DHANALAKSHMI SRINIVASAN

COLLEGE OF ENGINEERINGAND TECHNOLOGY
MAMALLAPURAM, CHENNAI - 603 104.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CUSTOMER CARE REGISTRY

A PROJECT REPORT

Submitted by

SHARMILAD	**310519104301**
SHOBANA M	**310519104120**
SEVVANTHI D	**310519104118**
THATCHAYANI I	**310519104133**
NAVIN KUMAR D	**310519104079**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY::CHENNAI-600025

AUG2022-DEC2022

ANNA UNIVERSITY::CHENNAI– 600 025

BONAFIDECERTIFICATE

Certificate that this project report **"CUSTOMER CARE REGISTRY"** is the bonafide work of "**SHARMILA D (310519104301**), **SHOBANA M (310519104120), SEVVANTHI D (310519104133), THATCHAYANI I (310519104133), NAVIN KUMAR D (310519104098)"** who carried out the project work under my supervision.

SIGNATURE **SIGNATURE**

HEAD OF THE DEPARTMENT **SUPERVISOR**

DepartmentofComputerScience andEngineering,

DhanalakshmiSrinivasanCollege of Engineering & Technology, Mamallapuram.

Department of Computer Science and Engineering,

Dhanalakshmi Srinivasan College of Engineering & Technology, Mamallapuram.

Submitted for the Practical Examination held on_____

INTERNAL EXAMINAR

EXTERNALEXAMINAR

ABSTRACT

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

ACKNOWLEDGEMENT

First of all,we thank,**Our Almighty** for his blessings upon us to strengthen our mind and soul to take up this project. We owe a great many thanks to a great many people who helped and supported us in this project.

We thank Our **Chairman, THIRU. A. SRINIVASAN**, who allowed us to do the project.

We are also thankful to Our **Principal**, **Dr. R SARAVANAN, (Ph.D)** for his constant support to do project.

We extremely thank to Our **Vice Principal**, **Dr. V. JANAKIRAMAN,(Ph.D),** for his constant support in selecting the project.

We are grateful to Our **Head of the department**, **Dr.P.MALTHI**,(**Ph.D),** who expressed his interest and guide in our work and supplied with some useful ideas.

We would like to thank Assistant Professor **Mrs. Shenbagam Kaliappan**, for following our project with interest and forgiving me constant support. She taught us not only how to do the project,but also how to enjoy project.

We wish to extend our grateful acknowledgement and sincere thanks to our project coordinators, Mr.Prabakaran MV**,** and Dr.Malathi P**,**for their constant encouragement and kind support in completing the project

Furthermore, we would like to thank all our **Teaching Faculty and Non-teaching Faculty** for their timely help in solving any project queries.

Finally, we would like to thank our **Parents** for their blessings, support and encouragement through out our life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

1	CCR	-	Customer Care Registry
2	TOS	-	Tracking of Service
3	CQ	-	Customer Queries
4	CRM		Customer Relationship Management
5	CSAT	-	Customer Satisfication
6	CX	-	Customer Experience
7	QA	-	Quality Assurance
8	UI	-	User Interface
9	CF		Customer Feedback
10	WA		Watson Assistent

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

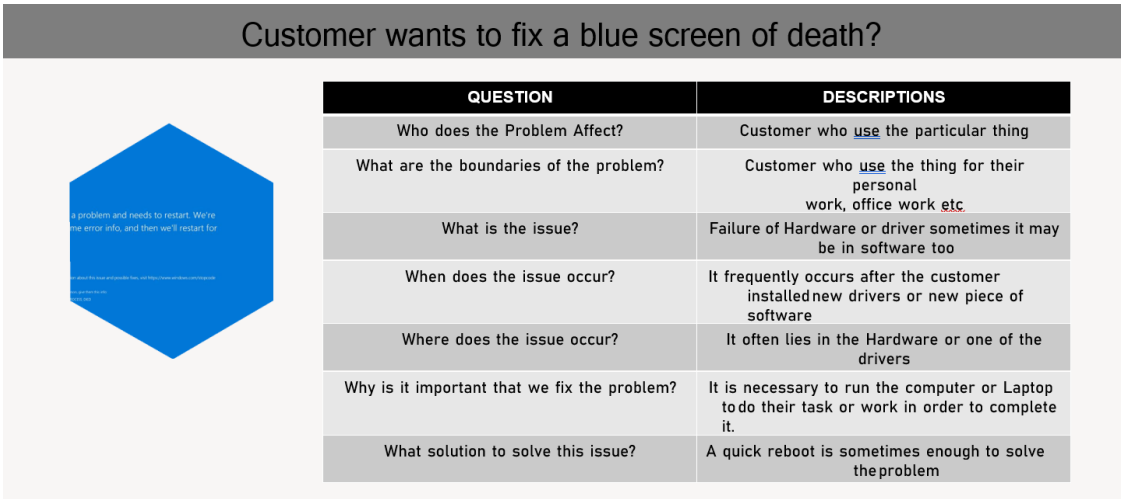
Customer is that the center of attention of each business. The terrible existence of business depends on client satisfaction. Client expects high-quality services, even willing to pay a premium for higher service.From a client perspective, smart service quality ends up in semi permanent client relationships Measuredby re-patronage and cross sales, additionally client advocates the service to others. Services are completely different from manufacturing, this distinction contributes to the accumulated completeness of service quality.

Corporations so build all efforts to produce high-quality services to please customers. However,despite best efforts,associate occasional criticism is inevitable. However,an honest recovery will flip angry, discontent customers into loyal ones, again. The key to success lies in recognizing the importance of responding fairly and effectively to client complaints. Complaints are usually a treasuring hoarded wealth of knowledge, resulting in constructive concepts for rising and upgrading services in the future. Researches show that solely many discontent customers really complain and provide the corporate a chance to correct itself. Others shift their loyalties. Hence, it becomes necessary to resolve complaints truthfully at the earliest, rather than taking a defensive approach. Structured client criticism management is one gospel for downside interference within the long run. This paper decides to develop one such customer care register model.

1.2 PROBLEM STATEMENT

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

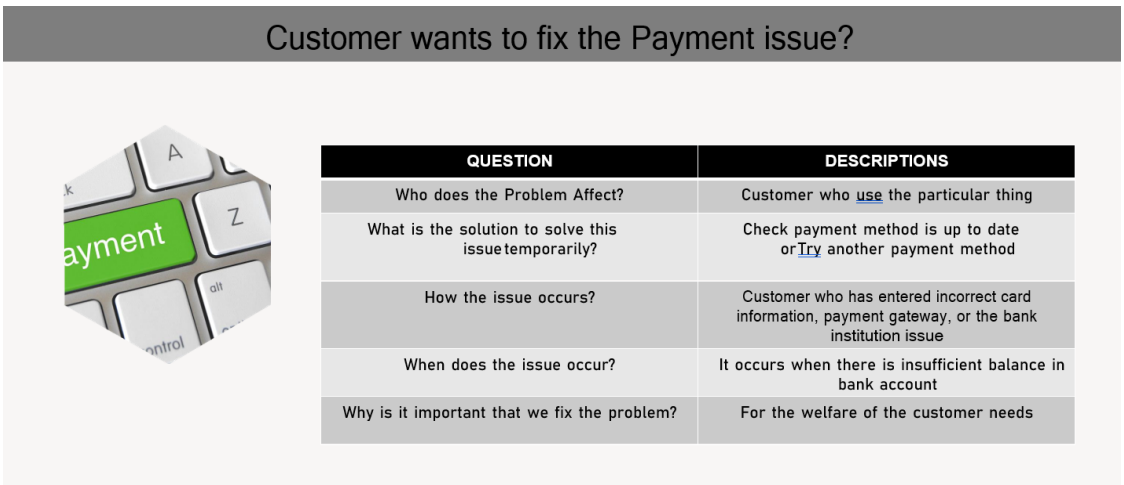
important.



Customer wants to fix a blue screen of death?

QUESTION	DESCRIPTIONS
Who does the Problem Affect?	Customer who use the particular thing
What are the boundaries of the problem?	Customer who use the thing for their personal work, office work etc
What is the issue?	Failure of Hardware or driver sometimes it may be in software too
When does the issue occur?	It frequently occurs after the customer installed new drivers or new piece of software
Where does the issue occur?	It often lies in the Hardware or one of the drivers
Why is it important that we fix the problem?	It is necessary to run the computer or Laptop to do their task or work in order to complete it.
What solution to solve this issue?	A quick reboot is sometimes enough to solve the problem

Figure 1.1 problem Statement of Blue screen



Customer wants to fix the Payment issue?

QUESTION	DESCRIPTIONS
Who does the Problem Affect?	Customer who use the particular thing
What is the solution to solve this issue temporarily?	Check payment method is up to date or Try another payment method
How the issue occurs?	Customer who has entered incorrect card information, payment gateway, or the bank institution issue
When does the issue occur?	It occurs when there is insufficient balance in bank account
Why is it important that we fix the problem?	For the welfare of the customer needs

Figure 1.2 Problem Statement of Payment Issue

1.3 WORKFLOW OF THE PROJECT

The Application has been developed to help the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to a customer, they will be notified with an email alert.Customers can view the status of the ticket till the service is provided. The main role and responsibility of the admin are to take care of the whole process. Starting from Admin login followed by the agent creation and assigning the customer's complaints. Finally, He will be able to track the work assigned to the agent and a notification will be sent to the customer. Customer can register for an

account.

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

1.4 CUSTOMER CARE REGISTRY

Customer service representatives are critically important to meeting your business goals and objectives, as well as ensuring the customers have a positive experience with your company. Customer service representatives listen to customer concerns, answer customer questions and provide information about the company's products and services. In some cases, customer service representatives may also take orders and set up new customer accounts. Given their prominent customer-facing role in the company, it is important to have a job description carefully tailored to attract candidates who have the necessary skills.

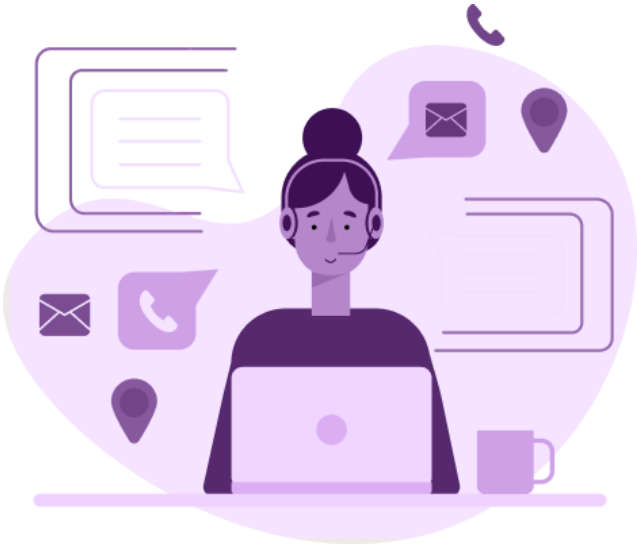


_Fig. 1.3 Customer Care Registry

1.5 USER LOGIN AND REGISTRATION

A registered user is a user of a website, program, or other systems who has previously registered. Registered users normally provide some sort of credentials to the system in order to prove their identity.

