PROJECT REPORT

1. INTRODUCTION

1.1 Project Overview

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

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

An Online Comprehensive Customer Care Solution is to manage Customer interaction and complaints with the Services providers over the phone or through mail. The system should have capability to integrate with any service providers from any domain or industry like Banking, Telecom and many social Media.

2. LITERATURE SURVEY

2.1 Existing Problem

S.nO	PAPER TITLE	PAPER CONCEPT	ADVANTAGE	DISADVANTAGE
1.	Shafiq Darwish Alabri,Suzilawti Kamaruddin,Abdul Rehman Gilal, Jafreezal Jaafar,Izzatdin Abdul Aziz,"The moderation influence of power Distance on the relationship between technological factors and the successful implementation of citizen relationship management in the public sector",IEEE Access, pp.132446-132465,2020.	Customer relationship management is Curently an important Strategic tool used by organisations to gain competitive advantages.However since the implementation of a CRM system is not risk free, it is important to know about the factors that influence its success.	The advantage of this approch is that at the end of each iteration the result is a running system which allows for imediate feedback.It is important because very hard if possible at all, to gain a relatively precise and complete specification of a system in one big step.	The disadvantage of the system is costly implementation of the huge cost spent by the business.CRM data can be obtained and missused by other parties(Third party access)
2.	Muthuswamy Shanmugaraja,Muthuswamy Nataraj,Nallaswamy Kunasekaran. "Customer care management model for service industries" 2010,2,145-155	This paper describes a model for Customer care Management in an automotive service industries.It as developed as a structured complaint management practice which warrants the timely responce to customer complaints and speedy resolution for survival in todays customer driven market.	The Advantages of QFD, is one of the important analyses for companies to keep competitive edge.	The disadvantage of In this new approch , the inter relationships between solution factors or not concederd.
3.	Josephine D.German, Aaron David	Motor cycle Taxi Application (MTA) are alternative used by commuter at	A CSI value of 76.86% was also computed, Signifing that	That suggest the features and services in the application can still

	H.Cabacungan,"Customer awarness andsaisfaction analysis on the use of motor cycle taxi application in the phillippines",2021.IEEE 8th International confrence on industrial Engineering and Application(ICIEA), CPP.637-642,2021.	present to awoid experiencing traffic congestion and break down in public transfortation.The customer Satisfaction index(CSI) model was used to measure the customer satisfaction.	customer in the phillipines were satisfied in using the various empty application.	we further improved to continue providing adequet service and increase customer satisfaction.
4.	Saria safdar,Shoab Ahmadkhan,Arslan shaukat, "Customer Experience Management for automation,data collection and methodology",2019. International confrence on information and communication Technology convergence(ICTC) pp.1354-1358,2019.	The rapid growth of information in all field in the era of globalization allows everyone will need the information Technology. This service will contributes to companies,organization or institution that utilize it.	Public Service to the better in every way, so that the public service or customer of a company until the institution need to be considered. Can acess information easily, quickly and anyware.	Servie was Slow in the current era and cannot be anticipated will have less impact the development of the business. Society can be served well ad didn't know the time.
5.	Si-Ahmed Nass, Stephan Sigg," Real time emotion recognition for Sales" and "Customer Satisfaction Servey",2018. 16th international Confrence on mobility,sensing and netwoking(MSN).pp.584-591,2018.	This paper proposes an emotional agent model for the analysis of the CSS. The agent capture and aggregates the feedback of each item of the servey individually and produces a comprehensive emotional state. It deploys an emotinal agent to abserve customer feedback of a particular product or service.	Customer Satisfction survey which one of the best method to find out whether the customer are satisfied is asking them.	The ability of the software agent do not know the type of condition that generate emotional state regarding a particular behavior or event requirea speific domain knowledge and emotion mechanism.

2.2 References:

[1]. Afriliana, I., Munadia, H.,Hasta,I,D(20180. EKUPEL: E-Custromer Satisfication Questionnaries at PT.PLN (Persero) Rayon Tegal Timer.ICT Journal: Information Communication and Technology.

[2]. M. Baye, Managerial Economics & Business Strategy McGraw-Hill Education, London, Abacus:
The Undercover Economist, vol. 2013, pp. 12-23, 2017.

[3[Agussalim, M., Ayu Rezhikana putari, M.,&Ali,H (2016) Analysis work disipline and work sprit towards prformance of employees(case study tax prataama two padang).

[4]. J. Obliquity Kay, why our goals are best achieved indirectly, London:
Profile Book, pp. 15-67, 2011.

[5]. P. Keatand P.K. Young, Managerial Economics Global Edition, London:
Pearson, pp. 23-46, 2014.

[6]. Bai changhongand Liu Chi, "study on customer loyalty of service enterprises and its determinants [J]", nankaibusiness review, no. 06, pp. 64-69, 2002.

[7]. Chip R. Bell, The service edge: 101 companies that profit from customer care by Ron Zemke with Dick Schaaf, New York:NewAmerican Library, pp. 584, 1989

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.

