PROJECT REPORT
CUSTOMER CARE REGISTRY APPLICATION
CLOUD APPLICATION
TEAM ID : PNT2022TMID29808

Submitted by

MANJUPRIYA K	(610519104059)
GOWSALYA A	(610519104030)
JANANI PRIYA E	(610519104039)
JANANIPRIYA S	(610519104038)

1.INTRODUCTION

1.1Project Overview

Customer care service is the support you offer your customers both before and after they buy and use your products or services that helps them have an easy and enjoyable experience with you. Offering amazing customer service is important if you want to retain customers and grow your business. Today's customer service goes far beyond the traditional telephone support agent. It's available via email, web, text message, and social media. Many companies also provide self-service support, so customers can find their own answers at any time day or night. Customer support is more than just providing answers; it's an important part of the promise your brand makes to its customer. 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 importantif the purchase relates to a service as opposed to a product. Customer Service may be provided by a Person or Sales & Service Representatives Customer Service is normally an integral part of a company's customer value proposition.

1.2 Purpose

It involves looking after customers to best ensure a delightful interaction and satisfaction witha business as well as its goods, services, and brand. Instead of just making a sale, good customer care ensures that customers are cared for, their needs are listened to, and they get help in finding the right solution. In many cases, customer care moves a step beyond basic customer service by building an emotional connection. Good customer care means helping customers in an efficient manner that goes beyond their expectations. The success of a business and customer care are intertwined and this is one of the reasons why many businesses are focusing more on offering their customers with excellent services. Be sure you do all that you can to keep your customers informed, on the move, and happy. If your company is responsive, friendly and offers relevant information when needed by a consumer, you'll be in a position to build a reputation for good customer care. Your duty doesn't end once you've made a sale. In fact, this is the stage in the process when your business ought to pay attention to customer satisfaction and a hassle-free journey for your consumers. In a time where consumers often complain about lack of service and technology allows for those experiences to be relayed through varied outlets, providing excellent customer care can set you apart from your competitors. By highlighting customer care in your

marketing strategy, you'll create a sense of uniqueness about your business by setting yourself apart from companies that deliver on their promises. The main purpose of the customer care is Increased customer satisfaction, Growth in profits,High employee motivation and satisfaction, Better-quality customer service skills. 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.

2.LITERATURE SURVEY

Customer relationship management (CRM) refers to the principles, practices, and guidelines that an organization follows when interacting with its customers. From the organization's point of view, this entire relationship encompasses direct interactions with customers, such as sales and service-related processes, forecasting, and the analysis of customer trends and behaviors. Ultimately, CRM serves to enhance the customer's overall experience. Customer relationship management includes the principles, practices, and guidelines an organization follows when interacting with its customers. CRM is often used to refer to technology companies and systems that help manage external interactions with customers. Major areas of growth in CRM technology include software, cloud computing, and artificial intelligence. This research background is a number of problems faced by the company can make a bad company image and reduced levels of consumer loyalty. To avoid harming the image of the company then the company must focus on service to consumers. Service to consumers is very important in increasing the satisfaction of its customers, due to the company customers is the most important asset in which consumers provide andilo is significant in the development of the company's reputation.Customer Relationship Management (CRM) to improve customer loyalty and good image .

2.1Existing problem

There is an existing problem in the fashion industry that needs to be addressed. A lot of designers are using unethical practices in order to make money. This includes using cheap materials and cutting corners in order to increase profits. This is not fair to the consumers, and it needs to stop. We need better, more ethical fashion brands out there, and we need them fast.

There is an existing problem with fashion recommender systems that dictates that they are often ineffective and unreliable. Fashion recommendation systems rely on data from users to create personalized recommendations, but this data can be unreliable. This means that the system may recommend items that are not in line

with the user's interests or style. Additionally, the system may not be able to recommend items that the user is actively looking for. This can lead to frustrating experiences when trying to find clothing that fits well or finding new styles to try.

The current systems rely on user feedback and ratings to generate recommendations, but this method is flawed because it does not take into account the user's own style. This means that the recommendations are ineffective for people who prefer different styles of clothing than the average person.

2.2 References

1 . https://www.investopedia.com/terms/c/customer_relation_management.asp

2.https://www.researchgate.net/publication/336878065_The_Role_Of_Customer_Service_Through_Customer_Relationship_Management_CRM_To_Increase_Customer_Loyalty_And_Good_Image

3.https://www.researchgate.net/publication/344889386_Design_And_Implementation_Of_Customer_Service_Complaint_Portal

4.https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.tandfonline.com/doi/full/10.1080/1331677X.2019.1676283&ved=2ahUKEwjQ65HQmpb6AhUwZWwGHaITBGE4ChAWegQIERAB&usg=AOvVaw25DMFhWG-bYj73eamWrkL2

2.3 Problem Statement Definition:

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. 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. 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

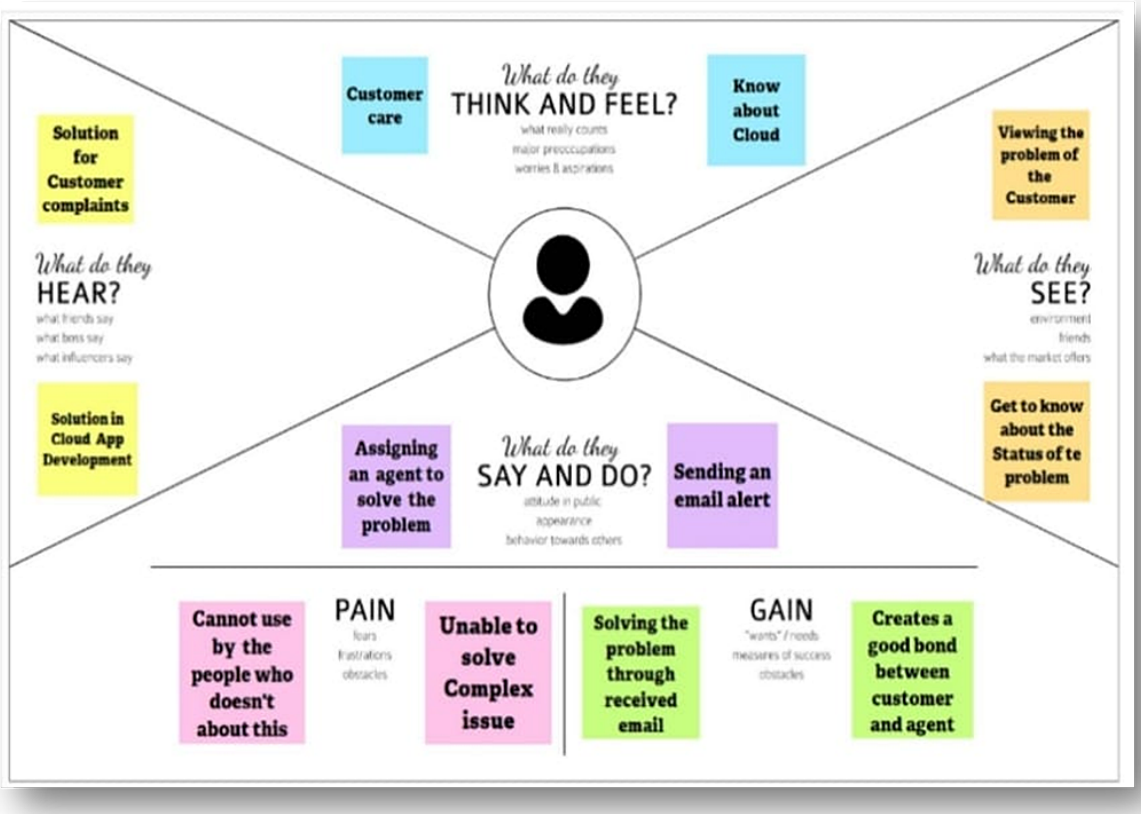
3.IDEATION & PROPOSED SOLUTION:

3.1Empathy Map Canvas:

An **empathy map** is a collaborative visualization used to articulate what we know about a particular type of user. It externalizes knowledge about users in order to

1) create a shared understanding of user needs, and

2) aid in decision making.