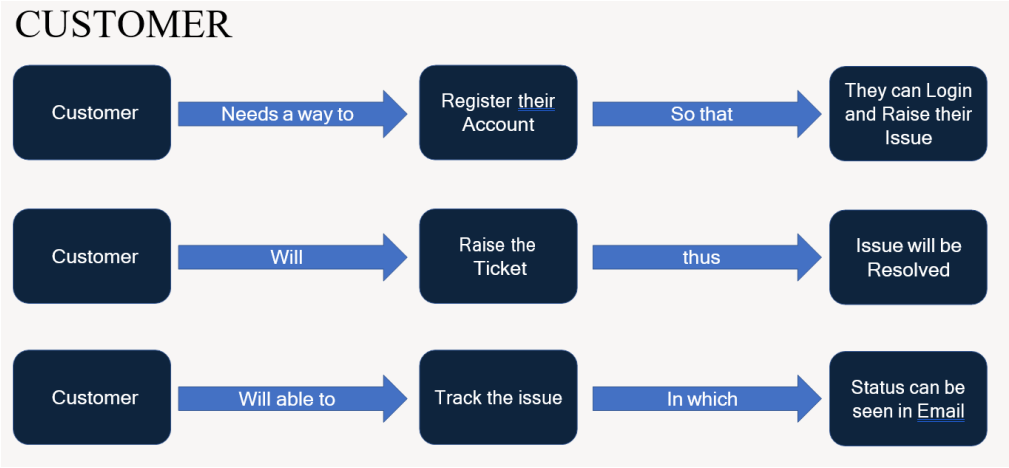


Figure 1.4 Problem statement of Customer

1.6 AGENT OR ADMIN PAGE

Admin is the role with the highest level of access to your website. Admins can add content on all pages and access all items in the Admin Toolbar. This means that Admins can control site-wide settings like the design of your website and the homepage layout.

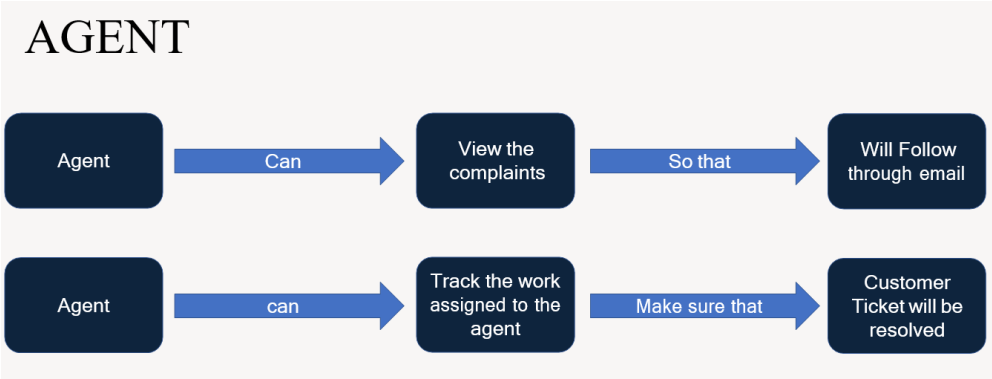


Figure 1.5 Problem statement of Agent

1.7 MERITS OF CUSTOMER CARE REGISTRY

- Customer loyalty
- Increase profit
- Customer Recommendation
- Increase conversion
- Improve public image

1.8 DEMERITS OF CUSTOMER CARE REGISTRY

- Lack of human approach
- The success of a technology depends on its design
- Unable to resolve complex issues

1.9 SOFTWARE REQUIRED

- Python,
- Flask,
- Docker.

1.10 SYSTEM REQUIRED

- 8GB RAM,
- Intel Core i3,
- OS-Windows/Linux/MAC,
- Laptop or Desktop

1.11 SUMMARY

This chapter provide the introduction of the customer care registry and the workflow of an project.

CHAPTER 2
LITERATURE REVIEW

2.1 INTRODUCTION

This chapter gives an overall description of literature survey that depicts various techniques to protect and store data on the network.

2.2 LITERATURE SURVEY

The following are some of the existing works relevant to the collaborative customer care registry.

REAL WORLD SMART CHATBOT FOR CUSTOMER CARE USING A SOFTWARE AS A SERVICE (SAAS) ARCHITECTURE

This journal employee chatbot for customer care. This is done by providing a human way interaction using LUIS and cognitive services. The tools and algorithms they were used AWS Public Cloud, AWS Lambda, API Gateway, LUIS, Ejabberd Chatbot. Cloud Computing, Machine Learning are the technologies implemented in formal. This proposes a robust,scalable, and extensible architecture with a technology stack consisting of the Ejabberd Server. The Ejabberd server makes creates the room functionality where the customer needs to be persistent over time in that room.

AN INTELLIGENT CLOUD BASED CUSTOMER RELATIONSHIP MANAGEMENT SYSTEM TODETERMINE FLEXIBLE PRICING FOR CUSTOMER RETENTION

This paper proposes that the customers are categorized based on purchase behaviors, historical ordering patterns and frequency of purchase customize customer care and promotions are given. The tools and algorithms they were used Intelligent Cloud- based Customer Relationship Management. Cloud Computing, Artificial intelligence are the technologies implemented in formal. Customer care is given based upon purchase behaviors, features of the product purchased without any interaction.

CHATBOT FOR CUSTOMER SERVICE

In this paper customer trust chatbots to provide the required support. Chatbots represent a potential means for automating customer service. The tools and algorithms they were used Chatbot and Java-script. Cloud Computing, Artificial intelligence and Machine Learning are the technologies implemented in formal. This provides automated customer service with the use of the cloud.

ARTIFICIAL INTELLIGENCE REPLACING HUMAN CUSTOMER SERVICE

This journal Chatbots for customer care registry using Artificial intelligence. This assists consumers in decision making. Based on the computers-are- social-actors paradigm. The tools and algorithms they were used Chatbots, Python, Mongo_DB. Cloud Computing, Artificial intelligence and Machine Learning are the technologies implemented in formal. Maintain Flexibility and focus on their customers. The use of chatbots in service interactions may raise greater consumer concerns regarding privacy risk issues.

IMPLEMENTING CONTINUOUS CUSTOMERCARE

In this paper, we employ the software as a service(SaaS) model which introduces drastic improvement to the situation, as the service provider can now have direct access to the user data and analyze it if agreed appropriately with the customer. The tools and algorithms they were used Java Script, HTML, Google Analytics. Cloud Computinf5rg and Machine Learning are the technologies implemented in formal. Feedback loops are used that allow the service provider to capture feedback at the point of experience. One way to discover is to conduct continual end-user experience monitoring to determine if users are happy. It is not always easy for SaaS providers to know what customers are experiencing.

2.3 SUMMARY

This chapter provides a brief description on the customer care registry. The various algorithm used are also briefly narrated.The issues and challenges in social network are also explained.

CHAPTER 3
INTRODUCTION

3.1 INTRODUCTION OF IDEATION AND DESIGN

This chapter gives an overall description of design concepts and the various methodologies and the Innovated ideas which should took place.

3.2 EMPATHY MAP

The empathy map canvas to capture the user Pains & Gains, Prepare list of problem statements.