PS- 1

PS-2

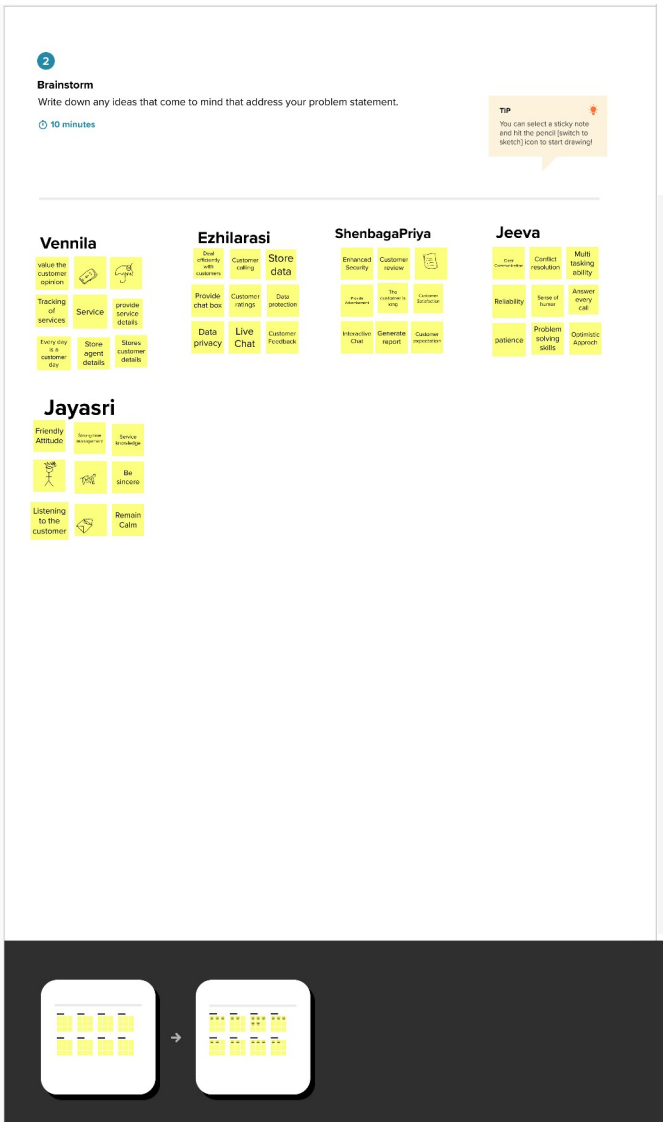
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A buyer	Buy a phone online	It takes a long time	The webpage is not responsive	annoyed
PS-2	A buyer	Add an item to my cart	But i cannot add the item to cart	The add to my cart button is faulty	disappointed

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation and 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, prioritisation volume over value,out-of-the-box ideas are welcome and build upon, and all participants are encourage to all collaborate, each other develop a rich amount of creative solution.


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!
Vennila
value the customer opinion
Tracking of services
Service
provide service details
Every day is a customer day
Store agent details
Stores customer details
Ezhilarasi
Deal efficiently with customers
Customer calling
Store data
Provide chat box
Customer ratings
Data protection
Data privacy
Live Chat
Customer Feedback
ShenbagaPriya
Enhanced Security
Customer review
The customer is king
Interactive Chat
Generate report
Customer experience
Jeeva
Conflict resolution
Multi tasking ability
Reliability
Sense of humor
Answer every call
patience
Problem solving skills
Optimistic Approch
Jayasri
Friendly Attitude
Service knowledge
Be sincere
Listening to the customer
Remain Calm
3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, 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
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.
customer
Customer Queries
Customer Satisfaction
Customer Expectation
Customer Ratings
Services
Stores Customer Details
Store Agent Details
Tracking of services
Provides service details
Security
Data Protection
Enhanced Security
Data Privacy
Stores data
Chatbox
Live Chat
Provide Chat box
Quick Chat resolution
Interactive chat


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.
We will solve the customer problems
Vennila
Ezhilarasi
ShenbagaPriya
Jeeva
Jayasri
customer
Services
Security
Chatbox
Improvement
Implement customer service program
Call your competitor
Timely response

3.3 Proposed Solution

Problem Statement:

To Solve Customer Issues Using Cloud Application development. Create problem Statement to understand your customer's point of view. It helps you figure out how your product or service will solve this problem for them.

The Statement helps you understand the experience you want to offer your Customers. It can also help you understand a new audience when creating a new product or service.

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

Idea/Solution Description:

Assigned Agent routing can be solved by directly routing to the specific agent about the issues using the specific Email.

Automated Ticket closure by using daily sync of the daily database. Status Shown to the status of the ticket to the customer. Regular data retrieval in the form of retrieving lost data.

Novelty and Uniqueness:

Assigned Agent Routing. Automated Ticket Closure, Status Shown to the Customer, and Backup data in case of failure.

Social impact and Satisfiction:

Upon implementation Customer Feel,

Customer Satisfaction,

Customer can track their status

Easy agent communication.

Business Model (Revenue Model):

Key Partners are Third-party applications, agents, and Customers.

Activities held as customer service, System Maintenance.

Key Resources support Engineers, Multi-channel.

Customer Relationship have 24/7 Email support, Knowledge-based channel.

