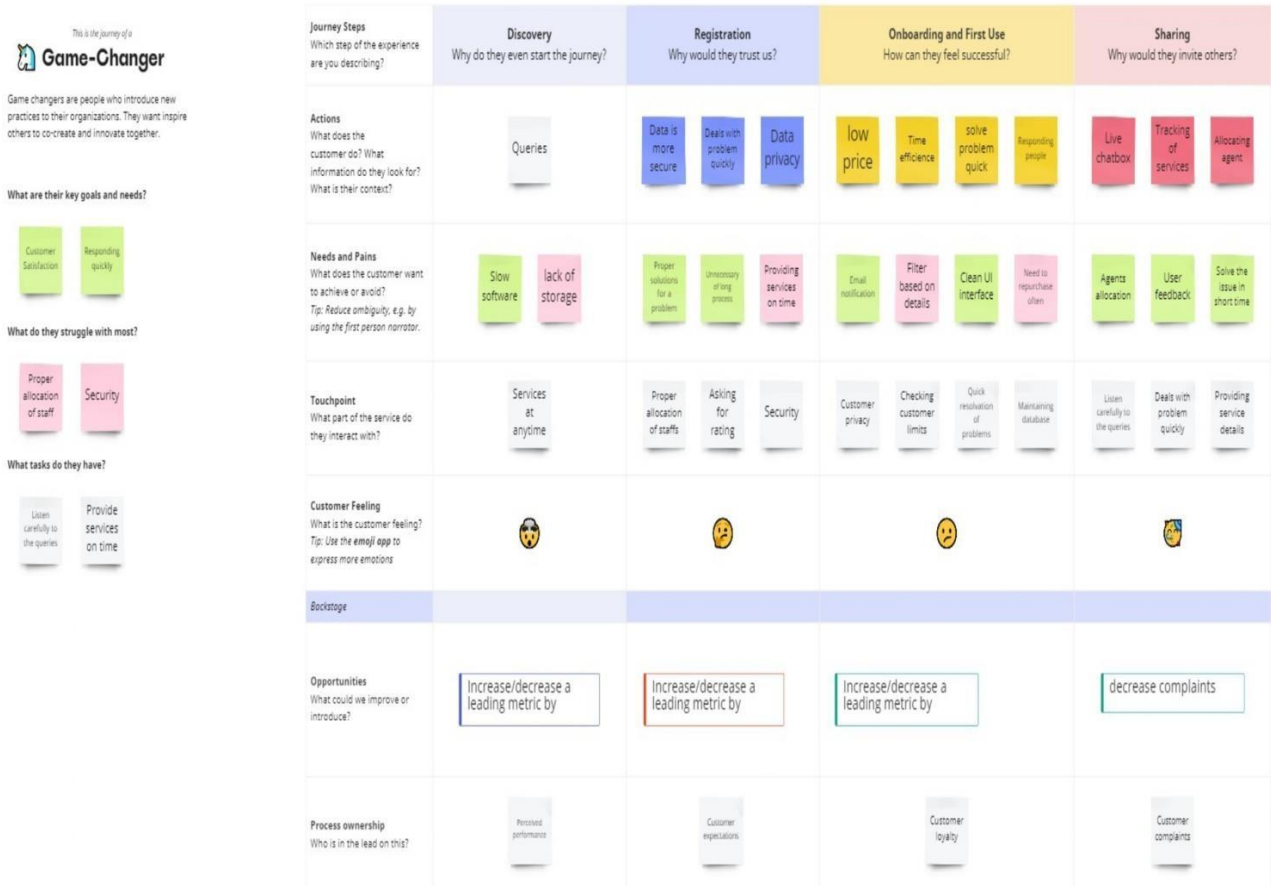
5.3 USER STORIES

The user will login into the website and go through the service available on the webpage.

The role of the admin is to check out the database about the availability and have a track off all the things that the users are going to service.

The user can directly talk to chat bot regarding the services. Get the recommendations based on information provided by the user.

Container of applications using docker, kubernetes and deployment the application. Create the documentation and final submit the application.



miro

Customer Reference:

https://miro.com/app/board/uXjVPOW6ggc=/?share_link_id=904722945180

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION,SCHEDULE

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement(Epic)	User Story Number	User Story/Task	Story points	Priority	Team Members
Sprint-1	User panel	USN-1	The user will login into the website and go through the services available on the webpage	2	High	Augusta S Fatima Stany J Yogeswari A Divyanandhi A
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the availability and have have a track of all the things that the user are going to services .	1	High	Yogeswari A Divyanandhi.A
Sprint-3	Chat bot	USN-3	The user can directly talk to chat bot regarding the services .get the	2	Low	Yogeswari A Divyanandhi A Fatima Stany J

			recommendation based on information provided by the user.			
Sprint-4	Final delivery	USN-4	Container of application using docker kubernetes and deployment the application create the documentation and final submit the applicatipn	2	Mediu m	Augusta S Fatima Stany J Yogeswari A Divyanandhi A

6.3 REPORT FROM JIRA

Go to our products page and choose the product you want to try..Fill in your details and select Agree and Sign up. Complete any further details in the signup process. It might take a few minutes for your site to be created.

CHAPTER 7

CODING & SOLUTIONING

7.1 FEATURE 1

Provide feedback

Send a complaint email

Use social media

File a complaint

CODING :

```
{ % block head % }
```

```
<title>Welcome</title>
```

```
{ % endblock % }
```

```
{ % block body % }
```

```
<body style="background-color:black;">
```

```
<div class="container-md" style="padding-top: 100px;">
```

```
<div class="row" style="height:100px;">
```

```
<h1 class="page-header text-primary mx-auto">WELCOME TO <span  
style="color:white;">ADMIN PAGE</span></h1>
```

```
</div>
```

```

<div class="row col-md-offset-3 col-md-5 ">
    {% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
    {% for category,message in messages %}
        <div class="alert alert-{{ category }}" text-center" style="color:
red;">{{ message }}</div>
    {% endfor %}
    {% endif %}
    {% endwith %}
</div>

</div>

<div class="row g-4 mb-4">

    <form action="{{ { url_for('remove') }}" method="POST">
        <div class="col mb-3">
            <button type="submit" class="btn btn-primary"><span
                style="color:black; font-weight: bolder;
">Delect</span></button>
        </div>
        <div class="col-3 form-group ">
            <input type="text" style="color: white; width: 150px;" class="form-
control" id="otpv" name="otpv"
                placeholder="Verification Code" required>
        </div>

```

</form>

<div class="col">

<a style="color: black; font-weight: bolder;" href="{{ url_for('home') }}"
class="btn btn-primary ">Logout

</div>

</div>

</div>

</div>

<div class="container-md">

<div id="accordion">

<div class="card">

<div class="card-header" id="headingOne">

<h5 class="mb-0">

<button class="btn btn-link" data-toggle="collapse" data-
target="#collapseOne"

aria-expanded="false" aria-controls="collapseOne">

<h2 style="color: black;">User Database</h2>

</button>

</h5>

</div>

<div id="collapseOne" class="collapse " aria-labelledby="headingOne"
data-parent="#accordion">

<div class="card-body">

<div class="row mx-auto" style="height: 50px;">

<input type="text" style="color: white; width: min-content;"
class="form-control"

id="myInput" name="myInput" onkeyup="myFunction()"
placeholder="Search for Names">

<div class="col" style="height:50px;">

<h2 style="color: black;"> TOTAL NUMBER OF USERS :

<span

style="color: black; font-weight: bolder; ">{ { message } }</h2>

</div>

</div>

<div class="row mx-auto ">

<table class="table table-striped table-light" id="myTable">

```

<thead class="thead-dark ">
  <tr>
    <th scope="row" onclick="sortTable(1)">DATE OF
JOIN</th>

    <th scope="row" onclick="sortTable(0)">ID</th>
    <th scope="row" onclick="sortTable(1)">NAME</th>
    <th scope="row" onclick="sortTable(0)">EMAIL</th>
    <th scope="row"
onclick="sortTable(1)">PASSWORD</th>
    <th scope="row"
onclick="sortTable(0)">PHONENUMBER</th>

    <th scope="row">DELECT</th>
  </tr>
</thead>
<tbody>
  { % for row in users % }
  <tr>
    <td style="color: black;">{ {row['DATE']}}</td>
    <td style="color: black;">{ {row['ID']}}</td>
    <td style="color: black;">{ {row["NAME"]}}</td>
    <td style="color: black;">{ {row["EMAIL"]}}</td>
    <td style="color: black;">{ {row["PASSWORD"]}}</td>
    <td style="color:
black;">{ {row['PHONENUMBER']}}</td>