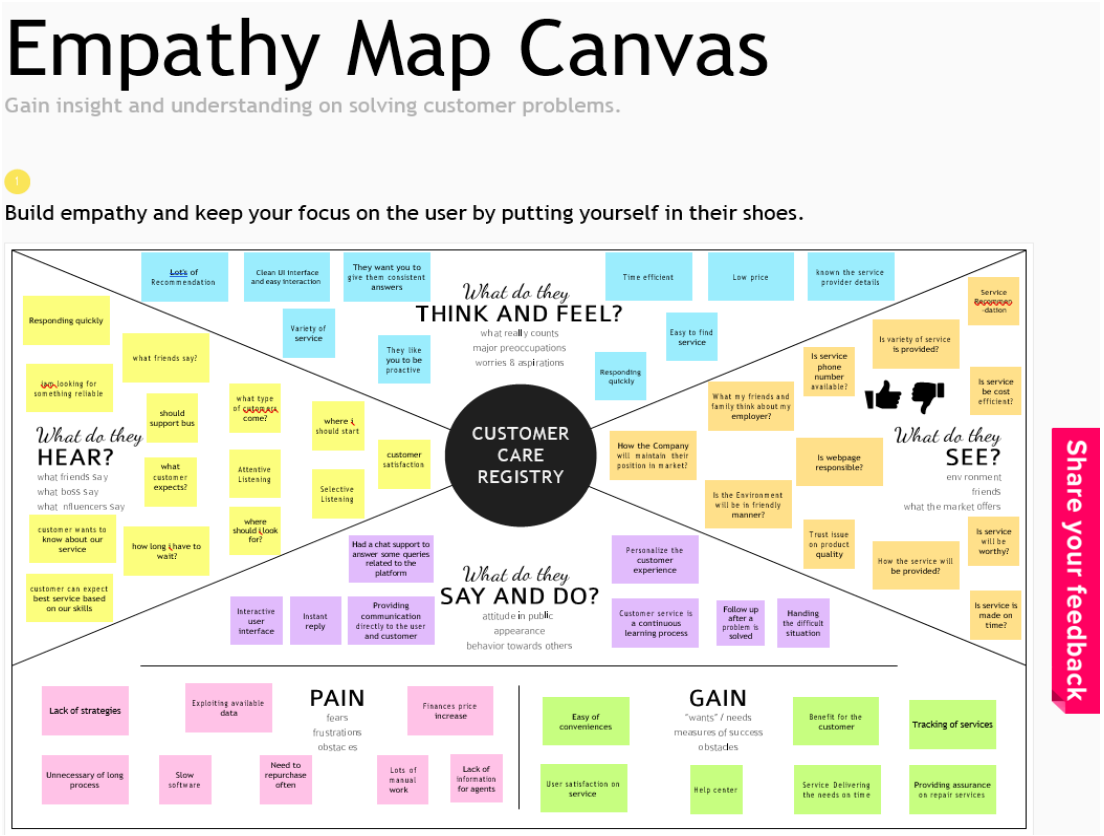


Figure 3.1 Empathy Map Canvas

3.3 BRAINSTROMING

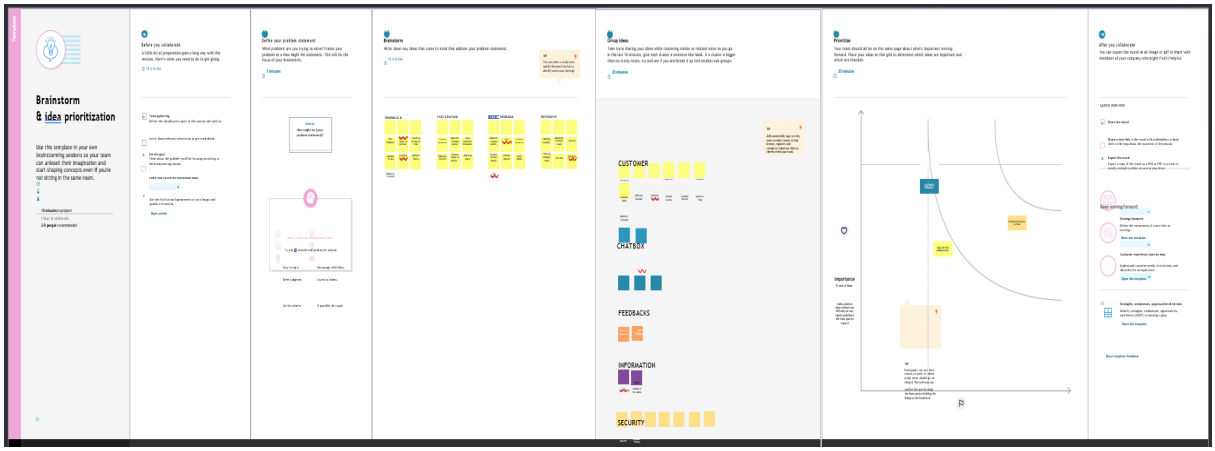


Figure 3.2 BrainStroming

3.4 PROBLEM SOLUTION FIT

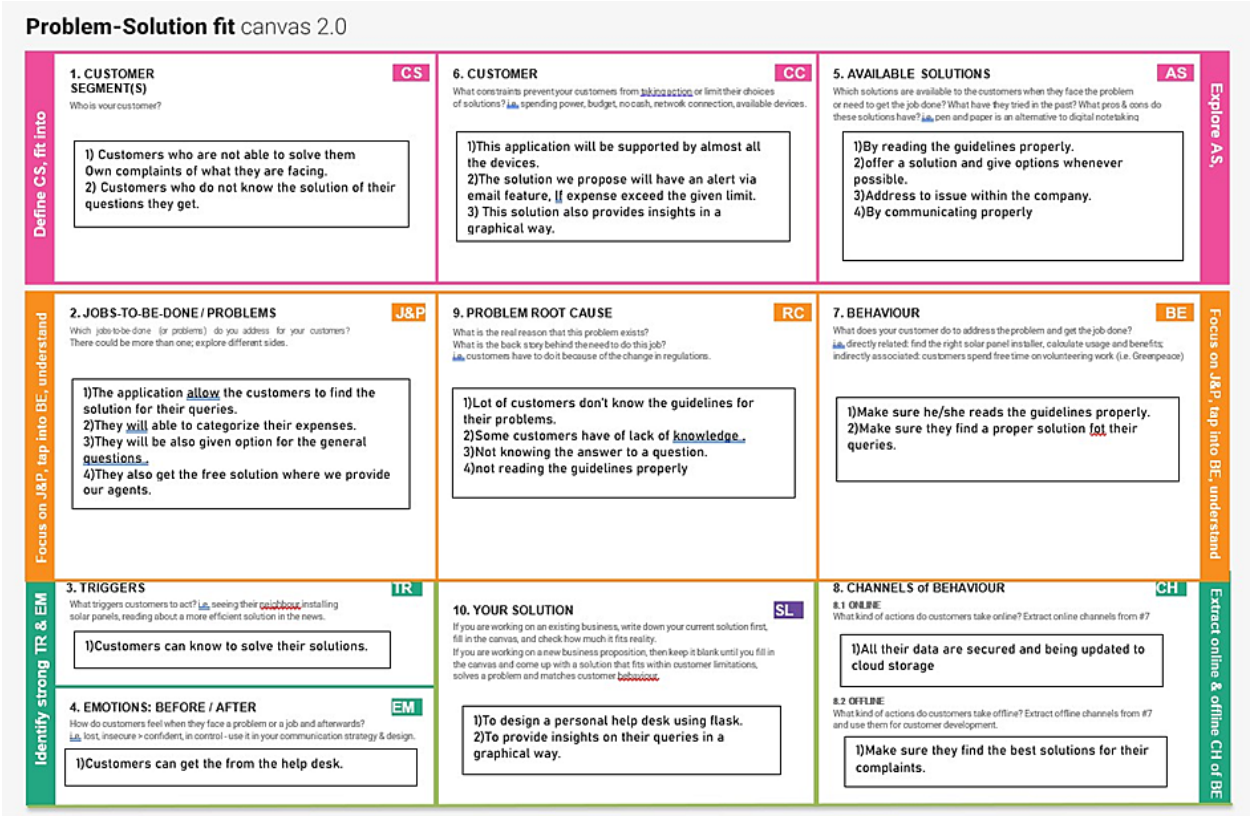


Figure 3.3 Problem Solution Fit

3.5 PROPOSED SOLUTION

Proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.

S.NO.	PARAMETER	DESCRIPTION
01	Problem Statement (Problem to be solved)	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.
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

3.6 SOLUTION ARCHITECTURE

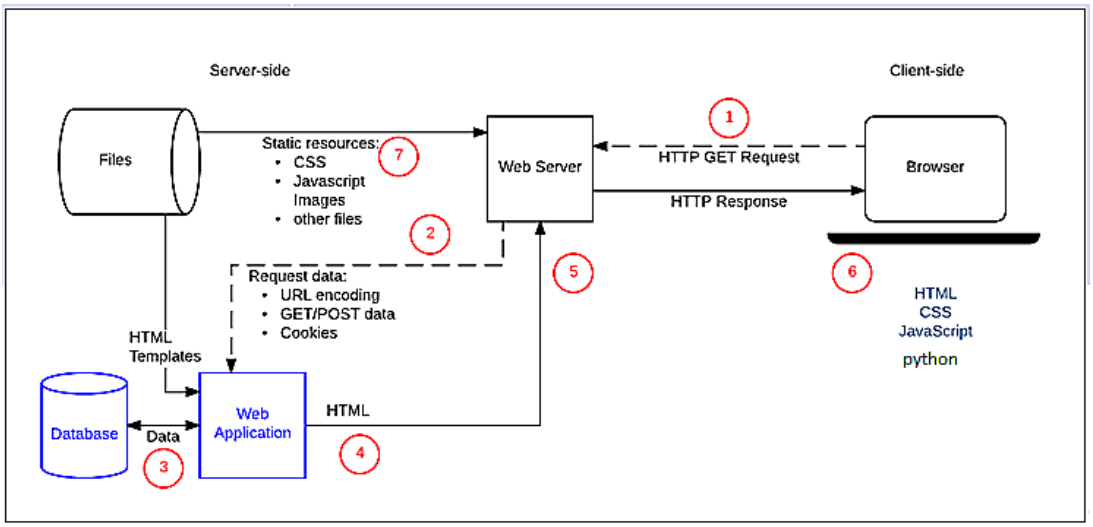


Figure 3.4 Solution Architecture

3.7 TECHNICAL ARCHITECTURE

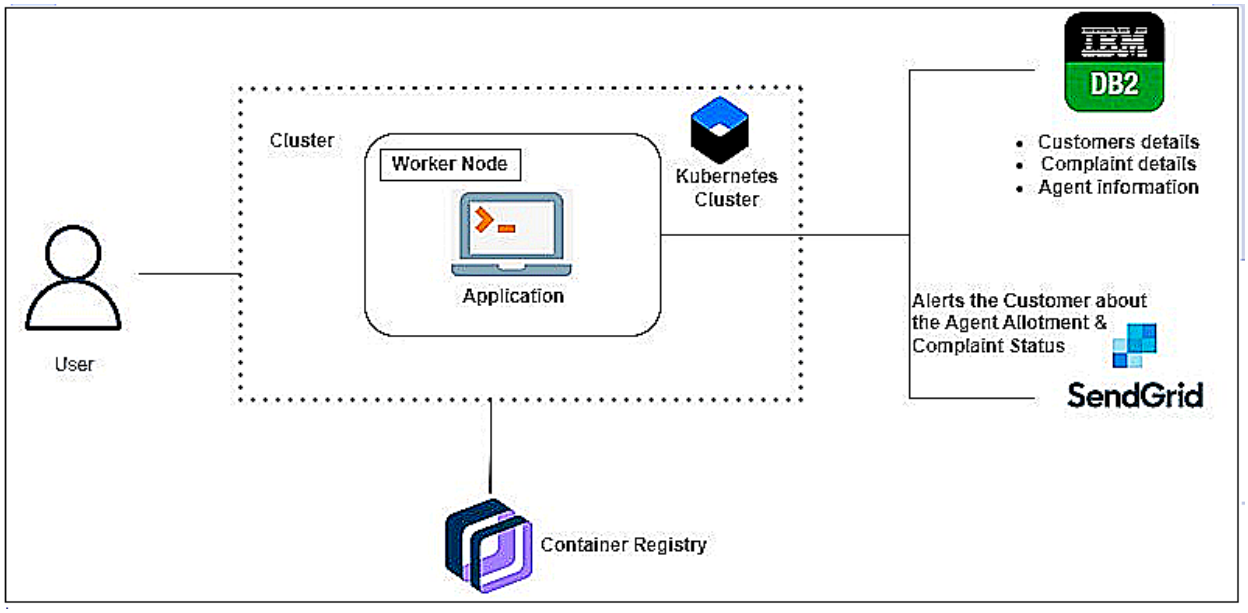


Figure 3.4 Technical Architecture

3.8 CUSTOMER JOURNEY

Customer journey maps to understand the user interactions & experiences with the application (entry to exit).

