INDEX

1. INTRODUCTION:

1.1 PROJECT OVERVIEW:

Developing a cloud application not only for solving customer complaints but also gives the satisfaction to customer to use the respective business products. This application helps a customer to raise the complaints for the issue they are facing in the respective products. The customers should give the detailed description and the priority level of the issues that they are facing. After the complaint reviewed by the admin, then the agents are assigned to the complaints raised by the customer. The respective customer of the complaints gets the email notifications of the process. And additionally, they can able to see the status of the complaints that they had raised.

Customer is that the centre of attention of each business. The terrible existence of business depends on client satisfaction. Client expects high-quality services, even willing to pay a premium for higher service. From a client perspective, smart service quality ends up in semipermanent client relationships measured by re-patronage and cross sales, additionally client advocates the service to others.

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.

1.2 PURPOSE:

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. This application helps a customer to raise the complaints for the issue they are facing in the respective products. The customers should give the detailed description and the priority level of the issues that they are facing. After the complaint reviewed by the admin, then the agents are assigned to the complaints raised by the customer.

2. LITERATURE SUREVEY:

Survey 1:

Stone. M. (1992):

ACHIEVING ELEVATED LEVELS OF CUSTOMER CARE:

Achieving elevated levels of customer care has noticeable benefit for the consumers. They get better service and often better products. Caring for customers is an effortless idea. It means looking after customers and meeting there needs and expectations of the customer. Globally most businesses want to meet the need of there customers. Customer service is now an accepted part of the businesses vocabulary therefore many organizations now have the customer care and service teams which are much more than a complaints department. However for some businesses consumers well being is a clear objective, this is true of many none for profit organizations and public sector bodies. On the other hand customer well being may be a very influential. If implied. objectives of the business are to meet its formal organization, such as profit. professional satisfaction or election to power. Customer Care is not just about handling Complaint, it's about ensuring that customers do not need to complain".

Survey 2:

LaLonde & Zinser (1976):

SIMPLIFING THE SALE AND USAGE USING FTS MODEL:

Services are the kind of activities between the organization and customers to improve or simplify sale and using of products. They involve also operations of producers provided for customers during the whole transaction. He improved this model by replacing complicated max-min composition operations with simplified arithmetic operations. A Heuristic Gaussian cloud transformation was integrated with an FTS model to forecast water quality. Services present important activities from the purchase order of customer to delivery of products. The activities are customer-oriented and depend on the kind of product and type of customer.

Survey 3:

Lehtinen, J.R (2007):

LONG-LASTING VIEW OF A PRODUCTS OR SERVICES:

Services are the system organized to assure continuity between the time of purchase order delivery of goods. The aim is to satisfy customer needs from the long-lasting view. Services are all activities connected with assuring relationship with customer – from product delivery to different ways of help by its using. Perception of customer services shows the differences

changes during years and confirms current tendency to be in a very deep touch with customers. It means to have a long-term relationship with clientele as well. Focus on customers is also one of main parts of market orientation. Customers are usually evaluated as the most important stakeholders. Therefore, companies generally pay high attention to them.

Survey 4:

Allmendinger, G., & Lombreglia, R. (2005):

FOCUSING THE CUSTOMERS THROUGH SMART SERVICES:

Smart services are strictly based on field intelligence. The field intelligence refers to the concept that connected systems and devices pave the way to intelligence that is higher than the intelligence of the individual parts. It is enabled by context information and high dynamics. Support from technology such as information and communications technology, as well as the ability to react to an individual's context and its changes make up smart service Intelligent sensors (i.e., sensors that not only collect data, but also prepare and preprocess them) are often used to determine the current contexts. Individual customer needs are not mentioned as precondition because they must be considered to be able to offer individual smart services. Additionally, customer needs often are the result of data analyses what forms part of the definition.

Survey 5:

GILLIG AND SAILER (2012):

VALUE CO-CREATED VIA INTERACTIONS IN ALL PHASES:

Although a characteristic of smart services is that value is co-created via interactions between the service provider and the customer, the role of the customer in the literature has not been as well explored as would be expected. Research has addressed the question of how to involve the customer in the innovation process but customer involvement in the operation and improvement phases is relatively unexamined. While exploratory case studies have already indicated the importance of the customer, general conclusions across different applications and industries are still missing. A systematic overview of the customer's role across all life-cycle phases of a smart service would help those engaged in the practice to improve their processes. A theoretical framework presenting the role of the customer from a more general perspective would contribute to academic knowledge. Another aspect regarding the customer's role would be to measure and predict their behaviour. Investigating in detail how usage behaviour influences smart services in all phases of the life cycle would provide a better understanding of smart services.

Survey 6:

MASSINK ET AL. (2010):

INVOLMENT OF ENVIRONMENT IN OPERATIONAL SMART

SERVICES:

The interaction between customer and provider is necessary, in addition to the service offered by the technology itself. Through collaboration the service provider knows the current needs and thus can adapt the smart service constantly. It is suggested that value co-creation does not require direct input from a customer because functionalities should be provided in a convenient way. Nevertheless, the present indicates that the customer and the environment are involved and form an important part in all phases from a strategic development to the improvement of operational smart services. This interaction can be direct,
e.g., in form of feedback, or indirect, e.g., by providing accurate information.

2.1 EXISTING SOLUTION:

- Already existing solutions uses the Support Vector Machine, Random Forest algorithms which is available on online platforms
- These algorithms are highly complicated and time consuming for processing and classifying images.
- Several key parameters should be correctly set to achieve best classification result.
- So, the existing solutions need more computing power, time and more information to provide accurate results.

2.2 REFERENCES:

1. Simon Haykin, “Bird classification using CNN: a comprehensive foundation,” Prentice Hall PTR, 1994.

2. Paul Viola, Michael Jones, “Classification and Grading of Image Using Texture Based Block-Wise Local Binary Patterns” CVPR (1) 1 (2001), 511–518, 2001.

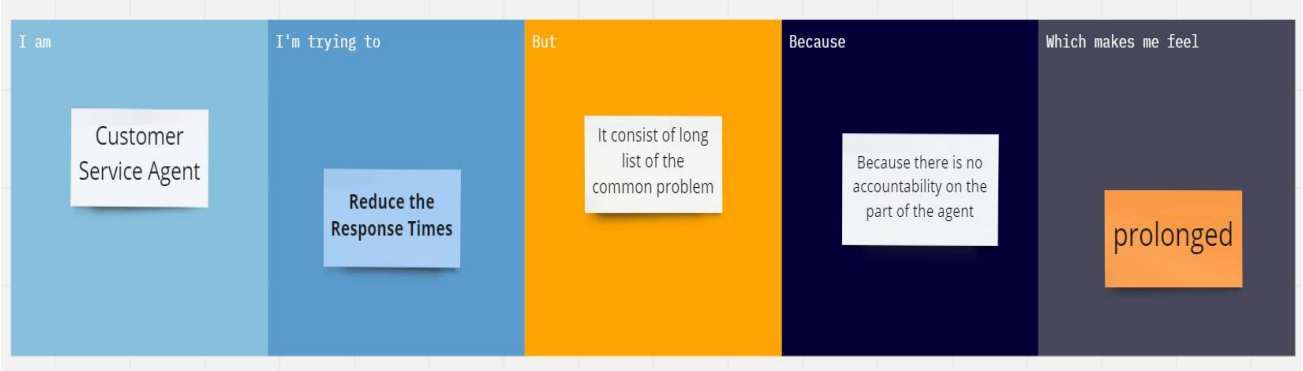
3. Gary Bradski and Adrian Kaehler. “Texture Classification from Random Features”, 2008.

