IBM NALAIYATHIRAN PROJECT REPORT

CUSTOMER CARE REGISTRY

Submitted by

TEAM ID : PNT2022TMID47409

Team Members	Register number
Abidarsh Mohan(TL)	910019104002
Ritchard M	910019104034
Annamalai S	910019104004
Srinath A	910019104042

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Customer care and customer service together help create a positive customer experience, or the overall impression a person has when interacting with your company. Both are vital, but there are subtle differences in how they are implemented. High-quality customer care is proactive. The needs of customers throughout the buyer's journey are anticipated, making customers feel supported. That, in turn, helps create an emotional connection between the customer and the company. Customer service is reactive. Here, the focus is on helping customers solve problems or answer questions before purchase, either in a self-serve fashion or via the customer support team. Customer care is more than just providing great customer service. It's a proactive approach to providing information, tools, and services to customers at each point they interact with a brand. If a company neglects customer care, it can negatively impact the customer service experience. For example, when a website chatbot can't provide key information about a product, customers are more likely to get frustrated and reach out to a customer service agent for help. Consumer expectations are

extremely high, putting increased pressure on companies to improve their customer relationships. This can lead to lost information when the same person reaches out via multiple channels. When a customer service agent doesn’t know the whole story and the customer has to repeatedly share the problem, it leaves both people frustrated. They can register for an account. After the login, they can create a complaint with a description of the problem they are facing. Each user will be assigned an agent. They can view the status of their complaint. • Customers get the insights they need to make an informed purchase. • Customer satisfaction can increase and customer loyalty can improve. • Customer service agents spend less time on routine tasks and answering commonly asked questions, enabling agents to do more meaningful task.

1.2 PURPOSE

There are two sides to customer service objectives. First, there are the goals and KPIs customer service teams attempt to achieve. Then, there’s customer service resume objectives. It’s important to understand the connection between the two: Writing a strong customer service resume objective starts with understanding the objectives of the field and its depth and possibilities. To provide insight into both levels of customer service objectives. The prime objective of customer service is to answer customer questions quickly and effectively, resolve issues with empathy and care, document pain points to share with internal teams, nurture relationships, and improve brand credibility. Great customer service can make people loyal to your brand, products, and services for years to come. 4 A strong customer service resume objective underscores your skills and experiences in contributing to customer service's overall goals and objectives. Meeting key customer service KPIs doesn’t just involve answering phones and emails. It's a whole world of solutions development, intuition, empathy, brand management, time management-and the soft skills that help connect people and create trust. I guide my team toward giving the best service possible. Sometimes, we’re not delivering good news. But the objective is to do that with compassion and empathy and in a way that we give the customer constructive next steps to move forward.

2 . LITERATURE SURVEY

2.1 EXISTING PROBLEM

The existing system is a semi-automated at where the information is stored in the form of excel sheets in disk drives. The information sharing to the Volunteers, Group members, etc. is through mailing feature only. The information storage and maintenance is more critical in this system. Tracking the member's activities and progress of the work is a tedious job here. This system cannot provide the information sharing by 24x7 days.

2.2 REFERENCES

1] Uddi Executive Overview: Enabling Service Oriented Architecture, 2004 Oct.

[2] "Web Services Architecture", http://www.w3.org/TR/ws-arch/. Date Accessed: 11/02/2004.

[3] UDDI V3 Specification. http://uddi.org/pubs/uddi-v3.00-published-20020719.html. 19/07/2002.

[4] Christensen E, Curbera F, Meredith G, Weerawarana S, Web Services Description Language (WSDL) 1.1, W3C Note, 2001

[5] Ali A S, Rana O F, Ali R A, Walker D W, UDDIe: an extended registry for Web services, SAINT-w'03 Proceedings of the 2003 Symposium on Applications and the Internet Workshops, 2003 Jan, pp .85-89.

[6] Tsai W T, Paul R, Cao Z, Yu L, Saimi A, Xiao B, Verification of Web Services Using an Enhanced UDDI Server ,Proceedings of The Eighth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems ,2003 Jan, pp 131-38.

[7] Liu J, Gu N, Zong Y Ding Z, Zhang S, Zhang Q Service Registration and Discovery in a

Domain-Oriented UDDI Registry , 6 Proceedings of the 2005 The Fifth International Conference on Computer and Information Technology (CIT’05),2005, pp .276-83.
[8] Jian W, Zhaohui W, Similarity-based Web Service Matchmaking Proceedings of the 2005 IEEE International Conference on Services Computing (SCC’05), 2005.
[9] Tretola G, Zimeo E , Structure Matching for Enhancing UDDI Queries Results , IEEE International Conference on Service-Oriented Computing and Applications(SOCA'07),2007.
[10] Ayorak E, Bener a, Super Peer Web Service Discovery Architecture, IEEE Proceedings - International Conference on Data Engineering, 2007 pp .287-94.
[11] Libing He W, Wu Y, Jianqun D A novel interoperable model of distributed UDDI - proceedings of the 2008 IEEE International Conference on Networking, Architecture, and Storage - IEEE NAS 2008 Jun, pp .153-54.
[12] Nawaz F, Qadir K, Ahmad H F, and SEMREG-Pro: A Semantic based Registry for Proactive Web Service Discovery using Publish Subscribe Model, Fourth International Conference on Semantics, Knowledge and Grid, IEEE Xplore, 2008 Dec, pp .301-08. [13] LIANG1 Q, CHUNG2 J A Federated UDDI System for Concurrent Access to Service Data, IEEE International Conference on eBusiness Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, and DKEEE’08, 2008 Oct, pp .71- 78.
[14] Vandan T, Nirmal D, InderjeetGarg S, Garg N, Soni P. An Improved Discovery Engine for Efficient and Intelligent discovery of Web Service with publication facility, SERVICES 2009 - 5th 2009 World Congress on Services, 2009, pp .63-70.
[15] Ma C, Song M, Xu K, Zhang X.Web Service Discovery Research and Implementation Based on Semantic Search Engine, IEEE 2nd Symposium on Web Society, Beijing. 2010 Aug 16-17, pp .672-77.
[16] Rajendran T.Balasubramanie P, Flexible and Intelligent Architecture for Quality Based Web Service Discovery with an Agent- Based Approach, International Conference on Communication and Computational Intelligence (INCOCCI), Erode. 2010 Dec 27-29, pp .617- 22.

2.3 PROBLEM STATEMENT DEFINITION

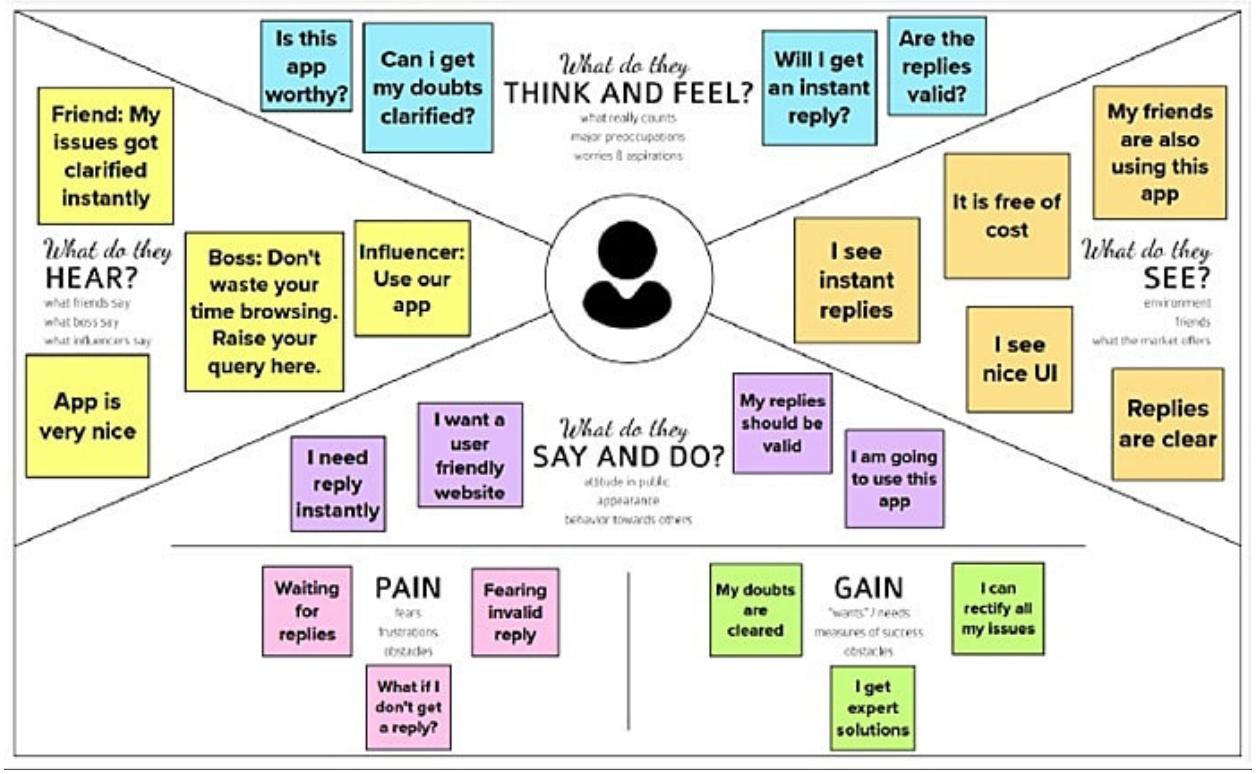
A problem statement is a concise description of the problem or issues a project seeks to address. The problem statement identifies the current state, the desired future state and any gaps between the two. A problem statement is an important communication tool that can help ensure everyone working on a project knows what the problem they need to address is and why the project is important.