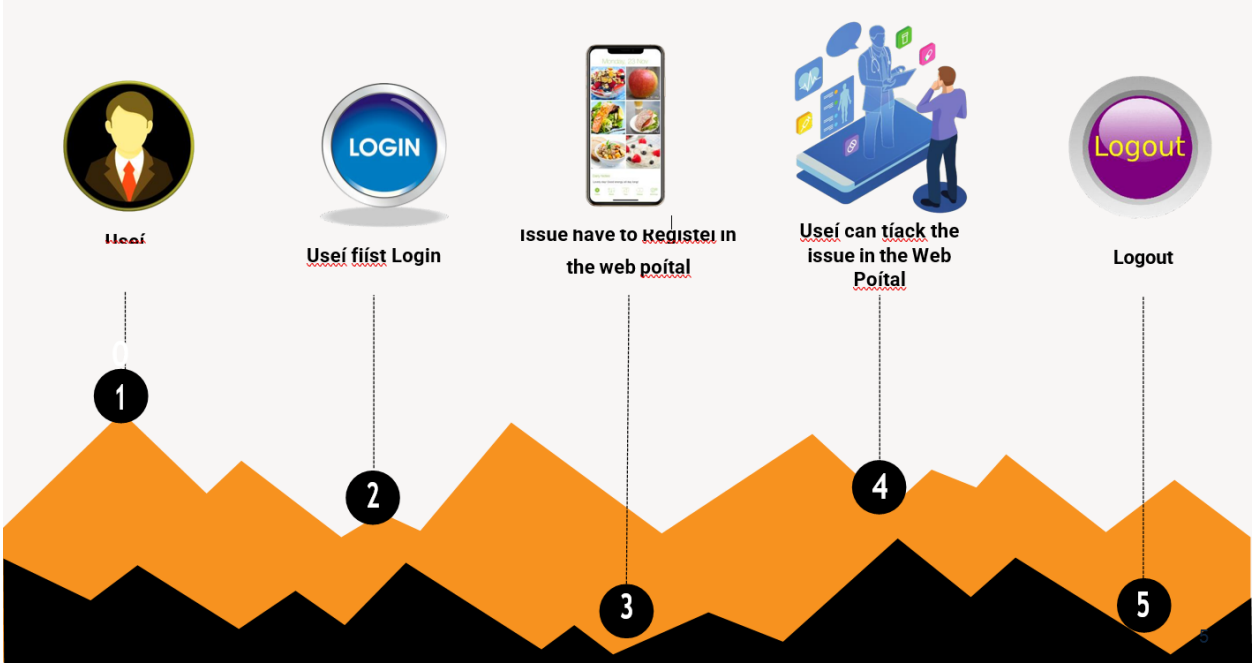


Figure 3.5 Customer Journey Map

STAGE	AWARENESS	CONSIDERATION	DECISION	SERVICE	LOYALTY
CUSTOMER ACTIVITIES	see social media campaign Hear about from friends	Conduct reach, compare features and pricing	Make a purchase	Contact customer service, Documentation,read product and service	Share the experience
TOUCHPOINTS	Social media, Traditional media , word of mouth	Social media, Websites	Website, Mobile app	Chatbot, Email notification	Social media,word of mouth Review sites
CUSTOMER EXPERIENCE	Interested, Hesitant	Curious, Excited	Excited	Frustrated	Satisfied, Excited
KPIS	customer feedback	New website visitors	Conversional rate	Waiting time, customer service score	Customer satisfaction score
RESPONSIBLE	Communications	Communications	Customer service	Customer service	Customer service, Customer success

3.9 DATAFLOW DIAGRAM

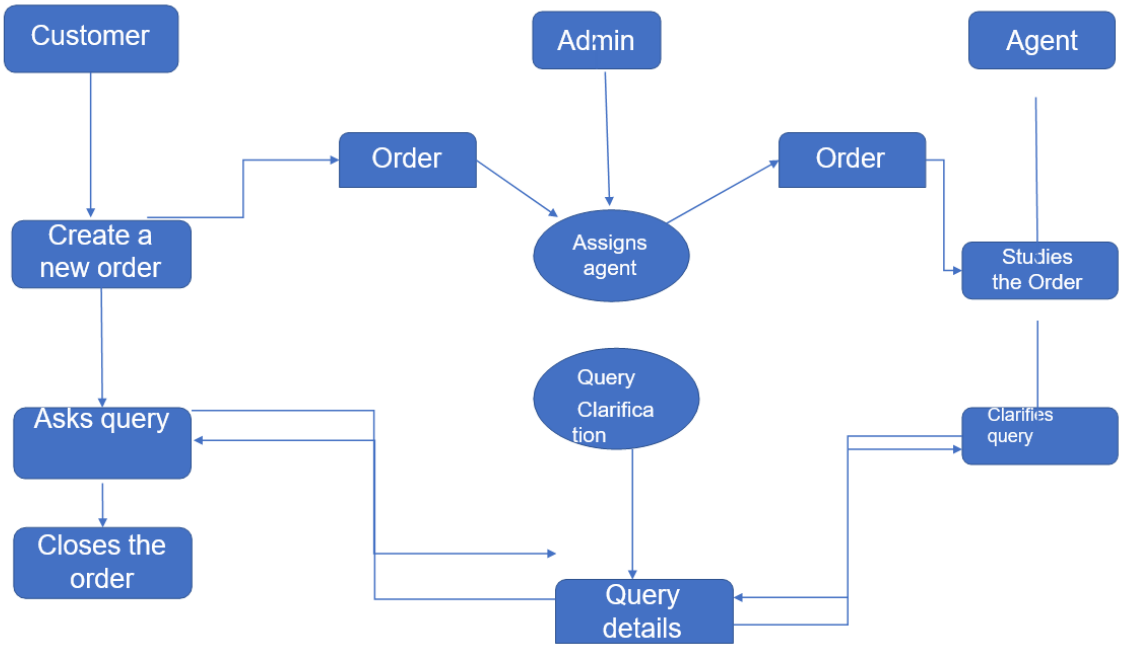


Figure 3.7 Data Flow Diagram

3.10 USER STORIES

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

3.11 TECHNOLOGY ARCHITECTURE

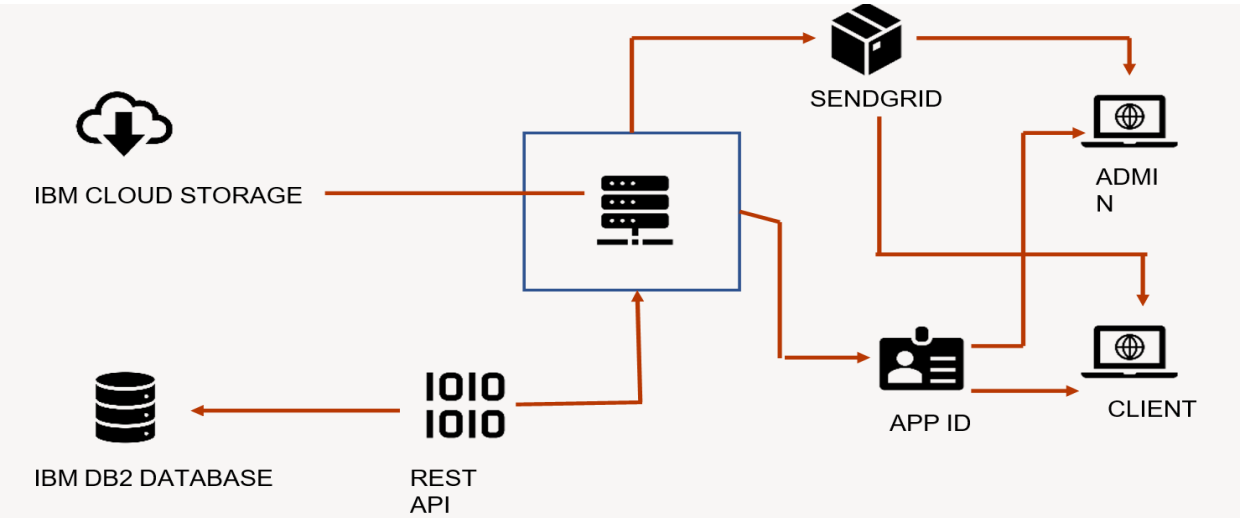


Figure 3.7 Technology Architecture

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 / ReactJs etc.
2.	Application Logic1	Logic for a process in the application	Python
3.	Application Logic2	Logic for a process in the application	IBM WatsonSTT service
4.	Application Logic3	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 Cloud-ant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Serviceor LocalFilesystem

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

3.12 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution

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

3.13 NON-FUNCTIONAL REQUIREMENTS

Following are the non-functional requirements of the proposed solution

FR No	Non-Functional Requirement	Description
1	Usability	To provide the solution to the problem
2	Security	Track of login authentication
3	Reliability	Tracking of decade status through email
4	Performance	Effective development of web application
5	Availability	24/7 service
6	Scalability	Agents scalability as per the number of customers

3.14 APPLICATION CHARACTERISTICS

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

3.15 SUMMARY

This chapter describes the existing and proposed formation all design solution.

CHAPTER 4
INTRODUCTION

4.1 INTRODUCTION OF PLANNING AND WEB CONNECTIVITY

This chapter gives an overall description planning of project and web connection to , IBMDB2, send-grid and kubernet.

4.2 Implementing Web Application

Implement Modules of a project

- Registration -User, Admin, Agent
- Login -User, Admin, Agent Create Complaint/Ticket
- Dashboard to show all the Tickets
- Assign the agent to Ticket/Complaint
- Send an email alert to the user on the Ticket/Complaint status

4.3 Create UI To Interact With Application

Create UI to interact with the application

- Registration Page -User, Admin, Agent
- Login Page -User, Admin, Agent
- Forgot Password page
- Dashboard to show Tickets
- Ticket Details page

Create the IBM Db2 service in the IBM cloud and connect the python code with DB.

Integrating SendGrid service to the application.

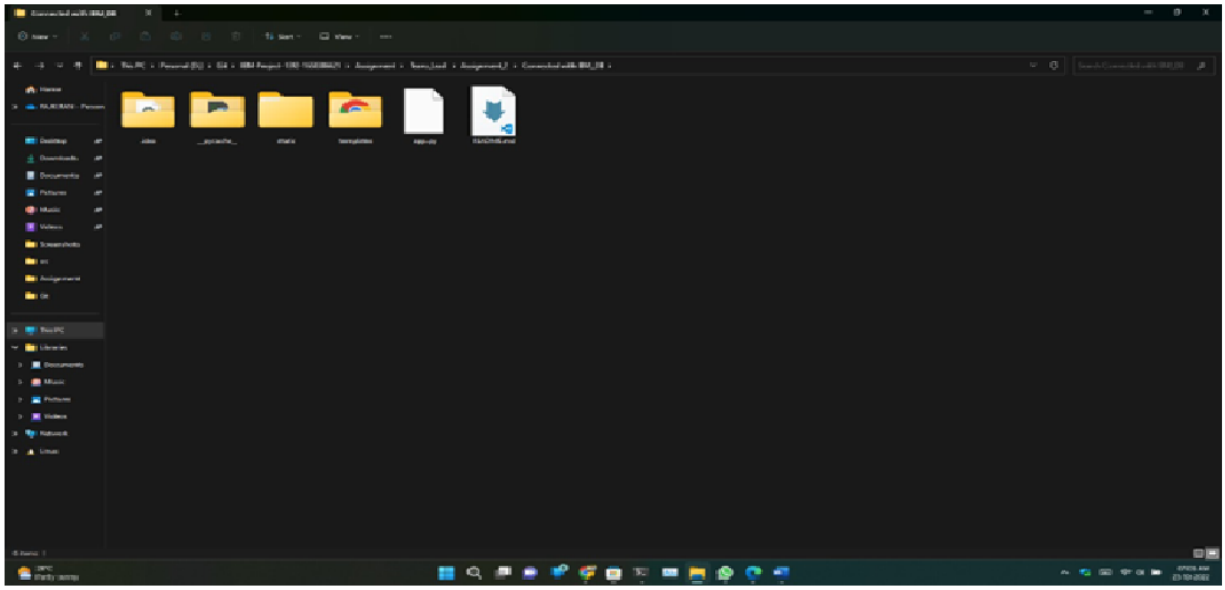
Containerize a Flask application by using Docker and deploy it to the IBM Cloud Kubernets Service