```

```

        <td><a style="color: black;" href="/delete/{ {row['ID']}}
"
        class="btn btn-primary">DELETE {{
row['ID']}}</a>
        </td>
    </tr>
    { % endfor % }
</tbody>
</table>

</div>
</div>
</div>
</div>
<div class="card">
    <div class="card-header" id="headingTwo">
        <h5 class="mb-0">
            <button class="btn btn-link collapsed" data-toggle="collapse" data-
target="#collapseTwo"
                aria-expanded="false" aria-controls="collapseTwo">
                <h2 style="color: black;">AGENT DATABASE</h2>
            </button>
        </h5>
    </div>

```



```
<div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-parent="#accordion">
```

```
<div class="card-body">
```

```
<div class="row mx-auto" style="height: 50px;">
```

```
<input type="text" style="color: white; width: min-content;" class="form-control"
```

```
id="myInput1" name="myInput1" onkeyup="myFunction1()" placeholder="Search for Names">
```

```
<div class="col" style="height:50px;">
```

```
<h2 style="color: black;"> TOTAL NUMBER OF AGENT :
```

```
<span
```

```
style="color: black; font-weight: bolder;">{{ msgagent }}</span></h2>
```

```
</div>
```

```
</div>
```

```
<div class="row mx-auto ">
```

```
<table class="table table-striped table-light " id="myTable">
```

```
<thead class="thead-dark ">
```

```
<tr>
```

```
<th scope="row" onclick="sortTable(1)">DATE</th>
```

```
<th scope="row" onclick="sortTable(0)">ID</th>
```

```
<th scope="row" onclick="sortTable(1)">NAME</th>
```

```

        <th scope="row" onclick="sortTable(0)">EMAIL</th>
        <th
                                scope="row"
onclick="sortTable(1)">PASSWORD</th>
        <th    scope="row"    onclick="sortTable(0)">PHONE
NUMBER</th>
        <th
                                scope="row"
onclick="sortTable(1)">SERVICE_AGENT</th>
        <th
                                scope="row"
onclick="sortTable(0)">ADDRESS</th>
        <th scope="row" onclick="sortTable(1)">CITY</th>
        <th scope="row" onclick="sortTable(0)">STATE</th>
        <th    scope="row"    onclick="sortTable(1)">RESUME
LINK</th>

        <th scope="row">DELECT</th>
    </tr>
</thead>
<tbody>
    { % for row in agents % }
    <tr>
        <td style="color: black;">{ {row['DATE']}}</td>
        <td style="color: black;">{ {row['ID']}}</td>
        <td style="color: black;">{ {row["NAME"]}}</td>
        <td style="color: black;">{ {row["EMAIL"]}}</td>
        <td style="color: black;">{ {row["PASSWORD"]}}</td>
        <td
                                style="color:
black;">{ {row['PHONENUMBER']}}</td>

```

```

                <td
                                style="color:
black;">{{ row['SERVICE_AGENT'] }}</td>
                <td style="color: black;">{{ row['ADDRESS'] }}</td>
                <td style="color: black;">{{ row['CITY'] }}</td>
                <td style="color: black;">{{ row['STATE'] }}</td>
                <td
                                style="color:
black;">{{ row['RESUME_LINK'] }}</td>

```

```

                <td><a
                                style="color:
                                black;"
href="/agentdelete/{{ row['ID'] }}"
                                class="btn
                                btn-primary">DELETE
{{ row['ID'] }}</a>

```

```

        </td>
    </tr>
    {% endfor %}
</tbody>
</table>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

<div class="card">

```

```

<div class="card-header" id="headingThree">
  <h5 class="mb-0">
    <button class="btn btn-link" data-toggle="collapse" data-
target="#collapseThree"
    aria-expanded="false" aria-controls="collapseThree">
    <h2 style="color: black;">Complaint Datatbase</h2>
  </button>
</h5>
</div>

```

```

<div id="collapseThree" class="collapse" aria-
labelledby="headingThree" data-parent="#accordion">

```

```

  <div class="card-body">

```

```

    <div class="row mx-auto" style="height: 50px;">

```

```

      <input type="text" style="color: white; width: min-content;"
class="form-control"

```

```

        id="myInput" name="myInput" onkeyup="myFunction()"
placeholder="Search for Names">

```

```

    <div class="col" style="height:50px;">

```

```

      <h2 style="color: black;"> TOTAL NUMBER OF
COMPLAINT : <span
        style="color: black; font-weight: bolder;
">{{ issue }}</span></h2>

```

</div>

</div>

<div class="row mx-auto ">

<table class="table table-striped table-light " id="myTable">

<thead class="thead-dark ">

<tr>

<th scope="row" onclick="sortTable(0)">ID</th>

<th scope="row"

onclick="sortTable(1)">CUSTOMER_ID</th>

<th scope="row" onclick="sortTable(0)">DATE</th>

<th scope="row" onclick="sortTable(1)">EMAIL</th>

<th scope="row"

onclick="sortTable(0)">PHONE_NUMBER</th>

<th scope="row" onclick="sortTable(1)">TOPIC</th>

<th scope="row"

onclick="sortTable(0)">DESCRIPTION</th>

<th scope="row"

onclick="sortTable(1)">SERVICE_TYPE</th>

<th scope="row"

onclick="sortTable(0)">SERVICE_AGENT</th>

<th scope="row"

onclick="sortTable(1)">ADDRESS</th>

<th scope="row" onclick="sortTable(0)">STATE</th>

```

                                <th                                scope="row"
onclick="sortTable(1)">IMAGE_LINK</th>
                                <th scope="row" onclick="sortTable(0)">STATUS</th>
                                <th scope="row" onclick="sortTable(1)">DELETE</th>
                                <th                                scope="row"
onclick="sortTable(0)">ALLOCATE</th>

                                </tr>
                                </thead>
                                <tbody>
                                { % for row in complaint % }
                                <tr>
                                <td style="color: black;">{ {row['ID']}}</td>
                                <td                                style="color:
black;">{ {row["CUSTOMER_ID"]}}</td>
                                <td style="color: black;">{ {row["DATE"]}}</td>
                                <td style="color: black;">{ {row["EMAIL"]}}</td>
                                <td                                style="color:
black;">{ {row['PHONENUMBER']}}</td>
                                <td style="color: black;">{ {row["TOPIC"]}}</td>
                                <td                                style="color:
black;">{ {row['DESCRIPTION']}}</td>
                                <td                                style="color:
black;">{ {row['SERVICE_TYPE']}}</td>
                                <td                                style="color:
black;">{ {row['SERVICE_AGENT']}}</td>

```

```

<td style="color: black;">{{row['ADDRESS']}}</td>
<td style="color: black;">{{row['STATE']}}</td>
<td style="color: black;">{{row['IMAGE_LINK']}}</td>

{% if 'Completed' == row.STATUS %}
<td style="color: black;"><button type="button"
class="btn btn-success text-white"> {{row['STATUS']}} </button></td>
{% elif 'Agent Alloted' == row.STATUS %}
<td style="color: black;"><button type="button"
class="btn btn-warning text-white"> {{row['STATUS']}} </button></td>
{% elif 'Processing' == row.STATUS %}
<td style="color: black;"><button type="button"
class="btn btn-danger text-white"> {{row['STATUS']}} </button></td>
{% endif %}

<td><a style="color: white;"
href="/deletecomplaint/{{row['ID']}}"
class="btn btn-dark">DELECT {{row['ID']}}</a>
</td>
<td><a style="color: black;"
href="/viewagent/{{row['ID']}}" class="btn btn-primary">AGENT
{{row['ID']}}</a>
</td>
</tr>
{% endfor %}