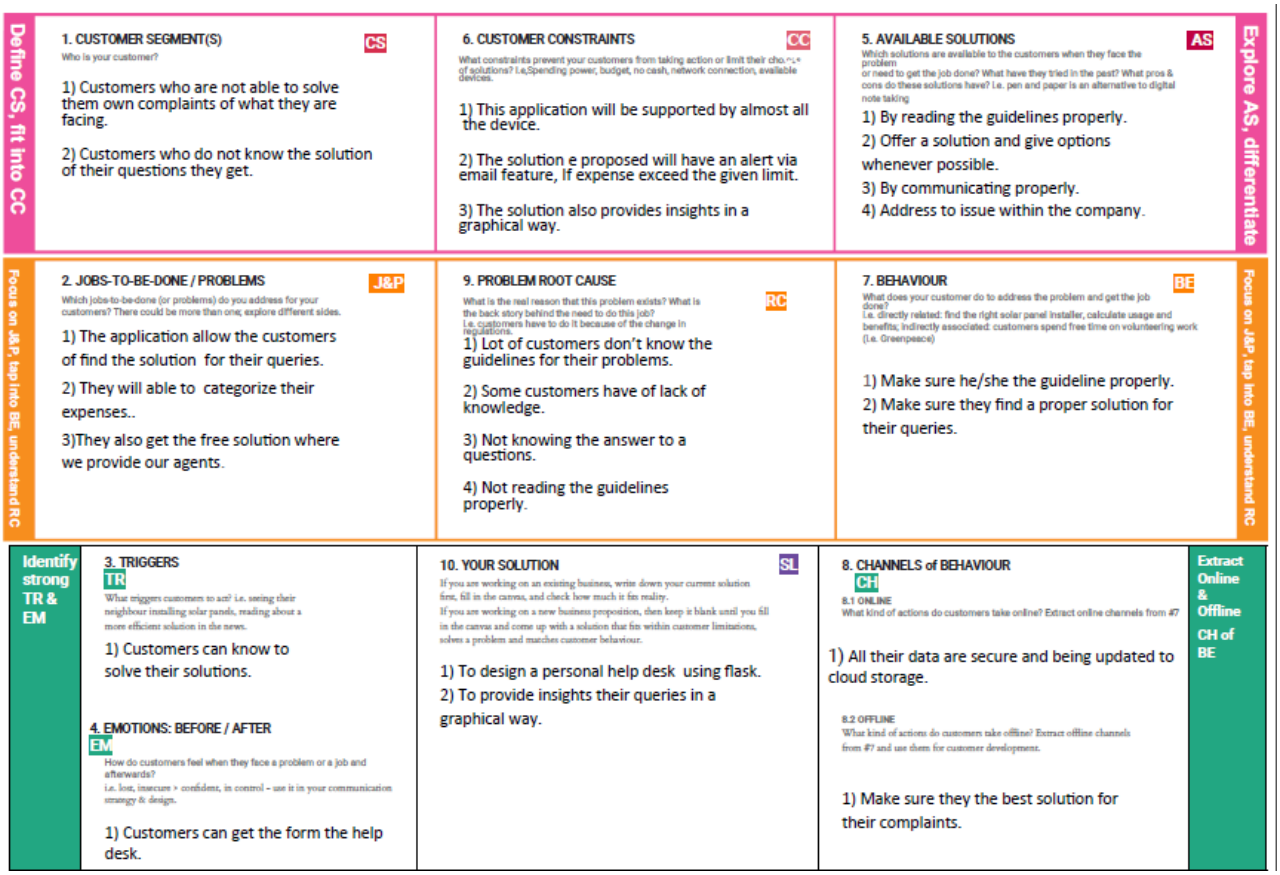
Cost Structure expresses Cloud Platform, Office.

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 grunt work and more time on actually resolving critical customer issues.

3.3 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 via Email. 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