4. Schmid Huber J, “Adapted approach for Species Classification: An Overview Neural Networks” 61: 85-117, 2015.

5. Haibing Wu and Xiaodong Gu, “Detection and Classification of images using Detection Line” 71,1–10, 2015.

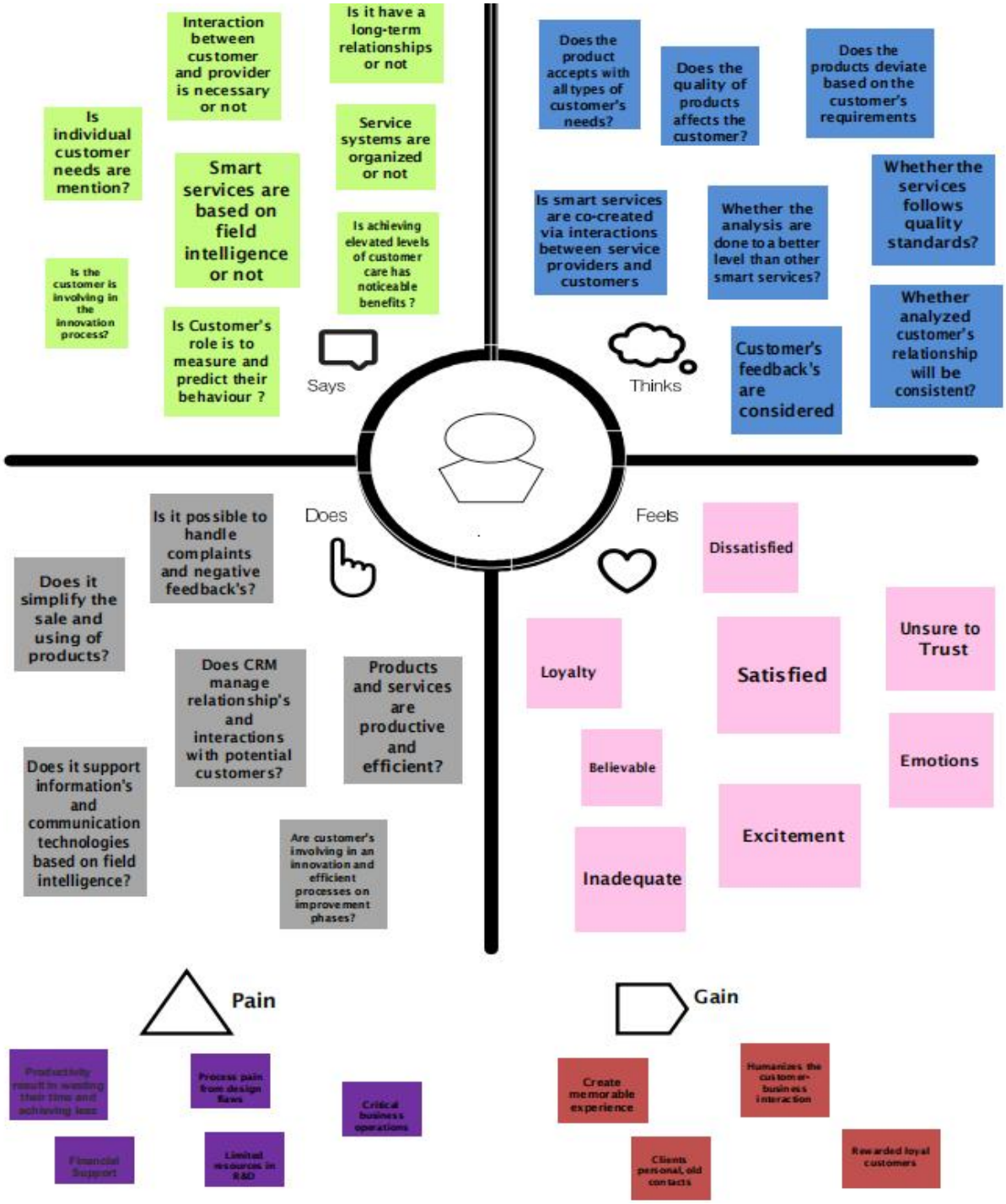
PROBLEM STATEMENT & DEFINITIO:

Problem Statement:

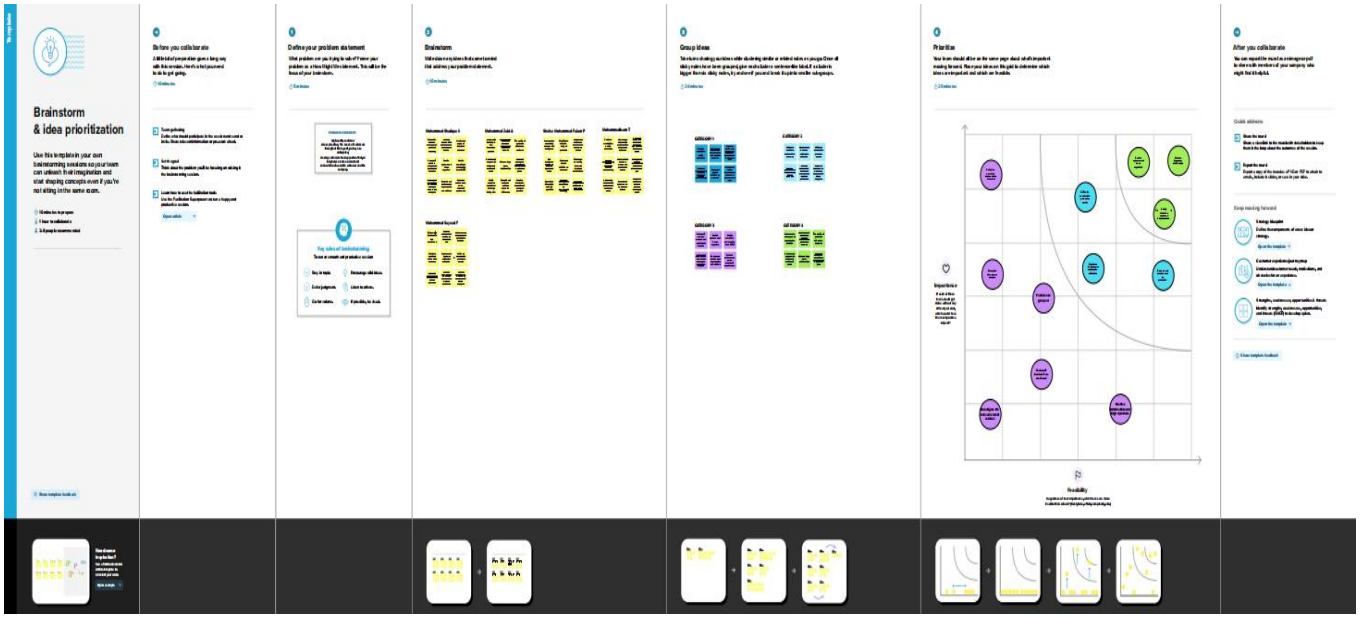


3. IDEATION & PROPOSED SOLUTION:

3.1 EMPATHY MAP:


Interaction between customer and provider is necessary or not
Is it have a long-term relationships or not
Is individual customer needs are mention?
Service systems are organized or not
Smart services are based on field intelligence or not
Is achieving elevated levels of customer care has noticeable benefits ?
Is the customer is involving in the innovation process?
Is Customer's role is to measure and predict their behaviour ?
Says
Does the product accepts with all types of customer's needs?
Does the quality of products affects the customer?
Does the products deviate based on the customer's requirements
Whether the services follows quality standards?
Is smart services are co-created via interactions between service providers and customers
Whether the analysis are done to a better level than other smart services?
Whether analyzed customer's relationship will be consistent?
Customer's feedback's are considered
Thinks
Does
Is it possible to handle complaints and negative feedback's?
Does it simplify the sale and using of products?
Does CRM manage relationship's and interactions with potential customers?
Products and services are productive and efficient?
Does it support information's and communication technologies based on field intelligence?
Are customer's involving in an innovation and efficient processes on improvement phases?
Feels
Dissatisfied
Loyalty
Satisfied
Unsure to Trust
Believable
Emotions
Excitement
Inadequate
Pain
Process pain from design flaws
Critical business operations
Financial Support
Limited resources in R&D
Gain
Create memorable experience
Humanizes the customer-business interaction
Clients personal, old contacts
Rewarded loyal customers