3 . IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

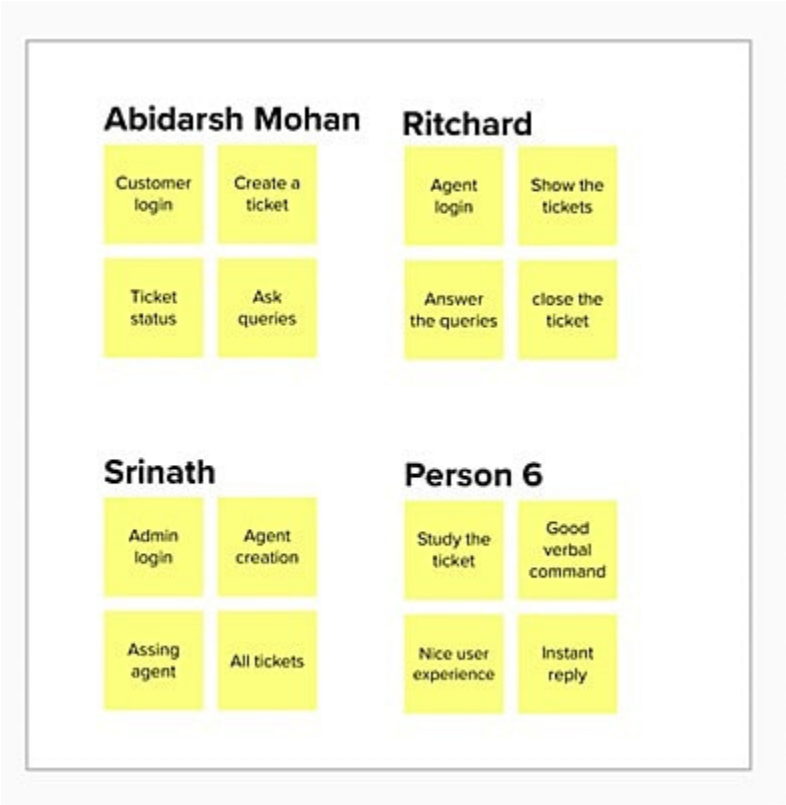
Empathy Map for Customer Care Registry:



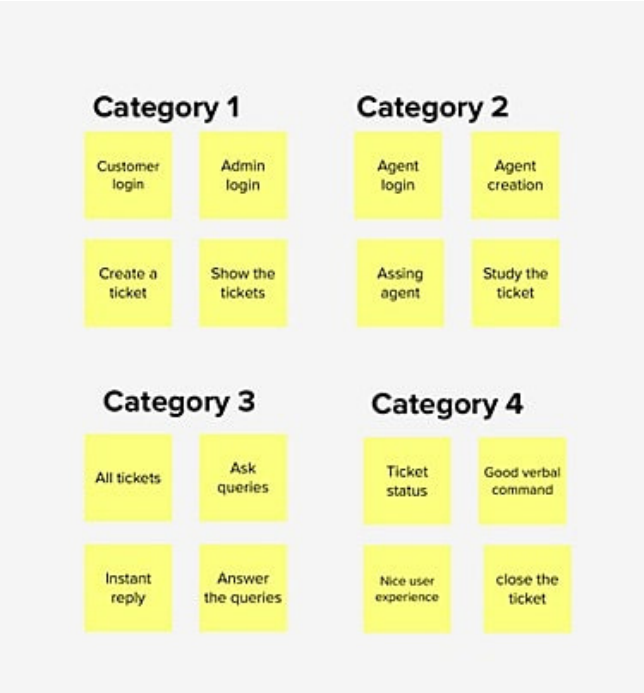
3.2 . IDEATION AND BRAINSTORMING

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

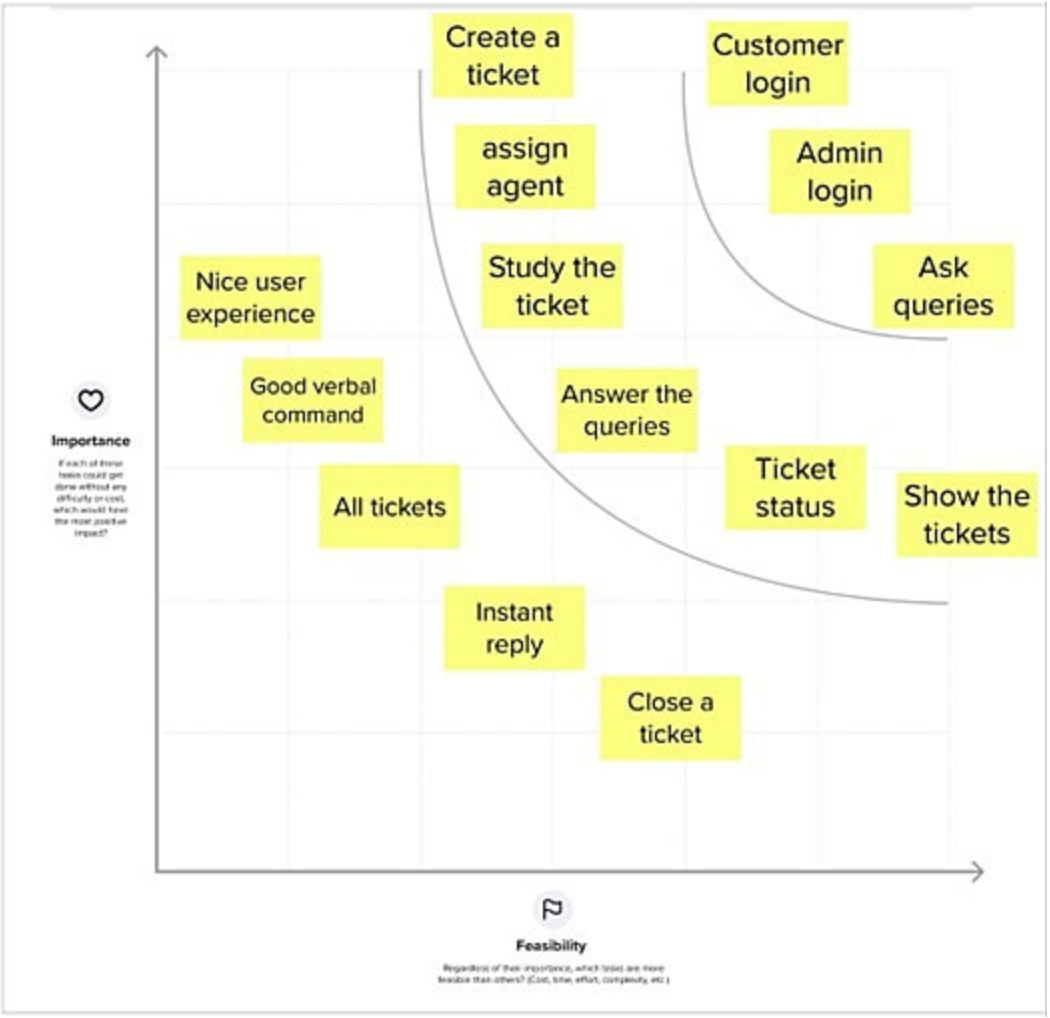
Brainstorm, Idea Listing



Grouping ideas



Idea Prioritization



3.3 . PROPOSED SOLUTION

Allotted Agent routing can be resolved by directly routing to a dedicated agent about the issue using the email. Automated Ticket closure by using sync of the cloud database. Status shown to the customer can display the tickets to the customer. The goal of the customer care service is to provide the platform that will allow the customer specialist to be efficient. And the solve the query with less time.

S.No.	Parameter	Description
1.	Problem Statement (Problemto be solved)	Customers facing problems in the product purchased or service providedis very common. For getting resolved these issues customers need to visit physically the regarding organisation which makes it an inconvenient process
2.	Idea/ Solution description	To create a customised application whichallows customers who can raisetheir issue which will be forwarded toadmin who assigns agents to rectify the issue and customer can also keep track of issue to know the current status.
3.	Novelty / Uniqueness	The problems can be rectified online. Automated assignment of problems toavailable agents in a distributed manner.Status shown to the customer.
4.	Social Impact/ Customer Satisfaction	Drastically reduces time for a problem to be rectified which in turn increases the user experience and theyaremore likely to trust and be loyalto the company.
5.	Business Model (Revenue Model)	Introducing subscription plans toprovide premium services to paidusers by which organization can create revenue.
6.	Scalability of the Solution	The real goalof scaling customer service is providing an environment that will allowyour customer servicespecialists to be efficient as possible.