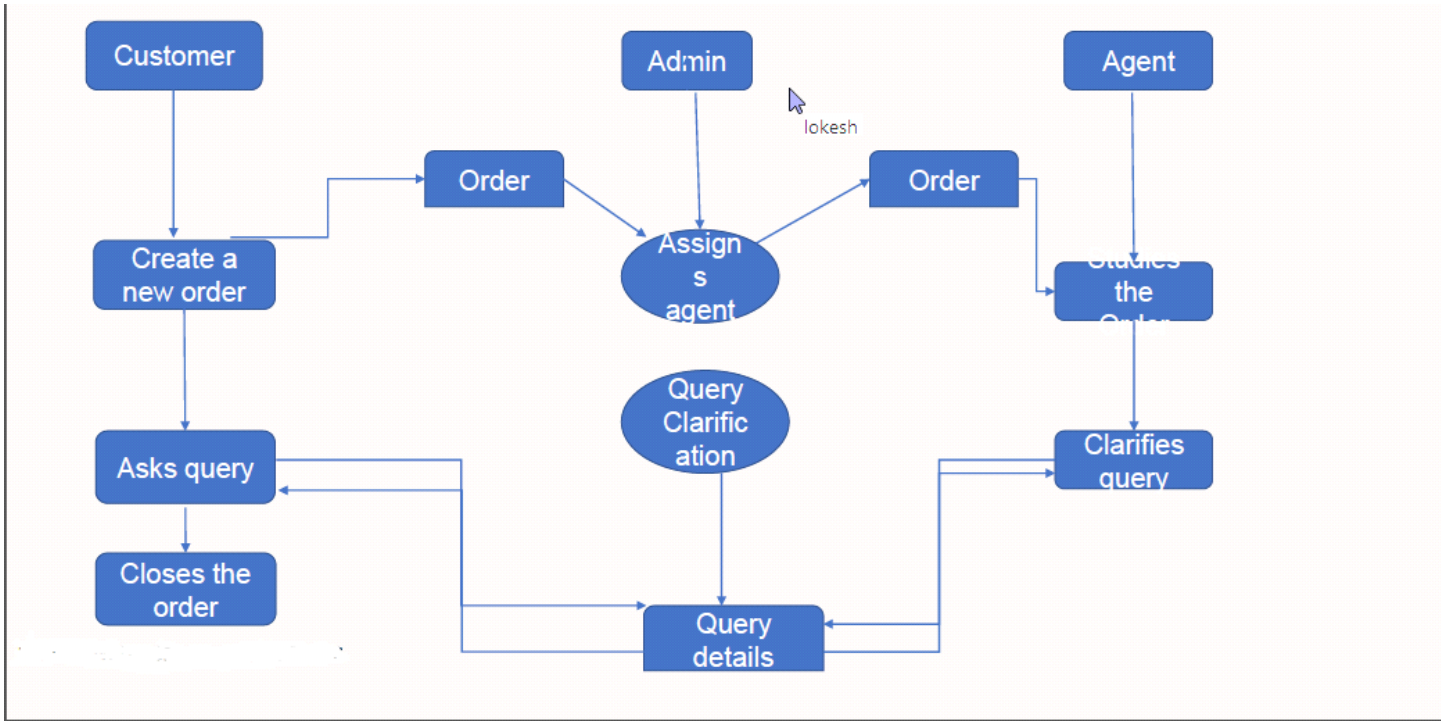
4.2 Non-functional Requirements:

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 decades status through Email
4	**Performance**	Effective development of web application
5	**Availability**	24/7 Service
6	**Scalability**	Agents Scalability as per the number of customers

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 &Technical Architecture

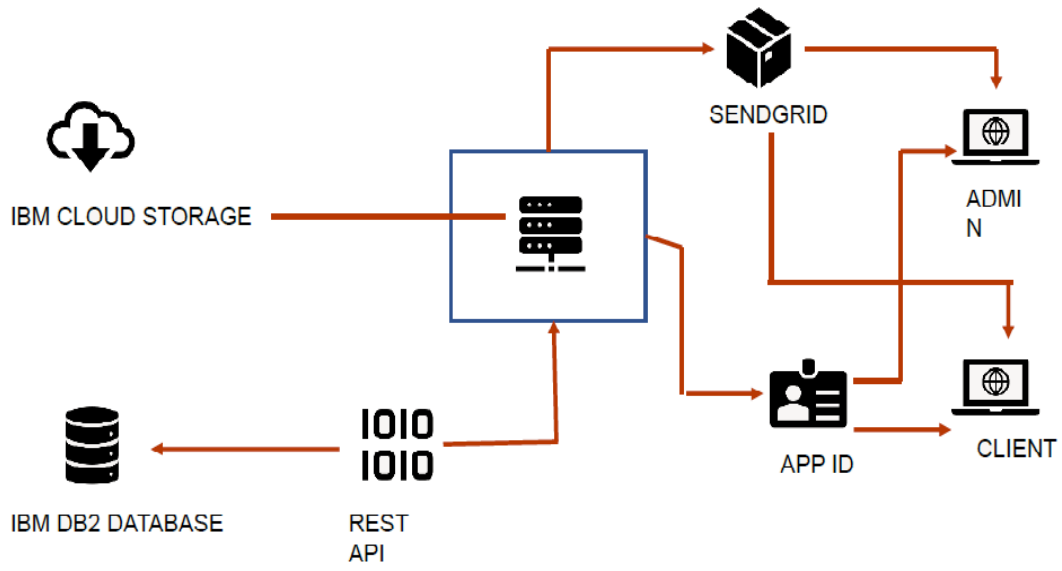
Solution Architecture is a Complex Process-- with many sub-process-- that bridges the gap between problems and technical solutions. Its goals are to:

1. Find best solution to solve existing business problems.

2. Describe the structure, characteristics, behaviour and other aspects of the software to project stakeholders.

3. Define features, development phase and solution requirements.

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



5.3 User Stories

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	login	USN-2	As a customer, I can login to the application by entering correct email and password.	I can access my account/dashboard.	High	Sprint-1
	Dashboard	USN-3	As a customer, I can see all the orders raised by me.	I get all the info needed in my dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can place my order with the detailed description of my query	I can ask my query	Medium	Sprint-2
	Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	My queries are clarified.	High	Sprint-3
	Forgot password	USN-6	As a customer, I can reset my password by this option incase I forgot my old password.	I get access to my account again	Medium	Sprint-4
	Order details	USN-7	As a Customer ,I can see the current stats of order.	I get abetter understanding	Medium	Sprint-4
Agent (web user)	Login	USN-1	As an agent I can login to the application by entering Correct email and password.	I can access my account / dashboard.	High	Sprint-3
	Dashboard	USN-2	As an agent, I can see the order details assigned to me by admin.	I can see the tickets to which I could answer.	High	Sprint-3
	Address column	USN-3	As an agent, I get to have conversations with the customer and clear his/er dobuts	I can clarify the issues.	High	Sprint-3
	Forgot password	USN-4	As an agent I can reset my password by this option in case I forgot my old password.	I get access to my account again.	Medium	Sprint-4

Admin (Mobile user)	Login	USN-1	As a admin, I can login to the appliaction by entering Correct email and password	I can access my account/dashboard	High	Sprint-1
	Dashboard	USN-2	As an admin I can see all the orders raised in the entire system and lot more	I can assign agents by seeing those order.	High	Sprint-1
	Agent creation	USN-3	As an admin I can create an agent for clarifying the customers queries	I can create agents.	High	Sprint-2
	Assignment agent	USN-4	As an admin I can assign an agent for each order created by the customer.	Enable agent to clarify the queries.	High	Sprint-1
	Forgot password	USN-5	As an admin I can reset my password by this option in case I forgot my old password.	I get access to my account.	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation Schedule

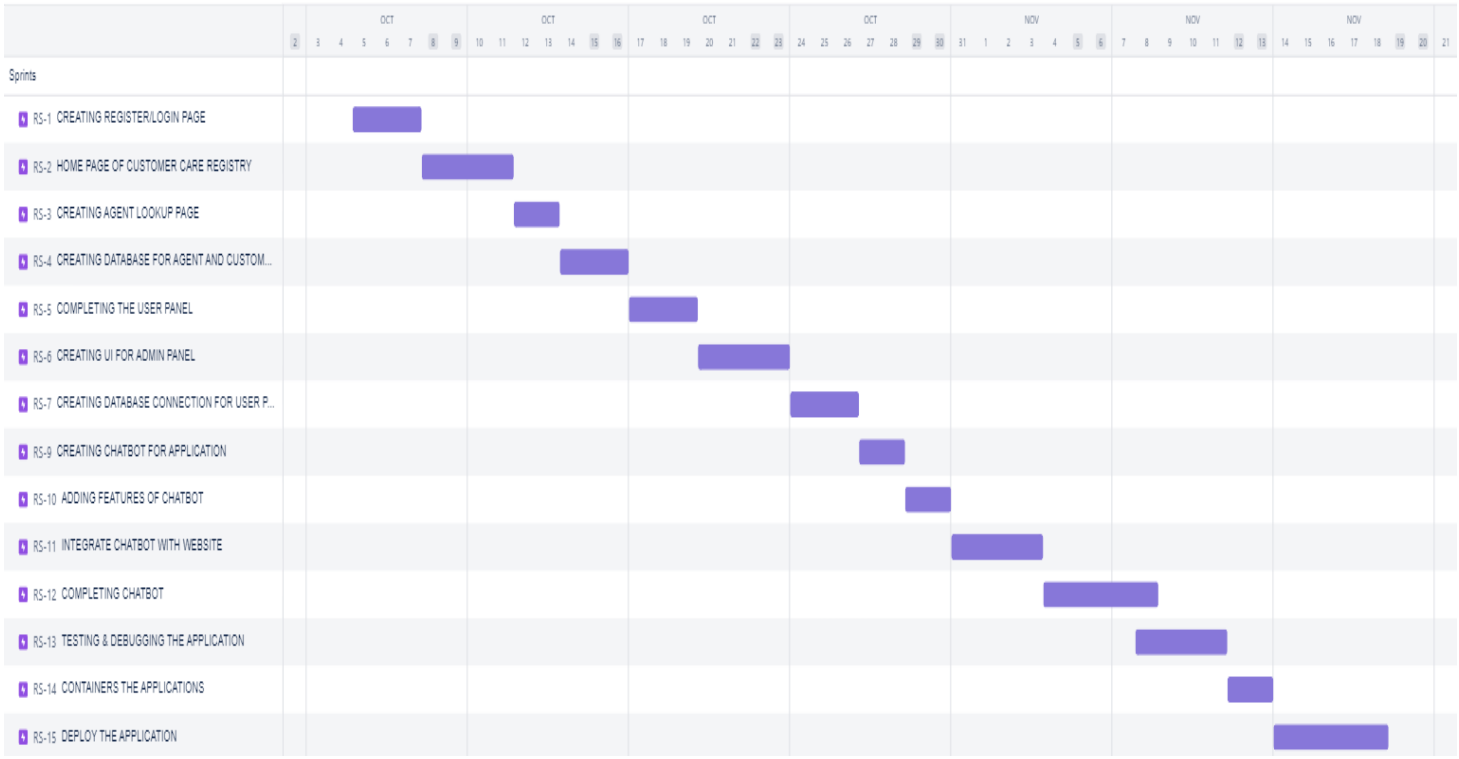
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and gothrough the services available on the webpage	20	High	SHENBAGAPRIYA JEEVA JAYASRI
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the availability and have a track of all the things that the users are going to services.	20	High	VENNILA EZHILARASI
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	EZHILARASI JEEVA
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	VENNILA SHENBAGAPRIYA JAYASRI

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6Days	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

6.3 Reports From JIRA



8. TESTING

8.1 Test Cases

FUNCTIONAL TESTING

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

WHITE BOX TESTING:

Testing based on an analysis of internal workings and structure of a piece of software. This testing can be done sing the percentage value of load and energy. The

tester should know what exactly is done in the internal program. Includes techniques such as Branch Testing and Path Testing. Also known as Structural Testing and Glass Box Testing.