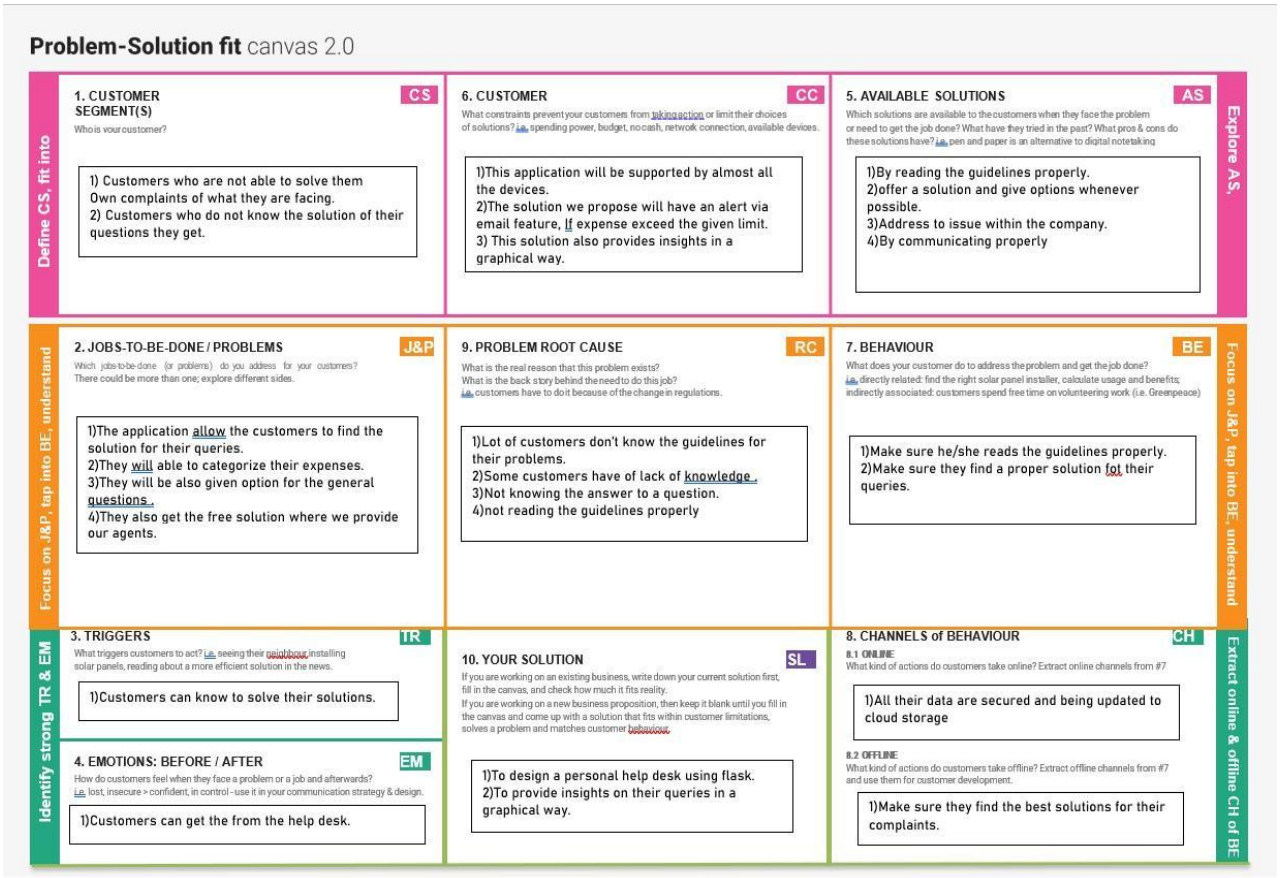
3.2 IDEATION & BRAINSTROMMING:



3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To solve customer issues using Cloud Application Development.
2.	Idea / 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.	Novelty / Uniqueness	Assigned Agent Routing, Automated Ticket Closure, Status Shown to the Customer, and Backup data in case of failures.
4.	Social Impact / Customer Satisfaction	Customer Satisfaction, Customer can track their status and Easy agent communication.
5.	Business Model (Revenue Model)	◆ Key Partners are Third-party applications, agents, andcustomers. ◆ Activities held as Customer Service, System Maintenance. ◆ Key Resources support Engineers, Multi-channel. ◆ Customer Relationship have 24/7 Email Support, Knowledge-based channel.
6.	Scalability of the Solution	The real goal of scaling customer service is providing an environment that will allow your customer service specialists to be as efficient as possible. An environment where they will be able to spend less time on grant work and more time on actually resolving critical customer issues

3.4 PROBLEM SOLUTION FIT:



4. REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENTS:

FR No	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
1	User Registration	Registration through Form Registration through Gmail Registration through Google
2	User Confirmation	Confirmation viaEmail Confirmation via OTP
3	User Login	Login via Google Login with Email id and Password
4	Admin Login	Login via Google Login with Email id and Password
5	Query Form	Description of the issues Contact information
6	E-mail	Login alertness
7	Feedback	Customer feedback

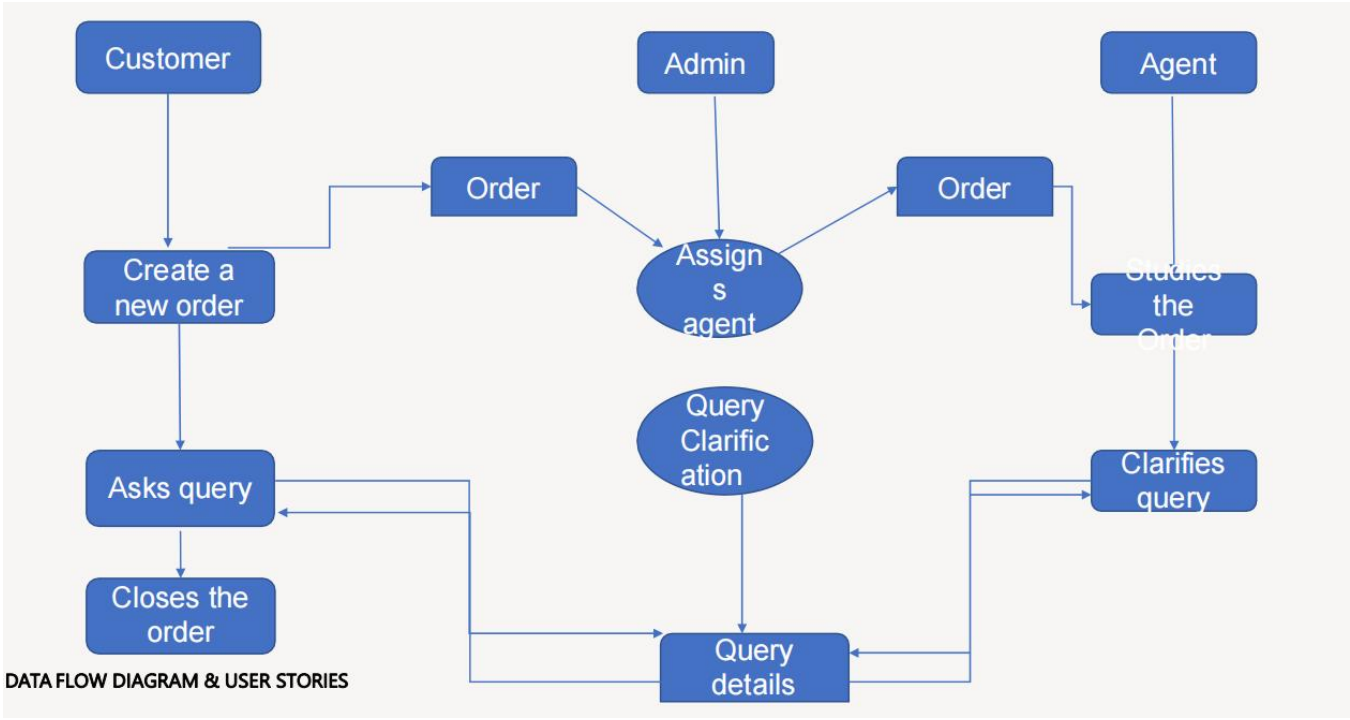
3.2 FUNCTIONAL REQUIREMENTS:

FR No	Non-Functional Requirement	Description
1	Usability	To provide the solution to the problem
2	Security	Track of login authentication
3	Reliability	Tracking of decade status through email
4	Performance	Effective development of web application

5. PROJECT DESIGN:

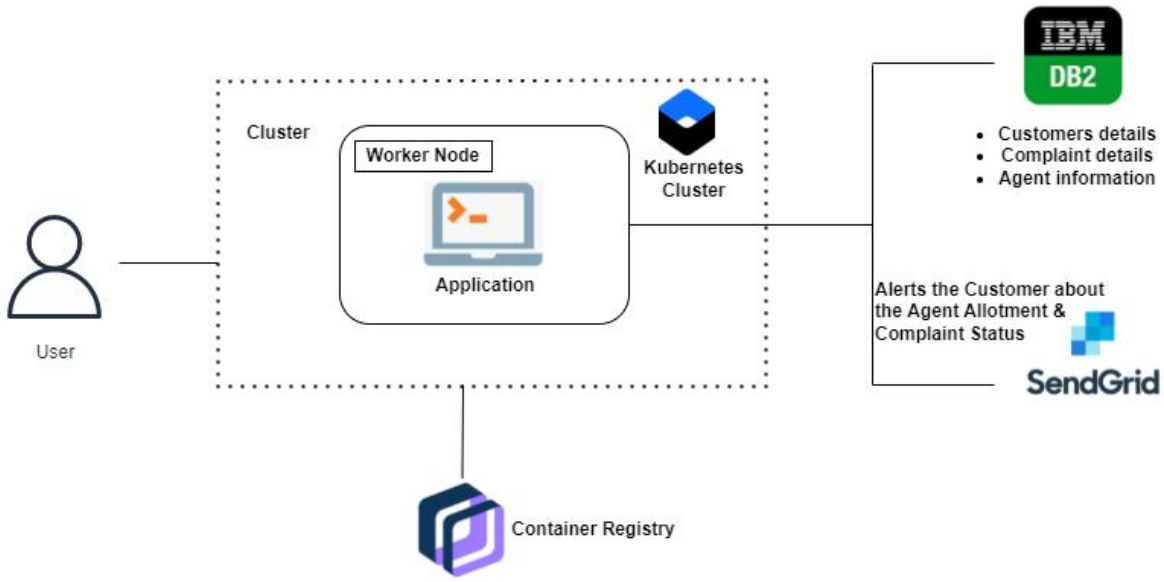
5.1 DATA FLOW DIAGRAMS:

DFD LEVEL 0 DIAGRAM:

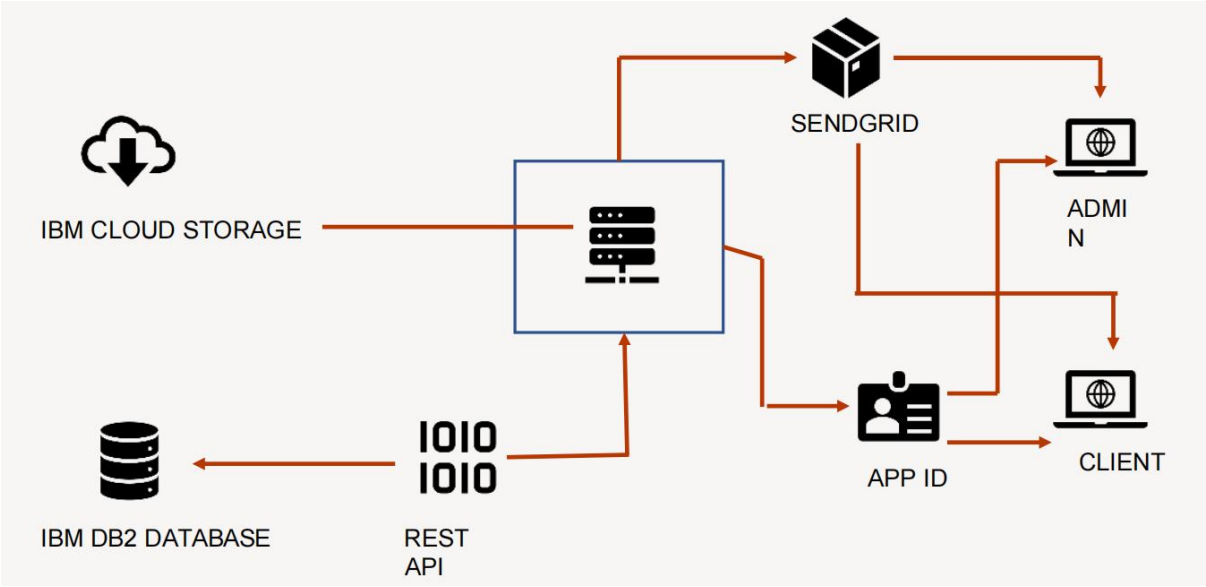


6.SOLUTION AND TECHNICAL ARCHITECTURE:

6.1 SOLUTION ARCHITECTURE:



6.2 TECHNOLOGY STACK:



COMPONENTS & TECHNOLOGIES:

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

APPLICATION CHARACTERISTICS:

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

6.PROJECT PLANNING & SCHEDULING:

6.1 SPRINT PLANNING AND ESTIMATION:

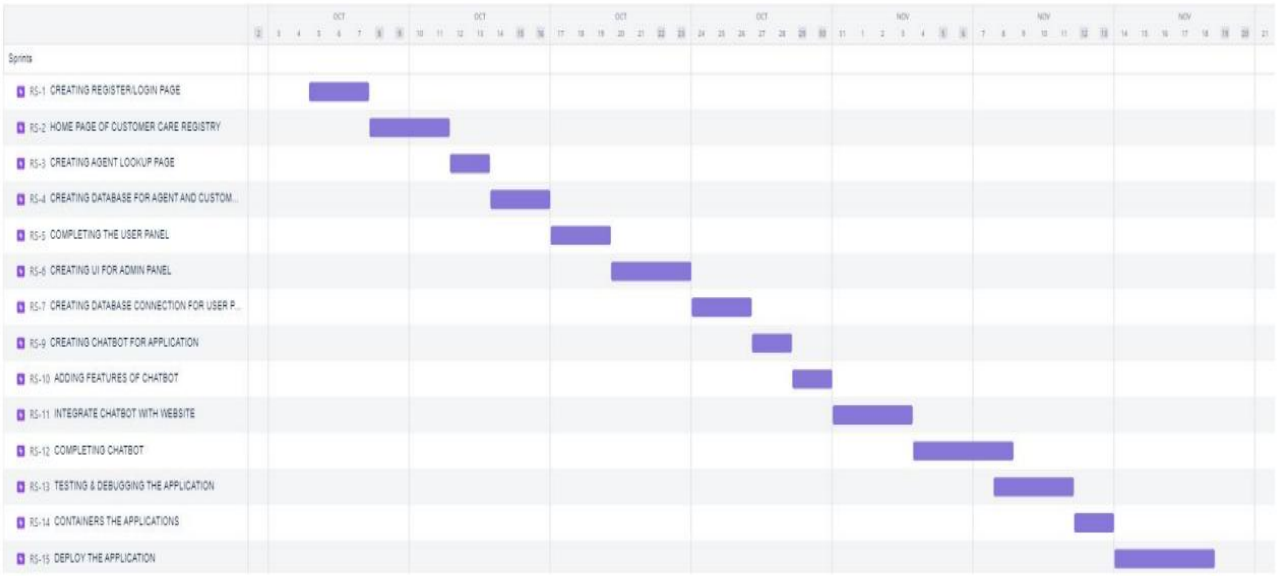
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		19 Nov 2022

6.2 SPRINT DELIVERY 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	20	High	MOHAMMED SHARIQUE MOHAMMED SAYEED
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the availability and have a track of all the things that the users are going to service	20	High	MOHAMMED ZAID SHAFEE MD FAIZAN MOHAMMED SAYEED
Sprint-3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding the services. Get the recommendations based on information provided by the user.	20	High	MOHAMMED ISAM MOHAMMED ZAID
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes and deployment the application.Create the documentation and final submit the application	20	High	MOHAMMED SHARIQUE MOHAMMED ISAM SHAFEE MD FAIZAN

6.3 REPORTS FROM JIRA:

BURNDOWN CHART:



6. CODING AND SOLUTIONING

Index.Html

```
<!DOCTYPE html>
<html>
<head>
    <title>Customer Care Registry</title>
    <!-- <link rel = "stylesheet" type = "text/css" href = "https://website-static.s3.jp-tok.cloud-
object-storage.appdomain.cloud/css/style.css"> -->
    <link rel = "stylesheet" type = "text/css" href = "/style.css">
    <link rel = "preconnect" href = "https://fonts.gstatic.com">
    <link rel="icon" type="image/x-icon" href="favicon.ico">

    <link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
<link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png">
<link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png">
<link rel="manifest" href="/site.webmanifest">

  <link href =
"https://fonts.googleapis.com/css2?family=Josefin+Sans:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,
700;1,100;1,200;1,300;1,400;1,500;1,600;1,700&display=swap" rel="stylesheet">
```

```
  <link rel="stylesheet" href = "https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css" integrity = "sha512-
iBBXm8fW90+nuLcSKlbmrPcLa0OT92xO1BIsZ+ywDWZCvqsWgccV3gFoRBv0z+8dLJgyAHIhR35VZc2oM/gI1w=="
crossorigin="anonymous" referrerpolicy="no-referrer" />

  <script>
    window.watsonAssistantChatOptions = {
      integrationID: "66576f0c-5408-4edc-803b-d9de1f553e8b",  // The ID of this integration.
      region: "eu-gb",  // The region your integration is hosted in.
      serviceInstanceID: "2607efc7-375b-465c-9e61-399a0f694519",  // The ID of your service instance.
      onLoad: function(instance) { instance.render(); }
    };
    setTimeout(function() {
      const t = document.createElement('script');
      t.src = "https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
      document.head.appendChild(t);
    });
  </script>
</head>
</head>
    <div class="hero">
        <nav>
            <h2 class="logo">PIT<span>TU</span></h2>
            <a class="btn" href = "">OUR TOP AGENTS</a>
            <a href="#MyRes" class="btn">RESUME</a>
            <a href="#AboutMe" class="btn">ABOUT</a>
            <a href="#ContactMe" class="btn">CONTACT</a>
            <div class="row text-center mx-auto mb-3 ">
                <a href="{{ url_for('signinpage') }}" class="btn btn-primary btn-block "
role="button"><span
                        style="color:black; font-weight: bolder; height: 100px;">Sign in</span> </a>
            </div>
            <div class="row text-center mx-auto mb-5 ">
                <a href="{{ url_for('signuppage') }}" class="btn btn-primary btn-block"
role="button"><span
                        style="color:black; font-weight: bolder; height: 100px;">Sign up</span></a>
            </div>
        </nav>

        <h1 class = "CCR"> <span class="CCR_fl">C</span>USTOMER <span class="CCR_fl">C</span>ARE <span
class="CCR_fl">R</span>EGISTRY </h1>

    </div>
    <!----About section start---->
    <section class="about">
        <h1 id="AboutMe"></h1>
        <div class="main">
            <img src="https://website-static.s3.jp-tok.cloud-object-
storage.appdomain.cloud/img/BG.jpeg" >

            <div class="about-text">
                <h2>About Me</h2>
                <h5>Developer <span>& Designer</span></h5>
                <p>Undergraduate!! I'm a Computer Science Student, <br> Aspiring for Software
Development Engineer. </p>
                <p>Programmer, Problem Solver, Web developer and Android Application developer are the
fields that I have
                    experienced with. Developed and designed a static websites and android
application.</p>
```

```
                <p class="mb-5">Experienced with the real-world Hands-on projects and have virtual
experiences with the Software Development
                    Engineering in JP Morgan & Chase co. and a Software Developer Internship program in
Accenture. </p>
            </div>
        </div>
    </section>
    <!-----service section start---------->
    <div class="service">
        <div class="title">
            <h2>Our Services</h2>
        </div>
        <div class="box">
            <div class="card">
                <i class="fas fa-bars"></i>
                <h5>Web Development</h5>
                <div class="pra">
                    <p>Web development refers to the building, creating, and maintaining of websites.
It includes aspects such as web design, web publishing, web programming, and database managements. It
is the creation of an applications that works over the internet i.e. websites.</p>
                    <p style="text-align: center;">
                        <a class="button" href="#">Read More</a>
                    </p>
                </div>
            </div>
            <div class="card">
                <i class="far fa-user"></i>
                <h5>App Development</h5>
                <div class="pra">
                    <p>App development is an act or process by which a mobile app is developed for
mobile devices, such as personal digital assistants, enterprise digital assistants or mobile phones.
These software applications are designed to run on devices, such as a smartphone or tablet
computer.</p>
                    <p style="text-align: center;">
                        <a class="button" href="#">Read More</a>
                    </p>
                </div>
            </div>
            <div class="card">
                <i class="far fa-bell"></i>
                <h5>Web & App Management</h5>
                <div class="pra">
                    <p>Application management provides a wide variety of application services,
processes and methodologies for maintaining, enhancing and managing custom applications, packaged
software applications or network-delivered applications and to remain up-to-date with the latest
softwares.</p>
                    <p style="text-align: center;">
                        <a class="button" href="#">Read More</a>
                    </p>
                </div>
            </div>
        </div>
    </div>
```

```
    <!------Contact Me------>
    <div class="contact-me">
        <p>Let Me Get You A Beautiful Website.</p>
        <a class="button-two" href="#">Hire Me</a>
    </div>
```

```
    <section class="site-section " id="section-resume">
        <div class="container">
            <div class="row">
                <div class="col-md-12 mb-5">
                    <div class="section-heading text-center">
                        <h1 id="MyRes"></h1>
                        <h2>My <strong>Resume</strong></h2>
                    </div>
                </div>
                <div class="col-md-6">
                    <h2 class="mb-5">Educations</h2>
                    <div class="resume-item mb-4">
                        <span class="date"><span class="icon-calendar"></span> March 2019 -
Present</span>
                        <h3>B. E in Computer Science Engineering (CSE)</h3>
                        <p>
                            C Abdul Hakeem College of Engineering & Technology ANNA UNIVERSITY.<br>
                            CGPA: 8.62
                        </p>
                        <span class="school">Ranipet</span>
                    </div>
                    <div class="resume-item mb-4">
                        <span class="date"><span class="icon-calendar"></span> March 2018 - March
2019</span>
                        <h3>High School (HSC STATE Board)</h3>
                        <p>
                            KH MATRICULATION SCHOOL<br>
                            Percentage 70 %</p>
                        <span class="school">Ranipet</span>
                    </div>
```

```
                    <div class="resume-item mb-4">
                        <span class="date"><span class="icon-calendar"></span> March 2016 - March
2017</span>
                        <h3>High School (SSLC STATE Board)</h3>
                        <p> KH MATRICULATION SCHOOL<br>
                            Percentage 84.4 %</p>
                        </p>
                        <span class="school">Ranipet</span>
                    </div>
                </div>
                <div class="col-md-6">
                    <h2 class="mb-5">Experience</h2>
                    <div class="resume-item mb-4">
                        <span class="date"><span class="icon-calendar"></span> October 2022 -
Present</span>
                        <h3>Software Development Intern - Shiash Info Solution</h3>
                        <p>Developed and assisted with the project that is Developed using technologies
such as Java and Databases such as MySQL.</p>
                        <span class="school">Developed a Web based application that provides a services
which meets the client requirements.</span>
                    </div>
                </div>
                <div class="col-md-6">
                    <h2 class="mb-5">Skills</h2>
                    <div class="resume-item mb-4">
                        <!-- <span class="date"><span class="icon-calendar"></span> October 2022 -
Present</span> -->
                        <h3>Technical Skills - </h3>
                        Java   |    Python   |   MySQL    |   JavaScript
  |   React.js   |   HTML5 & CSS3  
```

```
                    <br><br>
                    <h3>Development Skills - </h3>
                    Website Development    |     Android Application Development   |
  Web based Application
                </div>
                <br>
                <a class="ResDoc"
href="https://drive.google.com/file/d/1P1Q4w62t1Xd4pv4OlVZOyk8z4ki9unbY/view?usp=share_link"> ⇲
Preview My Resume Here ↩  </a>
            </div>
        </div>
    </div>
</section> <!-- .section -->
<section class="site-section" id="section-contact">
    <div class="container">
        <div class="row">
            <div class="col-md-12 mb-5">
                <div class="section-heading text-center">
                    <h1 id="ContactMe"></h1>
                    <h2>Get <strong>In Touch</strong></h2>
                </div>
            </div>
            <div class="col-md-7 mb-5 mb-md-0">
                <form action="submit_form" class="site-form" method="post">
                    <h3 class="mb-5">Get In Touch</h3>
                    <div class="form-group">
                        <input name="name" type="text" class="form-control px-3 py-4"
placeholder="Your Name">
                    </div>
                    <div class="form-group">
                        <input name="email" type="email" class="form-control px-3 py-4"
placeholder="Your Email">
                    </div>
                    <div class="form-group mb-5">
                        <textarea name="message" class="form-control px-3 py-4" cols="80" rows="15"
                            placeholder="Write a Message"></textarea>
                    </div>
                    <div class="form-group">
                        <input type="submit" class="btn btn-primary  px-4 py-3" value="Send
Message">
                    </div>
                </form>
                <script>
                    function test() {
                        Swal.fire('Thanks For Contacting Us..')
                        window.scrollTo(0, document.body.scrollHeight);
                    }
                    { { code } }
                </script>
            </div>
            <div class="col-md-5 pl-md-5">
                <h3 class="mb-5">My Contact Details</h3>
                <ul class="site-contact-details">
                    <li>
                        <span class="text-uppercase">Email</span>
                        mdshariquek16@gmail.com
                    </li>
                    <li>
                        <span class="text-uppercase">Phone</span>
                        +91 90805 34237
                    </li>
```

```
                    <li>
                        <span class="text-uppercase">Address</span>
                        Usmanpet Street<br>
                        Melvisharam, Ranipet <br>
                        Tamil Nadu, India
                    </li>
                </ul>
            </div>
        </div>
    </div>
</section>
```

```
<!------footer start--------->
<footer>
    <p>MD SHARIQUE</p>
    <p>For more details! Visit My :</p>
    <div class="social">
        <a href="https://www.linkedin.com/in/md-shariquek16/"><i class="fab fa-linkedin"></i></a>
        <a href="https://github.com/Mdsharu"><i class="fab fa-github"></i></a>
        <a href="https://www.facebook.com/"><i class="fab fa-facebook-f"></i></a>
        <a href="https://www.instagram.com"><i class="fab fa-instagram"></i></a>
        <a href="#"><i class="fab fa-dribbble"></i></a>
    </div>
</footer>
</body>
</html>
```