3.4 . PROBLEM SOLUTION FIT

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Customers who face issues in any product or services and for this issues they can approach solution.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Unsatisfied service provided by the agent.No suitable tool available.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem
or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Help desk :
By understanding the issue properly and providing better solution.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

Giving instant replies to the customer to their queries.

Updating the status of their queries to the customers.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Increasing complaints and no proper environment to manage,solve and track those issues.
Communication lag

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Customer should say the problem in effective manner and keep in touch with the agent until the issues resolved.

Focus on J&P, tap into BE, understand RC

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

The satisfying solution with proper resolving issues triggers.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Anxiety
Tensed
Anger
Frustuated
Disappointed

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

Creating a user-friendly application.

Individual agents will be assigned to each customer and also check their status of their queries.

Live chat option will be provided.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

Online:
Search on google.
Search on Help desk

offline:
Do by themselves.
Asking friends.

4 . REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

S No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	User Registration	Registration through Form Registration through Gmail Registration throughLinkedin.
2	User Confirmation	Confirmation via Email Confirmation viaOTP.
3	User Login	Login via Google Login with Email Id and password.
4	Complaint Registration	Registration complaint usingthe query formavailable in the dashboard.
5	Tracking Status	Fetching the statusof query usingunique id.
6	E-mail	Receiving email from the executive.

4.2 NON FUNCTIONAL REQUIREMENTS

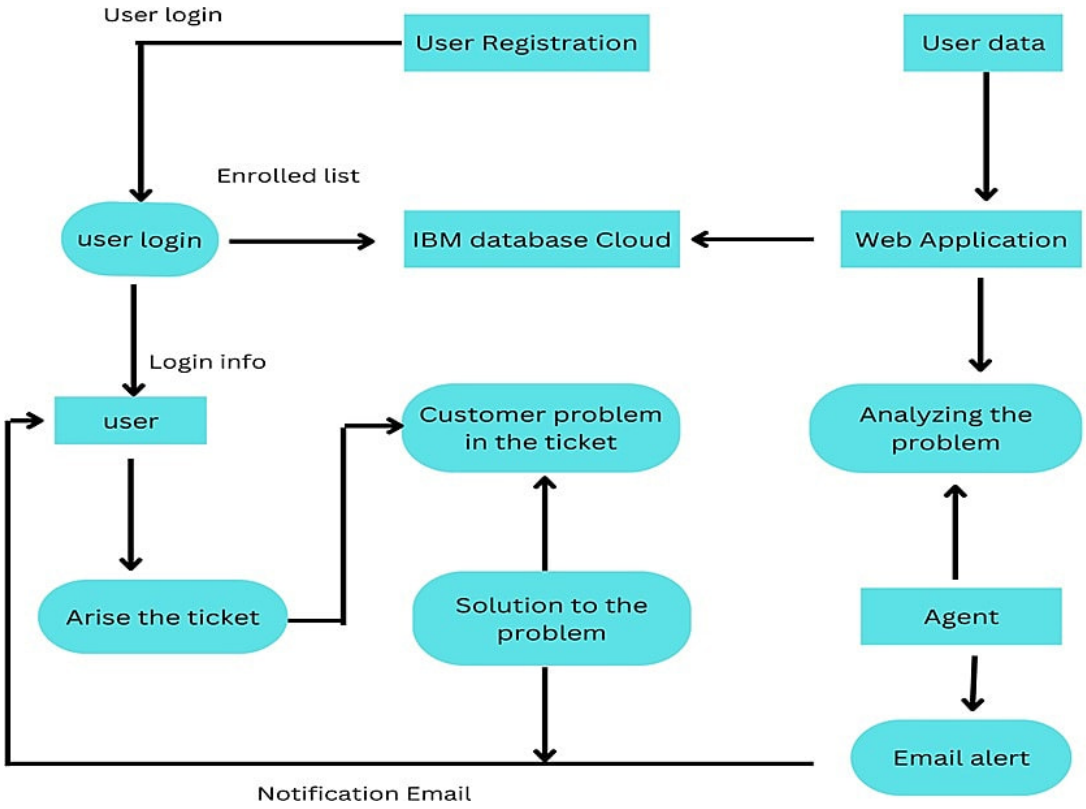
Following are the non functional requirements of the proposed solution.

S No.	Non-Functional Requirement	Description
1	Usability	To provide the solution to the problem.
2	Security	Encryption standards mustbe used in database.
3	Reliability	Tracking of decade status through E-mail.
4	Performance	Effective development of web application.
5	Availability	24/7 customer service multiple servers to avoid traffic.
6	Scalability	Agents scalability as per the number of customers.

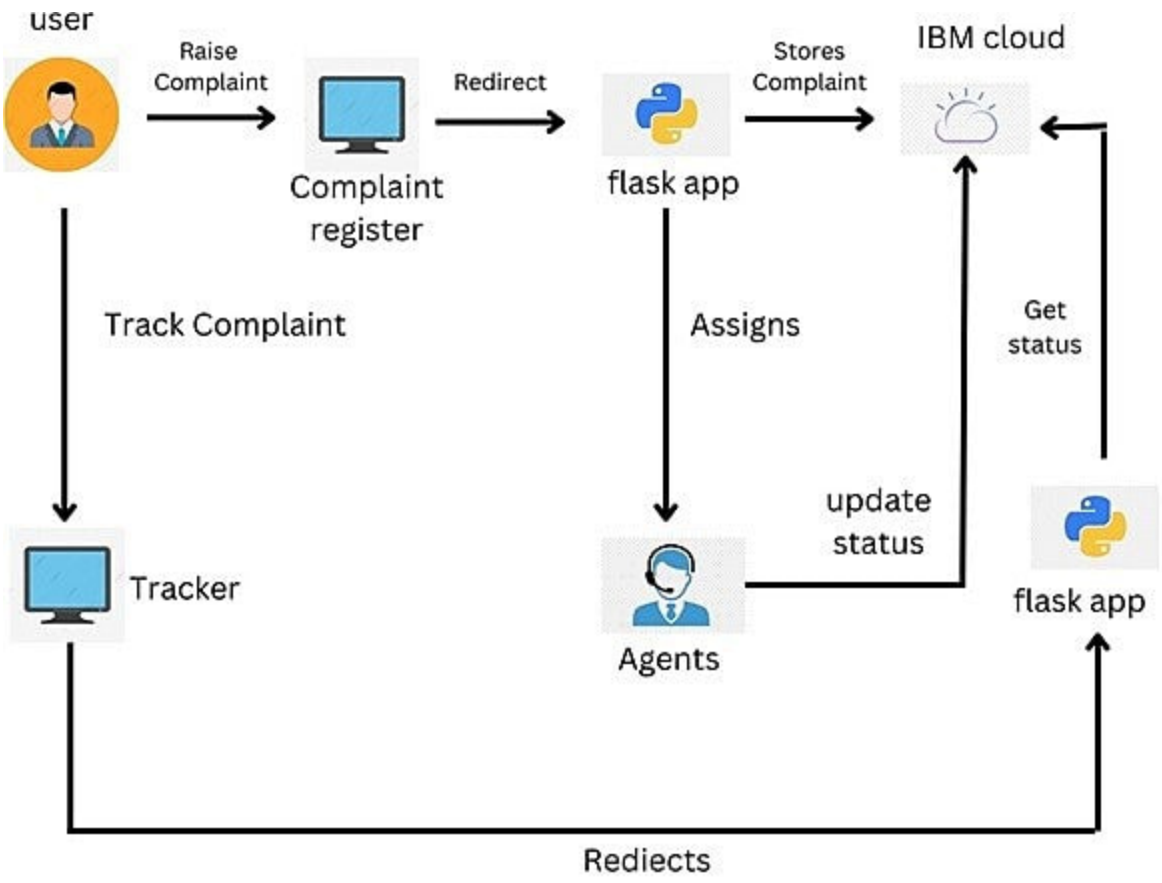
5 . PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

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 systemrequirement graphically. It shows how data entersand leaves the system, what changes the information, and where data is stored.