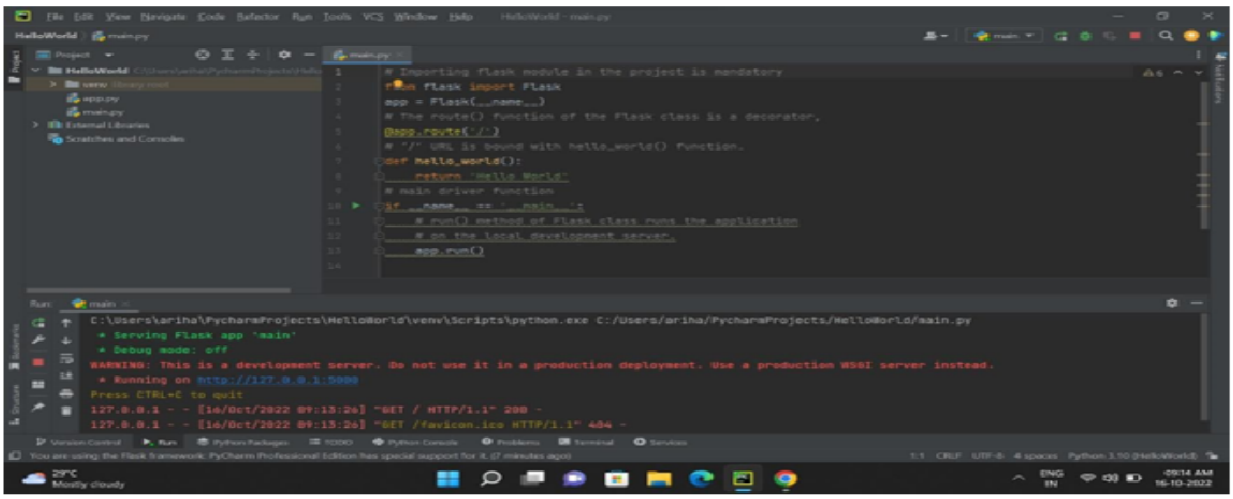
Upload the Image to IBM Container Registry.

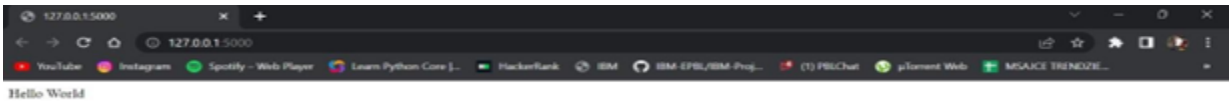
Once the image is uploaded to the IBM container registry deploy the image to IBM Kubernetes Cluster

4.4 ENVIRONMENTAL SETUP

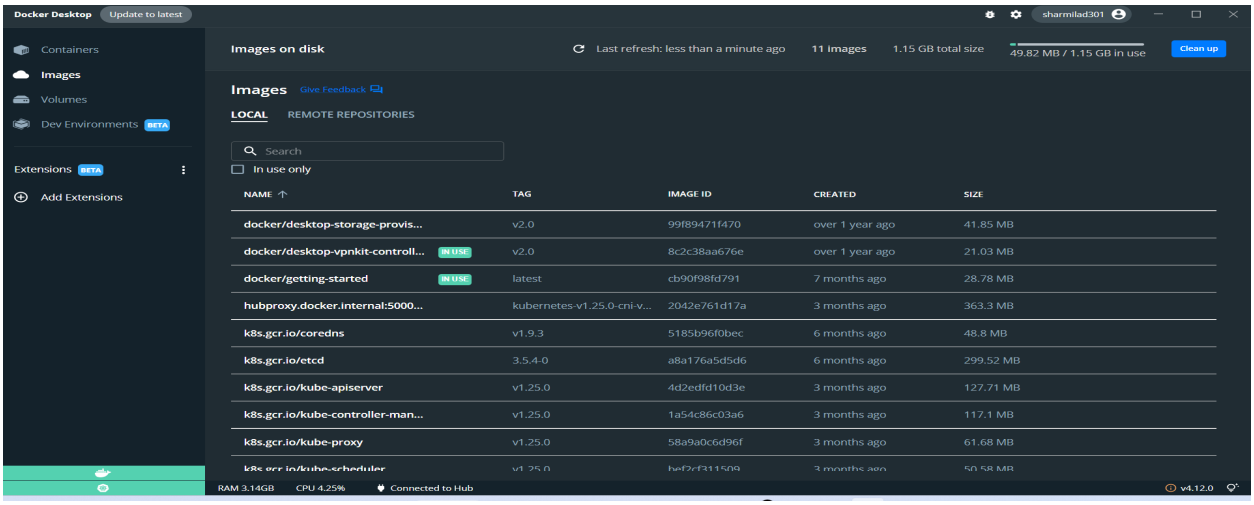
a. FLASK



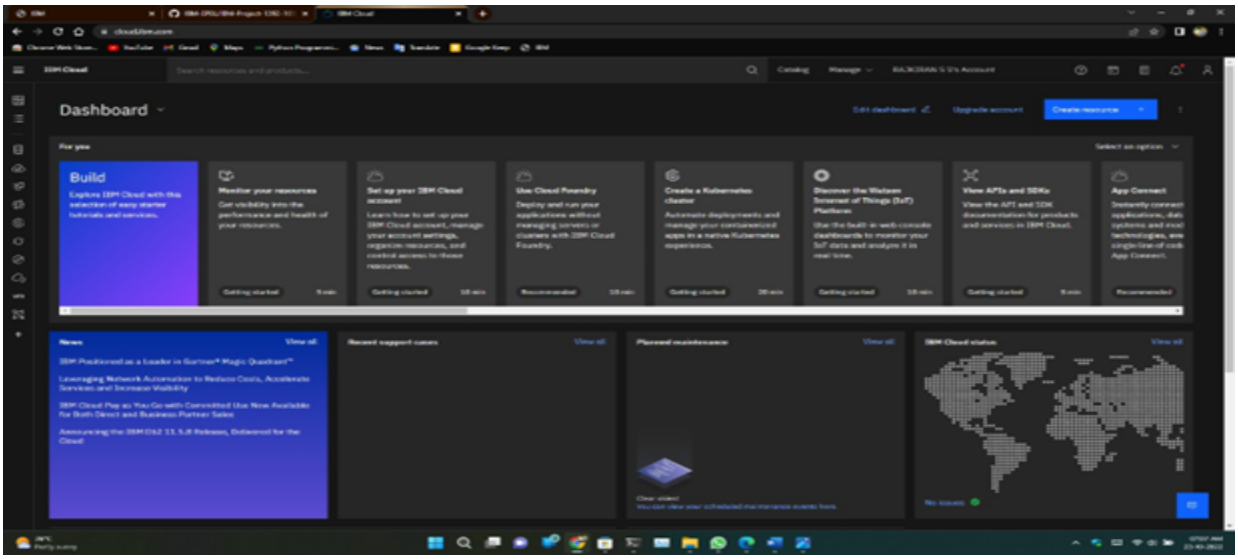




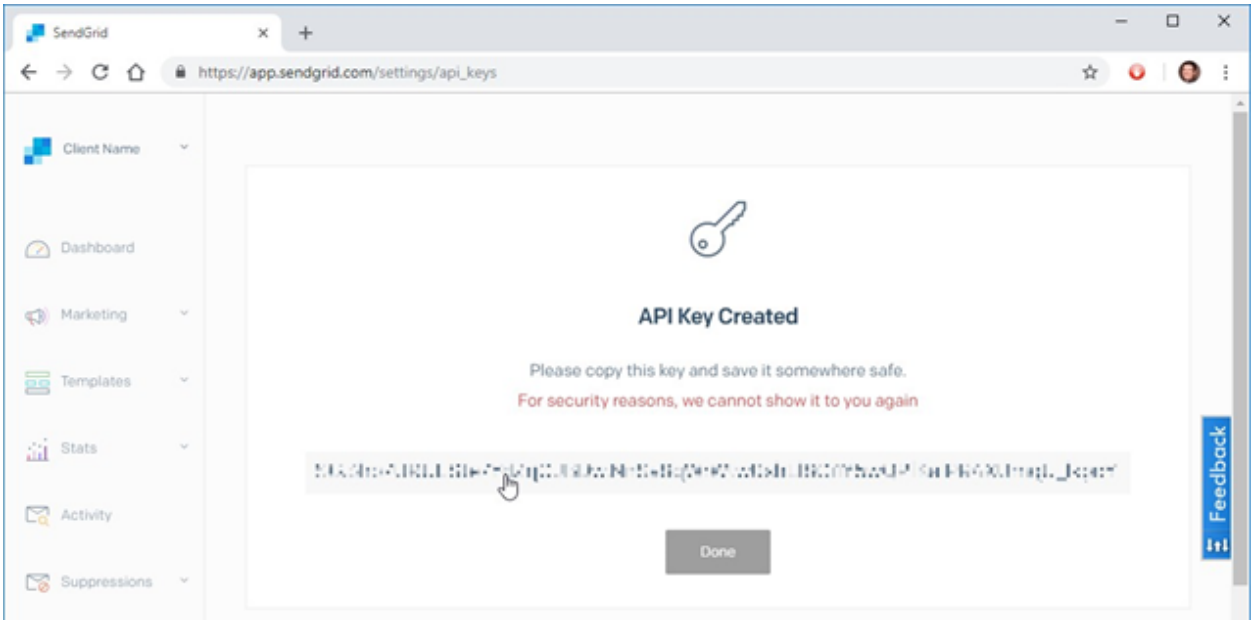
b. DOCKER SETUP



c. COLUD ACCOUNT CREATION



d. SENDGRID ACCOUNTCREATION



4.5 MILESTONE AND ACTIVITY LIST

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gatheringinformation by referring the,technical papers, research publications etc.	28 SEPTEMBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvasto capture the user Pains & Gains, Prepare list of problemstatements	24 SEPTEMBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top three ideas based on the feasibility andimportance.	25 SEPTEMBER 2022

Proposed Solution	Prepare the proposed solution document, which includes thenovelty, feasibility of idea, business model, social impact,scalability of solution, etc.	23 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	30 SEPTEMBER 2022
Solution Architecture	Prepare solution architecture document.	28 SEPTEMBER 2022

Customer Journey	Prepare the customer journey maps to understand the user interactions and experiences with the application (entry to exit).	20 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	8 OCTOBER2022
Data FlowDiagrams	Draw the data flow diagrams and submit for review.	9 OCTOBER2022
Technology Architecture	Prepare the technology architecture diagram.	10 OCTOBER 2022
Prepare Milestone and ActivityList	Prepare the milestones &activity list of the project.	22 OCTOBER2022
Project Development - Delivery of Sprint-1, 2, 3 &4	Develop and submit the developed code by testing it.	16 NOVEMBER 2022
Final Deliverables	Documentation and demo with web link	19 NOVEMBER 2022

4.6 PRODUCT BACKLOG,SPRINT SCHEDULE, AND ESTIMATION

SPRINT	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priori ty	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	SHARMILA D, SEVVANTHI.
sprint 2	Admin panel	USN-2	The role of the admin is to check out the database about the availability and have a track of all the things that the users are going service	20	High	SHARMILA D, THATCHAYANI.
sprint 3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding theservices. Get the recommendations based on information provided by the user.	20	High	SHARMILA D, NAVIN KUMAR.
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes anddeployment the application.Create the documentation and final submit the application	20	High	SHARMILA D, SHOBANA.

