TEAM ID: PNT2022TMID34000

IBM-Project-40567-1660631493
CUSTOMER CARE REGISTRY

TEAM DETAILS:

Team ID **:** PNT2022TMID34000

College Name : ARUNACHALACOLLEGE OF ENGINEERING FOR WOMEN

Department **:** COMPUTER SCIENCE & ENGINEERING

TEAM MEMBERS :

B.FARHANA(960219104042)
T.BHUVANESHWARI(960219104034)
A.K.AFANA FATHIMA(960219104006)
R.AJISHA(960219104009)

TEAM ID: PNT2022TMID34000

CONTENT

TEAM ID: PNT2022TMID34000

7. CODING & SOLUTIONING

a. Feature 1

b. Feature 2

8.RESULTS
9.ADVANTAGES & DISADVANTAGES
10.CONCLUSION

TEAM ID: PNT2022TMID34000

1.INTRODUCTION

INTRODUCTION TO PROJECT

The Customer Service Desk is a web based project.

Customer Service also known as Client Service is the provision of service to customers' Its significance varies by product, industry and domain. In many cases customer services is more important if the information relates to a service as opposed to a Customer.

Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition.

PURPOSE OF THE PROJECT

An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through and e-mail. The system should have capability to integrate with any Service Provider from any domain or industry like Banking. Telecom Insurance. etc.

Customer Service also known as Client Service is the provision of service to customers Its significance varies by product industry and domain. In many cases customer services is more important if the information relates to a service as opposed to as Customer

Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition

TEAM ID: PNT2022TMID34000

2.LITERATURE SURVEY

Customer Care Registry implementing on Web development based on Cloud Application

Development.

ABSTRACT

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.

ADVANTAGES

Improves customer services:

It makes it possible to develop a more personal and close relationship with customers and to offer them a personalized service, that fulfills their expectations, promoting affective ties between the customer and the company.

It promotes more effective marketing strategies:

Through the management and collection of data on customers, it is easier to discover their tastes, needs, expectations, and opinions. This information will help you design personalized strategies, adapted to your customers, which will be far more effective and profitable.

Increases sales volume and revenue:

The design of marketing campaigns tailored to customers translates into higher sales and, as a rule, more revenue.

Promotes more efficient communication within the company:

TEAM ID: PNT2022TMID34000

This solution promotes the creation of more effective communication channels between departments, which allows a better understanding of what is happening in each section involved and the detection and analysis of possible incidents that may arise.

Increases customer loyalty:

Just by meeting customer needs and expectations, we could be able to gain their retention and loyalty.

DISADVANTAGES

The price:

Some customer care registry plans are very expensive, even investing in a custom customer care registry software requires a significant investment but there are many free ones that even though they may not totally fit the needs of your company, we suggest you try them to know exactly which are the features that work best for you to build your own once you have the budget for it.

Training:

With a variety of options available, it is common for some customer care registrys to be more complex than others so they require some training in order to get the most out of them. The problem is that getting familiar with these programs requires a considerable amount of time and not everyone is willing to make that commitment.

Use:

If you are not going to use it then don't waste your time and money, just don't buy it. A customer care registry must be useful and for that.

Change:

Not everyone likes change, especially when it comes to their workflow, so by introducing a customer care registry to your company's processes, not everyone will be able to swiftly adapt to the new changes.

EXISTING SYSTEM

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.