5.2 . SOLUTION AND TECHNICAL ARCHITECTURE



5.3 . USER STORIES

User Type	Functiona Requirem ent	User sto ry Num	User Story/ Task	Acceptance criteria	Priori ty	Relea se
Customer (Mobileuser)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account /dashboard	High	Sprint-1
		USN-2	As a user,I will receiveconfirmation email once I haveregistered for theapplication	I can receive confirmation email& click confirm	High	Sprint-1
		USN-3	As a user, I can register for the applicationthrough Facebook	I can register & access thedashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user,I can register for the application throughGmail	I can register with email and access thedashboard	Medi um	Sprint-1
	Login	USN-5	As a user,I can log into the application byentering email &password	I can register with Emailand password	High	Sprint-1
	Dashboard	USN-6	As a user,Ican arise the ticket to say the problem	I can manually analyzing the problem.	High	Sprint-1
Customer (Webuser)	user	USN-7	As a customer, I can say my problemin the ticket	To provide good solutionshould analyzing be careful	High	Sprint-1
Customer Care Executive	Technical support	USN-8	Requested the detailed description of the problem.	The customer will satisfied with the givensolution.	High	Sprint-1
Administrator	Creator	USN-9	An Agent can control the process	Inform aboutlevel of solution in mail notification.	Medi um	Sprint-1

6 . PROJECT PLANNING & SCHEDULING

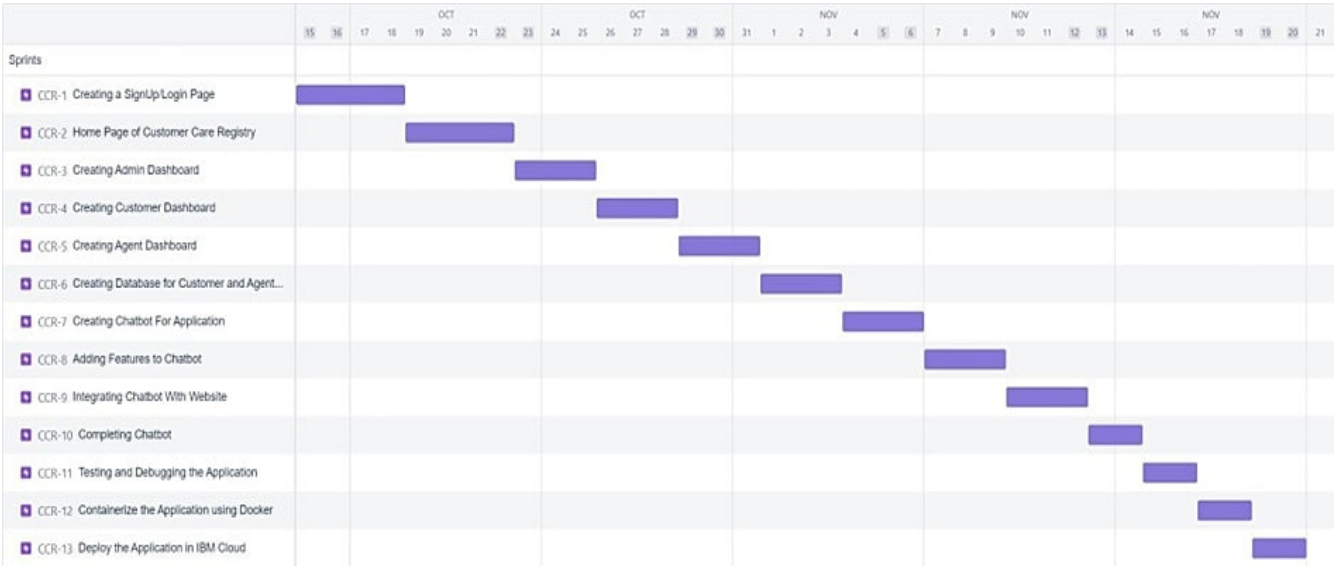
6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Customer Panel	USN-1	As a Customer, I can register for the application by entering my email, password, and confirming my password and I will be able to Access my dashboard for creating a Query Order.	2	High	Abidarsh Mohan, Annamalai
Sprint-1	Admin Panel	USN-2	As an admin, I can Login to the Application by entering correct login credentials and I will be able to Access My dashboard to create Agents and Assign an Agent to a Query Order.	2	High	Richar d, Srinath
Sprint-2	Agent Panel	USN-3	As an agent, I can Login to the Application by entering correct login credentials and I will be able to Access my Dashboard to check the Query Order and I can Clarify the Issues.	2	High	Abidarsh Mohan, Richard
Sprint-3	Chat Bot	USN-4	The Customer can directly Interact to the Chatbot regarding the services offered by the Web Portal and get recommendations based on information provided by them.	2	Medium	Srinath,Annamalai
Sprint-4	Final Delivery	USN-5	Container of applications using Docker kubernetes and deployment the application.Create the documentation and final submit the application	2	High	Abidarsh Mohan,Srinath, Annamalai,Richard

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	4 Days	28 Oct 2022	01 Nov 2022		01 Nov 2022
Sprint-2	20	7 Days	31 Oct 2022	06 Nov 2022		06 Nov 2022
Sprint-3	20	8 Days	07 Nov 2022	14 Nov 2022		14 Nov 2022
Sprint-4	20	7 Days	14 Nov 2022	21 Nov 2022		21 Nov 2022