BLACK BOX TESTING:

Testing without knowledge of the internal workings of the item being tested. Tests are usually functional. This testing can be done by the user who has no knowledge of how the shortest path is found.

8.2 User Acceptance Testing

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	Severity1	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

3.Test Case Analysis

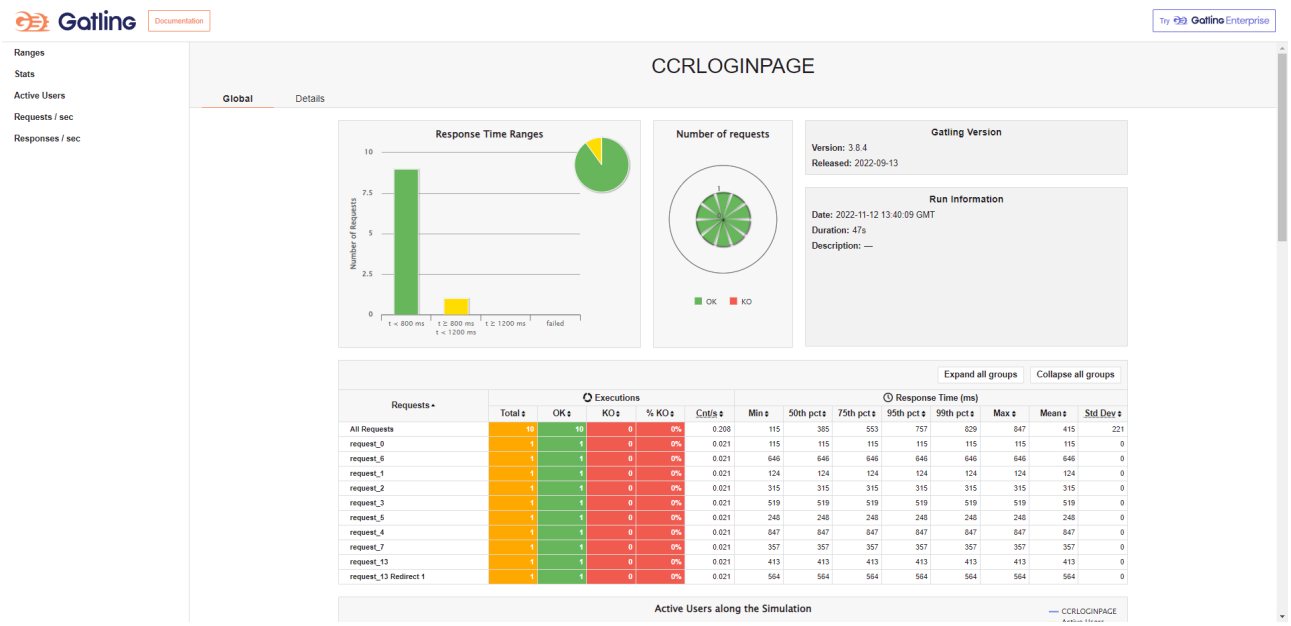
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	40	0	0	40
Security	5	0	0	5
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