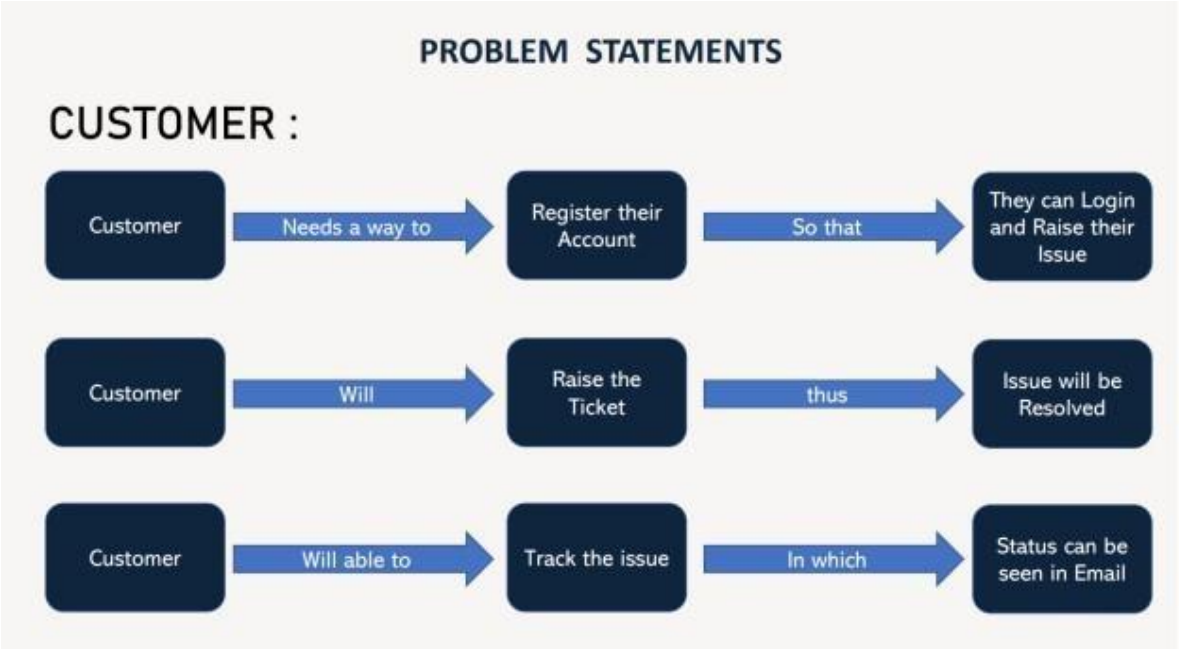
References

1. help desk

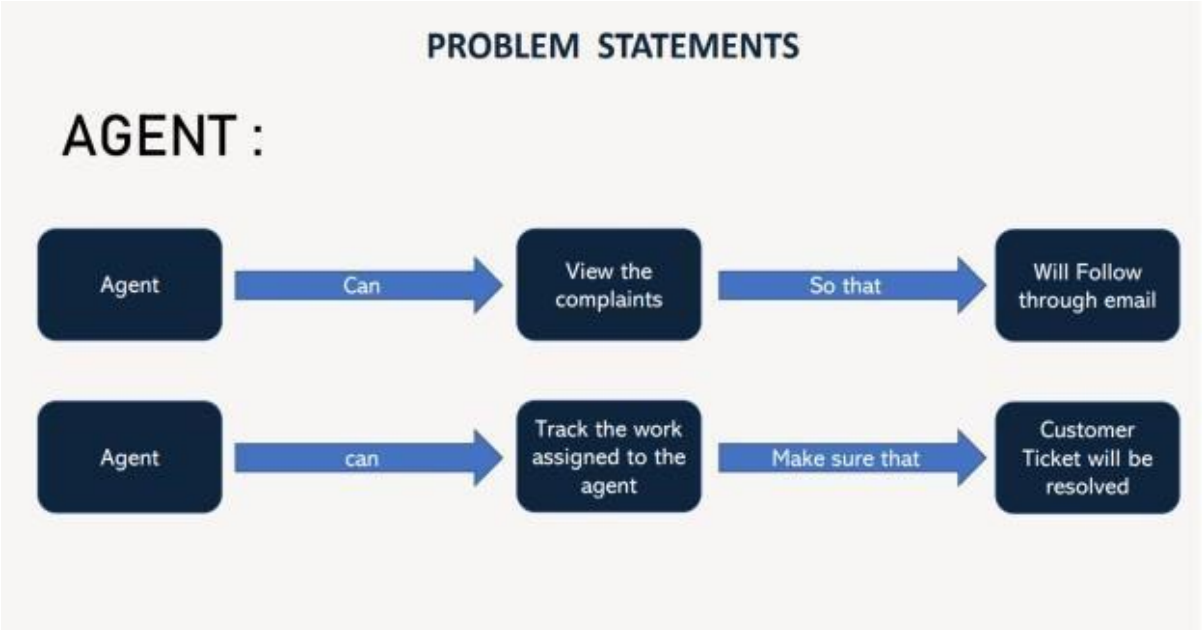
2. live chat box support

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.