```

```
        </tbody>
    </table>
```

```
    </div>
</div>
</div>
</div>
```

```
</div>
```

```
<script>
```

```
function myFunction() {
    var input, filter, table, tr, td, i, txtValue;
    input = document.getElementById("myInput");
    filter = input.value.toUpperCase();
    table = document.getElementById("myTable");
    tr = table.getElementsByTagName("tr");
    for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[2];
        if (td) {
            txtValue = td.textContent || td.innerText;
            if (txtValue.toUpperCase().indexOf(filter) > -1) {
                tr[i].style.display = "";
            } else {
                tr[i].style.display = "none";
            }
        }
    }
}
```



```
    }  
  }  
}
```

```
function myFunction1() {  
  var input, filter, table, tr, td, i, txtValue;  
  input = document.getElementById("myInput1");  
  filter = input.value.toUpperCase();  
  table = document.getElementById("myTable1");  
  tr = table.getElementsByTagName("tr");  
  for (i = 0; i < tr.length; i++) {  
    td = tr[i].getElementsByTagName("td")[2];  
    if (td) {  
      txtValue = td.textContent || td.innerText;  
      if (txtValue.toUpperCase().indexOf(filter) > -1) {  
        tr[i].style.display = "";  
      } else {  
        tr[i].style.display = "none";  
      }  
    }  
  }  
}
```

```
function sortTable(n) {  
  var table, rows, switching, i, x, y, shouldSwitch, dir, switchcount = 0;  
  table = document.getElementById("myTable");
```

```

switching = true;
//Set the sorting direction to ascending:
dir = "asc";
/*Make a loop that will continue until
no switching has been done:*/
while (switching) {
    //start by saying: no switching is done:
    switching = false;
    rows = table.rows;
    /*Loop through all table rows (except the
    first, which contains table headers):*/
    for (i = 1; i < (rows.length - 1); i++) {
        //start by saying there should be no switching:
        shouldSwitch = false;
        /*Get the two elements you want to compare,
        one from current row and one from the next:*/
        x = rows[i].getElementsByTagName("TD")[n];
        y = rows[i + 1].getElementsByTagName("TD")[n];
        /*check if the two rows should switch place,
        based on the direction, asc or desc:*/
        if (dir == "asc") {
            if (x.innerHTML.toLowerCase() > y.innerHTML.toLowerCase())
{
                //if so, mark as a switch and break the loop:
                shouldSwitch = true;
                break;

```

```

    }
} else if (dir == "desc") {
    if (x.innerHTML.toLowerCase() < y.innerHTML.toLowerCase())
{
    //if so, mark as a switch and break the loop:
    shouldSwitch = true;
    break;
}
}
}
if (shouldSwitch) {
    /*If a switch has been marked, make the switch
    and mark that a switch has been done:*/
    rows[i].parentNode.insertBefore(rows[i + 1], rows[i]);
    switching = true;
    //Each time a switch is done, increase this count by 1:
    switchcount++;
} else {
    /*If no switching has been done AND the direction is "asc",
    set the direction to "desc" and run the while loop again.*/
    if (switchcount == 0 && dir == "asc") {
        dir = "desc";
        switching = true;
    }
}
}
}

```

```
}  
</script>
```

```
{% endblock %}  
</body>
```

7.2 FEATURE 2

Make your request into a question

Explain the problem

CODINGS:

```
{% block head %}  
<title>Welcome</title>
```

```
{% endblock %}
```

```
{% block body %}
```

```
<body style="background-color:black;">
```

```
<div class="container-md" style="padding-top: 100px;">
```

```
<div class="row" style="height:100px;">
```

```
<h1 class="page-header text-primary mx-auto">WELCOME TO <span
style="color:white;">ADMIN PAGE</span></h1>
```

```
</div>
```

```
<div class="row col-md-offset-3 col-md-5 ">
```

```
{% with messages = get_flashed_messages(with_categories=true) %}
```

```
{% if messages %}
```

```
{% for category,message in messages %}
```

```
<div class="alert alert-{{category}} text-center" style="color:
red;">{{message}}</div>
```

```
{% endfor %}
```

```
{% endif %}
```

```
{% endwith %}
```

```
</div>
```

```
</div>
```

```
<div class="row g-4 mb-4">
```

```
<form action="{{ url_for('remove') }}" method="POST">
```

```
<div class="col mb-3">
```

```
<button type="submit" class="btn btn-primary"><span
```

```
style="color:black; font-weight: bolder;
```

```
</div>
```

```
<div class="col-3 form-group ">
```

```
        <input type="text" style="color: white; width: 150px;" class="form-  
control" id="otpv" name="otpv"
```

```
        placeholder="Verification Code" required>
```

```
    </div>
```

```
</form>
```

```
<div class="col">
```

```
    <a style="color: black; font-weight: bolder;" href="{ { url_for('home') } }"
```

```
    class="btn btn-primary ">Logout</a>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div class="container-md">
```

```
    <div id="accordion">
```

```
        <div class="card">
```

```
            <div class="card-header" id="headingOne">
```

```
                <h5 class="mb-0">
```

```
<button class="btn btn-link" data-toggle="collapse" data-
target="#collapseOne"
aria-expanded="false" aria-controls="collapseOne">
<h2 style="color: black;">User Database</h2>
</button>
</h5>
</div>
```

```
<div id="collapseOne" class="collapse " aria-labelledby="headingOne"
data-parent="#accordion">
<div class="card-body">
```

```
<div class="row mx-auto" style="height: 50px;">
```

```
<input type="text" style="color: white; width: min-content;"
class="form-control"
id="myInput" name="myInput" onkeyup="myFunction()"
placeholder="Search for Names">
```

```
<div class="col" style="height:50px;">
<h2 style="color: black;"> TOTAL NUMBER OF USERS :
<span
style="color: black; font-weight: bolder;
">{{ message }}</span></h2>
</div>
</div>
```

```

<div class="row mx-auto ">
  <table class="table table-striped table-light" id="myTable">
    <thead class="thead-dark ">
      <tr>
        <th scope="row" onclick="sortTable(1)">DATE OF
JOIN</th>
        <th scope="row" onclick="sortTable(0)">ID</th>
        <th scope="row" onclick="sortTable(1)">NAME</th>
        <th scope="row" onclick="sortTable(0)">EMAIL</th>
        <th scope="row"
onclick="sortTable(1)">PASSWORD</th>
        <th scope="row"
onclick="sortTable(0)">PHONENUMBER</th>
        <th scope="row">DELECT</th>
      </tr>
    </thead>
    <tbody>
      { % for row in users % }
      <tr>
        <td style="color: black;">{{ row['DATE'] }}</td>
        <td style="color: black;">{{ row['ID'] }}</td>

```



```

        <td style="color: black;">{{ row["NAME"] }}</td>
        <td style="color: black;">{{ row["EMAIL"] }}</td>
        <td style="color: black;">{{ row["PASSWORD"] }}</td>
        <td
                                style="color:
black;">{{ row["PHONENUMBER"] }}</td>

        <td><a style="color: black;" href="/delete/{{ row['ID'] }}"
"
                                class="btn
                                btn-primary">DELETE
                                {{
row['ID'] }}</a>

        </td>
    </tr>
    {% endfor %}
</tbody>
</table>

</div>
</div>
</div>
</div>
<div class="card">
    <div class="card-header" id="headingTwo">
        <h5 class="mb-0">
            <button class="btn btn-link collapsed" data-toggle="collapse" data-
target="#collapseTwo"
                aria-expanded="false" aria-controls="collapseTwo">

```

```
<h2 style="color: black;">AGENT DATABASE</h2>
</button>
</h5>
</div>
```

```
<div id="collapseTwo" class="collapse" aria-labelledby="headingTwo"
data-parent="#accordion">
  <div class="card-body">
```

```
    <div class="row mx-auto" style="height: 50px;">
      <input type="text" style="color: white; width: min-content;"
class="form-control"
      id="myInput1" name="myInput1" onkeyup="myFunction1()"
placeholder="Search for Names">
      <div class="col" style="height:50px;">
```

```
    <h2 style="color: black;"> TOTAL NUMBER OF AGENT :
<span
      style="color:      black;      font-weight:      bolder;
">{{ msgagent }}</span></h2>
    </div>
  </div>
```

```
<div class="row mx-auto ">
  <table class="table table-striped table-light " id="myTable">
```

```

<thead class="thead-dark ">
  <tr>
    <th scope="row" onclick="sortTable(1)">DATE</th>
    <th scope="row" onclick="sortTable(0)">ID</th>
    <th scope="row" onclick="sortTable(1)">NAME</th>
    <th scope="row" onclick="sortTable(0)">EMAIL</th>
    <th
                                scope="row"
onclick="sortTable(1)">PASSWORD</th>
    <th    scope="row"    onclick="sortTable(0)">PHONE
NUMBER</th>
    <th
                                scope="row"
onclick="sortTable(1)">SERVICE_AGENT</th>
    <th
                                scope="row"
onclick="sortTable(0)">ADDRESS</th>
    <th scope="row" onclick="sortTable(1)">CITY</th>
    <th scope="row" onclick="sortTable(0)">STATE</th>
    <th    scope="row"    onclick="sortTable(1)">RESUME
LINK</th>
    <th scope="row">DELECT</th>
  </tr>
</thead>
<tbody>
  { % for row in agents % }
  <tr>
    <td style="color: black;">{ {row['DATE']}}</td>
    <td style="color: black;">{ {row['ID']}}</td>