6.3 REPORTS FROM JIRA



7. CODING & SOLUTIONING

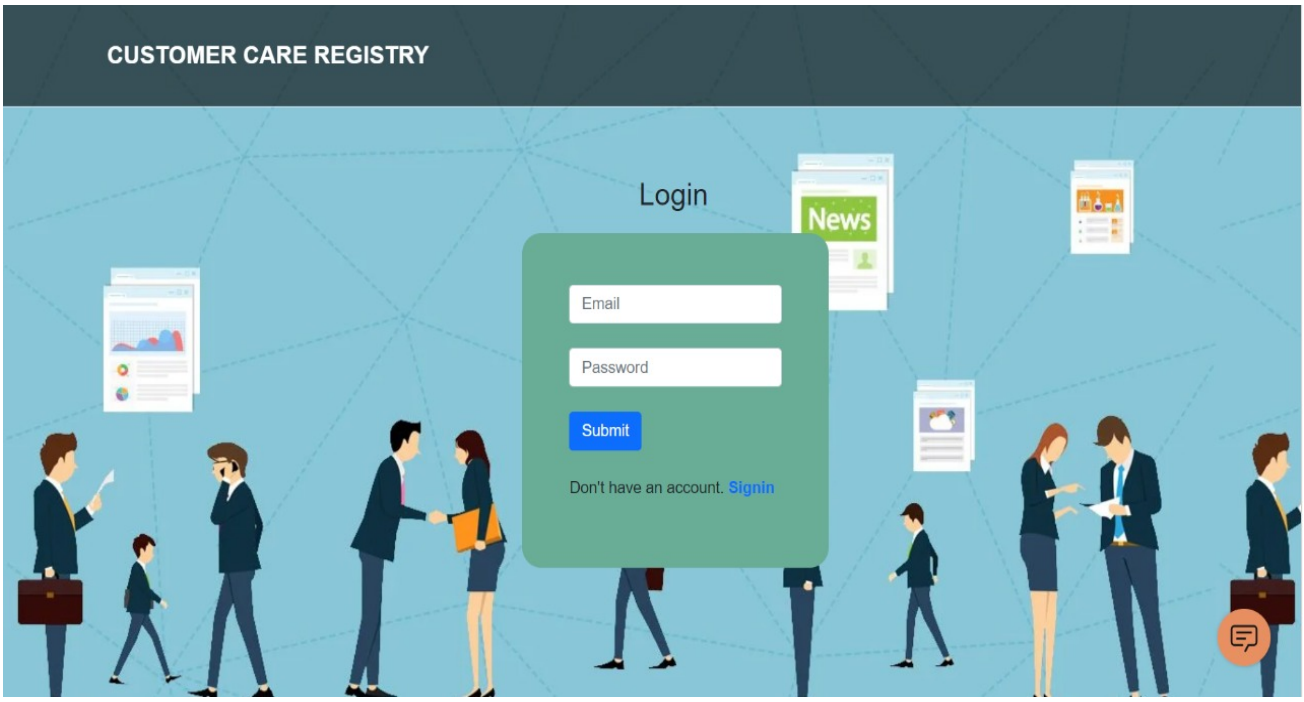
7.1 FEATURE 1

7 Main types of customer needs:

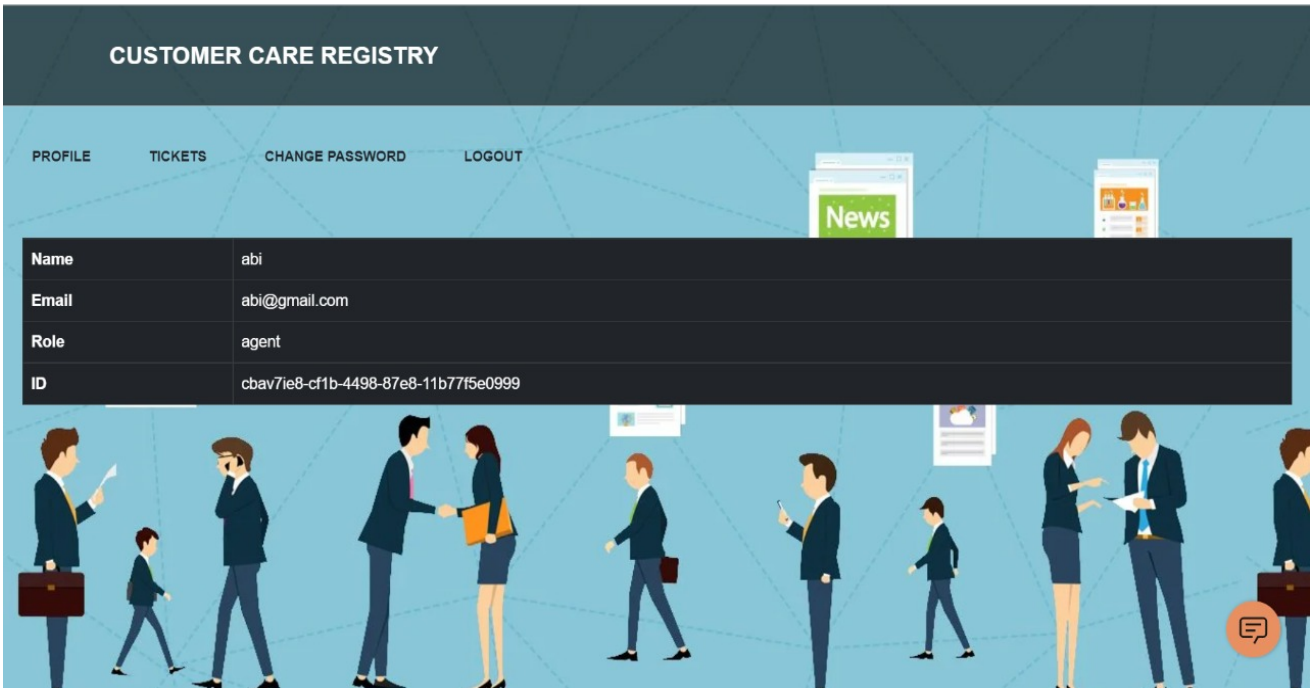
- Friendlines
- Empathy
- Fairness
- Control
- Alternatives
- Information