TEAM ID: PNT2022TMID34000

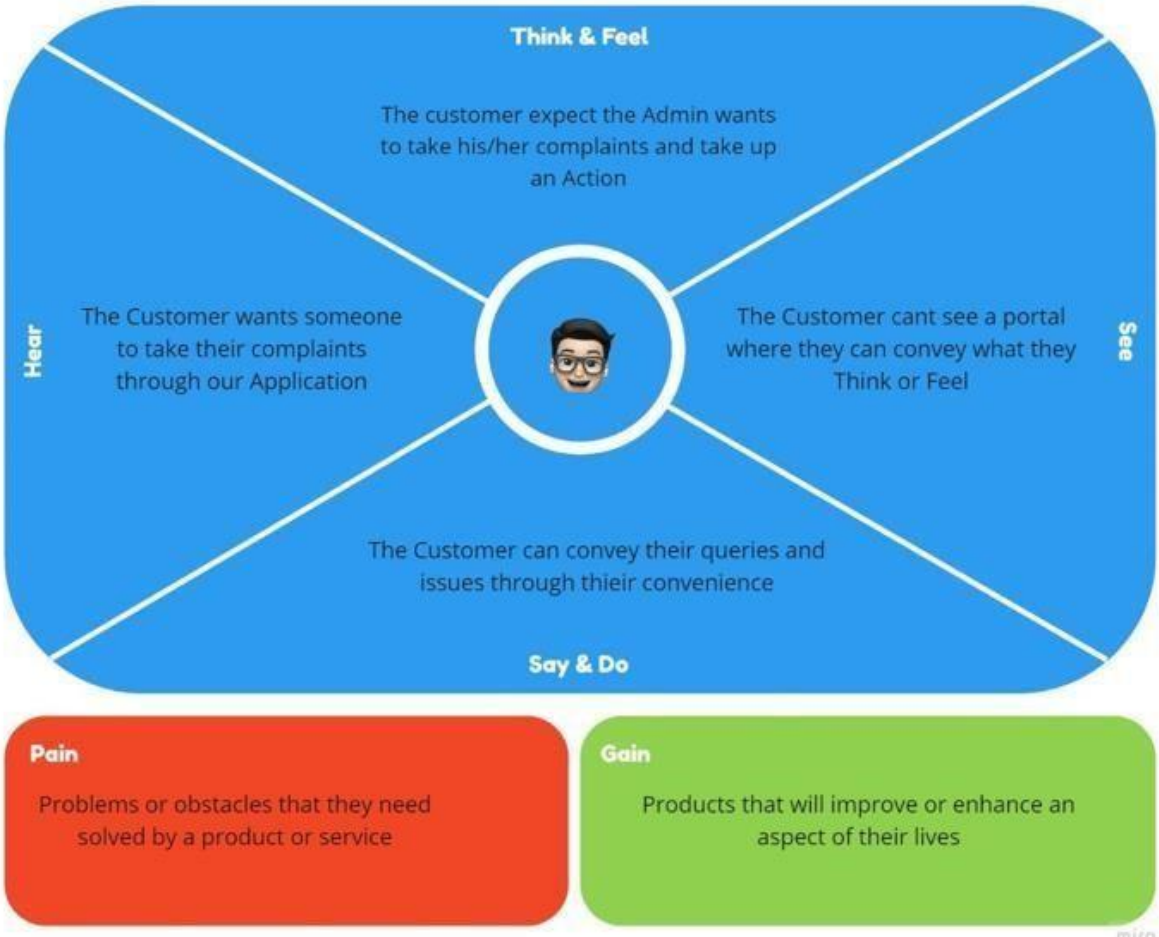


TEAM ID: PNT2022TMID34000

3.IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

An empathy map is **a collaborative tool teams can use to gain a deeper insight into their customers**. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.



TEAM ID: PNT2022TMID34000

Ideation & Brainstorming

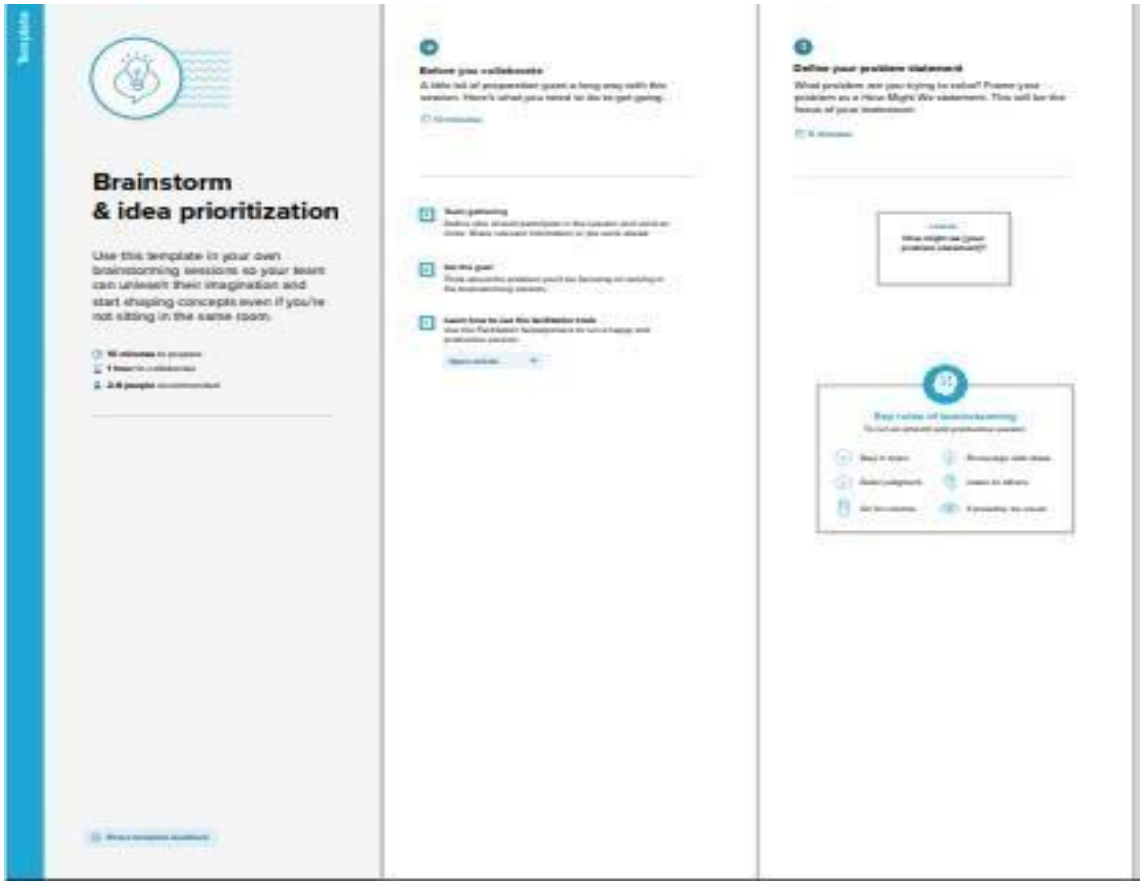
Ideation and the practise of brainstorming, a particular method for coming up with fresh ideas, are frequently closely related. The main distinction between ideation and brainstorming is that whereas brainstorming is nearly often done in groups, ideation is typically seen as being more of a solitary endeavour. A group of people are frequently gathered for a brainstorming session to generate either fresh, general ideas or solutions to specific problems or circumstances.

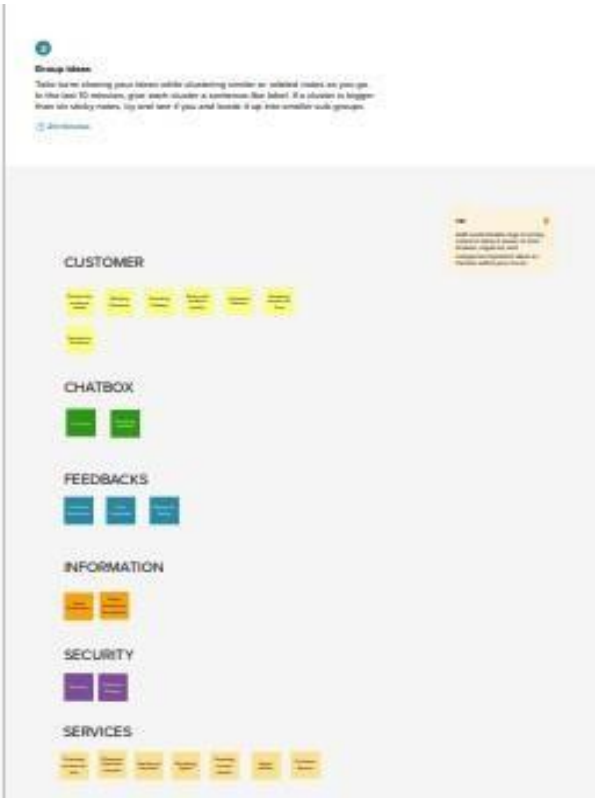
On instance, a large firm that has discovered it is the target of a significant lawsuit might wish to consult with its top executives to come up with ideas for how to publicly respond to the case being filed.