```

```

        <td style="color: black;">{{ row["NAME"]}}</td>
        <td style="color: black;">{{ row["EMAIL"]}}</td>
        <td style="color: black;">{{ row["PASSWORD"]}}</td>
        <td
                                style="color:
black;">{{ row['PHONENUMBER']}}</td>
        <td
                                style="color:
black;">{{ row['SERVICE_AGENT']}}</td>
        <td style="color: black;">{{ row['ADDRESS']}}</td>
        <td style="color: black;">{{ row['CITY']}}</td>
        <td style="color: black;">{{ row['STATE']}}</td>
        <td
                                style="color:
black;">{{ row['RESUME_LINK']}}</td>

        <td><a
                                style="color:
                                black;"
href="/agentdelete/{{ row['ID']}} "
                                class="btn
                                btn-primary">DELETE
{{ row['ID']}}</a>

        </td>
    </tr>
    {% endfor %}
</tbody>
</table>

</div>

```

</div>

</div>

</div>

<div class="card">

<div class="card-header" id="headingThree">

<h5 class="mb-0">

<button class="btn btn-link" data-toggle="collapse" data-target="#collapseThree"

aria-expanded="false" aria-controls="collapseThree">

<h2 style="color: black;">Complaint Datatbase</h2>

</button>

</h5>

</div>

<div id="collapseThree" class="collapse" aria-labelledby="headingThree" data-parent="#accordion">

<div class="card-body">

<div class="row mx-auto" style="height: 50px;">

<input type="text" style="color: white; width: min-content;" class="form-control"

id="myInput" name="myInput" onkeyup="myFunction()" placeholder="Search for Names">

```

<div class="col" style="height:50px;">
    <h2 style="color: black;"> TOTAL NUMBER OF
COMPLAINT : <span
    style="color: black; font-weight: bolder;
">{{ issue }}</span></h2>
</div>
</div>

```

```

<div class="row mx-auto ">
    <table class="table table-striped table-light " id="myTable">
        <thead class="thead-dark ">
            <tr>
                <th scope="row" onclick="sortTable(0)">ID</th>
                <th scope="row"
onclick="sortTable(1)">CUSTOMER_ID</th>
                <th scope="row" onclick="sortTable(0)">DATE</th>
                <th scope="row" onclick="sortTable(1)">EMAIL</th>
                <th scope="row"
onclick="sortTable(0)">PHONE_NUMBER</th>
                <th scope="row" onclick="sortTable(1)">TOPIC</th>
                <th scope="row"
onclick="sortTable(0)">DESCRIPTION</th>
                <th scope="row"
onclick="sortTable(1)">SERVICE_TYPE</th>

```

```
 scope="row" onclick="sortTable(0)">SERVICE_AGENT</th>  scope="row" onclick="sortTable(1)">ADDRESS</th>  scope="row" onclick="sortTable(0)">STATE</th>  scope="row" onclick="sortTable(1)">IMAGE_LINK</th>  scope="row" onclick="sortTable(0)">STATUS</th>  scope="row" onclick="sortTable(1)">DELECT</th>  scope="row" onclick="sortTable(0)">ALLOCATE</th> | | | | | | |
```

```

</tr>
</thead>
<tbody>
{ % for row in complaint % }
<tr>
<td style="color: black;">{ {row['ID']}}</td>
<td style="color:
black;">{ {row["CUSTOMER_ID"]}}</td>
<td style="color: black;">{ {row["DATE"]}}</td>
<td style="color: black;">{ {row["EMAIL"]}}</td>
<td style="color:
black;">{ {row['PHONENUMBER']}}</td>
<td style="color: black;">{ {row["TOPIC"]}}</td>

```

```

                                <td                                style="color:
black;">{{row['DESCRIPTION']}}</td>

                                <td                                style="color:
black;">{{row['SERVICE_TYPE']}}</td>

                                <td                                style="color:
black;">{{row['SERVICE_AGENT']}}</td>

                                <td style="color: black;">{{row['ADDRESS']}}</td>
                                <td style="color: black;">{{row['STATE']}}</td>
                                <td style="color: black;">{{row['IMAGE_LINK']}}</td>

                                {% if 'Completed' == row.STATUS %}
                                <td    style="color:    black;"><button    type="button"
class="btn btn-success text-white"> {{row['STATUS']}} </button></td>
                                {% elif 'Agent Alloted' == row.STATUS %}
                                <td    style="color:    black;"><button    type="button"
class="btn btn-warning text-white"> {{row['STATUS']}} </button></td>
                                {% elif 'Processing' == row.STATUS %}
                                <td    style="color:    black;"><button    type="button"
class="btn btn-danger text-white"> {{row['STATUS']}} </button></td>
                                {% endif %}

                                <td><a                                style="color:                                white;"
href="/deletecomplaint/{{row['ID']}} "
                                class="btn btn-dark">DELECT {{row['ID']}}</a>
                                </td>

```



```

                <td><a
                    style="color:
                        black;"
href="/viewagent/{ {row['ID']} }"
                    class="btn
                        btn-primary">AGENT
{ {row['ID']} }</a>

```

```

            </td>

```

```

        </tr>

```

```

    { % endfor % }

```

```

</tbody>

```

```

</table>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

<script>

```

```

function myFunction() {
    var input, filter, table, tr, td, i, txtValue;
    input = document.getElementById("myInput");
    filter = input.value.toUpperCase();
    table = document.getElementById("myTable");
    tr = table.getElementsByTagName("tr");
    for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[2];
        if (td) {

```

```

        txtValue = td.textContent || td.innerText;
        if (txtValue.toUpperCase().indexOf(filter) > -1) {
            tr[i].style.display = "";
        } else {
            tr[i].style.display = "none";
        }
    }
}
}

```

```

function myFunction1() {
    var input, filter, table, tr, td, i, txtValue;
    input = document.getElementById("myInput1");
    filter = input.value.toUpperCase();
    table = document.getElementById("myTable1");
    tr = table.getElementsByTagName("tr");
    for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[2];
        if (td) {
            txtValue = td.textContent || td.innerText;
            if (txtValue.toUpperCase().indexOf(filter) > -1) {
                tr[i].style.display = "";
            } else {
                tr[i].style.display = "none";
            }
        }
    }
}

```

```
}  
}
```

```
function sortTable(n) {  
    var table, rows, switching, i, x, y, shouldSwitch, dir, switchcount = 0;  
    table = document.getElementById("myTable");  
    switching = true;  
    //Set the sorting direction to ascending:  
    dir = "asc";  
    /*Make a loop that will continue until  
    no switching has been done:*/  
    while (switching) {  
        //start by saying: no switching is done:  
        switching = false;  
        rows = table.rows;  
        /*Loop through all table rows (except the  
        first, which contains table headers):*/  
        for (i = 1; i < (rows.length - 1); i++) {  
            //start by saying there should be no switching:  
            shouldSwitch = false;  
            /*Get the two elements you want to compare,  
            one from current row and one from the next:*/  
            x = rows[i].getElementsByTagName("TD")[n];  
            y = rows[i + 1].getElementsByTagName("TD")[n];  
            /*check if the two rows should switch place,  
            based on the direction, asc or desc:*/
```

```

if (dir == "asc") {
    if (x.innerHTML.toLowerCase() > y.innerHTML.toLowerCase())
    {
        //if so, mark as a switch and break the loop:
        shouldSwitch = true;
        break;
    }
} else if (dir == "desc") {
    if (x.innerHTML.toLowerCase() < y.innerHTML.toLowerCase())
    {
        //if so, mark as a switch and break the loop:
        shouldSwitch = true;
        break;
    }
}
}

if (shouldSwitch) {
    /*If a switch has been marked, make the switch
    and mark that a switch has been done:*/
    rows[i].parentNode.insertBefore(rows[i + 1], rows[i]);
    switching = true;
    //Each time a switch is done, increase this count by 1:
    switchcount++;
} else {
    /*If no switching has been done AND the direction is "asc",
    set the direction to "desc" and run the while loop again.*/

```

```
        if (switchcount == 0 && dir == "asc") {  
            dir = "desc";  
            switching = true;  
        }  
    }  
}  
}  
    </script>  
{% endblock %}
```

CHAPTER 8

TESTING

8.1 TEST CASES

Verify that all the required fields – username, email, password, confirm password, etc are present on the registration page. Verify that on passing valid values, a user should get registered and the same should be allowed to login to the application. Verify that if a user tries to register an existing username then an error message should get displayed.

TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
User_Page_TC_O1	Functional	USER PAGE	Verify user is able to see the Show Complaint popup when user clicked on popup	1.Enter URL and click go 2.Scroll down 3.Verify login/signup popup displayed or not	http://169.51.204.215:30106/	Show Complaint popup should display	Working as expected	PASS	Successful			Augusta S
User_Page_TC_O2	UI	USER PAGE	Verify the User has No Complaint	Click on the Uri and go to user page by giving Correct Credentials	http://169.51.204.215:30106/	No Complaint should shown	Working as expected	PASS	Successful			Fatima Stany J
User_Page_TC_O3	UI	USER PAGE	Verify User Total Complaint is Zero	Click on the Uri and go to user page by giving Correct Credentials	http://169.51.204.215:30106/	Total Number of Complaint is Zero	Working as expected	PASS	Successful			Dhyananidhi A

Admin_Page_TC_OO4	Functional	Admin Page	Admin can see the Agent Database	1 Enter URL(http://169.51.204.215:30106) and click go 2 Enter the Credentials for the admin page and submit	http://169.51.204.215:30106/	Agent Database should display on show agent database.	Working as expected	PASS	Successful			Yogeshwari A
Admin_Page_TC_OO5	Functional	Admin Page	Admin can delete the Agent Database	1 Enter URL(http://169.51.204.215:30106) and click go 2 Click on submit by giving correct credentials to the admin Page	http://169.51.204.215:30106/	Delete the agent Database	Working as expected	PASS	Successful			Yogeshwari A
Admin_Page_TC_OO6	Functional	Admin Page	Verify the overall Delete the database for User	1 Enter URL(http://169.51.204.215:30106) and click go 2 Click on submit by giving correct credentials to the admin Page 3 After type the "A" in the Text box for the agent database delete	http://169.51.204.215:30106/	Delete the overall Agent database delete.	Working as expected	PASS	Successful			Yogeshwari A

Test Case (SPRINT 02)

4

Agent_Register_TC_O01	Functional	AGENT REGISTER	Verify Id sent to customer email address	1 Enter URL(http://169.51.204.215:30106) and click go 1.Register the account by giving credentials 2. Click on button Submit	http://169.51.204.215:30106/	Email sent successfully	Working as expected	PASS	Successful			Augusta S
Web_Chat_TC_O01	Functional	WEB CHAT	Click on the Web chat button	1 Enter URL(http://169.51.204.215:30106) and click go 1 Click on the Web Chat Button	http://169.51.204.215:30106/	Web chat popup	Working as expected	PASS	Successful			Augusta S
Web_chat_TC_O02	UI	WEB CHAT	Web chat button visible	1 Enter URL(http://169.51.204.215:30106) and click go 1 shown on the Web Chat Button	http://169.51.204.215:30106/	Web chat visible	Working as expected	PASS	Successful			Fatima Stany J
Admin_Login_TC_O03	Functional	AGENT LOGIN	Verify user is able to get login id on emails	1 Enter URL(http://169.51.204.215:30106) and click go 2. To the Agent Login page getting of emails	http://169.51.204.215:30106/	Get Notified by Emails	Working as expected	PASS	Successful			Divyanandhi A

Test Case (SPRINT 02)

6

Agent_Login_TC_014	UI	AGENT Login	Visible for text field for enter email id	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and see your testfields	http://169.51.204.215:30106/	Text Fields for Email in Agent Page	Working as expected	PASS	Successful			Augusta S
LoginPage_TC_015	UI	USER Login	Visible for text field for enter email id	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and see your testfields	http://169.51.204.215:30106/	Text Fields for Email in Agent Page	Working as expected	PASS	Successful			Fatima Stany J
Agent_Login_TC_016	Functional	AGENT Login	Visible for Password on Forget Password	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the Agent Forget Page after verification Password should Visible	http://169.51.204.215:30106/	Password should Visible	Working as expected	PASS	Successful			Divyanandhi A

8.2 USER ACCEPTING TESTING

Checks whether a product is the right one for the end users. It has other names, e.g., end-user testing, operational, application, beta testing, or validation but they describe the same thing. In quality assurance, it's important to distinguish between validation and verification.

1. Purpose of Document

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

2. Defect Analysis

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

Resolution	Severty 1	Severty 2	Severty 3	Severty 4	Subttoal
BY design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduce	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

3. Test Case Analysis

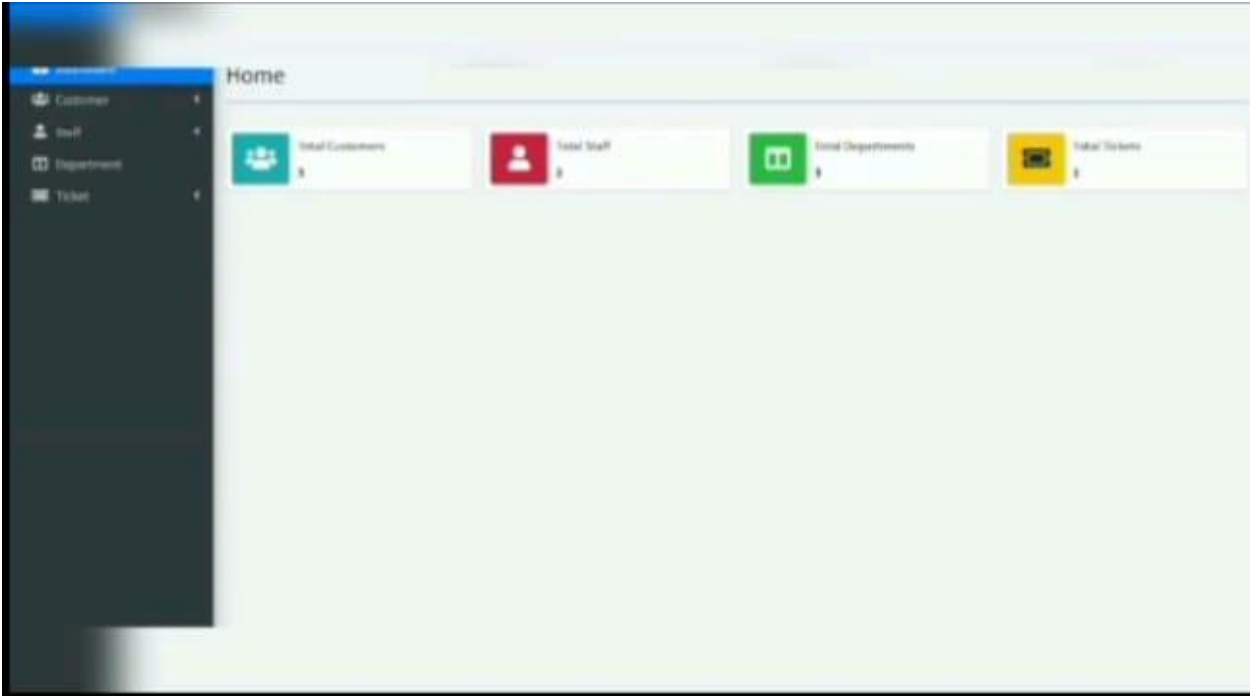
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource	3	0	0	3

Shipping				
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

CHAPTER 9
RESULTS

9.1 PERFORMANCE METRICS



Administrator Customer Support System

New Customer

Personal Information

First Name

Middle Name

Last Name

Contact No.