app.py // used to functioning of the application //

```
from __future__ import print_function

import datetime
from audioop import add
from pprint import pprint
from unicodedata import name
import ibm_db
import sib_api_v3_sdk
from flask import *
from flask import (Flask, flash, redirect, render_template, request, session,
                   url_for)
from markupsafe import escape
from sib_api_v3_sdk.rest import ApiException
from init import hello, id, randomnumber
conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME="";PORT="";SECURITY=SSL;SSLServerCertificate="";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('/signinpage', methods=['POST', 'GET'])
def signinpage():
    return render_template('signinpage.html')

@app.route('/agentsignin', methods=['POST', 'GET'])
def agentsignin():
    return render_template('signinpageagent.html')

@app.route('/signuppage', methods=['POST', 'GET'])
def signuppage():
    return render_template('signuppage.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('/newissue/<name>', methods=['POST', 'GET'])
def newissue(name):
    name = name
    return render_template('complaint.html',msg=name)

@app.route('/forgot', methods=['POST', 'GET'])
def forgot():
    try:
        global randomnumber
        ida = request.form['custid']
        print(ida)
        global id
        id = ida
        sql = "SELECT EMAIL,NAME FROM Customer 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
        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 = "Verification for Password"
        html_content = "<html><body><h1>Your verification Code is : <h2>" + \
            str(randomnumber)+"</h2> </h1> </body></html>"
        sender = {"name": "IBM CUSTOMER CARE REGISTRY",
                  "email": "ibmdemo6@yahoo.com"}
        to = [{"email": e, "name": n}]
        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)
        pprint(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('/verifyemail', methods=['POST', 'GET'])
def verifyemail():
    try:
        email = request.form['verifyemail']
        sql = "SELECT ID,NAME FROM Customer WHERE email=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        emailf = ibm_db.fetch_both(stmt)
        while emailf != False:
            id = emailf[0]
            name = emailf[1]
            break
        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 = "Regarding of your Customer Id"
        html_content = "<html><body><h1>Your Customer Id  is : <h2>" + \
            str(id)+"</h2> </h1> </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)
        pprint(api_response)
        message = "Email send to:"+email+" 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("Database not found in mail! Please Register Your account.", "danger")
    finally:
        return render_template('signinpage.html')

@app.route('/otp', methods=['POST', 'GET'])
def otp():
```

```
    try:
        otp = request.form['otp']
        cusid = id
        print(id)
        sql = "SELECT PASSWORD FROM Customer 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('/admin', methods=['POST', 'GET'])
def admin():
    userdatabase = []
    sql = "SELECT * FROM customer"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        userdatabase.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if userdatabase:
        sql = "SELECT COUNT(*) FROM customer;"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    users = []
    sql = "select * from ISSUE"
    stmt = ibm_db.exec_immediate(conn, sql)
    dict = ibm_db.fetch_both(stmt)
    while dict != False:
        users.append(dict)
        dict = ibm_db.fetch_both(stmt)
    if users:
        sql = "SELECT COUNT(*) FROM ISSUE;"
        stmt = ibm_db.exec_immediate(conn, sql)
        count = ibm_db.fetch_both(stmt)
    agent = []
    sql = "SELECT * FROM AGENT"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        agent.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if agent:
        sql = "SELECT COUNT(*) FROM AGENT;"
        stmt = ibm_db.exec_immediate(conn, sql)
        cot = ibm_db.fetch_both(stmt)
    return
render_template("admin.html",complaint=users,users=userdatabase,agents=agent,message=user[0],issue=cou
nt[0],msgagent = cot[0])
```

```python
@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(prep_stmt)
            flash("delected successfully the Customer", "success")
        except:
            flash("No data found in Customer", "danger")
        finally:
            return redirect(url_for('signuppage'))
    if otp == 'A':
        try:
            insert_sql = f"delete from AGENT"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prep_stmt)
            flash("delected 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(prep_stmt)
            flash("delected successfully the Complaints", "success")
        except:
            flash("No data found in Complaints", "danger")
        finally:
            return redirect(url_for('signuppage'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        try:
            id = request.form['idn']
            global hello
            hello = id
            password = request.form['password']
            print(id, password)
            if id == '1111' and password == '1111':
                return redirect(url_for('admin'))
            sql = f"select * from customer 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("welcome"))
            else:
                flash("Mismatch in credetials", "danger")
        except:
            flash("Error in Insertion operation", "danger")
    return render_template('signinpage.html')

@app.route('/welcome', methods=['POST', 'GET'])
def welcome():
```

```
    try:
        id = hello
        sql = "SELECT ID,DATE,TOPIC,SERVICE_TYPE,SERVICE_AGENT,DESCRIPTION,STATUS 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 credetials", "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"))
```

8. TESTING:

8.1 TEST CASES:

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	1.Enter URL and click go 2.Scroll down 3.Verify login/Singup popup displayed or not	http://169.51.204.215:30100/	Login/Signup popup should display	Working as expected	PASS	Successfull			SHARIQUE
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	1. Enter URL and click go 2.Click on Sigup button for User 3. Verify login/Singup popup with below UI elements: a.id text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	http://169.51.204.215:30100/	Application should show below UI elements: a.email text box b.password text box c. Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	PASS	Successful			SHAFEE FAIZAN MD ZAID
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1.Enter URL(https://bhopenzer.com/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter valid password in password text box 5.Click on login button	ID: 5106 password: Testing123	User should navigate to user account homepage	Working as expected	PASS	Successful			MD SAYEED MD ISAM
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL (http://169.51.204.215:30100/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter valid password in password text box 5.Click on login button	ID: 5106 password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			SHAFEE FAIZAN
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL (http://169.51.204.215:30100/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	ID: 5106 password: Testing1236	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			MD ISAM
LoginPage_TC_OO6	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL (http://169.51.204.215:30100/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	ID: 5342 password: Testing16	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			MD ZAID

LoginPage_TC_OO7	Functional	Login page	Verify User is able to log into application with Valid Credentials	1.Enter URL (http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	ID: 5106 password: Testing16	Application should show 'correct email or password' validation message.	Working as expected	PASS	Successful			MD SAYEED SHARIQUE
LoginPage_TC_OO8	Functional	Login page for ADMIN	Verify User is able to log into application with Valid Credentials	1.Enter URL (http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4.Enter valid password in password text box 5.Click on login button	ID: 1616 password: 5106	Application should show 'correct email or password' validation message.	Working as expected	PASS	Successful			MD ZAID
LoginPage_TC_OO9	UI	ADMIN PAGE	Verify all the Customer database is visible	1.Enter URL (http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	http://169.51.204.215:30106/	Customer database is visible	Working as expected	PASS	Successful			SHARIQUE
LoginPage_TC_OO7	UI	Home page for USER	Verify user is able to see the User home page when user finish on submitting Credentials	1.Enter URL (http://169.51.204.215:3 0106/) and click go 2. To the User Login page and submit Your Credentials n	http://169.51.204.215:30106/	USER Home Page popup should display .	Working as expected	PASS	Successful			MD SAYEED SHAFEE FAIZAN
LoginPage_TC_OO8	UI	Home page for ADMIN	Verify user is able to see the ADMIN home page when user finish on submitting Credentials	1.Enter URL (http://169.51.204.215:3 0106/) and click go 2. To the User Login page and submit Your Credentials 5.	http://169.51.204.215:30106/8	ADMIN Home Page popup should display .	Working as expected	PASS	Successful			MD ISAM
LoginPage_TC_OO9	Functional	AGENT PAGE	On delete Button the user Credentials will be delected	1.Enter URL (http://169.51.204.215:3 0106/) and click go 2. To the Admin Page and delect on User Credentials	http://169.51.204.215:30106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			SHARIQUE

8.1 USER ACCEPTANCE TESTING:

DEFECT ANALYSIS:

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	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
Totals	32	17	17	18	84