In a brainstorming session, participants are encouraged to freely share any ideas that may come to mind. According to the theory, by coming up with a lot of ideas, the brainstorming group is more likely to find a workable solution to the problem they are trying to solve.

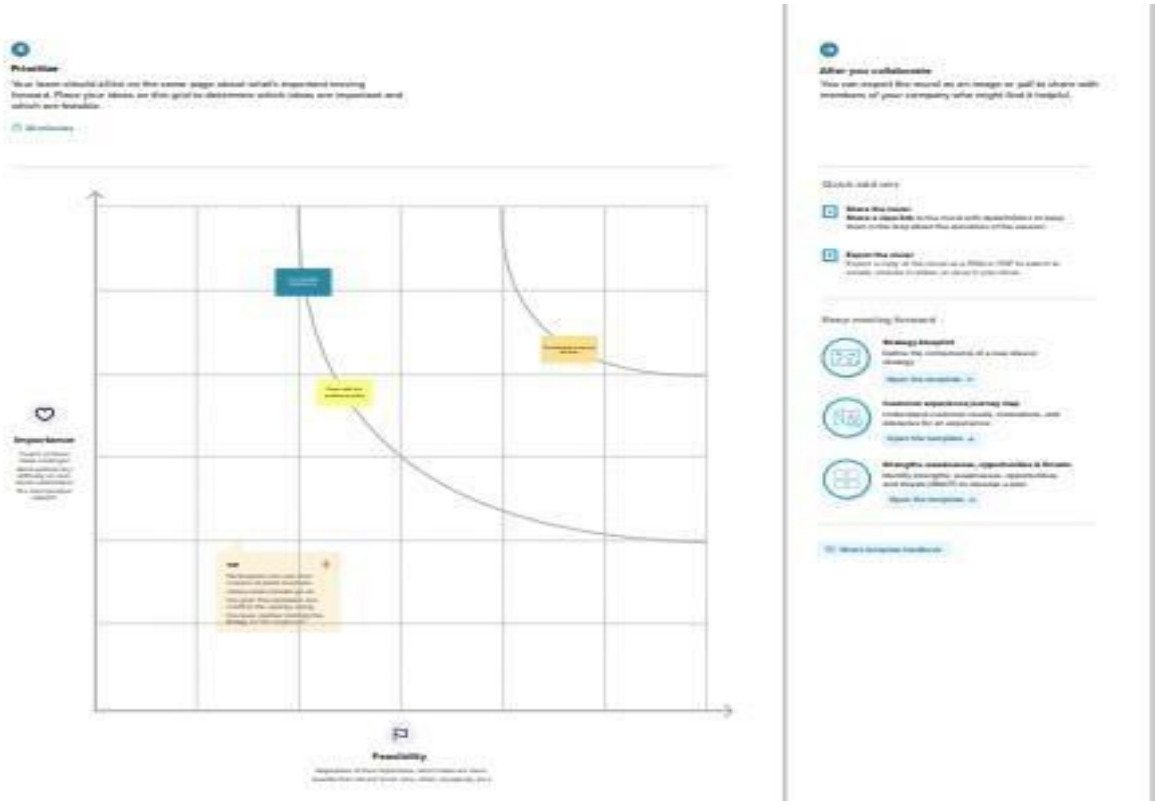
With the creation of various brainstorming software tools, such Brightidea and Idea wake, the distinction between ideation and brainstorming has gotten a little bit more hazy. These software applications are made to inspire staff members to come up with fresh suggestions for enhancing business operations and, eventually, bottom-line profitability. The applications frequently mix the ideation and brainstorming processes in that they can be used by individual employees, but businesses can replicate brainstorming sessions by having multiple employees use the software to produce fresh ideas for a particular problem.

TEAM ID: PNT2022TMID34000





TEAM ID: PNT2022TMID34000



Proposed Solution

TEAM ID: PNT2022TMID34000

S.NO	PARAMETER	DESCRIPTION
01	Problem Statement	To solve customer issues using Cloud Application Development.
02	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.
03	Novelty / Uniqueness	Assigned Agent Routing, Automated Ticket Closure, Status Shown to the Customer, and Backup data in case of failures.

S.NO	PARAMETER	DESCRIPTION
04	Social Impact & Customer Satisfaction	Customer Satisfaction, Customer can track their status and Easy agent communication.
05	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, Offices
06	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

Problem Solution fit

Problem-Solution Fit - this occurs when you have evidence that customers care about certain jobs, pains, and gains. At this stage you've proved the existence of a problem and have designed a value proposition that addresses your customers' jobs, pains and gains. Unfortunately you still do not have clear evidence that your customer really care enough about your value proposition enough to buy it.

TEAM ID: PNT2022TMID34000

TEAM ID: PNT2022TMID34000

4.REQUIREMENT ANALYSIS

What is Requirement Analysis?

It is the process of determining user expectations for a system under consideration. These should be quantifiable and detailed.