Address

System Credentials

Email

Password

Confirm Password

Administrator Customer Support System

Department List

[New Department](#)

Show 1 of 3 entries

#	Name	Description	Action
1	Feedbacks	customer can share their expectations on improvement of support	Action
2	Issues	customer can raise their tickets and staffs will be assigned to help them	Action
3	Message Logging	delaying in resolving and sending msg	Action

Showing 1 to 3 of 3 entries

Previous 1 Next

The screenshot shows the 'New Ticket' form in the 'Customer Support System'. The left sidebar has a dark theme with a 'Ticket' menu item highlighted in blue. The main content area has a light blue header with the title 'New Ticket'. The form contains three input fields: 'Subject', 'Customer', and 'Department', each with a dropdown arrow. Below these is a 'Description' field with a rich text editor toolbar. The toolbar includes icons for bold, italic, underline, strikethrough, link, unlink, bulleted list, numbered list, indent, outdent, undo, redo, and a help icon. The 'Customer' and 'Department' fields have a placeholder text 'Please select from...'. The 'Description' field is currently empty.

The screenshot shows the 'View Ticket' page in the Customer Support System. The sidebar on the left contains the following items:

- Administrator
- Dashboard
- Customers
- Staff
- Manage tickets
- Tickets** (selected)

The main content area displays the following ticket information:

Ticket

Subject

- Issues on ID

Customer

- S. Angus Jones

Status

- Open

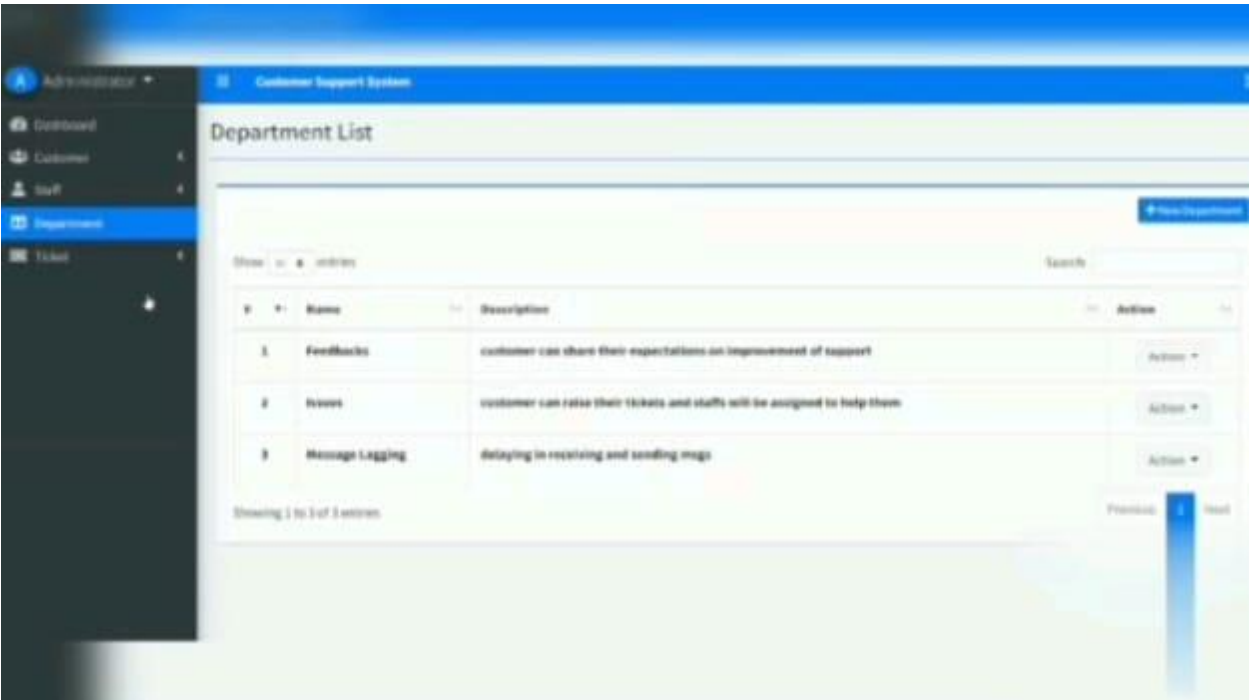
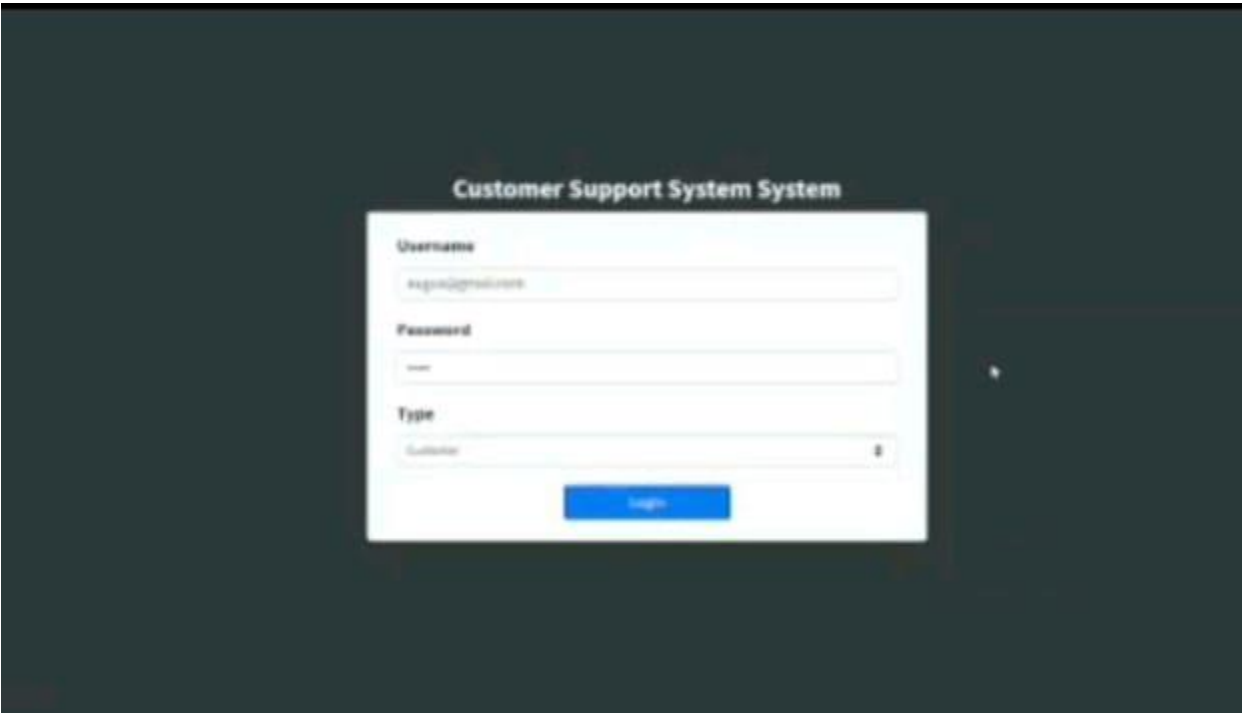
Next Support From