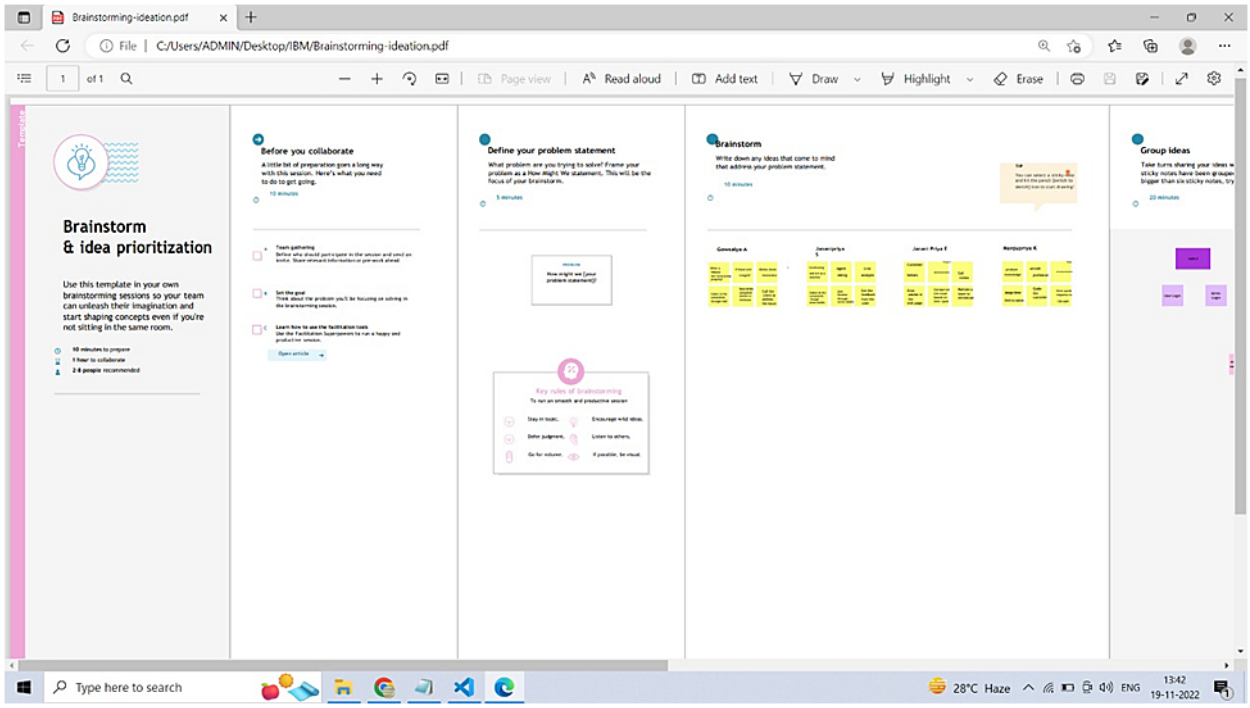


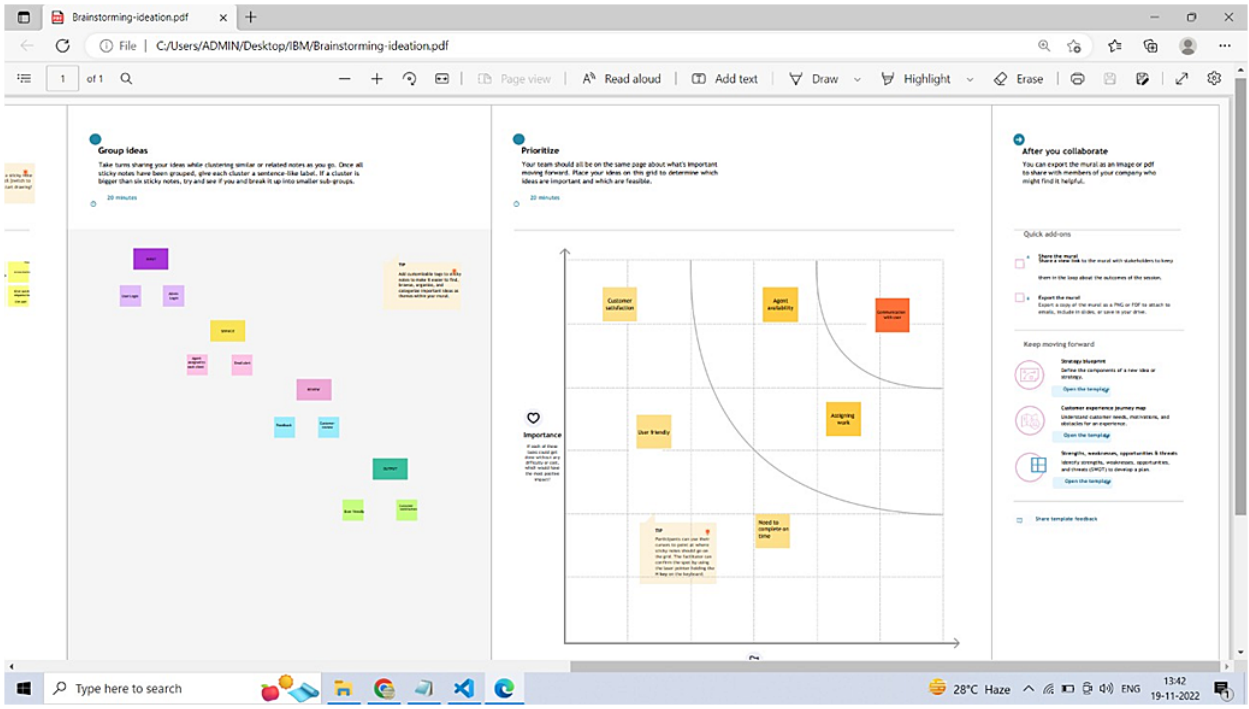
3.2Ideation & Brainstorming:

Ideation is a creative process where designers generate ideas in sessions (e.g., brainstorming, worst possible idea)

Brainstorming is a method design teams use to generate ideas to solve clearly defined design problems.

Brainstrom & Idea Listing and Grouping:





3.3Proposed Solution:

1.Problem Statement (Problem to be solved):

Satisfying the Customer's Problem Efficiently through Email

2.Idea / Solution description:

Customer Care Solution is to manage customer interaction and complaints with the Service Providers through e-mail and all the complaint details will be stored in Cloud. The system should have capability to integrate with any Service Provider from any domain or industry like Banking. Telecom, Insurance, etc. Customer Service may be provided by a Person or Sales & Service Representatives Customer Service is normally an integral part of a company's customer value proposition.

One of the best ways to offer a more personal customer experience is using the customer's name when talking with them through email, or when coming up with customer surveys. A friendlier approach that doesn't feel forced humanizes the consumer-business interaction. Using a friendly, informal, or familiar tone and style of writing using a personalized email to send the message instead of a generic corporate one ("john@business.com" instead of just "brandname@business.com").

3. Novelty / Uniqueness:

Actively Ask for Customers Feedback and sharing their feedbacks to all the agents which helps to solve the Customers problem.

4.Social Impact / Customer Satisfaction:

Customer can send their problem in their native language so they can create good bond between Customer and an Organisation.

5.Business Model (Revenue Model):

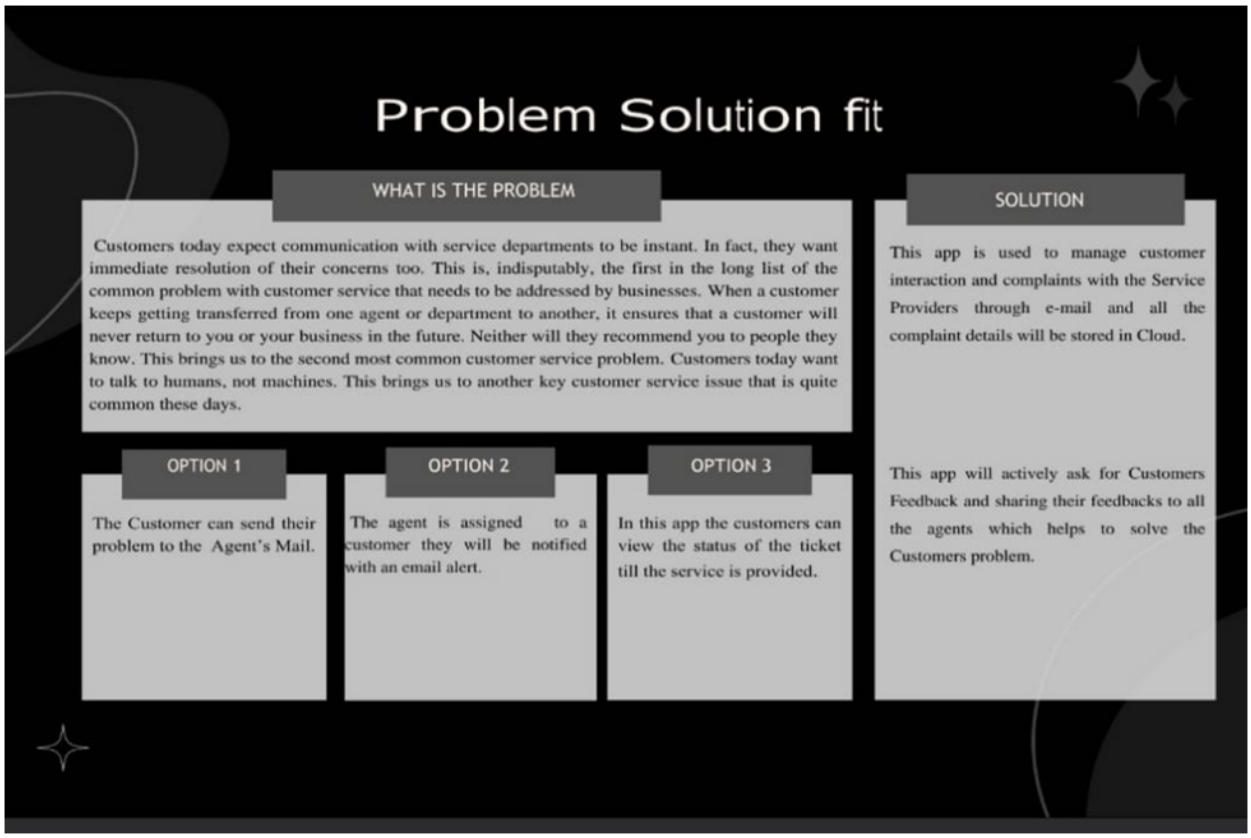
CX is an integral part of Customer Relationship Management (CRM) and the

reason why it’s important is because a customer who has a positive experience with a business is more likely to become a repeat and loyal customer . Customers can rate companies with a high customer experience score (i.e. 10/10).