4.7 PROJECT TRACKER,VELOCITY & BURNDOWNCHART

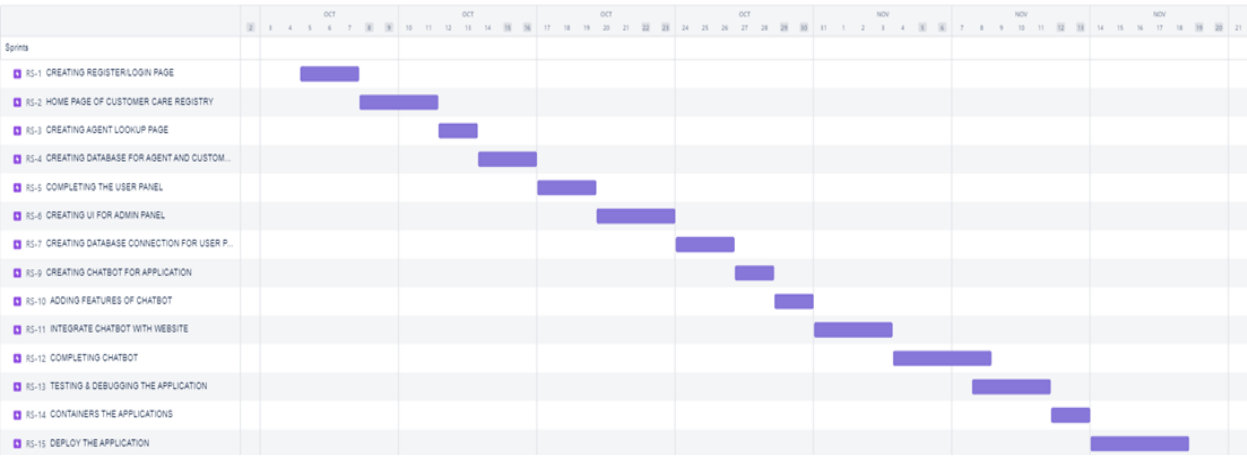
Sprint	Total StoryPoints	Duration	Sprint Start Date	SprintEnd Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	29 Oct 2022	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	05 Nov 2022	05 Nov 2022
Sprint-3	20	6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

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let us calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

4.8 BURNDOWN CHART



4.9 SUMMARY

the overall setup connectivity and planning of project is explained.

CHAPTER 5
SOURCE CODE AND OUTPUT

5.1 INTRODUCTION

This chapter gives an overall description of design concepts and the various methodologies and the Innovated ideas which should took place.

5.2 SOURCE CODE

```
from __future__ import print_function
from audio import add
import date time
from unicode data import name
from sib_api_v3_sdk.rest import ApiException
from print import print
from flask import Flask, render_template, request, redirect, url_for,
session, flash
from markup safe import escape
from flask import *
import ibm_db
import date time

conn =
ibm_db.connect("DATABASE=;HOSTNAME=;PORT=;SECURITY=SSL;SSLServerCertifi
cate=;UID=;PWD=", '', '')
print(conn)
print("connection successful...")

app = Flask(__name__)
app.secret_key = 'your secret key'


@app.route('/')
def home():
    message = "TEAM ID : PNT2022TMID26814" +" "+ "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 sign in -_page():
    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 signup page():
    return render_template('signuppage.html')


@app.route('/agentRegister', methods=['POST', 'GET'])
def agentRegister():
    return render_template('agentregister.htm')

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


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


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

    try:
        global random number
        ida = request.form['custid']
        print(ida)
        global id
        id = ida
        sql = "SELECT EMAIL,NAME FROM Customer WHERE id=?"
        stmt = ibm_db.prepare(conn, sql)
```

```
        ibm_db.bind_param(stmt, 1, ida)
        ibm_db.execute(stmt)
        emailf = ibm_db.fetch_both(stmt)
        while emailf != False:
            e = emailf[0]
            n = emailf[1]
            break

        configuration = sib_api_v3_sdk.Configuration()
        configuration.api_key['api-key'] =

        api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
            sib_api_v3_sdk.ApiClient(configuration))
        subject = "Verification for Password"
        html_content = "<html><body><h1>Your verification Code is :
<h2>" + \
            str(randomnumber)+"</h2> </h1> </body></html>"
        sender = {"name": "IBM CUSTOMER CARE REGISTRY",
                  "email": "ibmdemo6@yahoo.com"}
        to = [{"email": e, "name": n}]
        reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
        headers = {"Some-Custom-Name": "unique-id-1234"}
        params = {"parameter": "My param value",
                  "subject": "Email Verification"}
        send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
            to=to, reply_to=reply_to, headers=headers,
html_content=html_content, params=params, sender=sender,
subject=subject)

        api_response =
api_instance.send_transac_email(send_smtp_email)

        pprint(api_response)
        message = "Email send to:"+e+" for password"
        flash(message, "success")

    except ApiException as e:
        print("Exception when calling SMTPApi->send_transac_email:
%s\n" % e)
        flash("Error in sending mail")
    except:
        flash("Your didn't Signin with this account")
    finally:
        return render_template('forgot.html')
```

```
@app.route('/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 WHERE id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, ida)
        ibm_db.execute(stmt)
        emailf = ibm_db.fetch_both(stmt)
        while emailf != False:
            e = emailf[0]
            n = emailf[1]
            break

        configuration = sib_api_v3_sdk.Configuration()
        configuration.api_key['api-key'] =

        api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
            sib_api_v3_sdk.ApiClient(configuration))
        subject = "Verification for Password"
        html_content = "<html><body><h1>Your verification Code is :
<h2>" + \
            str(randomnumber)+"</h2> </h1> </body></html>"
        sender = {"name": "IBM CUSTOMER CARE REGISTRY",
                  "email": "ibmdemo6@yahoo.com"}
        to = [{"email": e, "name": n}]
        reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
        headers = {"Some-Custom-Name": "unique-id-1234"}
        params = {"parameter": "My param value",
                  "subject": "Email Verification"}
        send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
            to=to, reply_to=reply_to, headers=headers,
html_content=html_content, params=params, sender=sender,
subject=subject)

        api_response =
api_instance.send_transac_email(send_smtp_email)
```

```
        pprint(api_response)
        message = "Email send to:"+e+" for 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']
            global hello
            hello = id
            password = request.form['password']
            print(id, password)
            if id == '1111' and password == '1111':
                return redirect(url_for('admin'))

            sql = f"select * from customer where id='{escape(id)}' and
password='{escape(password)}'"
            stmt = ibm_db.exec_immediate(conn, sql)
            data = ibm_db.fetch_both(stmt)

            if data:
                session["name"] = escape(id)
                session["password"] = escape(password)
                return redirect(url_for("welcome"))

            else:
                flash("Mismatch in credetials", "danger")
        except:
            flash("Error in Insertion operation", "danger")

    return render_template('signinpage.html')

@app.route('/welcome', methods=['POST', 'GET'])
def welcome():
    try:
        id = hello
        sql = "SELECT
```

```
ID,DATE,TOPIC,SERVICE_TYPE,SERVICE_AGENT,DESCRIPTION,STATUS FROM ISSUE
WHERE CUSTOMER_ID =?"
        agent = []
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        otpf = ibm_db.fetch_both(stmt)
        while otpf != False:
            agent.append(otpf)
            otpf = ibm_db.fetch_both(stmt)

        sql = "SELECT COUNT(*) FROM ISSUE WHERE CUSTOMER_ID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        t = ibm_db.fetch_both(stmt)

        return
render_template("welcome.html",agent=agent,message=t[0])
    except:
        return render_template("welcome.html")

@app.route('/loginagent', methods=['GET', 'POST'])
def loginagent():
    if request.method == 'POST':
        try:
            global loginagent
            id = request.form['idn']
            loginagent = id
            password = request.form['password']

            sql = f"select * from AGENT where id='{escape(id)}' and
password='{escape(password)}'"
            stmt = ibm_db.exec_immediate(conn, sql)
            data = ibm_db.fetch_both(stmt)

            if data:
                session["name"] = escape(id)
                session["password"] = escape(password)
                return redirect(url_for("agentwelcome"))

            else:
                flash("Mismatch in credetials", "danger")
        except:
```

```
            flash("Error in Insertion operation", "danger")

    return render_template("signinpageagent.html")

@app.route('/delete/<ID>')
def delete(ID):
    sql = f"select * from customer where Id='{escape(ID)}'"
    print(sql)
    stmt = ibm_db.exec_immediate(conn, sql)
    student = ibm_db.fetch_row(stmt)
    if student:
        sql = f"delete from customer where id='{escape(ID)}'"
        stmt = ibm_db.exec_immediate(conn, sql)

        flash("Delected Successfully", "success")
        return redirect(url_for("admin"))



@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,COUN
TRY,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 = " <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>Customer Care
Registry</h3> </body></html>"
                sender = {"name": "IBM CUSTOMER CARE REGISTRY",
                  "email": "ibmdemo6@yahoo.com"}
                to = [{"email": email, "name": name1}]
                reply_to = {"email": "ibmdemo6@yahoo.com", "name":
"IBM"}
                headers = {"Some-Custom-Name": "unique-id-1234"}
```