HOME PAGE

LOGIN PAGE



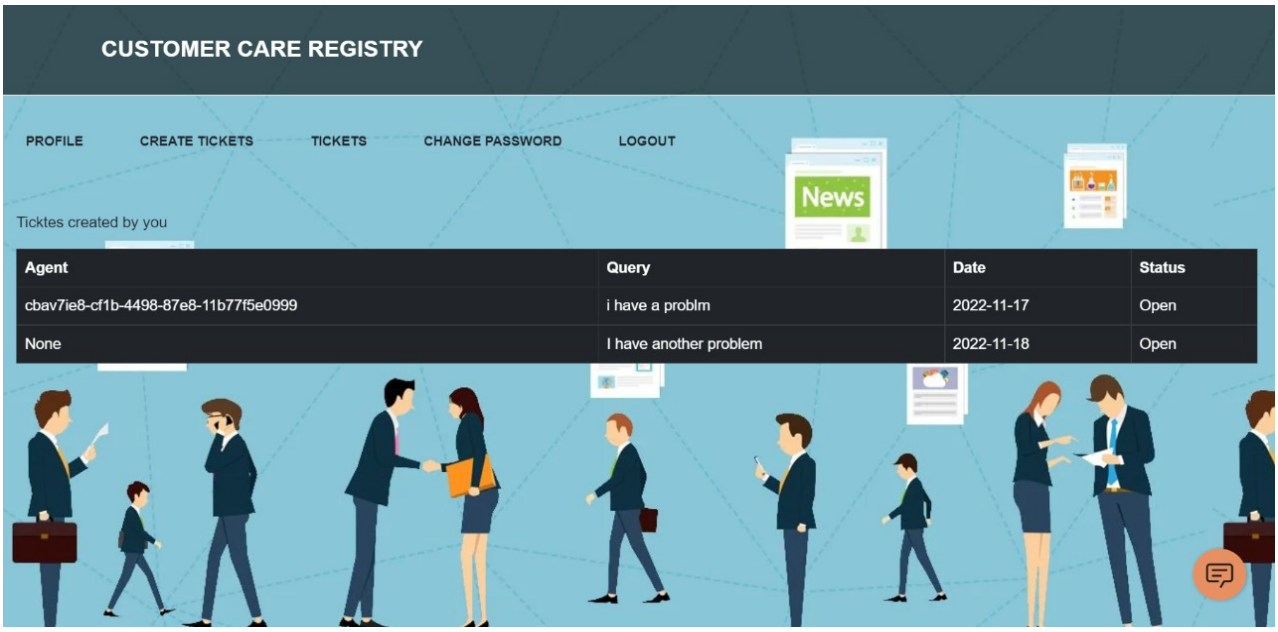
PROFILE PAGE



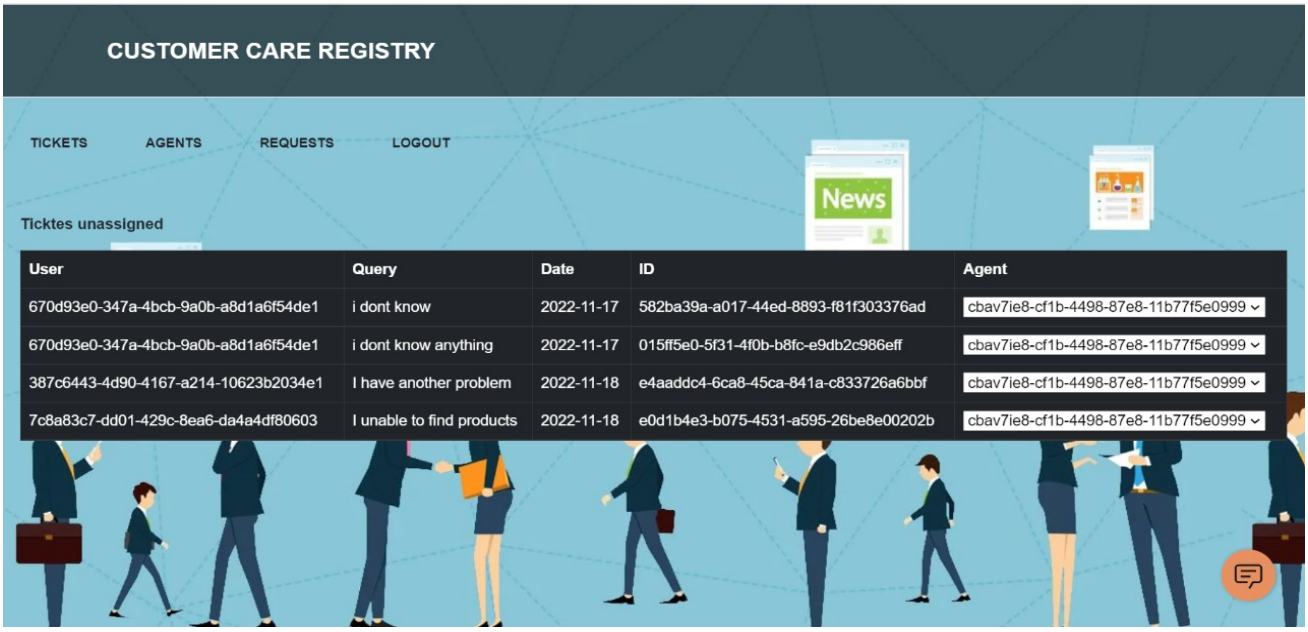
7.2 FEATURE 2

- Complaint Tracking
- Email Alert
- 24/7 Montoring

TICKET CREATION

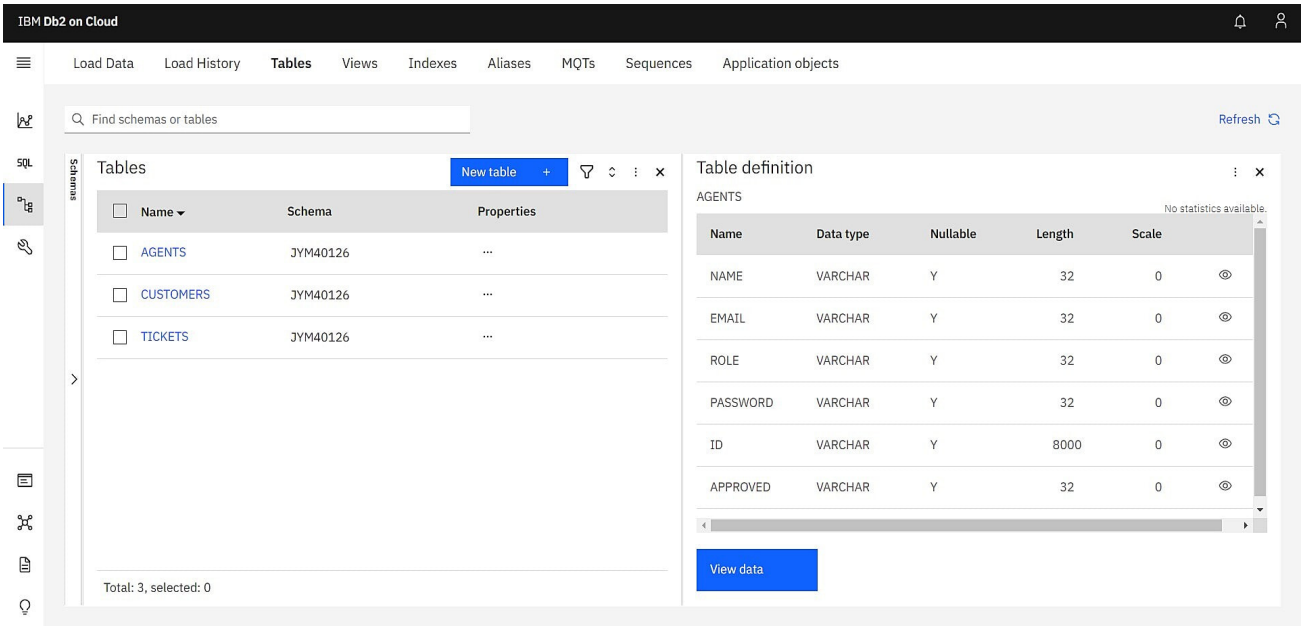


ADMIN PAGE

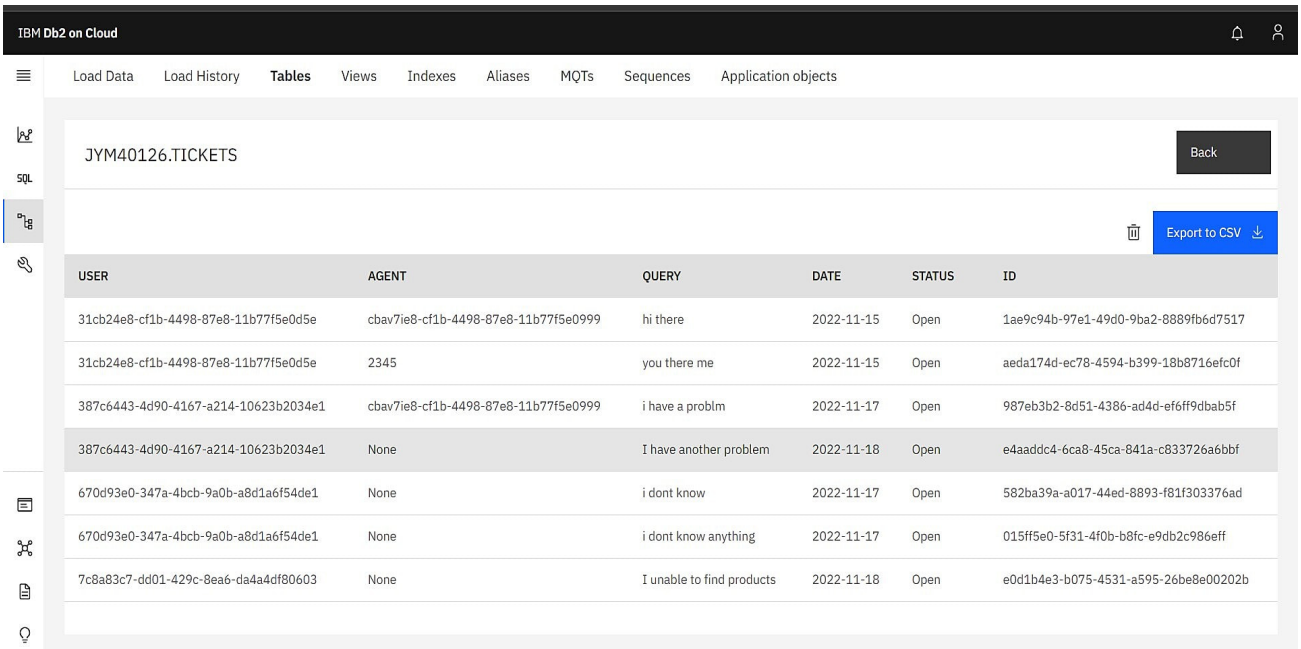


7.3 DATABASE SCHEMA

TABLES



QUERY DATA



USER	AGENT	QUERY	DATE	STATUS	ID
31cb24e8-cf1b-4498-87e8-11b77f5e0d5e	cbav7ie8-cf1b-4498-87e8-11b77f5e0999	hi there	2022-11-15	Open	1ae9c94b-97e1-49d0-9ba2-8889fb6d7517
31cb24e8-cf1b-4498-87e8-11b77f5e0d5e	2345	you there me	2022-11-15	Open	aeda174d-ec78-4594-b399-18b8716efc0f
387c6443-4d90-4167-a214-10623b2034e1	cbav7ie8-cf1b-4498-87e8-11b77f5e0999	i have a problm	2022-11-17	Open	987eb3b2-8d51-4386-ad4d-ef6ff9dbab5f
387c6443-4d90-4167-a214-10623b2034e1	None	I have another problem	2022-11-18	Open	e4aaddc4-6ca8-45ca-841a-c833726a6bbf
670d93e0-347a-4bcb-9a0b-a8d1a6f54de1	None	i dont know	2022-11-17	Open	582ba39a-a017-44ed-8893-f81f303376ad
670d93e0-347a-4bcb-9a0b-a8d1a6f54de1	None	i dont know anything	2022-11-17	Open	015ff5e0-5f31-4f0b-b8fc-e9db2c986eff
7c8a83c7-dd01-429c-8ea6-da4a4df80603	None	I unable to find products	2022-11-18	Open	e0d1b4e3-b075-4531-a595-26be8e00202b

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
User_Page_TC_OO1	Functional	USER PAGE	Verify user is able to see the Show Complaint popup when user clicked on popup	1.Enter URL and click go 2.Scroll down 3.Verify login/Singup popup displayed or not	http://169.51.204.215:30106/	Show Complaint popup should display	Working as expected	PASS	Successfull			ANNAMALALS
User_Page_TC_OO2	UI	USER PAGE	Verify the User has No Complaint	Click on the Url and go to user page by giving Correct Credentials	http://169.51.204.215:30106/	No Complaint should shown	Working as expected	PASS	Successful			RITCHARD.M
User_Page_TC_OO3	UI	USER PAGE	Verify User Total Complaint is Zero	Click on the Url and go to user page by giving Correct Credentials	http://169.51.204.215:30106/	Total Number of Complaint is Zero	Working as expected	PASS	Successful			ABIDARSH MOHAN
Agent_Login_TC_O4	UI	AGENT Login	Visible for text field for enter email id	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and seen your textfields	http://169.51.204.215:30106/	Text Fields for Email in Agent Page	Working as expected	PASS	Successful			RITCHARD.M
LoginPage_TC_O5	UI	USER Login	Visible for text field for enter email id	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and seen your textfields	http://169.51.204.215:30106/	Text Fields for Email in Agent Page	Working as expected	PASS	Successful			SRINATILA ABIDARSH MOHAN
Agent_Login_TC_O6	Functional	AGENT Login	Visible for Password on Forgot Password	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the Agent Forgot Page after verification Password should Visible	http://169.51.204.215:30106/	Password should Visible	Working as expected	PASS	Successful			ANNAMALALS

8.2 USER ACCEPTANCE TESTING

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 ornot	http://169.51.204.215:30106/	Login/Signup popup should display	Working as expected	PASS	Successfull	Y		ANNAMALALS
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	1.Enter URL and click go 2.Click on Signp 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://169.51.204.215:30106/	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	Y		RITCHARD.M ANNAMALALS
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1.Enter URL(https://shopenzer.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: 5342 password: Testing123	User should navigate to user account homepage	Working as expected	PASS	Successul	Y		SRINATH.A ABIDARSH.MOHAN
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) 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: 5342 password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful	Y		SRINATH.A ABIDARSH.MOHAN
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) 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: 5342 password: Testing12367868 6786876876	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful	Y		RITCHARD.M
LoginPage_TC_OO6	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) 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: 5342 password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful	Y		ANNAMALALS

9. RESULTS

9.1 PERFORMANCE METRICES

10 . ADVANTAGES&DISADVANTAGES

ADVANTAGES

- It retains the customer
- Gets you more references
- Increases profitability
- Gives you and your employees confidence
- Creates a holistic marketing scenario
- Competitive advantage

- Boost Customer Loyalty
- Enhance Brand Reputation
- Improve Products, Services, Procedures and Staff

DISADVANTAGES

• Higher staff wages from hiring employees who are experts in customer service.

• Paying for staff training.

• The extra services offered, such as refreshments.

• Higher wage costs from the extra time staff take to provide post-sales service.