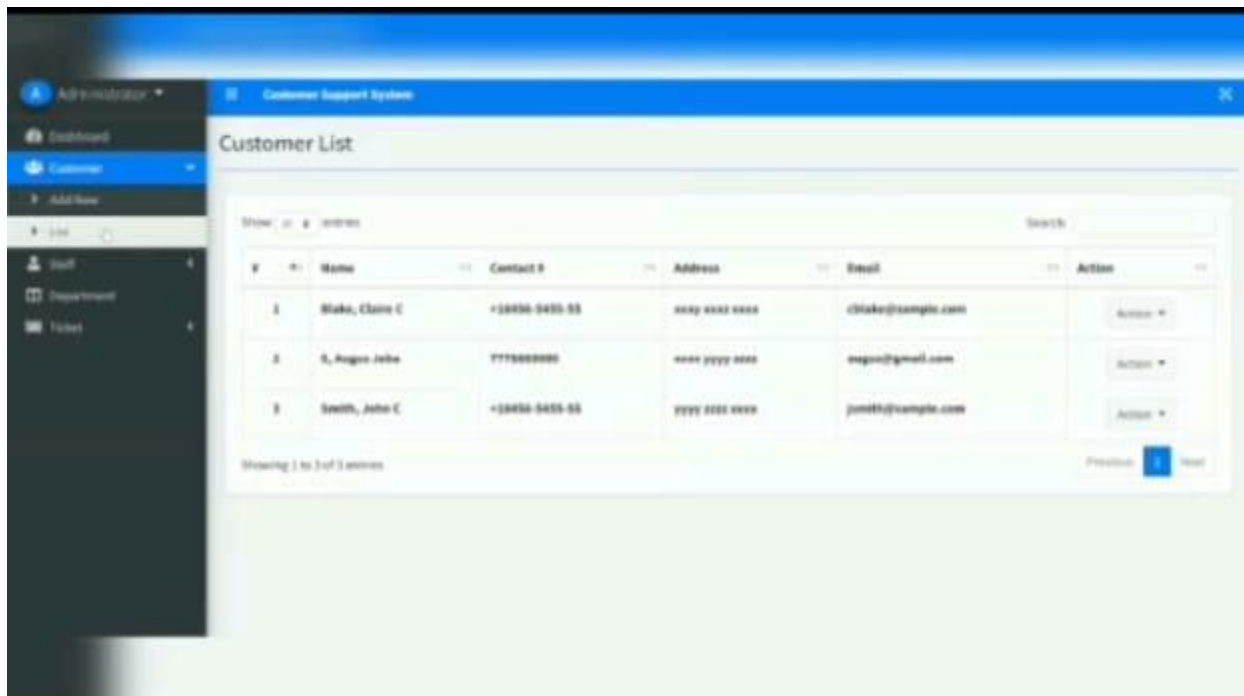
- Issues

Comment/s

New Comment

Issues on ID system and logging system...





CHAPTER 10

ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- Record and document customer conversations over a period of time
- Add a professional touch to your customer service using email signatures
- Automated email notifications can be used to update customers about the status of their issue or support ticket
- Easily attach relevant images, videos, docs, or other files

DISADVANTAGES:

- Delayed email responses can make customers feel frustrated
- Keeping track of emails can get challenging when you receive hundreds of them every day
- Typing long replies can be time-consuming

CHAPTER 11

CONCLUSION

Customer service is important to every business! Without customers, no business can survive. Customers will go where they are treated fairly and with respect, and even spend more money at such a business. Think the hardest thing is to get all employees to share in this goal.

CHAPTER 12

FUTURE SCOPE

- Self-service: Customers want to figure it out themselves, if they can do so without hassle or consequence. Community-based service: Customers helping one another figure it out is going to continue as a trend. Predictive support: This requires the innovative use of data, but providing “help” to customers before they know they need it is the holy grail of customer support, and it’s coming.

CHAPTER 13

APPENDIX

SOURCE CODE:

```
from audioop import add
import datetime
from unicodedata import name
from sib_api_v3_sdk.rest import ApiException
from pprint import pprint
from flask import Flask, render_template, request, redirect, url_for, session,
flash
from markupsafe import escape
from flask import *
import ibm_db
import sib_api_v3_sdk
from init import randomnumber
from init import id
from init import hello
import datetime
conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME=;PORT=;SECURITY=S
SL;;UID=;PWD=", "", "")
print(conn)
```



```

print("connection successful...")
app = Flask(__name__)
app.secret_key = 'your secret key'
@app.route('/')
def home():
    message = "TEAM ID : PNT2022TMID37544" +" "+ "BATCH ID : B1-1M3E "
    return render_template('index.html',mes=message)
@app.route('/home', methods=['POST', 'GET'])
def index():
    return render_template('index.html')
@app.route('/agentRegister', methods=['POST', 'GET'])
def agentRegister():
    return render_template('agentregister.html')

@app.route('/forgotpass', methods=['POST', 'GET'])
def forgotpass():
    return render_template('forgot.html')
@app.route('/forgot', methods=['POST', 'GET'])
def forgot():
    try:
        global randomnumber
        ida = request.form['custid']

        print(api_response)
        message = "Email send to:"+e+" for password"

```

```

        flash(message, "success")
except ApiException as e:
    print("Exception when calling SMTPApi->send_transac_email: %s\n" %
e)

    flash("Error in sending mail")
except:
    flash("Your didn't Signin with this account")
finally:
    return render_template('forgot.html')
@app.route('/agentforgot', methods=['POST', 'GET'])
def agentforgot():
    try:
        global randomnumber
        ida = request.form['custid']
        print(ida)
        global id
        id = ida
        sql = "SELECT EMAIL,NAME FROM AGENT WHERE id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, ida)
        ibm_db.execute(stmt)
        emailf = ibm_db.fetch_both(stmt)
        while emailf != False:
            e = emailf[0]
            n = emailf[1]
            Break

```

```

    print(api_response)
    message = "Email send to:"+e+" for OTP"
    flash(message, "success")
except ApiException as e:
    print("Exception when calling SMTPApi->send_transac_email: %s\n" %
e)
    flash("Error in sending mail")
except:
    flash("Your didn't Signin with this account")
finally:
    return render_template('forgot.html')
@app.route('/agentotp', methods=['POST', 'GET'])
def agentotp():
    try:
        otp = request.form['otp']
        cusid = id
        print(id)
        sql = "SELECT PASSWORD FROM AGENT WHERE ID=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, cusid)
        ibm_db.execute(stmt)
        otpf = ibm_db.fetch_both(stmt)
        while otpf != False:
            verify = otpf[0]
            Break

```

```

    if otp == str(randomnumber):
        msg = "Your Password is "+verify+"
        flash(msg, "success")
        return render_template('forgot.html')
    else:
        flash("Wrong Otp", "danger")
    finally:
        return render_template('forgot.html')
@app.route('/remove', methods=['POST', 'GET'])
def remove():
    otp = request.form['otpv']
    if otp == 'C':
        try:
            insert_sql = f"delete from customer"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prepare_stmt)
            flash("deleted successfully the Customer", "success")
        except:
            flash("No data found in Customer", "danger")
        finally:
            return redirect(url_for('signupage'))
    if otp == 'A':
        try:
            insert_sql = f"delete from AGENT"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prepare_stmt)

```

```

        flash("deleted successfully the Agents", "success")
    except:
        flash("No data found in Agents", "danger")
    finally:
        return redirect(url_for('signuppage'))
if otp == 'C':
    try:
        insert_sql = f"delete from AGENT"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.execute(prepare_stmt)
        flash("deleted successfully the Complaints", "success")
    except:
        flash("No data found in Complaints", "danger")
    finally:
        return redirect(url_for('signuppage'))
@app.route('/welcome', methods=['POST', 'GET'])
def welcome():
    try:
        id = hello
        sql = "SELECT
ID,DATE,TOPIC,SERVICE_TYPE,SERVICE_AGENT,DESCRIPTION,STA
TUS FROM ISSUE WHERE CUSTOMER_ID =?"
        agent = []
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)

```

```

    otpf = ibm_db.fetch_both(stmt)
    while otpf != False:
        agent.append(otpf)
        otpf = ibm_db.fetch_both(stmt)
    sql = "SELECT COUNT(*) FROM ISSUE WHERE CUSTOMER_ID =
?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, id)
    ibm_db.execute(stmt)
    t = ibm_db.fetch_both(stmt)

    return render_template("welcome.html", agent=agent, message=t[0])
except:
    return render_template("welcome.html")
@app.route('/loginagent', methods=['GET', 'POST'])
def loginagent():
    if request.method == 'POST':
        try:
            global loginagent
            id = request.form['idn']
            loginagent = id
            password = request.form['password']
            sql = f"select * from AGENT where id='{escape(id)}' and
password='{escape(password)}'"
            stmt = ibm_db.exec_immediate(conn, sql)
            data = ibm_db.fetch_both(stmt)