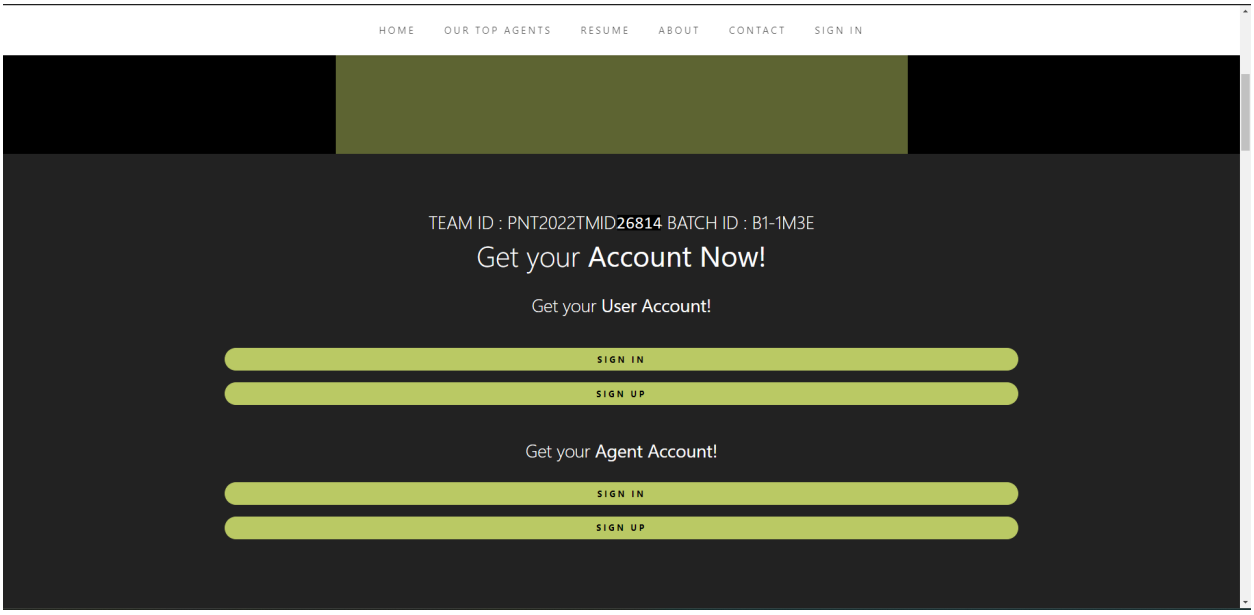
```
                params = {"parameter": "My param value",
                  "subject": "Email Verification"}
                send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
                to=to, reply_to=reply_to, headers=headers,
html_content=html_content, params=params, sender=sender,
subject=subject)

                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/<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=5000, debug=True)
```