• It can be particularly difficult for small businesses to cope with these costs

11. CONCLUSION

In conclusion, customer care, involves the use of basic ethics and any company who wants to have success and grow, needs to remember, that in order to do so, it must begin with establishing a code of ethics in regards to how each employee is to handle the dealing with customers. Customers 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 customercare. People need to treat others with respect and kindness; people should try to 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. Thereby,the customer care registry wouldbe far helpful and approachable. It offers easy tracking, recording and notificationthan any other means.

12 . FUTURE SCOPE

Machine learning (ML), emerging customer service trends 2022 can help businesses in improving overall CX. Chat applications powered by AI are trending. Large companies, as well as startups, are leveraging this to reduce costs and improve service for customers. Predictive analytics has particularly proved to be very useful. Through this, quarries that will result in a call for assistance can be predicted easily. Implementing ML in customer service trends will give you a significant difference in business growth.

13 . APPENDIX

MAIN.PY

```
from flask import Flask,render_template,redirect,url_for,request,session,flash,Blueprint

from re import fullmatch

from uuid import uuid4

from datetime import date

from flask_mail import Mail, Message

import ibm_db


conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=
SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=jym40126;PWD=1FiBds6gspiPRar
V","","")



app = Flask(__name__)

# instantiate the mail class
```

```
mail = Mail(app)
app.secret_key = 'hello'

from customer import customer
from agent import agent
from admin import admin

app.register_blueprint(customer)
app.register_blueprint(agent)
app.register_blueprint(admin)

@app.route('/home')
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/about')
def about():
    return render_template('about.html')

# configuration of mail
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'abidarsh75@gmail.com'
app.config['MAIL_PASSWORD'] = 'rkwm mrxa rnil glrp'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
```

```
@app.route("/mail")
def send_mail(header,recipient,content):
   msg = Message(
              header,
              sender ='noreply@email.com',
              recipients = [recipient]
             )
   msg.body = content
   mail.send(msg)
   return 'Sent'

@app.route('/tickets/query/mail',methods=['GET','POST'])
def query_mail():

    if request.method == 'GET':
        params = request.args.get('id')
        print(params)

        sql = 'SELECT * FROM tickets WHERE ID=?'
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,params)
        ibm_db.execute(stmt)
        tickets = ibm_db.fetch_assoc(stmt)

        print(tickets)
```

```
    return render_template('agent_mail.html',ticket=tickets)

elif request.method == 'POST':
    status = 'close'
    _id = request.form['id']
    answer = request.form['reply']
    receiver = request.form['to']
    subject = request.form['qn']
    try:
        send_mail(subject, receiver, subject)
    except:
        return 'Mail not sent'
    else:
        sql = 'UPDATE tickets SET STATUS=? WHERE ID=?'
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,status)
        ibm_db.bind_param(stmt,2,_id)
        ibm_db.execute(stmt)

        ticket = []
        status = 'open'
        sql = 'SELECT * FROM tickets WHERE STATUS=?'
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,status)
        ibm_db.execute(stmt)
        temp = ibm_db.fetch_assoc(stmt)
        while temp:
```

```
            ticket.append(temp)
            temp = ibm_db.fetch_assoc(stmt)
        print(ticket)
    return redirect(url_for('agent.agent_tickets',tickets=ticket))



if __name__ == '__main__':
    app.run(debug=True)
```

CUSTOMER.PY

```
from flask import Flask,render_template,redirect,url_for,request,session,flash,Blueprint
from re import fullmatch
from uuid import uuid4
from datetime import date
import ibm_db

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=
SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=jym40126;PWD=1FiBds6gspiPRar
V","","")

customer = Blueprint('customer',__name__)

@customer.route('/signin',methods=['GET','POST'])
def customer_signin():
    if request.method == 'GET':
        return render_template('signin.html')
```

```
elif request.method == 'POST':
    _id = str(uuid4())
    name = request.form['name']
    email = request.form['email']
    role = 'customer'
    password = request.form['password']
    confirm_password = request.form['confirm_password']

    regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
    special_chars = ('!','@','$','%','(',')','*','-','+','?','/','\\','.',',')
    flag = True

    if name=="" or email=="" or password=="" or confirm_password=="" or role==":
        flash('Fill empty details please!!!!')
        flag = False

    if not(fullmatch(regex,email)):
        flash('Enter valid email')
        flag = False

    if password != confirm_password:
        flash('Passwords doesn\'t match!!!')
        flag = False

    if not any(x.isupper() for x in password):
        flash('Use an uppercase in password')
        flag = False
```

```
if not any(x.isdigit() for x in password):
    flash('Use a number in password')
    flag = False

if not any(x in special_chars for x in password):
    flash('Use special characters in password!')
    flag = False

if flag:
    sql = "SELECT * FROM customers WHERE EMAIL=?"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    if account:
        flash('Email already exists,login please')
        return redirect(url_for('customer.login'))
    else:
        sql = 'INSERT INTO customers VALUES(?,?,?,?,?)'
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,name)
        ibm_db.bind_param(stmt,2,email)
        ibm_db.bind_param(stmt,3,role)
        ibm_db.bind_param(stmt,4,password)
        ibm_db.bind_param(stmt,5,_id)
        ibm_db.execute(stmt)
```

```
            flash('Account created successfully!!!!')
            return redirect(url_for('customer.customer_login'))
        else:
            return redirect(url_for('customer.customer_signin'))

@customer.route('/login',methods=['GET','POST'])
def customer_login():
    if request.method == 'GET':
        return render_template('login.html')
    elif request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        role = 'customer'

        sql = 'SELECT * FROM customers WHERE EMAIL=? AND PASSWORD=? AND
ROLE=?'
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.bind_param(stmt,3,role)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        session['user'] = account
        print(session['user'])
        if account:
            flash('logged in successfully')
            return render_template('customer_dashboard.html',name=session['user']['NAME'])
```

```
        else:
            flash('Wrong account credentials')
            return redirect(url_for('customer.customer_login'))

@customer.route('/profile',methods=['GET','POST'])
def profile():
    if request.method == 'GET':
        if session['user']:
            cur_user = session['user']
            print(cur_user)
            return render_template('customer_profile.html',user=cur_user)
        else:
            return redirect(url_for('customer.customer_login'))

@customer.route('/create_ticket',methods=['GET','POST'])
def create_ticket():

    if request.method == 'GET':
        return render_template('customer_create_ticket.html')

    elif request.method == 'POST':
        user = session['user']['ID']
        agent = 'None'
        query = request.form['query']
        date1 = str(date.today())
        _id = str(uuid4())
        status = 'Open'
```

```
        print(date1)

        if query == '':
            flash('Query cannot be empty')
            return redirect(url_for('customer.create_ticket'))
        else:
            sql = 'INSERT INTO tickets VALUES(?,?,?,?,?,?)'
            stmt = ibm_db.prepare(conn,sql)
            ibm_db.bind_param(stmt,1,user)
            ibm_db.bind_param(stmt,2,agent)
            ibm_db.bind_param(stmt,3,query)
            ibm_db.bind_param(stmt,4,date1)
            ibm_db.bind_param(stmt,5,status)
            ibm_db.bind_param(stmt,6,_id)
            ibm_db.execute(stmt)
            flash('Ticket created successfully')
            return redirect(url_for('customer.create_ticket'))

@customer.route('/tickets')
def tickets():
    ticket = []
    _id = session['user']['ID']
    sql = 'SELECT * FROM tickets'
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.execute(stmt)
    temp = ibm_db.fetch_assoc(stmt)
    while temp:
```

```
        if temp['USER'] == _id:
            ticket.append(temp)
        temp = ibm_db.fetch_assoc(stmt)
    print(len(ticket))
    return render_template('customer_tickets.html',tickets=ticket)

@customer.route('/change_password',methods=['GET','POST'])
def change_password():

    if request.method == 'GET':
        return render_template('customer_password.html')

    elif request.method == 'POST':
        password = request.form['password']
        confirm_password = request.form['confirm_password']
        _id = session['user']['ID']

        flag = True
        regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
        special_chars = ('!','@','$','%','(',')','*','-','+','?','/','\\','.',',')

        if password == '' or confirm_password == '':
            flag = False
            flash('Passwords cannot be empty')

        if password != confirm_password:
            flag = False
```

```
    flash('Passwords donot match')

if password == session['user']['PASSWORD']:
    flag = False
    flash('Same as old password!!!')

if not any(x.isupper() for x in password):
    flash('Use an uppercase in password')
    flag = False

if not any(x.isdigit() for x in password):
    flash('Use a number in password')
    flag = False

if not any(x in special_chars for x in password):
    flash('Use special characters in password!')
    flag = False

if flag:
    sql = 'UPDATE customers SET PASSWORD=? WHERE ID=?'
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,password)
    ibm_db.bind_param(stmt,2,_id)
    ibm_db.execute(stmt)
    flash('Password updated successfully')
    return redirect(url_for('customer.customer_logout'))
else:
```

```
        return redirect(url_for('customer.change_password'))


@customer.route('/logout')
def customer_logout():
    session.pop('user')
    return redirect(url_for('customer.customer_login'))
```

