CUSTOMER CARE REGISTRY

TEAM ID : PNT2022TMID07252
TEAM LEADER : KARTHIKEYAN S
TEAM MEMBERS :
NIKITHA A
ARTHIDEVI R
HARISHANKAR

INDEX

1. INTRODUCTION

1.1 PROJECT OVERVIEW

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

1.2 PURPOSE

An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through 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. It's significance varies by product, industry and domain. In many cases customer services is more important if the information relates to a service as opposed to a Customer. Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

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

2.2 REFERENCE

- https://www.wikipedia.org/
- https://www.expertsystem.com/
- https://www.edureka.co/
- https://www.forbes.com/
- https://www.google.com

2.3 PROBLEM STATEMENT DEFINITION

A customer problem statement outlines problems that your customers face. It helps you figure out how your product or service will solve this problem for them. The statement helps you understand the experience you want to offer the customers

3. IDEATION & PROPOSED SOLUTION

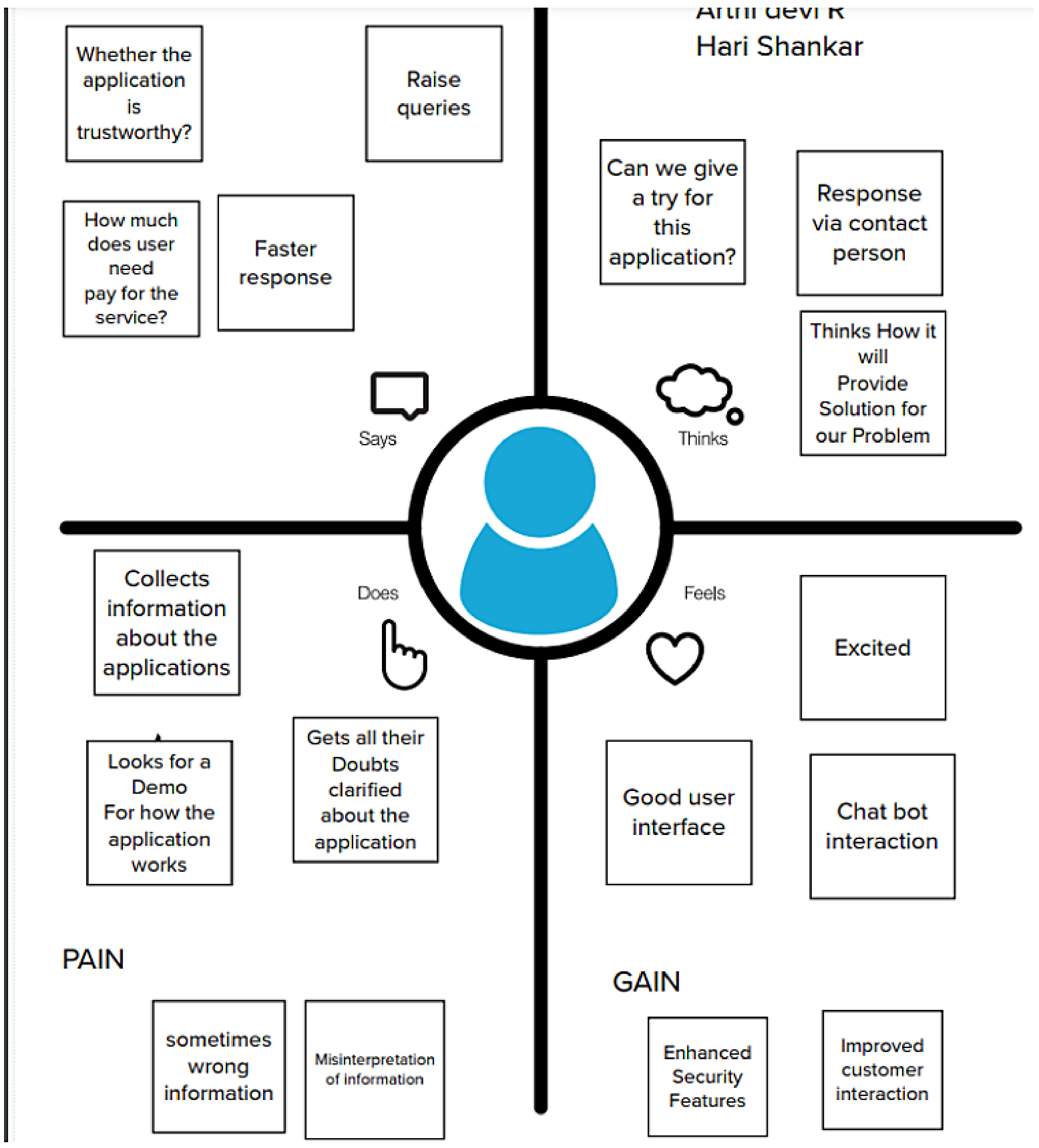
3.1 EMPATHY MAP CANVAS

A Customer Empathy Map is a tool used when collecting data about customers to better understand your target customer base. They allow you to visualize customer needs, condense customer data into a clear, simple chart, and help you see what customers want — not what you think they want. By following this map, you can systematically find answers, without playing a guessing game. When we look at empathy from a marketing perspective, we're talking about putting ourselves into our customers shoes, to be able to understand their needs and wants better. And thus, deliver a product or service that not only meets but exceeds their expectations! There are six key steps in a Customer Empathy Map that will allow you to collect important information about your ideal customer to be able to really understand them. The six different components you'll consider are:

1. What the customer thinks and feels
2. What the customer hears
3. What the customer sees
4. What the customer says and does
5. The customer's pains
6. The customer's gains

Customer Care Registry

Team ID :PNT2022TMID07252



Arthi devi R
Hari Shankar
Whether the application is trustworthy?
Raise queries
How much does user need pay for the service?
Faster response
Says
Can we give a try for this application?
Response via contact person
Thinks How it will Provide Solution for our Problem
Thinks
Does
Feels
Collects information about the applications
Excited
Looks for a Demo For how the application works
Gets all their Doubts clarified about the application
Good user interface
Chat bot interaction
PAIN
GAIN
sometimes wrong information
Misinterpretation of information
Enhanced Security Features
Improved customer interaction

3.2 IDEATION & BRAINSTROMING

After working with multiple CRM teams across various verticals, countries, team sizes and structures, I can certainty say that focused brainstorming sessions for CRM teams often represents an immense untapped opportunity. Even when CRM teams are conducting regular brainstorming sessions, they often lack three key characteristics:

1. They are not specific enough:

Often sessions are conducted around a broad topic such as 'How can we improve onboarding?' While the session may feel productive, it results in generic ideas which do not tackle user problems directly.

2. They are unstructured and outcomes unclear:

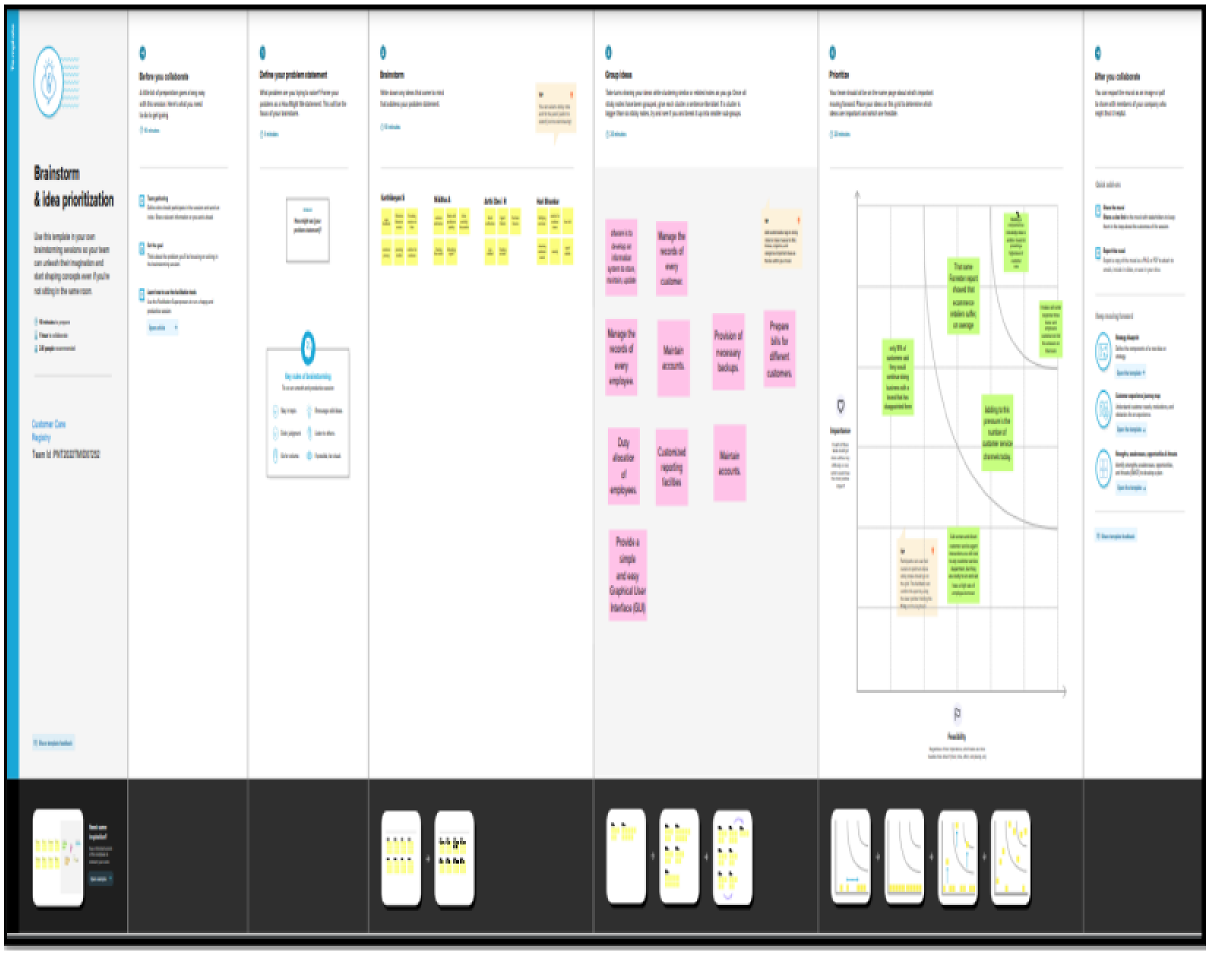
A typical brainstorm session is often unstructured to allow for multiple hypothesis, typically one problem/hypothesis is already provided to then come up with various solutions.