Requirement Analysis:

- Serves as a foundation for test plans and project plan
- Serves as an agreement between developer and customer
- Process to make stated and unstated requirements clear
- Process to validate requirement for completeness, ambiguity and feasibility

Functional requirement

Functional requirements specify what a system should be able to do through computations, technical details, data manipulation and processing, and other specialised functions. Use cases, which are used to represent behavioural requirements, explain all the instances in which the system makes use of the functional requirements. Non-functional requirements, commonly referred toas "quality requirements," which place restrictions on the design or execution, support functional requirements (such as performance requirements,security,or reliability). Non-functional requirements often take the form "system shall be," while functional needs are typically articulated in the form "system must do." While non-functional needs are defined in the system architecture, the plan for accomplishing functional requirements is detailed in the system design. Functional requirements, as used in requirements engineering, outline specified outcomes of a system

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it’s important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behaviour under specific conditions.

For example:
The system sends an approval request after the user enters personal information.

A search feature allows a user to hunt among various invoices if they want to credit an issued invoice.

The system sends a confirmation email when a new user account is created

Non-functional requirements

In general, non-functional requirements outline what a system is supposed tobe rather than what it should be able to perform. Functional requirements are typically expressed as "system shall do," an individual action or component of the system, maybe explicitly in terms of a mathematical function, or as a black box description of an input, output, process, and control functional model, also known as an IPO Model. Non-functional requirements, on the other hand, have the form of "system shall be," which refers to a general characteristic of the system as a whole or of a particular aspect rather than a specific function. The overall characteristics of the system frequently determine whether a development project is a success or a failure

Non-functional requirements, not related to the system functionality, rather define how the system should perform. Some examples are:

- The website pages should load in 3 seconds with the total number of simultaneous users <5 thousand.

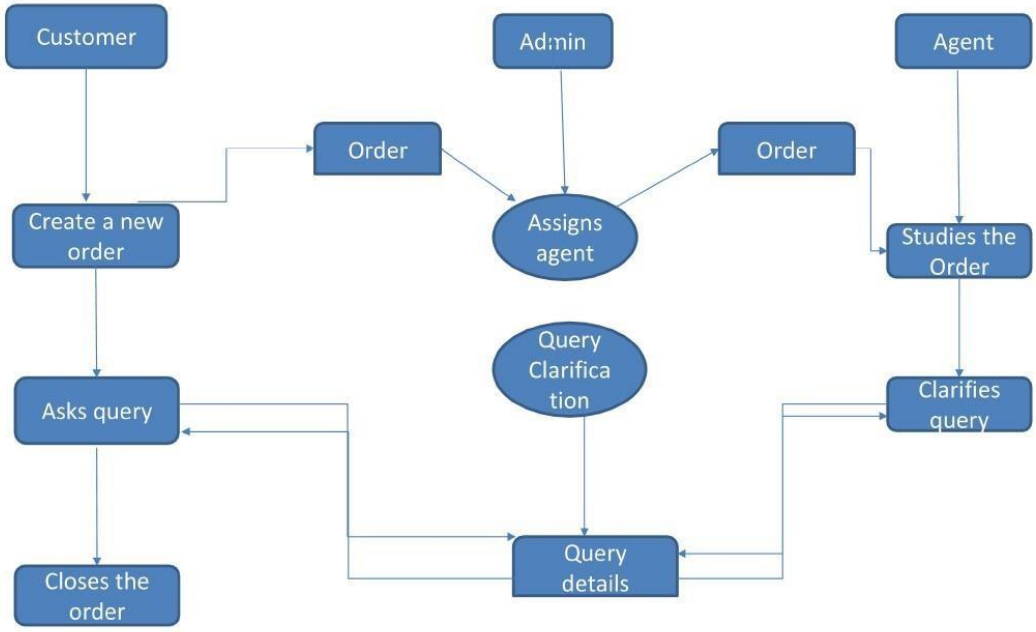
- The system should be able to handle 20 million users without performance deterioration.

- Here’s a brief comparison and then we’ll proceed to a more in-depth explanation of each group