6.Scalability of the Solution

Even though it is a brand or Customer’s fault, the agent will give a correct solution to the customer.

3.4Problem Solution fit:



4.REQUIREMENT ANALYSIS:

4.1Functional requirement:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User login	Login with Email ID and Password
FR-4	Dashboard	To view the Agent's profile, tracking the status.
FR-5	Chatbot	Add the problem information

4.2 Non-Functional requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Provides a efficient solution. Easy to use for customers.
NFR-2	Security	It is a very secured app so it cannot be accessed by unauthorized person.
NFR-3	Reliability	Tracks the status of the ticket.
NFR-4	Performance	Customers can rate companies with a high customer experience score (i.e. 10/10).

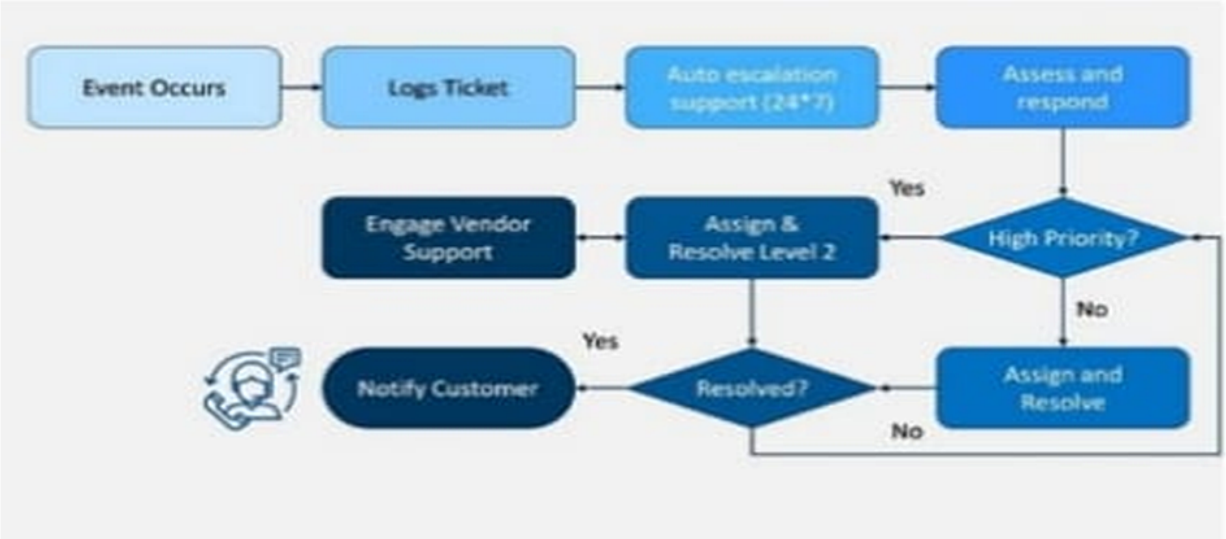
NFR-5	Availability	This ensures that all of its User and agent related data is available to the end-users at any time of the day, whenever and wherever required.
NFR-6	Scalability	24/7 customer support means customers can get help and find answers to questions as soon as they come up—24/7 and in real-time.

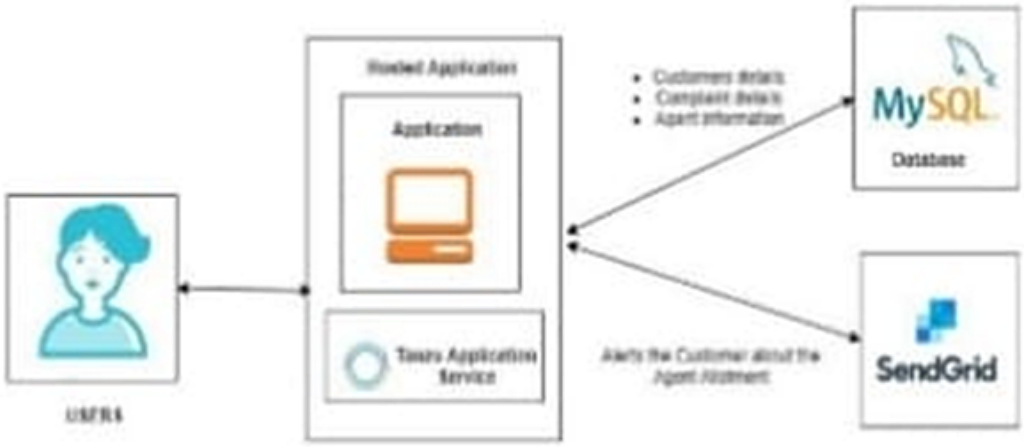
5.PROJECT DESIGN:

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.
Example: (Simplified)

–





5.2 Solution & Technical Architecture:

5.3User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	User Registration In the mobile app	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password in the app	I can access my account / dashboard	High	Sprint-1

	Confirmation mail from the admin	USN-2	As a user, I will receive confirmation email from the admin,once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
	Login in to the app using registered email	USN-3	As a user, I can register the complaint on an issue I'm facing	I can register & access the dashboard with Login	High	Sprint-2
	Dashboard	USN-4	As a user, receive an solution in email from the agent	I can receive the email alert from the agent regarding solution	Medium	Sprint-1
	Track	USN-5	As a user, I can track the status of their Complaint	I can track the status of the complaint in dashboard	High	Sprint-1
	Feedback	USN-6	As a User, I can share the feedback on the feedback form	Friendly helpful	High	
Customer (Web user)			Can access the overall data from the Organization		High	
Customer Care			They have the data in		High	

Executive			the Cloud			
Administrator			Access the Cloud easily		High	

6.PROJECT PLANNING & SCHEDULING

6.1Sprint Planning & Estimation

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**	**Release**
Customer (Mobile user)	User Registration In the mobile app	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password in the app	I can access my account / dashboard	High	Sprint-1
	Confirmation mail from the admin	USN-2	As a user, I will receive confirmation email from the admin,once I have registered for the	I can receive confirmation email & click confirm	High	Sprint-1

			application			
	Login in to the app using registered email	USN-3	As a user, I can register the complaint on an issue I'm facing	I can register & access the dashboard with Login	High	Sprint-2
	Dashboard	USN-4	As a user, receive an solution in email from the agent	I can receive the email alert from the agent regarding solution	Medi um	Sprint-1
	Track	USN-5	As a user, I can track the status of their Complaint	I can track the status of the complaint in dashboard	High	Sprint-1
	Feedback	USN-6	As a User, I can share the feedback on the feedback form	Friendly helpful	High	
Customer (Web user)			Can access the overall data from the Organizati		High	

			on			
Customer Care Executive			They have the data in the Cloud		High	
Administrator			Access the Cloud easily		High	

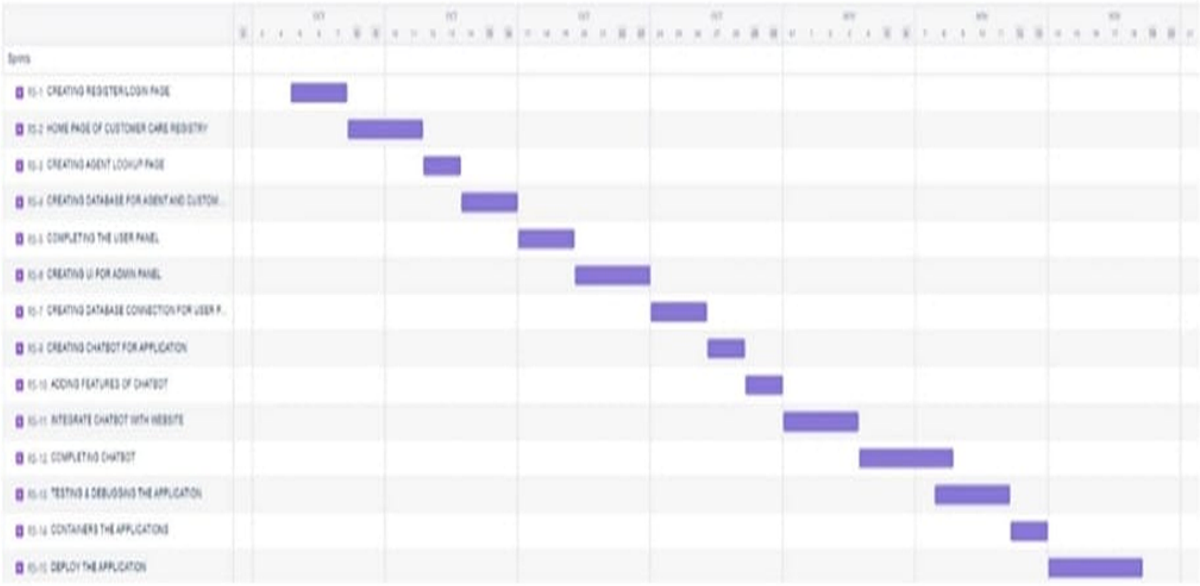
6.1Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and go through the services available on the webpage	20	High	Manjupriya Jananipriya.S Gowsalya
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database aboutthe availability and have a track of all the things that the users are going to service	20	High	Janani priya.E Manjupriya
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	Gowsalya Jananipriya.S
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	Jananipriya.S Gowsalya Janani priya.E