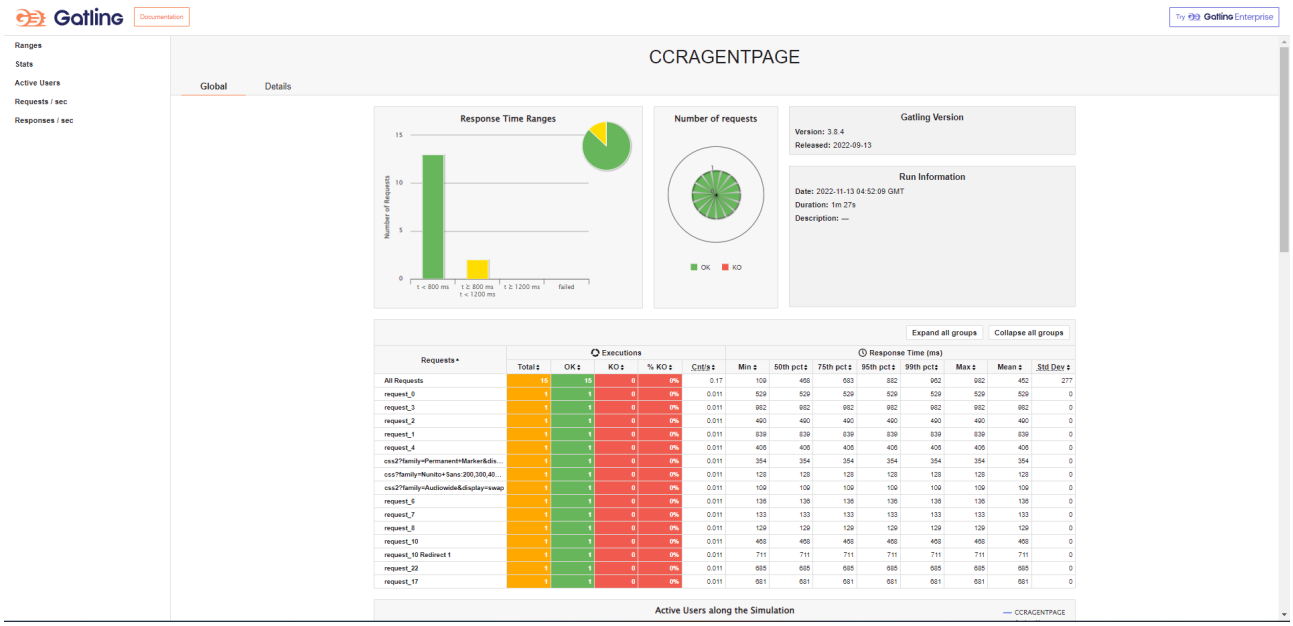
9. RESULT

9.1 Performance Metrics

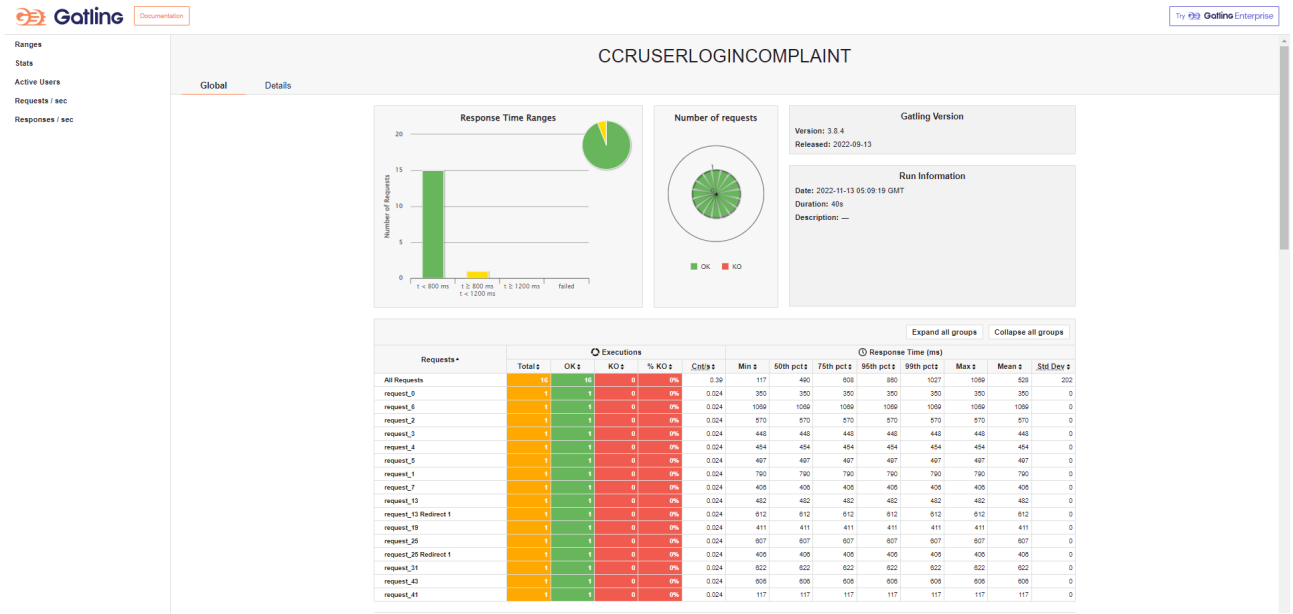
Login Page



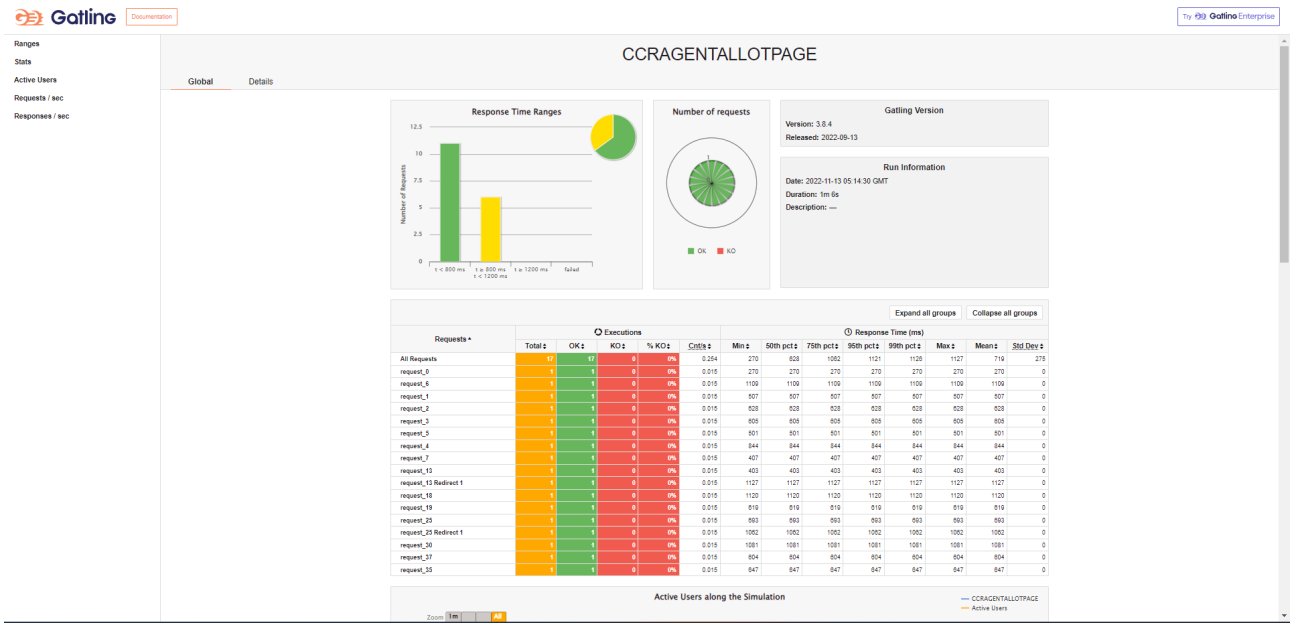
Agent Login Page



Complaint Page



User Alert Page



10. ADVANTAGES & DISADVANTAGES

Email Customer Services

Right from the first electronic mail send in 1971 to the billions of emails that are sent and receives every day, we have come a long way.

Among the different type of customer service available, Customers consider emails as a more trustworthy and professional channel. No wonder 12% of customers still choose email to register their requests.

The best part about email customer services is that it doesn't cost a fortune. Your agents get more time to respond, and they can use canned responses or email templates for faster replies.

One major challenge with email customer service is that after a point in time it becomes difficult keep track of every single email. In such a case, you can adopt customer email management software to convert emails into tickets and ensure they can never slip through the cracks.

ADVANTAGES:

Record and document customer conversations over a period of time

Add a professional touch to your customer service using email signature.

Automates email notifications can be used to update customers about the status of their issues or support ticket.

Easily attach relevant images, videos, docs, or other files.

DISADVANTAGES

Deploy 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.

Lack of real time human to human interaction.

11. CONCLUSION

In conclusion, Customer Care, involves the use of basic ethic and any company who wants to have success and grow, needs remainder, that in order to do so, it must begin with establishing a code of establishing a code of ethics in regards to how each employee is to handle the dealing with customers. Customer are at the heart of the company and its growth or decline. Customer care involves, the treatment, care, loyalty,trust the employee should extend to the consumer, as well in life. This concept can be applied to so much more than just customer care. People need treat others with respect and kindness, people should try take others into consideration when making any decision. If more people were to practice this policy, chances are the world would be a better, more understanding place for all to exist.

12.FUTURE SCOPE

The current state of customer care registry, in so many companies, looks something like this:

- Customer acquisition is prioritised over retention