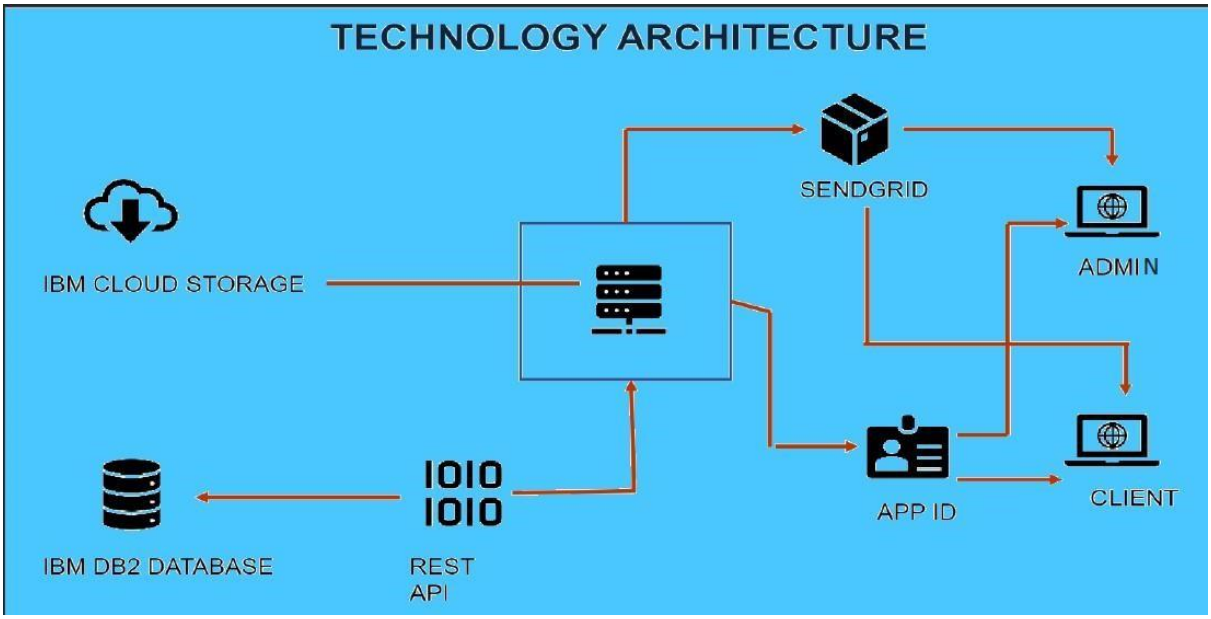
TEAM ID: PNT2022TMID34000

5.PROJECT DESIGN

Data Flow Diagrams



Solution & Technical Architecture



TECHNOLOGY ARCHITECTURE

TEAM ID: PNT2022TMID34000

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
8	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration	Local, Cloud Foundry, Kubernetes, etc...

APPLICATION CHARATERISTICS

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
8	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration	Local, Cloud Foundry, Kubernetes, etc...

USER STORIES

TEAM ID: PNT2022TMID34000

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

6. PROJECT PLANNING & SCHEDULING

'Project Planning and Scheduling', though separate, are two sides of the same coin in project management. Fundamentally, 'Project planning' is all about choosing and designing effective policies and methodologies to attain project objectives.

While 'Project scheduling' is a procedure of assigning tasks to get them completed by allocating appropriate resources within an estimated budget and time-frame.

The basis of project planning is the entire project. Unlikely, project scheduling focuses only on the project-related tasks, the project start/end dates and project dependencies.

Thus, a 'project plan' is a comprehensive document that contains the project aims, scope, costing, risks, and schedule. And a project schedule includes the estimated dates and sequential project tasks to be executed.

Project Planning

- Developing a project to make it ready for investment
- Determines the jobs/tasks required to attain project objectives

Sprint Planning & Estimation

- Sprint planning is an event in scrum that kicks off the sprint.
- The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved.
- Sprint planning is done in collaboration with the whole scrum team.
- In scrum, the sprint is a set period of time where all the work is done.

TEAM ID: PNT2022TMID34000

- However, before you can leap into action you have to set up the sprint.
- You need to decide on how long the time box is going to be, the sprint goal, and where you're going to start.
- The sprint planning session kicks off the sprint by setting the agenda and focus.
- If done correctly, it also creates an environment where the team is motivated, challenged, and can be successful.
- Bad sprint plans can derail the team by setting unrealistic expectations.

Sprint 1

HOMEPAGE
LOGIN PAGE(CUSTOMER)
ADMIN PAGE(CUSTOMER_LIST)
AGENT LOGIN PAGE

Sprint 2

HOMEPAGE
AGENT HOMEPAGE
CUSTOMER HOMEPAGE
ADMIN
WEB CHAT

Sprint 3

HOMEPAGE
CUSTOMER COMPLAINT_PAGE
CUSTOMER HOMEPAGE
ADMIN PAGE(COMPLAINT_LIST)
AGENT ALLOTMENT

Sprint 4

TESTING THE USER AND ADMIN LOGIN PAGE
TESTING THE SIGN IN AND SIGN UP
TESTING THE ALL PAGE

TEAM ID: PNT2022TMID34000

7.CODING & SOLUTIONING

College graduates with prior programming expertise or technical degrees are recruited and transitioned into professional positions with Alabama firms and organisations through the highly competitive CodingSolutions job accelerator and talent refinement programme at no cost to the graduates. We provide a pool of varied, well-trained, techs-savvy individuals that wants to launch and advance their career in Alabama.

The mission of veteran- and woman-owned CodingSolutions is to mobilise the next generation of IT talent and provide them the tools and resources they require to make your business successful. Innovative talent is necessary for innovative technologies.

We wish to provide CodingSolutions prospects to assist you expand your Alabama team.

Our applicants are swiftly hired at the top of the list by growing businesses for lucrative, long-term positions.

Features:

7 main types of customer needs:

- Friendliness
- Empathy
- Fairness
- Control
- Alternatives
- Information
- Time

TEAM ID: PNT2022TMID34000

1.Friendliness

This is the most basic customer need that's associated with things like courtesy and politeness. Friendly agents are a top indicator of a good customer experience, according to the customers surveyed in our 2021 Trends Report

2. Empathy

Customers need to know the organization understands and appreciates their needs and circumstances. In fact, 49% surveyed in our 2021 Trends Report said they want agents to be empathetic.

3. Fairness

Customers must feel that they're getting adequate attention and fair and reasonable answers.

4. Control

Customers want to feel like they have an influence on the outcome. You can empower your customers by listening to their feedback and using it to improve.