6.2 Sprint Delivery Schedule:

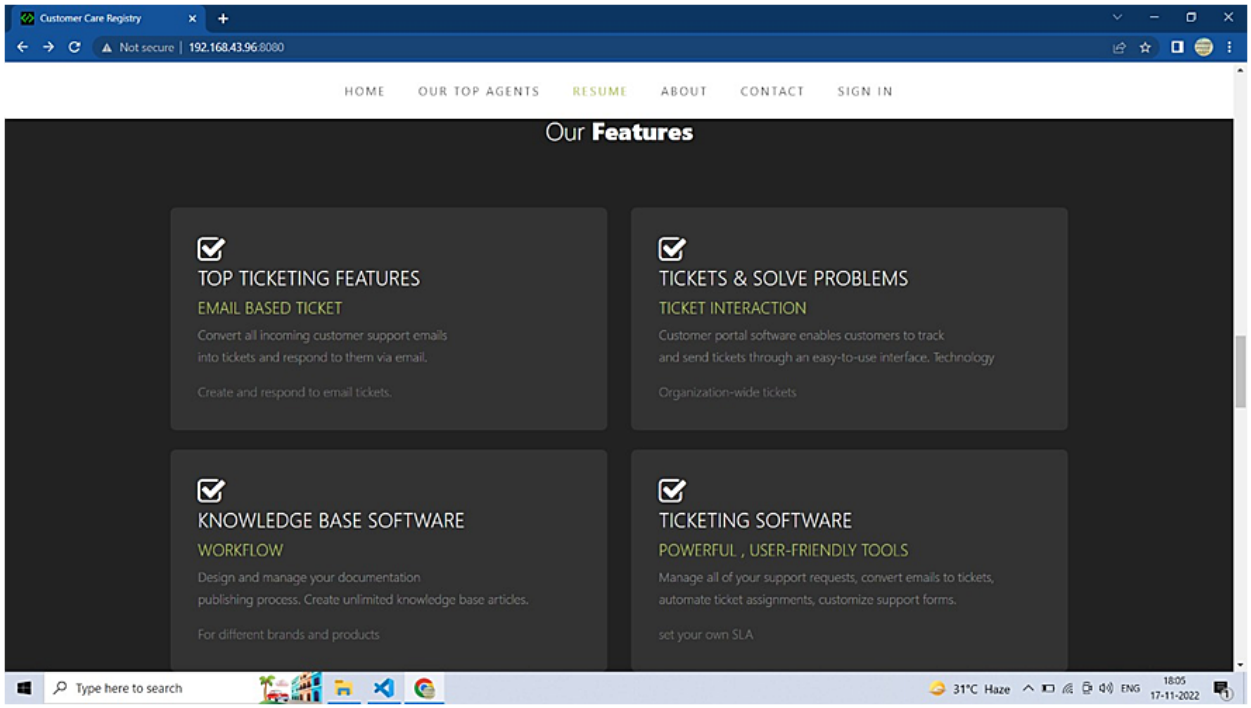
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12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

6.3Reports from JIRA:

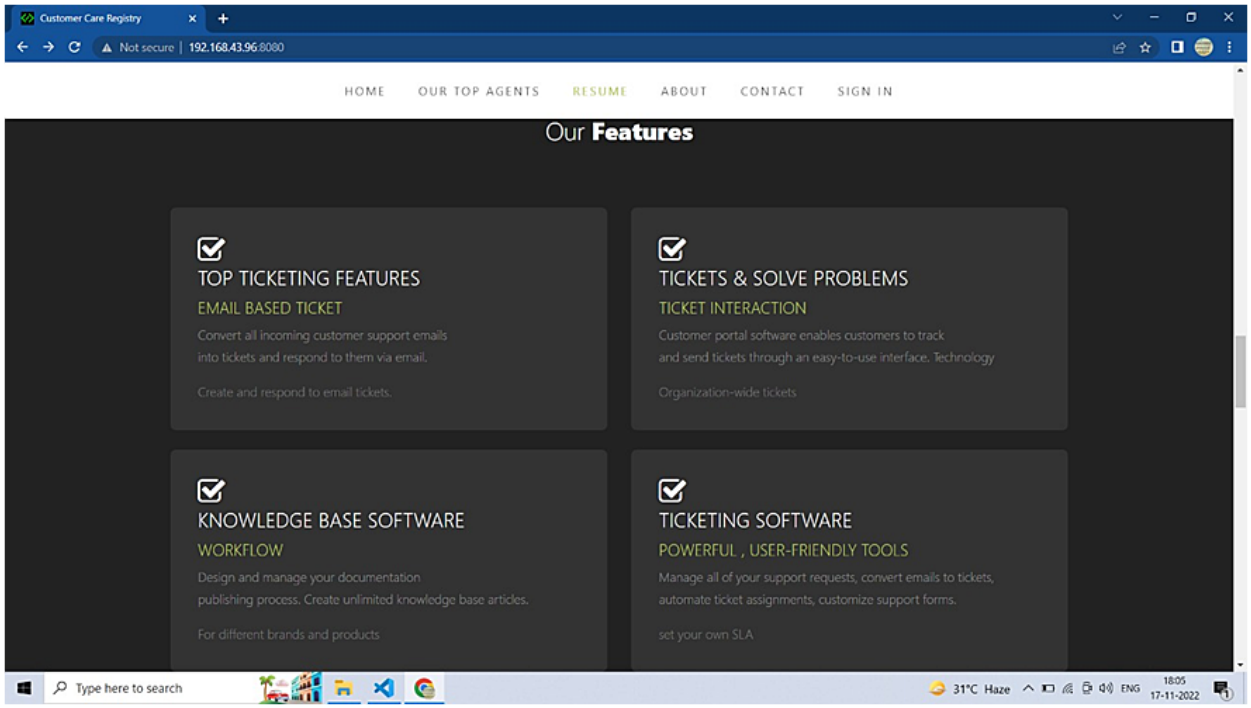


7. CODING & SOLUTIONING (Explain the features added in the project along with code

7.1 Feature 1:


Customer Care Registry
Not secure | 192.168.43.96:8080
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
Our Features
TOP TICKETING FEATURES
EMAIL BASED TICKET
Convert all incoming customer support emails into tickets and respond to them via email.
Create and respond to email tickets.
TICKETS & SOLVE PROBLEMS
TICKET INTERACTION
Customer portal software enables customers to track and send tickets through an easy-to-use interface. Technology
Organization-wide tickets
KNOWLEDGE BASE SOFTWARE
WORKFLOW
Design and manage your documentation publishing process. Create unlimited knowledge base articles.
For different brands and products
TICKETING SOFTWARE
POWERFUL , USER-FRIENDLY TOOLS
Manage all of your support requests, convert emails to tickets, automate ticket assignments, customize support forms.
set your own SLA
Type here to search
31°C Haze
ENG
18:05
17-11-2022

7.2 Feature 2 ：


Customer Care Registry
Not secure | 192.168.43.96:8080
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
Our Features
TOP TICKETING FEATURES
EMAIL BASED TICKET
Convert all incoming customer support emails into tickets and respond to them via email.
Create and respond to email tickets.
TICKETS & SOLVE PROBLEMS
TICKET INTERACTION
Customer portal software enables customers to track and send tickets through an easy-to-use interface. Technology
Organization-wide tickets
KNOWLEDGE BASE SOFTWARE
WORKFLOW
Design and manage your documentation publishing process. Create unlimited knowledge base articles.
For different brands and products
TICKETING SOFTWARE
POWERFUL , USER-FRIENDLY TOOLS
Manage all of your support requests, convert emails to tickets, automate ticket assignments, customize support forms.
set your own SLA
Type here to search
31°C Haze
ENG
18:05
17-11-2022

8 . TESTING :

8.1. Test case :

The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not. Test case designing includes preconditions, case name, input conditions, and expected result. A test case is a first level action and derived from test scenarios.Test case gives detailed information about testing strategy, testing process, preconditions, and expected output. These are executed during the testing process to check whether the software application is performing the task for that it was developed or not. Test case helps the tester in defect reporting by linking defect with test case ID. Detailed test case documentation works as a full proof guard for the testing team because if developer missed something, then it can be caught during execution of these full-proof test cases.To write the test case, we must have the requirements to derive the inputs, and the test scenarios must be written so that we do not miss out on any features for testing. Then we should have the test case template to maintain the uniformity, or every test engineer follows the same approach to prepare the test document

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	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	0	0	2	7
External	0	2	0	0	2
Fixed	12	11	35	45	103
Not Reproduced	0	5	0	0	5
Skipped	0	0	0	0	0
Totals	17	18	35	47	117