- Customer service investment projects are sidelined.
- Departmental efficiency is of highest priority.
- Businesses see employees in the customer service department as short-term and disposable. They are there to fulfil a specific, repetitive, purpose.
- Employees are considered unskilled and leaders hire accordingly.

- New agents view customer service as a ‘last resort’ or ‘short term’ job. People often see careers in customer support as unambitious.
- Agent training rarely goes beyond product and people skills.

In the next 3-5 years, we expect to see these **future customer care registry trends**:

- The shift from a primarily 'cost centre' to primarily 'growth centre' worldview.
- The job desk for a customer care registry director will focus more on leadership, innovation, and ability to drive company-wide improvement.
- Customer service will shift to become a strategic partner of marketing, sales, and product development. CS will help with direction, project prioritisation, and impact.
- A need for customer service leaders to take a highly strategic seat at the table. They’ll need to argue for investment in talent, technology, and innovation.
- A shift in performance metrics. Forget of resolved tickets. In the future, we'll measure performance based on of customers saved from the precipice of churn.
- A career in customer care registry will not be a last resort. Top graduates will prioritise getting an education in strategic customer interaction.

- Focus on ticket deflection will reduce because brands will view each customer interaction as an opportunity to learn, build a relationship, and grow profits. They deserve a well-trained, human touch.

  Modern and developing technology enables this future to exist.
  With new technology, administrative tasks will tend toward zero.

- The sole purpose of the customer service is to meet the expectations of the customers so that they are satisfied with the outcome. These services are also available to understand the queries of the customers and ensure that they enjoy a cost-effective experience after purchasing any product from the respective company.

13. APPENDIX

Source Code

```
 from __future__ import print_function

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=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)

  rerender_template("admin.html",complaint=users,users=userdatabase,agents=agent,mes
sage=user[0],issue=count[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"))
```

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-42901-1660710986.git