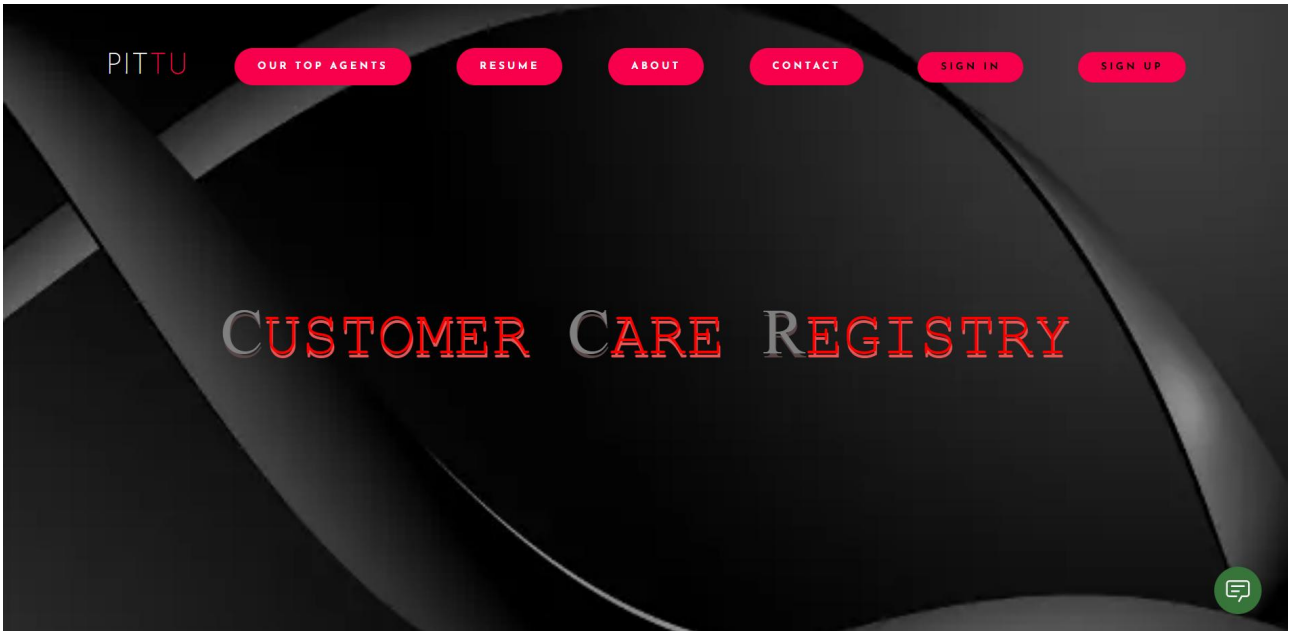
Test Case Analysis

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

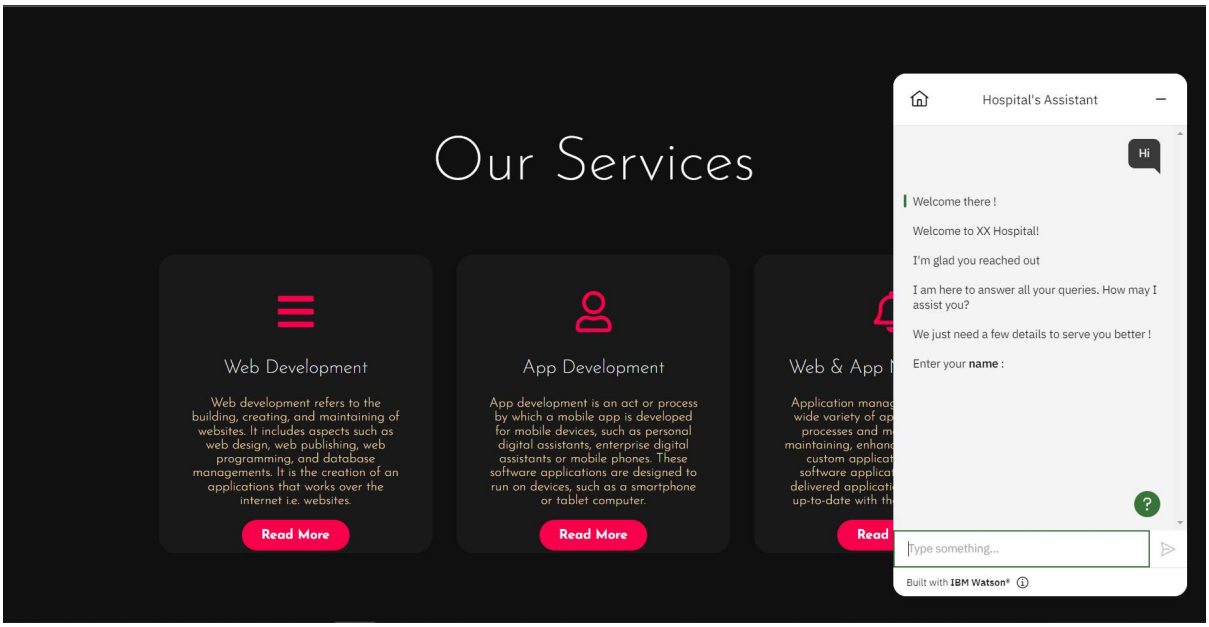
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	40	0	0	40
Security	5	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	10
Final Report Output	4	0	0	4
Version Control	4	0	0	4

9. RESULTS :

OUTPUT:



IBM WATSON - CHAT BOT:



9. ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

- Customers complaint are get solved effectively.
- Implementing of Chat Bot.
- It reduced the time period for customers to get solved their problems.
- Customers receiving an email regarding their complaint details.

DISADVANTAGES:

- Customers must be specific with their complaints to Chat Bot in order to get resolved.
- Difficult to implementing in all perspective.

11. CONCLUSION:

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

12. FUTURE SCOPE:

Pre-interaction

In the pre-interaction stage, support ticket analytics will soon enable:

- Understanding of particularly painful topics that need extra attention
- Flagging of topics that cause nasty customer outcomes (e.g.churn)
- Support ticket delegation of tricky tickets to experienced agents

Interaction

In the interaction stage, support ticket analytics will soon enable:

- Faster response times
- Auto-identified opportunities for up sell or growth
- Continuous training of agents and automatic quality assurance

Post-interaction

In the post-interaction stage, support ticket analytics will soon enable customer service teams to:

- Identify points of customer friction before they become pain points (i.e. root cause analytic)
- Get alerted to unexpected trending issues to make sure they don't affect more customers (i.e. issue prevention)
- Identify opportunities for product development