5. Alternatives

Customers want choice and flexibility from customer service; they want to know there is a range of options available to satisfy them. In fact, high-performing companies are more likely to provide customers with a choice of customer service channels. 50% of high performers have adopted an omnichannel support strategy, compared to 18% of their lower-performing peers

6. Information

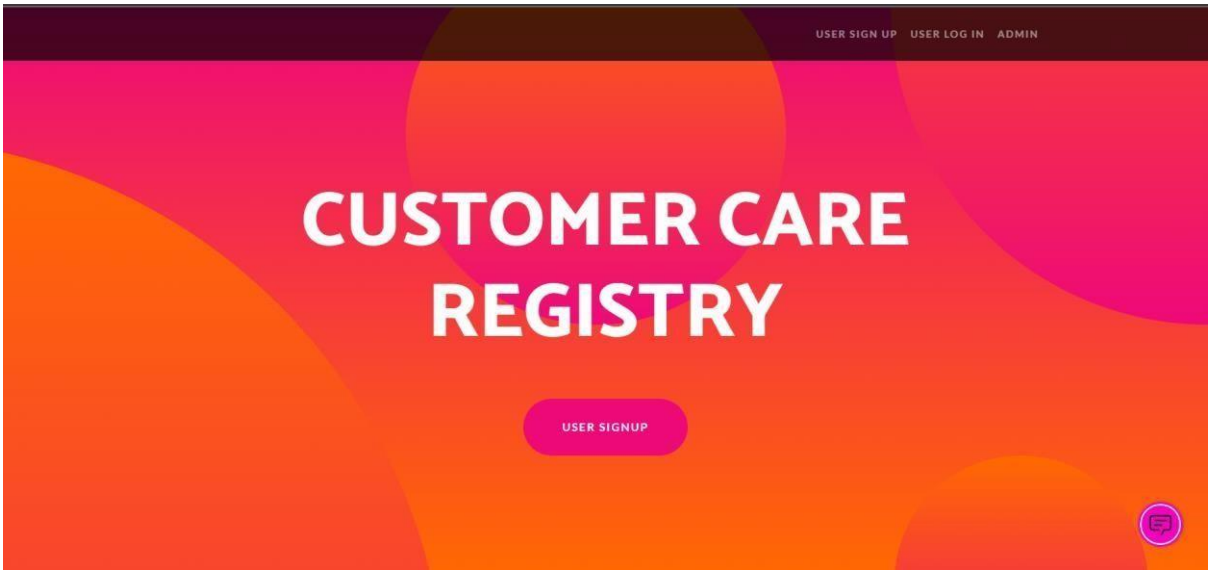
Customers want to know about products and services in a pertinent and time-sensitive manner; too much information and selling can be off-putting for them. A knowledge base is a great way to provide existing customers with the information they need, when they need it. And highperforming CX teams are more likely to offer a knowledge base, according to our research.

TEAM ID: PNT2022TMID34000

7. Time

Customers' time is valuable, and organizations need to treat it as such. 73% of customers said resolving their issues quickly is the top component of a good customer experience. To deliver on that expectation, CX teams need customer service software that arms them with tools to respond to customers quickly and effectively.

8.RESULTS



9.ADVANTAGES & DISADVANTAGES

ADVANTAGES :

1.Customer loyalty

Loyal customers have many benefits for businesses. 91% of customers say a positive customer service experience makes them more likely to make a further purchase (source: Salesforce Research). Also, investing in new customers is five times more expensive than retaining existing ones (source: Invesp). Creating loyal customers through good customer service can therefore provide businesses with lucrative long-term relationships.

2. Increase profits

These long-term customer relationships established through customer service can help businesses become more profitable. Businesses can grow revenues between 4% and 8% above their market when they prioritise better customer service experiences (source: Bain & Company). Creating a better customer service experience than those offered by competitors can help businesses to standout in their market place, and in turn make more sales.

3. Customer recommendations

Providing good customer service can create satisfied customers, who are then more likely to recommend the business to others. 94% of customers will recommend a company whose service they rate as “very good” (source: Qualtrics XM Institute). This is useful, as 90% of customers are influenced by positive reviews when buying a product (source: Zendesk). Customers recommending a company through word of mouth or online reviews can improve the credibility of the business.

4. Increase conversion

Good customer service can help businesses turn leads into sales. 78% of customers say they have backed out of a purchase due to a poor customer experience (source: Glance). It is therefore safe to assume that providing good customer service will help to increase customer confidence and in turn increase conversion.

5. Improve public image

Customer service can help businesses to improve the public perception of the brand, which can then provide protection if there is a slip up. 78% of customers will forgive a company for a mistake after receiving excellent service (source: Salesforce Research). Meanwhile, almost 90% of customers report trusting a company whose service they rate as “very good.” On the other hand, only 16% of those who give a “very poor” rating trust companies to the same degree(source: Qualtrics XM Institute). Creating positive customer experiences is vital in gaining customer trust and creating a strong public image.

Disadvantage :

The Consumer Protection Act in India has numerous restrictions and drawbacks, which are listed in this article.

Only services for which a particular payment has been made are covered under the consumer protection act. However, it does not protect medical professionals, or hospitals, and covers cases when this act does not apply to free medical care.

This act does not apply to mandatory services, such as water supply, that are provided by state agencies.

Only two clauses related to the supply of hazardous materials are covered by this act. Consumer redress is not given any power by the consumer protection act.

The consumer protection act focuses on the supply of ineffective products, but there are no strict regulations for those who produce it.

10.CONCLUSION

It is a web-enabled project.

- With this project the details about the product will be given to the customers in detail with in a short span of time.
- Queries regarding the product or the services will also be clarified.
- It provides more knowledge about the various technologies.