```

```

    if data:
        session["name"] = escape(id)
        session["password"] = escape(password)
        return redirect(url_for("agentwelcome"))
    else:
        flash("Mismatch in credentials", "danger")
except:
    flash("Error in Insertion operation", "danger")
return render_template("signinpageagent.html")
@app.route('/delete/<ID>')
def delete(ID):
    sql = f"select * from customer where Id='{escape(ID)}'"
    print(sql)
    stmt = ibm_db.exec_immediate(conn, sql)
    student = ibm_db.fetch_row(stmt)
    if student:
        sql = f"delete from customer where id='{escape(ID)}'"
        stmt = ibm_db.exec_immediate(conn, sql)

        flash("Delected Successfully", "success")
        return redirect(url_for("admin"))
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        try:

```

```

x = datetime.datetime.now()
y = x.strftime("%Y-%m-%d %H:%M:%S")
name = request.form['name']
email = request.form['email']
password = request.form['password']
phonenumber = request.form['phonenumber']
sql = "SELECT * FROM customer WHERE email = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, email)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
if account:
    flash("Record Already found", "success")
else:
    insert_sql = "insert into
customer(name,email,password,phonenumber,DATE)values(?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, name)
    ibm_db.bind_param(prepare_stmt, 2, email)
    ibm_db.bind_param(prepare_stmt, 3, password)
    ibm_db.bind_param(prepare_stmt, 4, phonenumber)
    ibm_db.bind_param(prepare_stmt, 5, y)
    ibm_db.execute(prepare_stmt)
    flash("Your Information Stored Successful. Kindly check mail for Id
!", "success")
    sql = "SELECT id FROM Customer WHERE email=?"

```



```

stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, email)
ibm_db.execute(stmt)
hi = ibm_db.fetch_tuple(stmt)
configuration = sib_api_v3_sdk.Configuration()
configuration.api_key['api-key'] = "
api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
sib_api_v3_sdk.ApiClient(configuration))
subject = "Registering Account"

html_content = " <html><body><h1>Thanks for Registering into
Customer Care Registry</h1> <h2>Your Account Id is
:"+str(hi[0])+ "</h2><h2>Please kindly login with this Id</h2> <h2>With
Regards:</h2><h3>Customer Care Registry</h3> </body></html>"

sender = {"name": "IBM CUSTOMER CARE REGISTRY",
"email": "ibmdemo6@yahoo.com"}
to = [{"email": email, "name": name}]
reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
headers = {"Some-Custom-Name": "unique-id-1234"}
params = {"parameter": "My param value",
"subject": "Email Verification"}
send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
to=to, reply_to=reply_to, headers=headers,
html_content=html_content, params=params, sender=sender, subject=subject)
api_response = api_instance.send_transac_email(send_smtp_email)
print(api_response)
except:

```

```

        flash("Error in Insertion Operation", "danger")
    finally:
        return redirect(url_for("signuppage"))
        con.close()

    return render_template('signuppage.html')
@app.route('/agentwelcome', methods=['POST', 'GET'])
def agentwelcome():
    # try:
        id = loginagent
        sql = "SELECT NAME FROM AGENT WHERE ID =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        hi = ibm_db.fetch_tuple(stmt)
        while hi != False:
            type = hi[0]
            name = type
            Break
        str = name+id
        sql = "SELECT
ISSUE.ID,ISSUE.DATE,ISSUE.TOPIC,ISSUE.SERVICE_TYPE,ISSUE.SER
VICE_AGENT,ISSUE.DESCRPTION,ISSUE.STATUS,ISSUE.ADDRESS,I
SSUE.CUSTOMER_ID,CUSTOMER.NAME,CUSTOMER.PHONENUMBE
R FROM ISSUE FULL OUTER JOIN CUSTOMER ON CUSTOMER.ID =
ISSUE.CUSTOMER_ID WHERE ISSUE.SERVICE_AGENT = ?"
        agent = []

```

```

    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, str)
    ibm_db.execute(stmt)
    otpf = ibm_db.fetch_both(stmt)
    while otpf != False:
        agent.append(otpf)
        otpf = ibm_db.fetch_both(stmt)

    sql = "SELECT COUNT(*) FROM ISSUE WHERE SERVICE_AGENT
= ?"
    stmt5 = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt5, 1, str)
    ibm_db.execute(stmt5)
    t = ibm_db.fetch_both(stmt5)

    return render_template("agentwelcome.html", agent=agent, message=t[0])
# except:
#     flash("No record found", "danger")
#     return render_template("agentwelcome.html")

@app.route('/viewagent/<ID>', methods=['GET', 'POST'])
def viewagent(ID):
    try:
        id = int(ID)
        global customerid
        customerid = id

```

```

idn = str(id)
global services
sql = "SELECT SERVICE_TYPE FROM ISSUE WHERE ID ="
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, id)
ibm_db.execute(stmt)
hi = ibm_db.fetch_tuple(stmt)
while hi != False:
    type = hi[0]
    services = type
    Break

sql = "SELECT NAME,ID FROM AGENT WHERE SERVICE_AGENT
="
agent = []
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, services)
ibm_db.execute(stmt)
otpf = ibm_db.fetch_both(stmt)
while otpf != False:
    agent.append(otpf)
    otpf = ibm_db.fetch_both(stmt)

flash("Successful","success")
return render_template("agentapply.html",agent=agent,id=idn)
except:

```

```

        flash("No record found","danger")
        return render_template('agentapply.html')
@app.route('/updatethis/<ID>', methods=['GET', 'POST'])
def updatethis(ID):

    agentid = ID
    print(customerid)
    print(agentid)
    status = "Agent Alloted"

    sql = "SELECT NAME,EMAIL FROM AGENT WHERE ID =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, agentid)
    ibm_db.execute(stmt)
    hi = ibm_db.fetch_tuple(stmt)
    while hi != False:
        msg = hi[0]
        email = hi[1]
        str1 = msg
        emailid = email

        Break
    mail = emailid
    print(mail)
    final = str1 + agentid
    sql = "UPDATE ISSUE SET SERVICE_AGENT = ?,STATUS = ?

```

WHERE ID = ? AND SERVICE_TYPE = ?"

```
stmt = ibm_db.prepare(conn, sql)
```

```
ibm_db.bind_param(stmt, 1, final)
```

```
ibm_db.bind_param(stmt,2,status)
```

```
ibm_db.bind_param(stmt,3,customerid)
```

```
ibm_db.bind_param(stmt,4,services)
```

```
ibm_db.execute(stmt)
```

```
flash("Successful","success")
```

```
configuration = sib_api_v3_sdk.Configuration()
```

```
configuration.api_key['api-key'] = "
```

```
api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
```

```
sib_api_v3_sdk.ApiClient(configuration))
```

```
subject = "Agent Alloted for you Account"
```

```
html_content = " <html><body><h1>Agent has be alloted for your  
Ticket</h1> <h2>Your Agent Id is :"+str(agentid)+"</h2> <div><h2>Your  
servicetype is:</h2>"+services+"<h3>Your Token id  
:"+str(customerid)+"</h3><h2>With Regards:</h2><h3>Customer Care  
Registry</h3> </body></html>"
```

```
sender = {"name": "IBM CUSTOMER CARE REGISTRY",
```

```
    "email": "ibmdemo6@yahoo.com"}
```

```
to = [{"email": mail, "name": "Agent"}]
```

```
reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
```

```
headers = {"Some-Custom-Name": "unique-id-1234"}
```

```
params = {"parameter": "My param value",
```

```
    "subject": "Email Verification"}
```

```
send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
```

```

        to=to, reply_to=reply_to, headers=headers, html_content=html_content,
        params=params, sender=sender, subject=subject)

        api_response = api_instance.send_transac_email(send_smtp_email)

        print(api_response)

        return redirect(url_for('admin'))

```

```

@app.route('/completed/<DESCRIPTION>', methods=['GET', 'POST'])
def completed(DESCRIPTION):
    status = "Completed"

    try:
        sql = "UPDATE ISSUE SET STATUS = ? WHERE DESCRIPTION =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,status)
        ibm_db.bind_param(stmt,2,DESCRIPTION)
        ibm_db.execute(stmt)
        flash("Successful","success")
        return redirect(url_for('agentwelcome'))
    except:
        flash("No record found","danger")
        return redirect(url_for('agentwelcome'))

if __name__ == '__main__':

```

GITHUB: <https://github.com/IBM-EPBL/IBM-Project-33480-1660221622>

DEMOLINK: <https://www.youtube.com/watch?v=rTZA7EzxIfk>