13. APPENDIX

13.1 SAMPLE CODE

```
<!DOCTYPE html>
<html>
<head>
    <title>Customer Care Registry</title>
    <!-- <link rel = "stylesheet" type = "text/css" href = "https://website-static.s3.jp-tok.cloud-
object-storage.appdomain.cloud/css/style.css"> -->
    <link rel = "stylesheet" type = "text/css" href = "/style.css">
    <link rel = "preconnect" href = "https://fonts.gstatic.com">
    <link rel="icon" type="image/x-icon" href="favicon.ico">

    <link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
<link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png">
<link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png">
<link rel="manifest" href="/site.webmanifest">

  <link href =
"https://fonts.googleapis.com/css2?family=Josefin+Sans:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,
700;1,100;1,200;1,300;1,400;1,500;1,600;1,700&display=swap" rel="stylesheet">
  <link rel="stylesheet" href = "https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css" integrity = "sha512-
```

```
iBBXm8fW90+nuLcSKlbmrPcLa0OT92xO1BIsZ+ywDWZCvqsWgccV3gFoRBv0z+8dLJgyAHIhR35VZc2oM/gI1w=="
crossorigin="anonymous" referrerpolicy="no-referrer" />

  <script>
    window.watsonAssistantChatOptions = {
      integrationID: "66576f0c-5408-4edc-803b-d9de1f553e8b",  // The ID of this integration.
      region: "eu-gb",  // The region your integration is hosted in.
      serviceInstanceID: "2607efc7-375b-465c-9e61-399a0f694519",  // The ID of your service instance.
      onLoad: function(instance) { instance.render(); }
    };
    setTimeout(function() {
      const t = document.createElement('script');
      t.src = "https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
      document.head.appendChild(t);
    });
  </script>
</head>
</head>
    <div class="hero">
        <nav>
            <h2 class="logo">PIT<span>TU</span></h2>
            <a class="btn" href = "">OUR TOP AGENTS</a>
            <a href="#MyRes" class="btn">RESUME</a>
            <a href="#AboutMe" class="btn">ABOUT</a>
            <a href="#ContactMe" class="btn">CONTACT</a>
            <div class="row text-center mx-auto mb-3 ">
                <a href="{{ url_for('signinpage') }}" class="btn btn-primary btn-block "
role="button"><span
                        style="color:black; font-weight: bolder; height: 100px;">Sign in</span> </a>
            </div>
            <div class="row text-center mx-auto mb-5 ">
                <a href="{{ url_for('signuppage') }}" class="btn btn-primary btn-block"
role="button"><span
                        style="color:black; font-weight: bolder; height: 100px;">Sign up</span></a>
            </div>
        </nav>

        <h1 class = "CCR"> <span class="CCR_fl">C</span>USTOMER <span class="CCR_fl">C</span>ARE <span
class="CCR_fl">R</span>EGISTRY </h1>

    </div>
    <!----About section start---->
    <section class="about">
        <h1 id="AboutMe"></h1>
        <div class="main">
            <img src="https://website-static.s3.jp-tok.cloud-object-
storage.appdomain.cloud/img/BG.jpeg" >
    </section>
    <!-----service section start----------->
    <div class="service">
        <div class="title">
            <h2>Our Services</h2>
        </div>
        <div class="box">
            <div class="card">
                <i class="fas fa-bars"></i>
                <h5>Web Development</h5>
            </div>
        </div>
    </div>
```

```
    <!------Contact Me------>
    <div class="contact-me">
        <p>Let Me Get You A Beautiful Website.</p>
        <a class="button-two" href="#">Hire Me</a>
    </div>

    <section class="site-section " id="section-resume">
        <div class="container">
            <div class="row">
                <div class="col-md-12 mb-5">
                    <div class="section-heading text-center">
                        <h1 id="MyRes"></h1>
                        <h2>My <strong>Resume</strong></h2>
                    </div>
                </div>
                <div class="col-md-6">
                    <h2 class="mb-5">Educations</h2>
                    <div class="resume-item mb-4">
                        <span class="date"><span class="icon-calendar"></span> March 2019 -
Present</span>
                        <h3>B. E in Computer Science Engineering (CSE)</h3>
                        <p>
                            C Abdul Hakeem College of Engineering & Technology ANNA UNIVERSITY.<br>
                            CGPA: 8.62
                        </p>
                        <span class="school">Ranipet</span>
                    </div>
                    <div class="resume-item mb-4">
                        <span class="date"><span class="icon-calendar"></span> March 2018 - March
2019</span>
                        <h3>High School (HSC STATE Board)</h3>
                        <p>
                            KH MATRICULATION SCHOOL<br>
                            Percentage 70 %</p>
                        <span class="school">Ranipet</span>
                    </div>
                    <div class="resume-item mb-4">
                        <span class="date"><span class="icon-calendar"></span> March 2016 - March
2017</span>
                        <h3>High School (SSLC STATE Board)</h3>
                        <p> KH MATRICULATION SCHOOL<br>
                            Percentage 84.4 %</p>
                        </p>
                        <span class="school">Ranipet</span>
                    </div>
                </div>
                <div class="col-md-6">
                    <h2 class="mb-5">Experience</h2>
                    <div class="resume-item mb-4">
                        <span class="date"><span class="icon-calendar"></span> October 2022 -
Present</span>
                        <h3>Software Development Intern - Shiash Info Solution</h3>
                        <p>Developed and assisted with the project that is Developed using technologies
such as Java and Databases such as MySQL.</p>
                        <span class="school">Developed a Web based application that provides a services
which meets the client requirements.</span>
                    </div>
                </div>
                <div class="col-md-6">
                    <h2 class="mb-5">Skills</h2>
                    <div class="resume-item mb-4">
```

```
                    <!-- <span class="date"><span class="icon-calendar"></span> October 2022 -
Present</span> -->
                    <h3>Technical Skills - </h3>
                    Java   |    Python   |   MySQL    |   JavaScript
  |   React.js   |   HTML5 & CSS3  
                    <br><br>
                    <h3>Development Skills - </h3>
                    Website Development    |     Android Application Development   |
  Web based Application
                </div>
                <br>
                <a class="ResDoc"
href="https://drive.google.com/file/d/1P1Q4w62t1Xd4pv4OlVZOyk8z4ki9unbY/view?usp=share_link"> ⇲
Preview My Resume Here ↩  </a>
            </div>
        </div>
    </div>
  </section> <!-- .section -->
</body>
</html>
```

```
from __future__ import print_function

import datetime
from audioop import add
from pprint import pprint
from unicodedata import name
import ibm_db
import sib_api_v3_sdk
from flask import *
from flask import (Flask, flash, redirect, render_template, request, session,
                   url_for)
from markupsafe import escape
from sib_api_v3_sdk.rest import ApiException
from init import hello, id, randomnumber
conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME="";PORT="";SECURITY=SSL;SSLServerCertificate="";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('/signinpage', methods=['POST', 'GET'])
def signinpage():
    return render_template('signinpage.html')
@app.route('/agentsignin', methods=['POST', 'GET'])
def agentsignin():
    return render_template('signinpageagent.html')
@app.route('/signuppage', methods=['POST', 'GET'])
def signuppage():
    return render_template('signuppage.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('/newissue/<name>', methods=['POST', 'GET'])
def newissue(name):
    name = name
    return render_template('complaint.html',msg=name)
@app.route('/forgot', methods=['POST', 'GET'])
def forgot():
    try:
        global randomnumber
        ida = request.form['custid']
        print(ida)
        global id
        id = ida
        sql = "SELECT EMAIL,NAME FROM Customer 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
        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 = "Verification for Password"
        html_content = "<html><body><h1>Your verification Code is : <h2>" + \
            str(randomnumber)+"</h2> </h1> </body></html>"
        sender = {"name": "IBM CUSTOMER CARE REGISTRY",
                  "email": "ibmdemo6@yahoo.com"}
        to = [{"email": e, "name": n}]
        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)
        pprint(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('/verifyemail', methods=['POST', 'GET'])
def verifyemail():
    try:
        email = request.form['verifyemail']
        sql = "SELECT ID,NAME FROM Customer WHERE email=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
```

```
        emailf = ibm_db.fetch_both(stmt)
        while emailf != False:
            id = emailf[0]
            name = emailf[1]
            break
        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 = "Regarding of your Customer Id"
        html_content = "<html><body><h1>Your Customer Id  is : <h2>" + \
            str(id)+"</h2> </h1> </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)
        pprint(api_response)
        message = "Email send to:"+email+" 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("Database not found in mail! Please Register Your account.", "danger")
    finally:
        return render_template('signinpage.html')
@app.route('/otp', methods=['POST', 'GET'])
def otp():
    try:
        otp = request.form['otp']
        cusid = id
        print(id)
        sql = "SELECT PASSWORD FROM Customer 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('/admin', methods=['POST', 'GET'])
def admin():
    userdatabase = []
    sql = "SELECT * FROM customer"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
```

```
    while dictionary != False:
        userdatabase.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if userdatabase:
        sql = "SELECT COUNT(*) FROM customer;"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    users = []
    sql = "select * from ISSUE"
    stmt = ibm_db.exec_immediate(conn, sql)
    dict = ibm_db.fetch_both(stmt)
    while dict != False:
        users.append(dict)
        dict = ibm_db.fetch_both(stmt)
    if users:
        sql = "SELECT COUNT(*) FROM ISSUE;"
        stmt = ibm_db.exec_immediate(conn, sql)
        count = ibm_db.fetch_both(stmt)
    agent = []
    sql = "SELECT * FROM AGENT"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        agent.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if agent:
        sql = "SELECT COUNT(*) FROM AGENT;"
        stmt = ibm_db.exec_immediate(conn, sql)
        cot = ibm_db.fetch_both(stmt)
    return
render_template("admin.html",complaint=users,users=userdatabase,agents=agent,message=user[0],issue=cou
nt[0],msgagent = cot[0])
@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(prep_stmt)
            flash("delected successfully the Customer", "success")
        except:
            flash("No data found in Customer", "danger")
        finally:
            return redirect(url_for('signuppage'))
    if otp == 'A':
        try:
            insert_sql = f"delete from AGENT"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prep_stmt)
            flash("delected 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(prep_stmt)
```

```
            flash("delected successfully the Complaints", "success")
        except:
            flash("No data found in Complaints", "danger")
        finally:
            return redirect(url_for('signuppage'))
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        try:
            id = request.form['idn']
            global hello
            hello = id
            password = request.form['password']
            print(id, password)
            if id == '1111' and password == '1111':
                return redirect(url_for('admin'))
            sql = f"select * from customer 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("welcome"))
            else:
                flash("Mismatch in credetials", "danger")
        except:
            flash("Error in Insertion operation", "danger")
    return render_template('signinpage.html')
@app.route('/welcome', methods=['POST', 'GET'])
def welcome():
    try:
        id = hello
        sql = "SELECT ID,DATE,TOPIC,SERVICE_TYPE,SERVICE_AGENT,DESCRIPTION,STATUS 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 credetials", "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"))
```

```
FROM python:3.10.4
WORKDIR /app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["python","./app.py"]
```

13.2 GITHUB LINK & PROJECT DEMO LINK

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-221-1658224643

PROJECT DEMO LINK: https://mdsharu.github.io/CCR/