3. Test Case Analysis

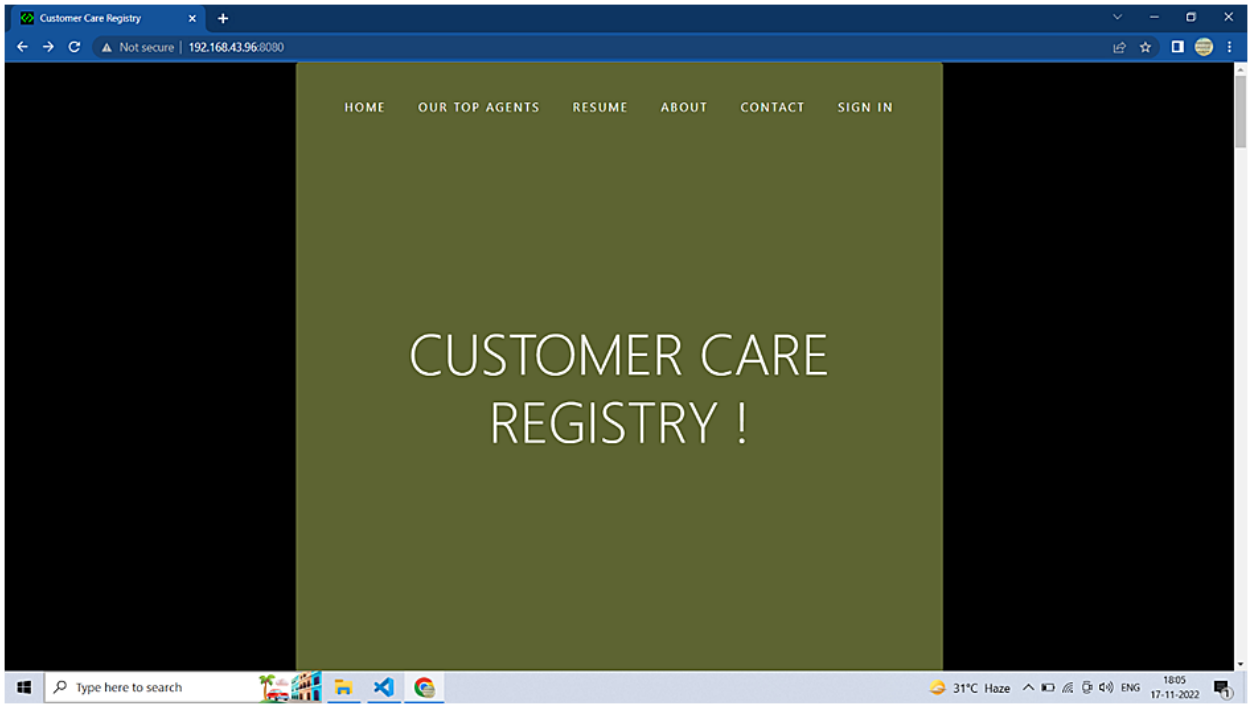
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Client Application	72	0	0	72
Security	7	0	0	7
Exception Reporting	5	0	0	5
Final Report Output	4	0	0	4

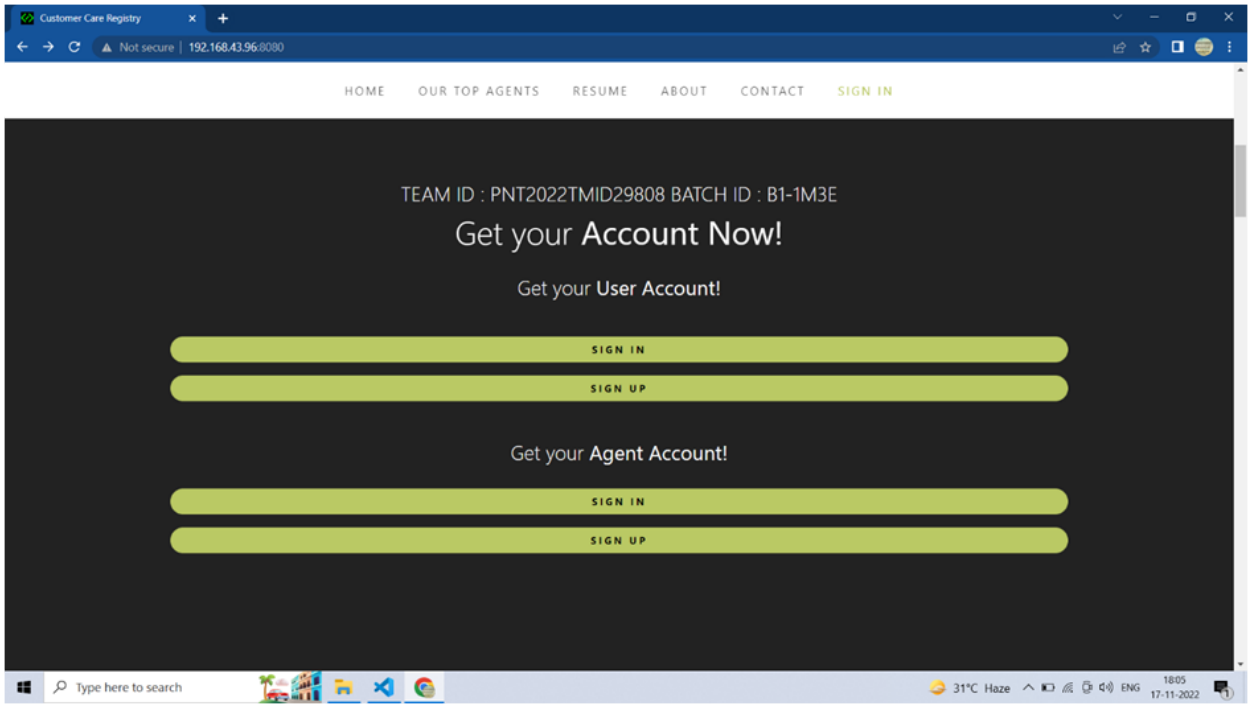
9.RESULT :

9.1 Performance Metrics:

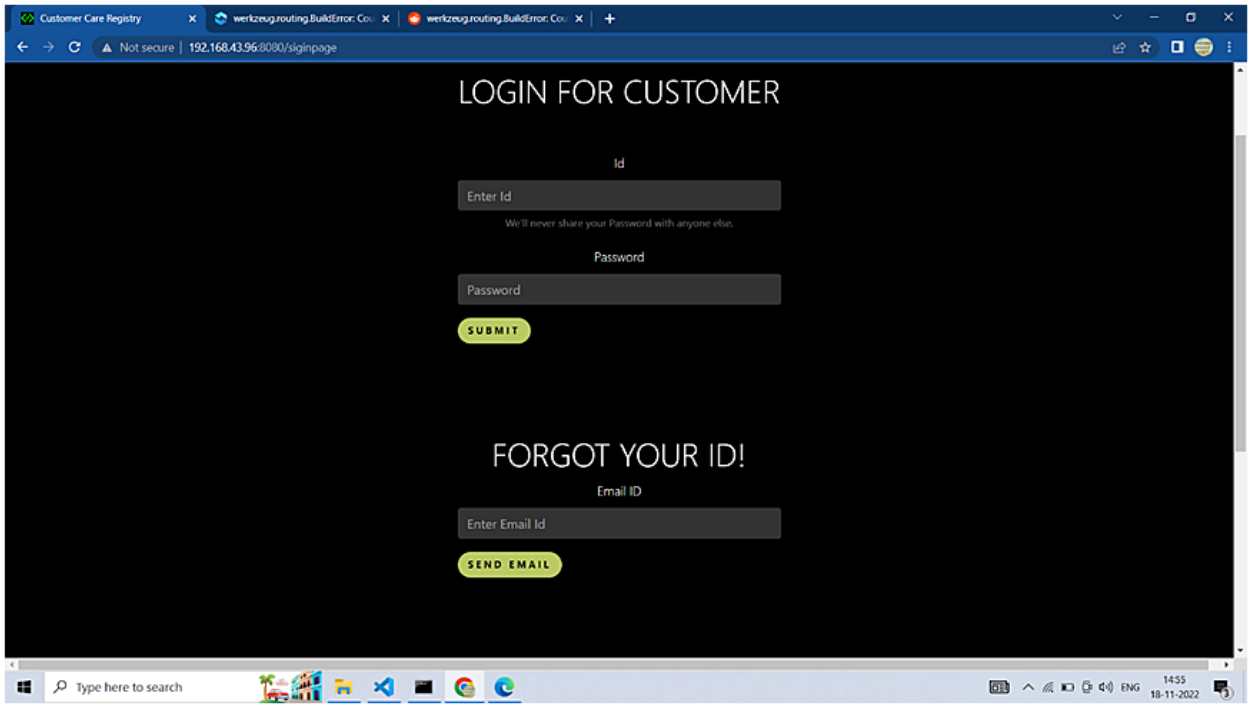
HOME PAGE:



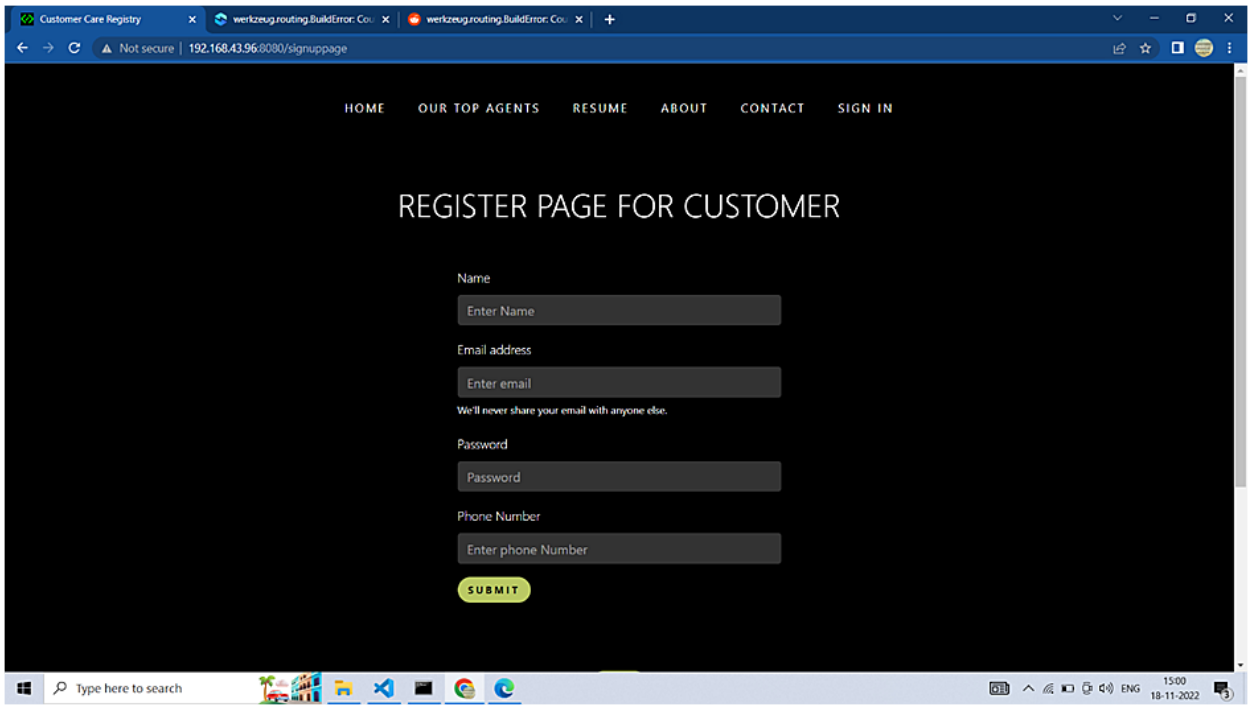
SIGN IN:


Customer Care Registry
Not secure | 192.168.43.96:8080
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
TEAM ID : PNT2022TMID29808 BATCH ID : B1-1M3E
Get your Account Now!
Get your User Account!
SIGN IN
SIGN UP
Get your Agent Account!
SIGN IN
SIGN UP
Type here to search
31°C Haze
ENG
18:05
17-11-2022

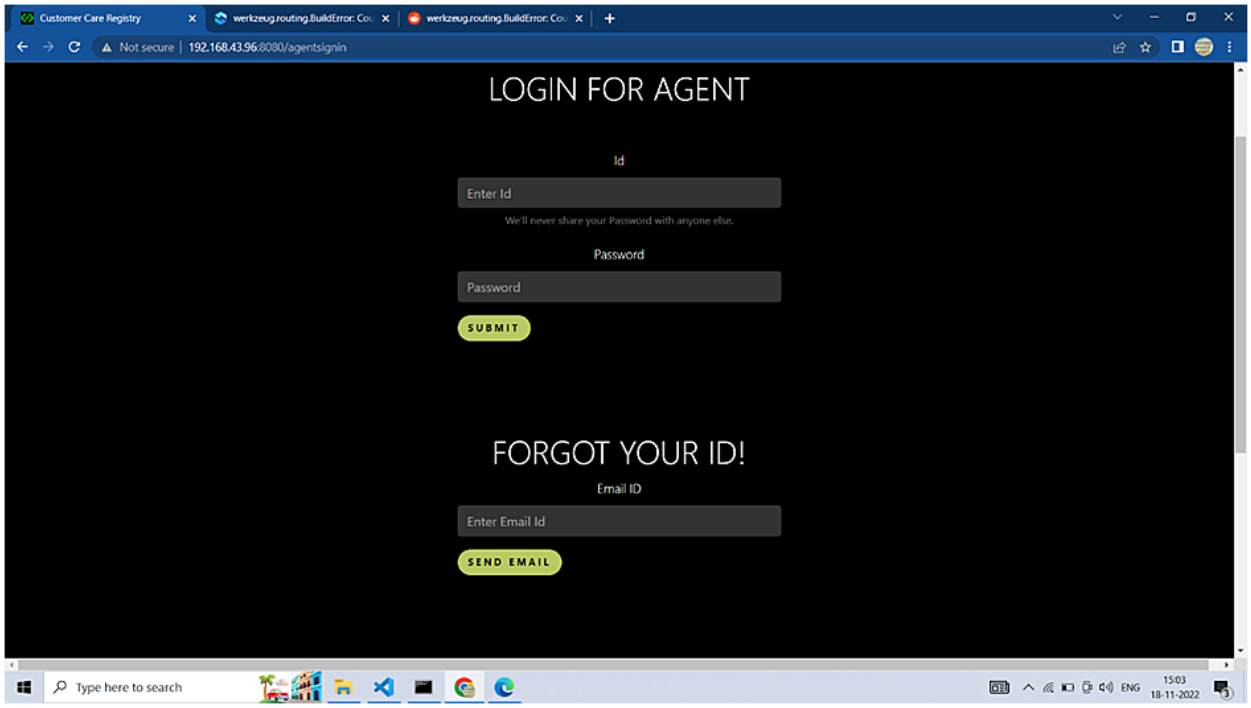
SIGN IN PAGE FOR CUSTOMER:



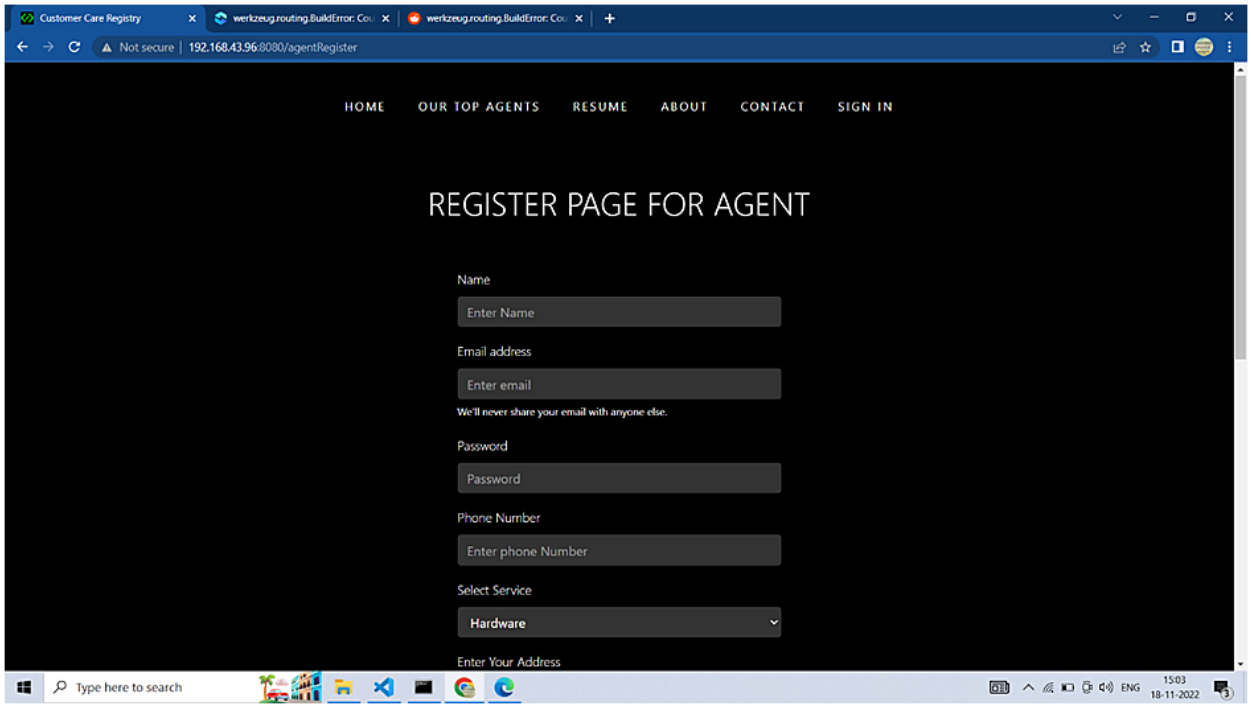
SIGN UP PAGE FOR CUSTOMER



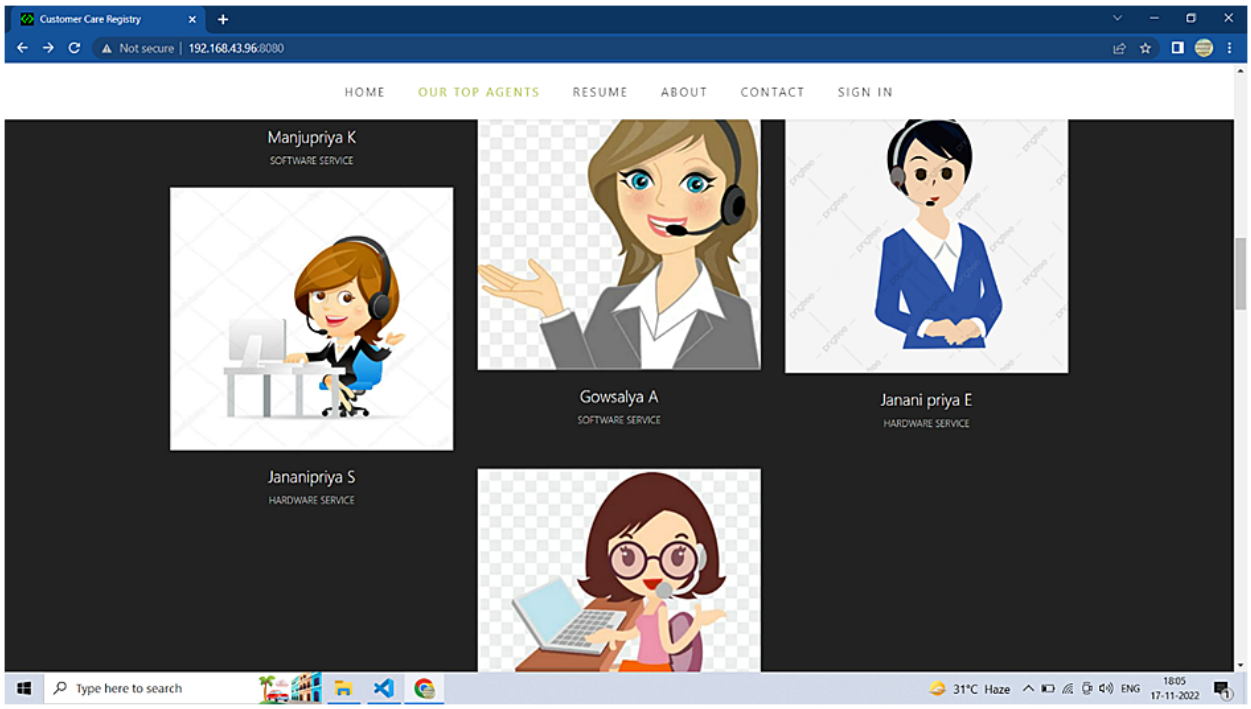
SIGN IN PAGE FOR AGENT



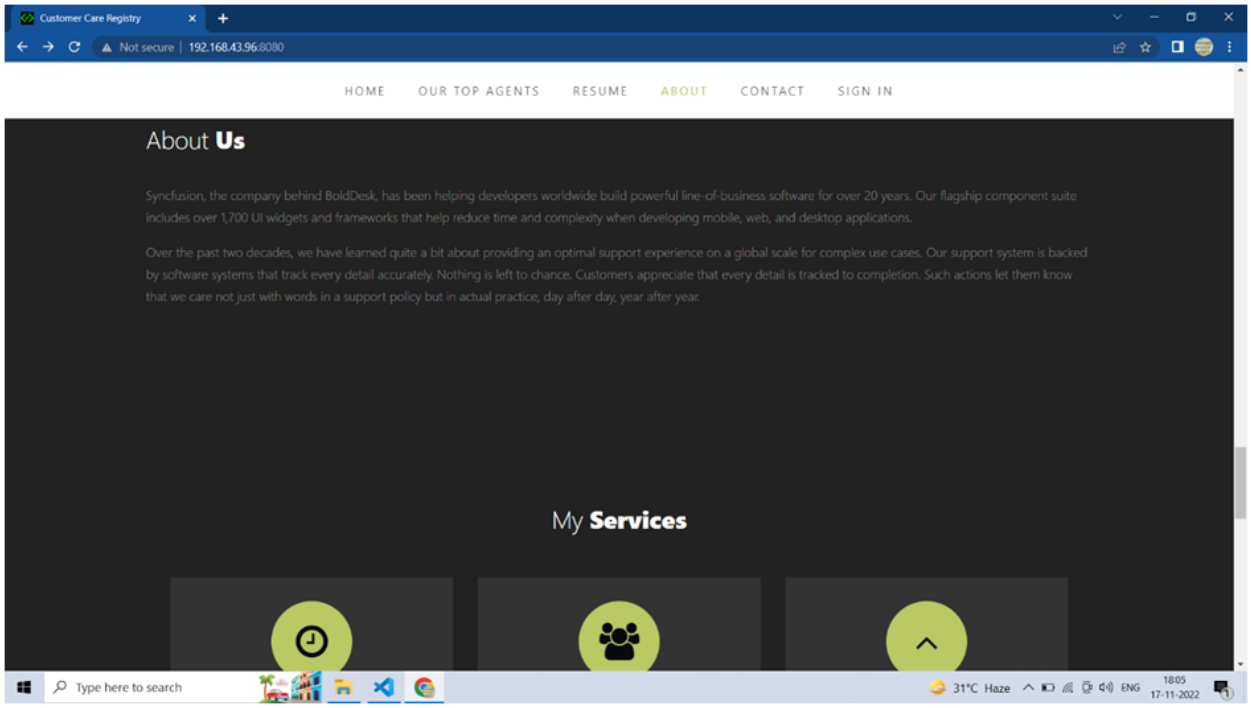
SIGN UP PAGE FOR AGENT



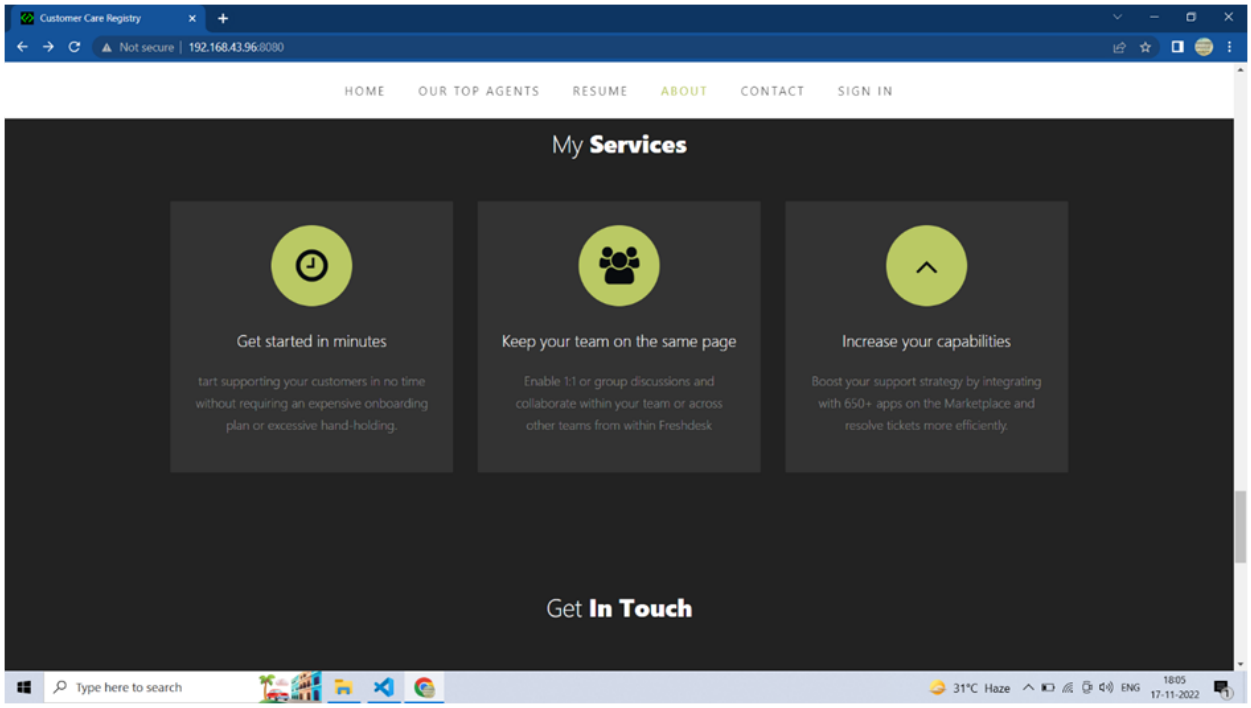
OURTOPAGENTS:



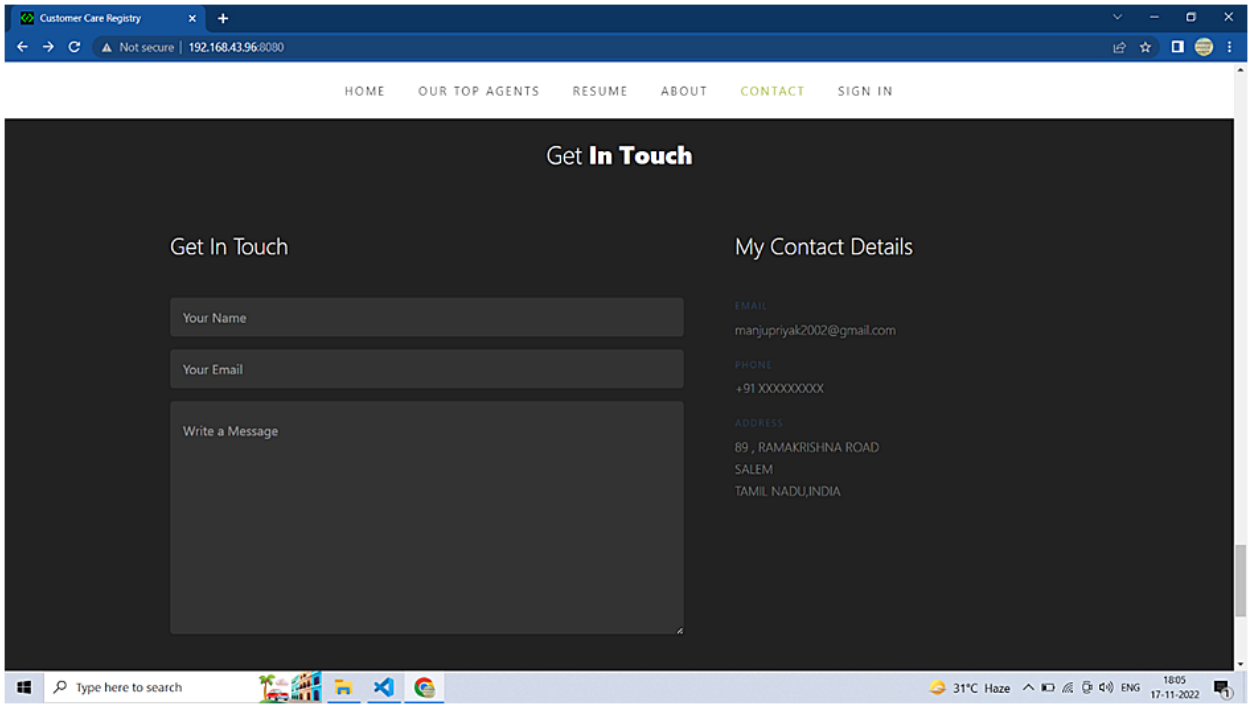
ABOUT US:


Customer Care Registry
Not secure | 192.168.43.96:8080
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
About Us
Syncfusion, the company behind BoldDesk, has been helping developers worldwide build powerful line-of-business software for over 20 years. Our flagship component suite includes over 1,700 UI widgets and frameworks that help reduce time and complexity when developing mobile, web, and desktop applications.
Over the past two decades, we have learned quite a bit about providing an optimal support experience on a global scale for complex use cases. Our support system is backed by software systems that track every detail accurately. Nothing is left to chance. Customers appreciate that every detail is tracked to completion. Such actions let them know that we care not just with words in a support policy but in actual practice, day after day, year after year.
My Services
Type here to search
31°C Haze
ENG
18:05
17-11-2022

MY SERVICES:


Customer Care Registry
Not secure | 192.168.43.96:8080
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
My Services
Get started in minutes
tart supporting your customers in no time without requiring an expensive onboarding plan or excessive hand-holding.
Keep your team on the same page
Enable 1:1 or group discussions and collaborate within your team or across other teams from within Freshdesk
Increase your capabilities
Boost your support strategy by integrating with 650+ apps on the Marketplace and resolve tickets more efficiently.
Get In Touch
Type here to search
31°C Haze
ENG
18:05
17-11-2022

GET IN TOUCH:



10 . ADVANTAGES & DISADVANTAGES :

ADVANTAGES :

Automated email notifications can be used to update customers about the status of their issue or support ticket . Create a multilingual knowledge base to cater to customers from different parts of the world . Have personalized, one-on-one interactions. When you use a customer support system, you can easily have a record of the complaints that your customers

make. This helps you to get to know your customer better and better anticipate their service needs to eliminate the issues.

DISADVANTAGES :

Lack of real-time human-to-human interaction . Keeping track of emails can get challenging when you receive hundreds of them every day . Agents have to be trained in multiple customer service areas such as technical support, etc.. It cannot be used to solved complex problem .

11 . CONCLUSION :

There are many customer service applications available on the internet. Noting down the structural components of those applications and we built a customer care registry application.It will be a web application build with Flask (Python micro-web framework), HTML, JavaScript. It will be a ticket-based customer service registry. Customers can register into the application using their email, password. Then, they can login to the system, and raise as tickets as they want in the form of their tickets.These tickets will be sent to the admin, for which an agent is assigned. Then, the assigned agent will have a one-to-one chat with the customer and the latter’s queries will be clarified. It is also the responsibility of the admin, to create an agent

12 . FUTURE SCOPE :

- Attracting and much more responsive UI throughout the application
- Releasing cross-platform mobile applications
- Incorporating automatic replies in the chat columns

- Deleting the account whenever customer wishes to
- Supporting multi-media in the chat columns
- Creating a community for our customers to interact with one another

13 . APPENDIX :

13.1 SOURCE CODE :

```
from__future_____import  print_function

from audioopimport add

import datetime

from   unicodedata   import   name

import sib_api_v3_sdk

from  sib_api_v3_sdk.rest  import  ApiException

from pprint import pprint

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

frommarkupsafe import escape

from  flask  import  *

import          ibm_db

importdatetime

conn          =          ibm_db.connect("DATABASE=bludb;HOSTNAME=1bbf73c5-d84a-4bb0-85b9-
ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=32286;SECURITY=SSL;SSL
ServerCertificate=;UID=zdb94008;PWD=9CbgTMAJjAF0Vqob", ", ")

print(conn)

print("connection successful...")



#app     =     Flask(___name___)_

app.secret_key = ''

app  =  Flask(_name_,_template_folder  =  'templates')
```

```
app.config['SECRET_KEY']                    =                    "
app.config['MAIL_SERVER']   =   'smtp.sendgrid.net'
app.config['MAIL_PORT']                    =                    587
app.config['MAIL_USE_TLS']              =              True
app.config['MAIL_USERNAME']         =         'apikey'
app.config['MAIL_PASSWORD'] = "
app.config['MAIL_DEFAULT_SENDER'] = 'manjupriyak2002@gmail.com'


@app.route('/')
def home():
    message = "TEAM ID : PNT2022TMID29808" +" "+"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('/siginpage', methods=['POST','GET']) def
siginpage():
    return render_template('siginpage.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.hinttml')


@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":   "manjupriyak2002@gmail.com"}
    to = [{"email": e, "name": n}]
    reply_to = {"email": "manjupriyak2002@gmail.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():


    try:
        global randomnumber
        ida   =   request.form['custid']
        print(ida)
        global    id
        id = ida
        sql = "SELECT EMAIL,NAME FROM AGENT WHEREid=?"
        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":   "manjupriyak2002@gmail.com"}
    to = [{"email": e, "name": n}]
    reply_to  =  {"email":  "manjupriyak2002@gmail,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 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) if
    users:
        sql = "SELECT COUNT(*) FROM ISSUE;"
        stmt = ibm_db.exec_immediate(conn,
        sql)count = ibm_db.fetch_both(stmt)

    agent = []
```

```
    sql = "SELECT * FROM AGENT"
    stmt      =      ibm_db.exec_immediate(conn,
    sql)dictionary   =   ibm_db.fetch_both(stmt)
    while dictionary != False:

        agent.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)


    if agent:
        sql = "SELECT COUNT(*)FROM AGENT;"
        stmt  =  ibm_db.exec_immediate(conn,  sql)
        cot = ibm_db.fetch_both(stmt)


    return
render_template("admin.html",complaint=users,users=userdatabase,agents=agent,message=user[0]
,issue=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']
            globalhello
            hello = id
            password    =    request.form['password']
            print(id, password)
            if id == '1111' and password == '1111':
                return redirect(url_for('admin'))


            sql = f"select * from customerwhere 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 WHERECUSTOMER_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"))


@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") 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 (NAME,EMAIL,PASSWORD,PHONENUMBER,SERVICE_AGENT,ADDRESS,CITY,STATE,COUNTRY,RESUM E_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 AGENTWHERE
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 CustomerCare Registry"
html_content = " <html><body><h1>Thanks for Registering into Customer Care Registry</h1> <h2>Your Account Id is :"+str(hi[0])+"</h2><h2>With Regards:</h2><h3>CustomerCare Registry</h3> </body></html>"
sender = {"name": "IBM CUSTOMER CARE REGISTRY",
  "email": "manjupriyak2002@gmail.com"}
```

```
            to = [{"email": email,"name": name1}]
            reply_to = {"email": "manjupriyak2002@gmail.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.clos
            e()


    return render_template('agentregister.html')



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

```
    try:

        sql = "UPDATE ISSUE SET STATUS= ? WHERE DESCRIPTION =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,status)
        ibm_db.bind_param(stmt,2,DESCRIPTION)
        ibm_db.execute(stmt)

        flash("Successful","success")
        return
    redirect(url_for('agentwelcome'))except:
        flash("No record found","danger") return
        redirect(url_for('agentwelcome'))

@app.route('/deletecomplaint/<ID>')
def deletecomplaint(ID):
    sql = f"select * from ISSUE 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 ISSUE where
        ID='{escape(ID)}'" stmt =
        ibm_db.exec_immediate(conn, sql)
        users = []
        flash("Delected Successfully", "success")
```

```
    return redirect(url_for("admin"))


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```

13 . 2 Github link :

https://github.com/IBM-EPBL/IBM-Project-36486-1660295375

13 . 3 Project demo link :

https://youtu.be/MKbI4SAMuZw