ADMIN.PY

```
from flask import Flask,render_template,redirect,url_for,request,session,flash,Blueprint
from re import fullmatch
from uuid import uuid4
from datetime import date
from customer import customer
import ibm_db
import json as JSON


conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=
SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=jym40126;PWD=1FiBds6gspiPRar
V","","")


admin = Blueprint('admin',__name__)


@admin.route('/admin',methods=['GET','POST'])
def admin_login():
        if request.method == 'GET':
                return render_template('admin_login.html')
        elif request.method == 'POST':
```

```
        email = request.form['email']
        password = request.form['password']

        if email == 'abidarsh75@gmail.com' and password == '12345':
            session['admin'] = {'email':'a@gmail.com'}
            return render_template('admin_dashboard.html')
        else:
            flash('wrong credentials!!!')
            return redirect(url_for(admin.admin_login))

@admin.route('/admin/tickets')
def admin_tickets():
    sql = 'SELECT * FROM tickets'
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.execute(stmt)
    ticket = ibm_db.fetch_assoc(stmt)
    unasign_tickets = []

    while ticket:
        if ticket['AGENT'] == 'None':
            unasign_tickets.append(ticket)

        ticket = ibm_db.fetch_assoc(stmt)

    sql = 'SELECT * FROM agents'
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.execute(stmt)
```

```
	agent = ibm_db.fetch_assoc(stmt)
	approved = []

	while agent:
		if agent['APPROVED'] != 'None':
			approved.append(agent)
		agent = ibm_db.fetch_assoc(stmt)

	return render_template('admin_tickets.html',tickets=unasign_tickets,agents=approved)

@admin.route('/admin/requests')
def admin_requests():
	sql = 'SELECT * FROM agents'
	stmt = ibm_db.prepare(conn,sql)
	ibm_db.execute(stmt)
	agent = ibm_db.fetch_assoc(stmt)
	unapproved = []

	while agent:
		if agent['APPROVED'] == 'None':
			unapproved.append(agent)
		agent = ibm_db.fetch_assoc(stmt)

	print(unapproved)
	return render_template('admin_requests.html',agents = unapproved)

@admin.route('/admin/approve/<approval>/<agent>')
```

```
def admin_approve(approval,agent):
	sql = 'UPDATE agents SET APPROVED=? WHERE ID=?'
	stmt = ibm_db.prepare(conn,sql)
	ibm_db.bind_param(stmt,1,approval)
	ibm_db.bind_param(stmt,2,agent)
	ibm_db.execute(stmt)
	return 'ok'

@admin.route('/admin/agents')
def admin_agents():
	sql = 'SELECT * FROM agents'
	stmt = ibm_db.prepare(conn,sql)
	ibm_db.execute(stmt)
	agent = ibm_db.fetch_assoc(stmt)
	agents = []

	while agent:
		if agent['APPROVED'] == 'yes':
			agents.append(agent)
		agent = ibm_db.fetch_assoc(stmt)

	print(agents)
	return render_template('admin_agents.html',agents=agents)

@admin.route('/admin/logout')
def admin_logout():
	session.pop('admin')
```

```
        return redirect(url_for('customer.customer_login'))

@admin.route('/admin/assign/<ticket>/<agent>')
def admin_assign(ticket,agent):

        print(ticket,agent)
        sql = 'UPDATE tickets SET AGENT=? WHERE ID=?'
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,agent)
        ibm_db.bind_param(stmt,2,ticket)
        ibm_db.execute(stmt)
        return "ok"
```

AGENT.PY

```
from flask import Flask,render_template,redirect,url_for,request,session,flash,Blueprint
from re import fullmatch
from uuid import uuid4
from datetime import date
import ibm_db

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=
SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=jym40126;PWD=1FiBds6gspiPRar
V","","")

agent = Blueprint('agent',__name__)

@agent.route('/signin/',methods=['GET','POST'])
```

```
def agent_signin():
    if request.method == 'GET':
        return render_template('agent_signin.html')

    elif request.method == 'POST':
        _id = str(uuid4())
        name = request.form['name']
        email = request.form['email']
        role = 'agent'
        password = request.form['password']
        confirm_password = request.form['confirm_password']

        regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
        special_chars = ('!','@','$','%','(',')','*','-','+','?','/','\\','.',',')
        flag = True

        if name==" or email==" or password==" or confirm_password==" or role==":
            flash('Fill empty details please!!!!')
            flag = False

        if not(fullmatch(regex,email)):
            flash('Enter valid email')
            flag = False

        if password != confirm_password:
            flash('Passwords doesn\'t match!!!')
            flag = False
```

```
if not any(x.isupper() for x in password):
    flash('Use an uppercase in password')
    flag = False

if not any(x.isdigit() for x in password):
    flash('Use a number in password')
    flag = False

if not any(x in special_chars for x in password):
    flash('Use special characters in password!')
    flag = False

    sql = "SELECT * FROM agents WHERE EMAIL=?"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    if account:
        flash('Email already exists,login please')
        return redirect(url_for('agent.agent_login'))
    else:
        sql = 'INSERT INTO agents VALUES(?,?,?,?,?,?)'
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,name)
        ibm_db.bind_param(stmt,2,email)
        ibm_db.bind_param(stmt,3,role)
```

```
            ibm_db.bind_param(stmt,4,password)
            ibm_db.bind_param(stmt,5,_id)
            ibm_db.bind_param(stmt,6,'None')
            ibm_db.execute(stmt)
            flash('Account created successfully!!!!')
            return redirect(url_for('agent.agent_login'))
        else:
            return redirect(url_for('agent.agent_signin'))

@agent.route('/login/',methods=['GET','POST'])
def agent_login():
    if request.method == 'GET':
        return render_template('login.html')
    elif request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        role = 'agent'

        sql = 'SELECT * FROM agents WHERE EMAIL=? AND PASSWORD=? AND ROLE=?'
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.bind_param(stmt,3,role)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        session['agent'] = account
        print(session['agent'])
```

```
        if account:
            if account['APPROVED'] == 'yes':
                print('here1')
                return render_template('agent_dashboard.html')
            elif account['APPROVED'] == 'None':
                print('here2')
                return "Your request is under process"
            elif account['APPROVED'] == 'no':
                print('here3')
                return "Sorry your request has been rejected"
        else:
            flash('Wrong account credentials')
            return redirect(url_for('agent.agent_login'))

@agent.route('/profile/',methods=['GET','POST'])
def agent_profile():
    if request.method == 'GET':
        if session['agent']:
            cur_user = session['agent']
            print(cur_user)
            return render_template('agent_profile.html',user=cur_user)
        else:
            return redirect(url_for('agent.agent_login'))

@agent.route('/tickets/')
def agent_tickets():
    ticket = []
```

```
    _id = session['agent']['ID']
    sql = 'SELECT * FROM tickets'
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.execute(stmt)
    temp = ibm_db.fetch_assoc(stmt)
    while temp:
        if temp['AGENT'] == _id:
            ticket.append(temp)
        temp = ibm_db.fetch_assoc(stmt)
    print(len(ticket))
    return render_template('agent_tickets.html',tickets=ticket)

@agent.route('/change_password/',methods=['GET','POST'])
def agent_change_password():

    if request.method == 'GET':
        return render_template('agent_password.html')

    elif request.method == 'POST':
        password = request.form['password']
        confirm_password = request.form['confirm_password']
        _id = session['agent']['ID']

        flag = True
        regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
        special_chars = ('!','@','$','%','(',')','*','-','+','?','/','\\','.',',')
```

```
if password == '' or confirm_password == '':
    flag = False
    flash('Passwords cannot be empty')

if password != confirm_password:
    flag = False
    flash('Passwords donot match')

if password == session['agent']['PASSWORD']:
    flag = False
    flash('Same as old password!!!')

if not any(x.isupper() for x in password):
    flash('Use an uppercase in password')
    flag = False

if not any(x.isdigit() for x in password):
    flash('Use a number in password')
    flag = False

if not any(x in special_chars for x in password):
    flash('Use special characters in password!')
    flag = False

if flag:
    sql = 'UPDATE agents SET PASSWORD=? WHERE ID=?'
    stmt = ibm_db.prepare(conn,sql)
```

```
        ibm_db.bind_param(stmt,1,password)
        ibm_db.bind_param(stmt,2,_id)
        ibm_db.execute(stmt)
        flash('Password updated successfully')
        return redirect(url_for('agent.agent_logout'))
    else:
        return redirect(url_for('agent.change_password'))

@agent.route('/logout/')
def agent_logout():
    session.pop('agent')
    return redirect(url_for('agent.agent_login'))
```

LINKS :

GITHUB LINK : IBM-EPBL/IBM-Project-6774-1658836915

DEMO LINK : https://www.dropbox.com/s/ie6nzx5lho7wo3f/Customer%20care%20registry.mp4?dl=0