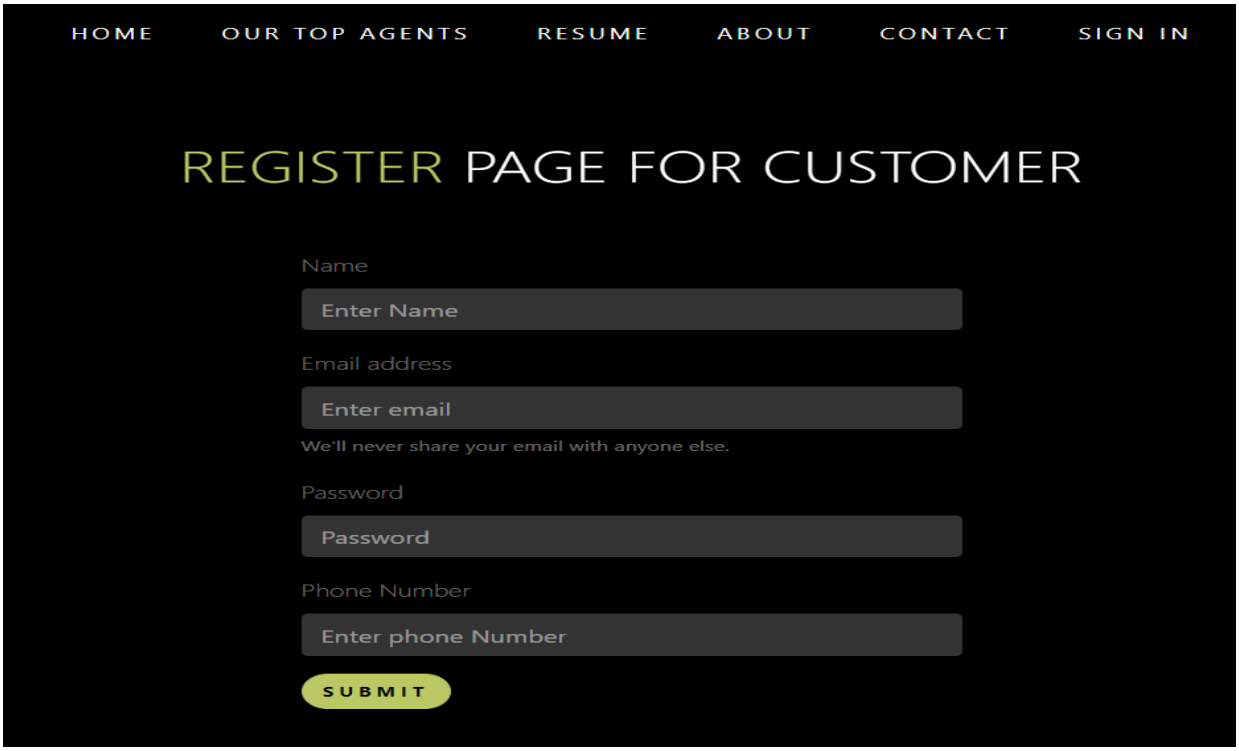
5.3 OUTPUT

1. website login

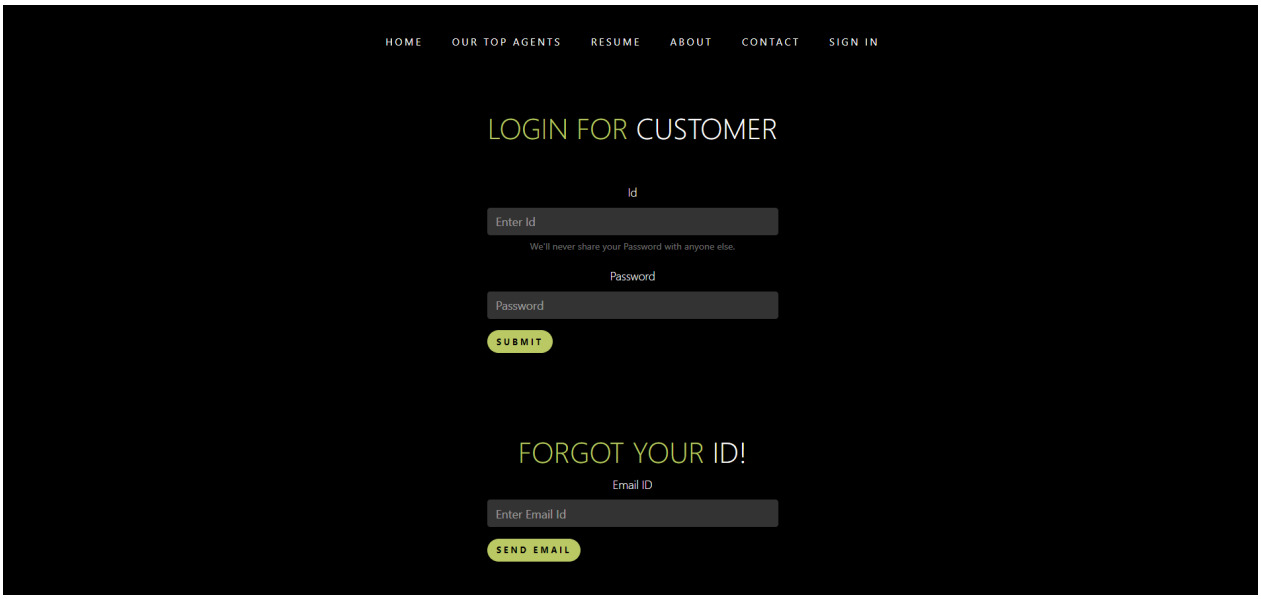
Customer Care Registry



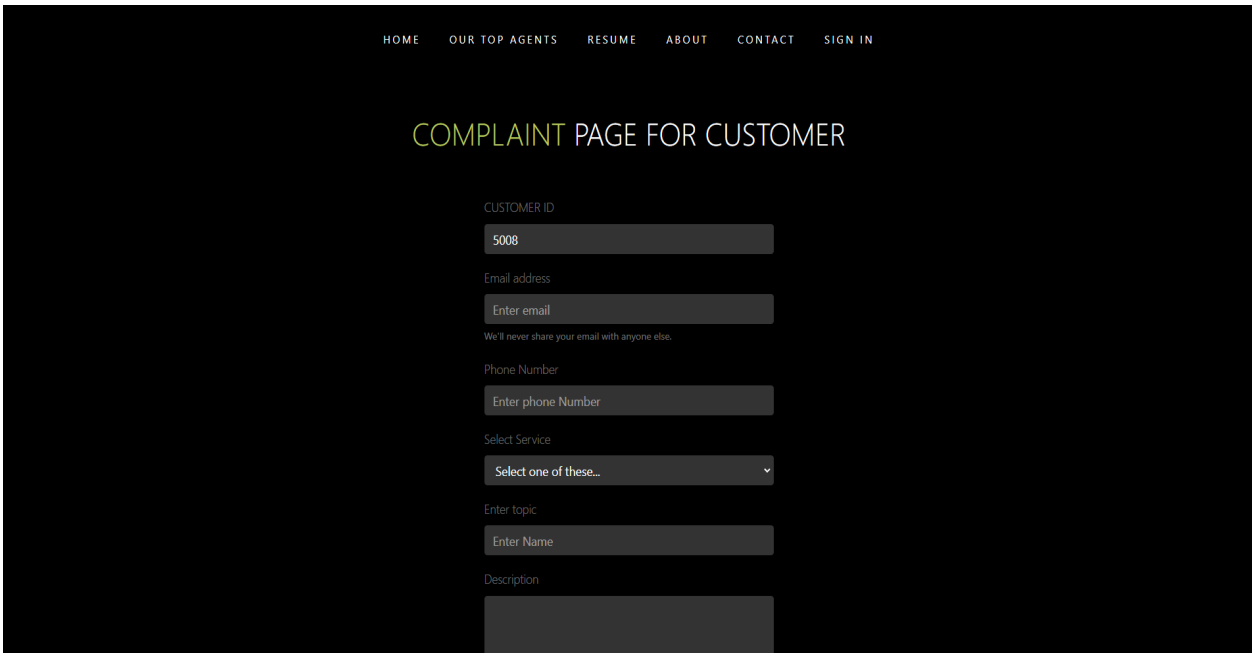
2. Registration of customer page



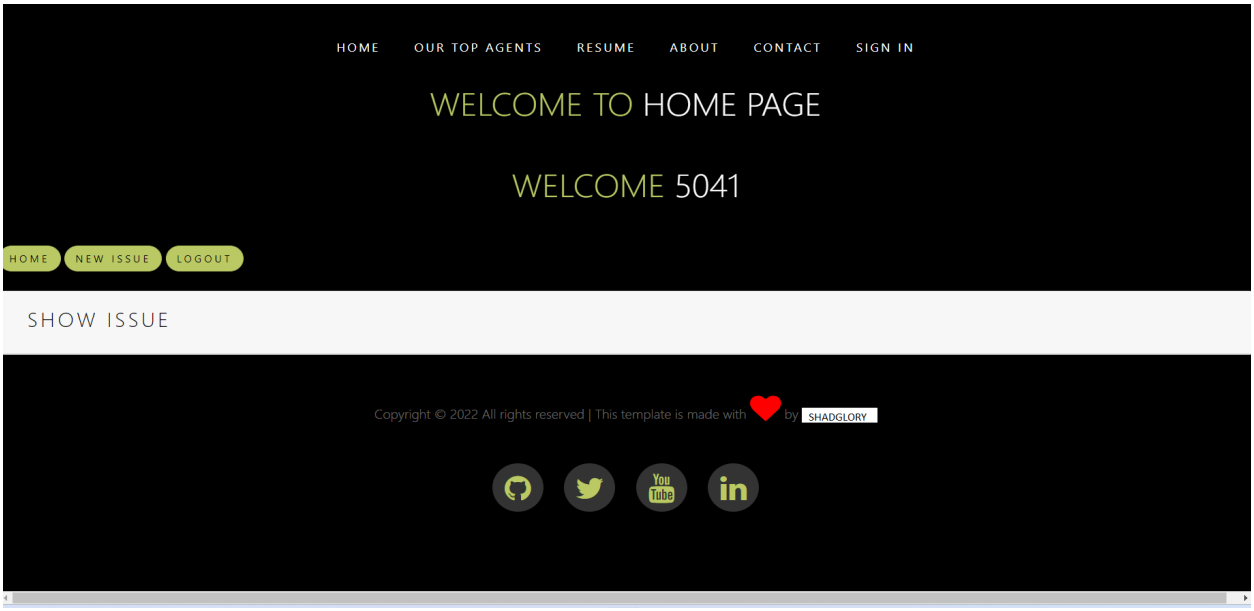
3. login for Customer



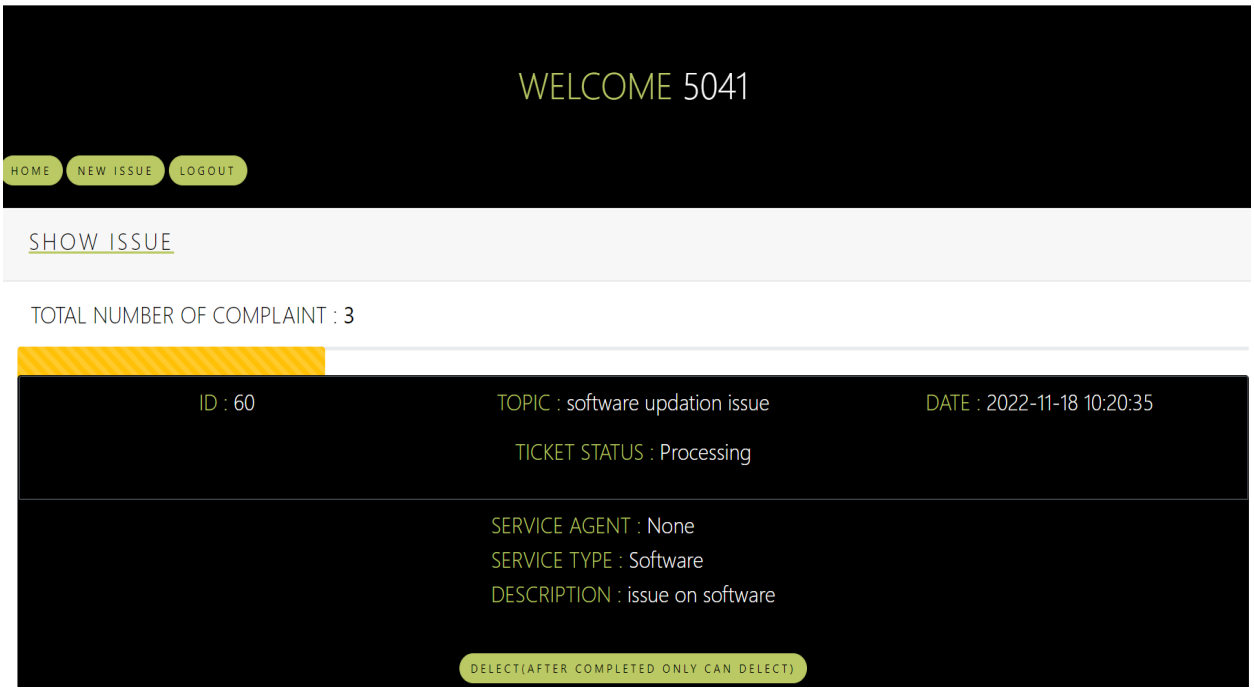
4. complaint page for customers



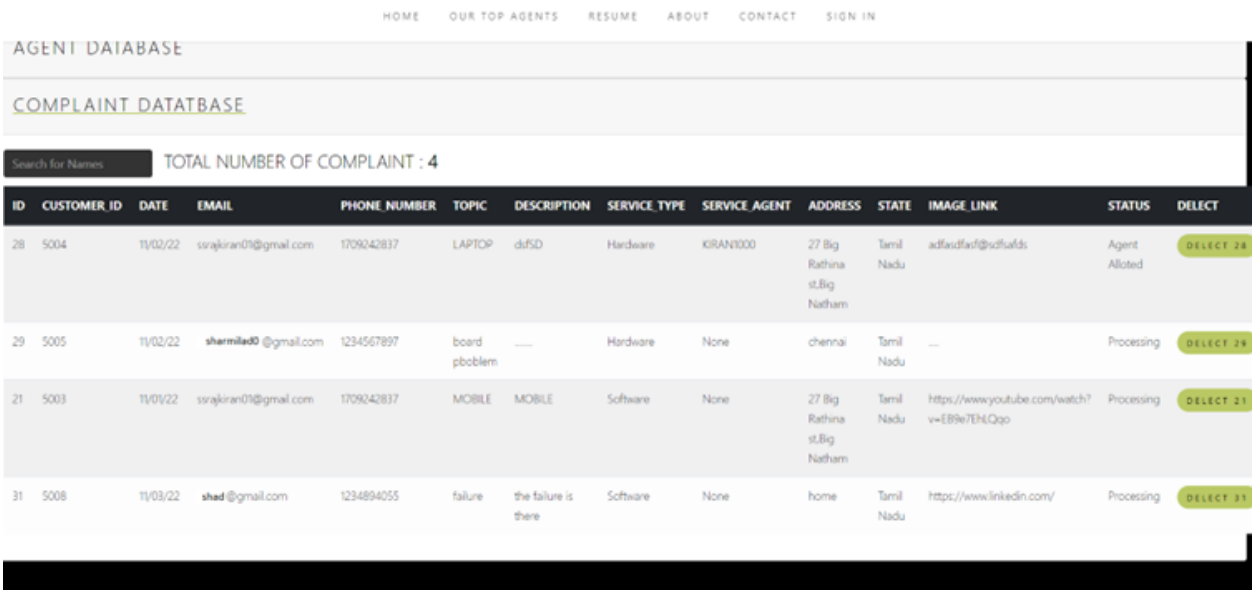
5. Home page of customer



6. Storing Information

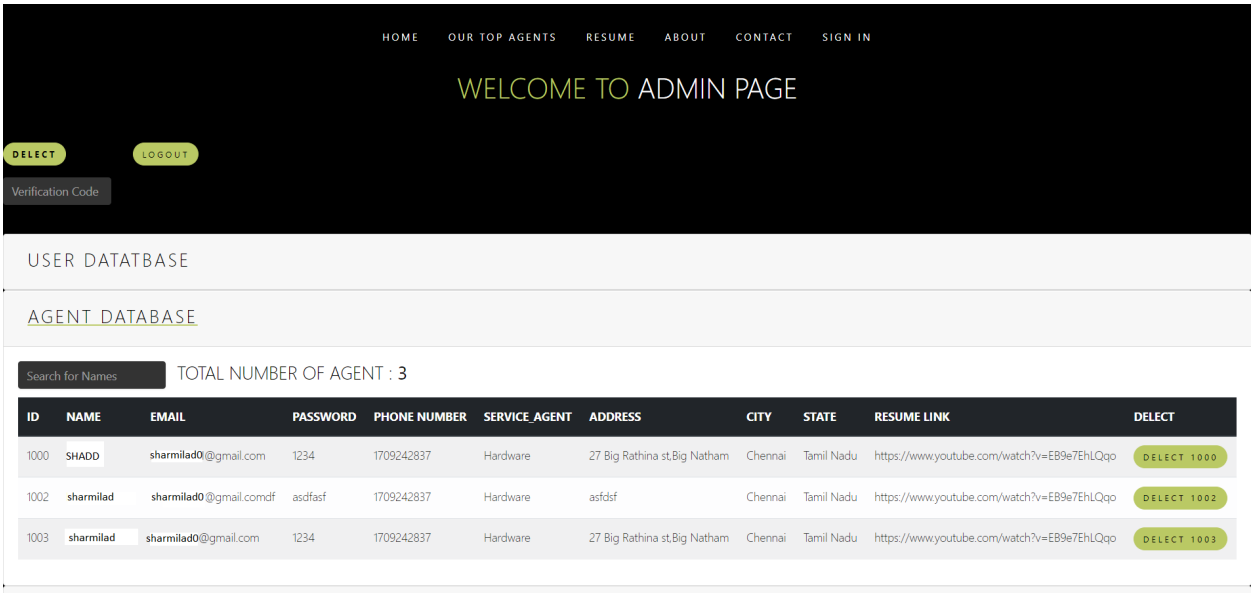


7. Agent database



ID	CUSTOMER_ID	DATE	EMAIL	PHONE_NUMBER	TOPIC	DESCRIPTION	SERVICE_TYPE	SERVICE_AGENT	ADDRESS	STATE	IMAGE_LINK	STATUS	DELECT
28	5004	11/02/22	ssrajkiran01@gmail.com	1709242837	LAPTOP	dsfSD	Hardware	KIRAN1000	27 Big Rathina st,Big Natham	Tamil Nadu	adfasdfasf@sdfsafds	Agent Alloted	DELECT 28
29	5005	11/02/22	**sharmilad0** @gmail.com	1234567897	board pboblem	.....	Hardware	None	chennai	Tamil Nadu	...	Processing	DELECT 29
21	5003	11/01/22	ssrajkiran01@gmail.com	1709242837	MOBILE	MOBILE	Software	None	27 Big Rathina st,Big Natham	Tamil Nadu	https://www.youtube.com/watch?v=E89e7EhLQqo	Processing	DELECT 21
31	5008	11/03/22	**shad** @gmail.com	1234894055	failure	the failure is there	Software	None	home	Tamil Nadu	https://www.linkedin.com/	Processing	DELECT 31

8. Admin page



5.4 SUMMARY

This chapter gives the description about the software used and the output details briefly.

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 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 customer care. 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 bea better, more understanding place for all to exist. The connectivity of IBM DB2 is evolved and associated with all formation.

6.2 FUTURE WORK

The proposed scheme be very useful in protecting users privacy detail like Email, password and name all are used by organization all in social network using IBM DB2, send-grid and Docker. Our Future work could be how to retrieve the details in both local and internet access efficiently. And the formation level to be very fast and secure.The details all are more secure in protection field.

REFRENCES

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

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

[3]. P. Keat and P.K. Young, Managerial EconomicsGlobal Edition, London:Pearson, pp. 23- 46, 2014.

[4]. Bai changhong and Liu Chi, "study on customer loyaltyof service enterprises and its determinants [J]", nankai Businessreview, no. 06, pp. 64-69,2002.

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