11.APPENDIX

Source code:

```
from __future__ import print_function
from audioop import add
import datetime
from unicodedata import name
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 datetime conn =
ibm_db.connect("DATABASE=;HOSTNAME=;PORT=;SECURITY;=SSL;SSLServerCertificat e=;UID=;PWD=", '',
'')
print(conn)
print("connection successful...")
app = Flask(__name__)
app.secret_key = 'your secret key'
@app.route('/')
def home():
   message = "TEAM ID : PNT2022TMID37544" +" "+ "BATCH ID : B1-1M3E "
   return render_template('index.html',mes=message)
@app.route('/home', methods=['POST', 'GET'])
def index():
  return render_template('index.html')
@app.route('/signinpage', methods=['POST', 'GET']) def signinpage(): 43     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/', 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] 44
     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 :

" + \ str(randomnumber)+"

```
  "        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('/agentforgot', methods=['POST', 'GET'])
def agentforgot(): 45
  try:
    global randomnumber
   ida = request.form['custid']
   print(ida)
  global id
  id = ida
  sql = "SELECT EMAIL,NAME FROM AGENT WHERE id=?"
  stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, ida)
   ibm_db.execute(stmt)

emailf = ibm_db.fetch_both(stmt)
  while emailf != False:
e = emailf[0]

   n = emailf[1]
  break
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 = "

# Your verification Code is :

## " + \            str(randomnumber)+"

   "        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) 46
api_response = api_instance.send_transac_email(send_smtp_email) pprint(api_response)        message =
"Email send to:"+e+" for OTP"
flash(message, "success")
except ApiException as e:
   print("Exception when calling SMTPApi->send_transac_email: %s\n" % e)
     flash("Error in sending mail")
```

```
except:
    flash("Your didn't Signin with this account")
  finally:
  return render_template('forgot.html')
@app.route('/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) 47
  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,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': 48
  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)}'" 49            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) 50
   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/')
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) 51
  if student:
  sql = f"delete from customer where id='{escape(ID)}'"
  stmt = ibm_db.exec_immediate(conn, sql)
flash("Delected Successfully", "success")
return redirect(url_for("admin"))
@app.route('/agentform', methods=['GET', 'POST'])
def agentform():
if request.method == 'POST':
  try:
x = datetime.datetime.now()
y = x.strftime("%Y-%m-%d %H:%M:%S")
```

TEAM ID: PNT2022TMID34000

```
name1 = request.form['name']
email = request.form['email']
   password = request.form['password']
      phonenumber = request.form['phonenumber']
service = request.form['service']
  address = request.form['address']
   city = request.form['city']
state = request.form['state']
   country = request.form['country']
   link = request.form['link']
sql = "SELECT * FROM AGENT WHERE EMAIL = ?"
     stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt, 1, email)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
if account:
  flash("Record Aldready found", "success")
else:
print("exec")
insert_sql = "INSERT INTO AGENT 52
(NAME,EMAIL,PASSWORD,PHONENUMBER,SERVICE_AGENT,ADDRESS,CITY,STATE,COU
NTRY,RESUME_LINK,DATE) VALUES(?,?,?,?,?,?,?,?,?,?,?)"
prep_stmt = ibm_db.prepare(conn, insert_sql)
  ibm_db.bind_param(prep_stmt, 1, name1)
ibm_db.bind_param(prep_stmt, 2, email)
  ibm_db.bind_param(prep_stmt, 3, password)
   ibm_db.bind_param(prep_stmt, 4, phonenumber)
  ibm_db.bind_param(prep_stmt, 5, service)
   ibm_db.bind_param(prep_stmt, 6, address)
  ibm_db.bind_param(prep_stmt, 7, city)
    ibm_db.bind_param(prep_stmt, 8, state)
ibm_db.bind_param(prep_stmt, 9, country)
   ibm_db.bind_param(prep_stmt, 10, link)
ibm_db.bind_param(prep_stmt, 11, y)
  ibm_db.execute(prep_stmt)
flash("Record stored Successfully", "success")
  sql = "SELECT ID FROM AGENT WHERE email=?"
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt, 1, email)
  ibm_db.execute(stmt)
  hi = ibm_db.fetch_tuple(stmt)
configuration = sib_api_v3_sdk.Configuration()
  configuration.api_key['api-key']
api_instance =
sib_api_v3_sdk.TransactionalEmailsApi(              sib_api_v3_sdk.ApiClient(configuration))
     subject = "Registering Account in Customer Care Registry"
  html_content = "
```

Thanks for Registering into Customer Care Registry

TEAM ID: PNT2022TMID34000

Your Account Id is :"+str(hi[0])+"

With Regards:

Customer Care Registry

```
"              sender = {"name": "IBM CUSTOMER CARE REGISTRY",              "email":
"ibmdemo6@yahoo.com"}              to = [{"email": email, "name": name1}] 53
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)
except:
  flash("Error in Insertion Operation", "danger")
finally:
 return redirect(url_for("agentRegister"))
  con.close()
return render_template('agentregister.html')
@app.route('/completed/', methods=['GET', 'POST'])
def completed(DESCRIPTION):
  status ="Completed"
  try:
sql = "UPDATE ISSUE SET STATUS = ? WHERE DESCRIPTION =?"
 stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt,1,status)       ibm_db.bind_param(stmt,2,DESCRIPTION)

 ibm_db.execute(stmt) flash("Successful","success")
 return redirect(url_for('agentwelcome'))
except:
flash("No record found","danger")
  return redirect(url_for('agentwelcome'))
@app.route('/deletecomplaint/')
def deletecomplaint(ID):
sql = f"select * from ISSUE where ID='{escape(ID)}'" 54     print(sql)
   stmt = ibm_db.exec_immediate(conn, sql)
student = ibm_db.fetch_row(stmt)
 if student:
    sql = f"delete from ISSUE where ID='{escape(ID)}'"

stmt = ibm_db.exec_immediate(conn, sql)
 users = []       flash("Delected Successfully", "success")
```

TEAM ID: PNT2022TMID34000

```
    return redirect(url_for("admin"))
            if __name__ == '__main__':     app.run(host='0.0.0.0', port=5000, debug=True
```

Github id: IBM-EPBL/IBM-Project-40567-1660631493

Github link:CLICK HERE

Demo link:

https://drive.google.com/file/d/156qCMcWwR0FB4QYAdVUrG3yGz3a-193j/view?usp=drivesdk