3. They lack follow-up on actions, results and learnings:

The most common result of brainstorming sessions I have seen are a bunch of ideas scribbled down on post-it notes and stuck to a wall to become part of the office decor. Yes, some ideas may be worked on once everyone's energy has returned, but the structure needed to bring those great ideas down to the ground and prioritize them for follow-up in a systematic way (using a framework like Reach Relevance Frequency, for example) is often missing. In the interest of saving time, I am going to assume that you are already aware of the foundations and basics of the brainstorming process. In case you want a refresher, I highly recommend checking out the following articles:

1. Basics of a brainstorming mindset: Brainstorming – IDEO U
2. Ways in which brainstorming can unlock value: Better Brainstorming in HBR
3. Some top techniques to generate ideas:

20 Brainstorming Techniques for Generating Better Ideas.


Brainstorm & idea prioritization
Before you collaborate
Define your problem statement
Brainstorm
Group ideas
Prioritize
After you collaborate
Manage the records of every customer
Manage the records of every employee
Maintain accounts.
Provision of necessary backups.
Prepare bills for different customers.
Duty allocation of employees.
Customized reporting facilities
Maintain accounts.
Provide a simple and easy Graphical User Interface (GUI)
Importance
Feasibility

3.3 PROPOSED SOLUTION

In the proposed solution the complaint go is an online complaint registration system using web services and android. Here a web portal is created which is used to process various complaints well supported with different web services. The online complaint management system provides an online way to solve the problems faced by the public by saving time and eradicate corruption. The main objective is to make complaint much easier to coordinate, monitor, track and resolve by tracking the status of complaint. The GPS based complaint redressal system develops an GPS based Complaint Redressal System (GPSCRS). The complaint is registered via a mobile application. Global Positioning System (GPS) sensor present in smart mobile devices is used to determine the exact location of the complaint. The technologies used in complaint go and online complaint management system is Google places API, Firebase cloud messaging, Javascript, Web, Cloud, GPS to track location and Phonegap. There are few disadvantages in complaint go and online complaint management system like:

- User can only register their complaints, there is no option to track the status of the registered complaint.
- Complaints are forwarded to the employees manually by the admin.
- This application provides a way to classify the complaints posted based on the GPS location and didn't handle resolving those complaints.

3.4 PROBLEM SOLUTION FIT

Problem-Solution Fit in 8 steps:

1. Who is your customer?

It is important to have a good picture of your customer, not only the demographics but preferably also sociographic data.

2. What is the problem of the customer?

Make assumptions of the problems you think your customer has and then go on to validate those assumptions. In the best case scenario, you can provide measurable insight into how big the customer's problem is.

3. Is the customer aware of the problem?

The more informed the customer is about their problem, the better. It may even be the case that your customer has realized an inefficient homegrown solution to try to solve the problem. Try as hard as you can to identify how well the customer is aware of the problem and when the customer will run into the problem.

4**. What is your product or service?**

Once you have found a validated problem you can realize your solution. Describe what the benefits and functionalities of your product or service are.

5. What are alternative solutions for the customer?

In a blue ocean, there may be few/no alternative solutions, but in a red ocean, you are bound to have competition trying to solve the same problem. Map out what the competition is and what they offer.

6. What are considerations your customer has while choosing a solution?

What are important go/no-go parts of your business, product or service? Map these out and, as much as possible, make sure your product is optimized for them.

7. Does your product or service actually solve the customer's problem?

What are important go/no-go parts of your business, product or service? Map these out and, as much as possible, Once you have found a quantifiable problem with the customer, it is fairly easy to measure the effectiveness of your solution. Make sure you have sufficient evidence that your product or service actually solves the customer's problem.

8. Build – Measure – Learn Cycle

If a you have found a bottleneck in one or more of the above 7 steps you can apply the Build-Measure-Learn cycle. In the end it remains a matter of making assumptions and validating them.

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

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 the 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. It’s significance varies by product, industry and domain.

MODULES:

- Admin
- Login
- Department
- Designation
- Domain
- Complaint
- Report
- Queries

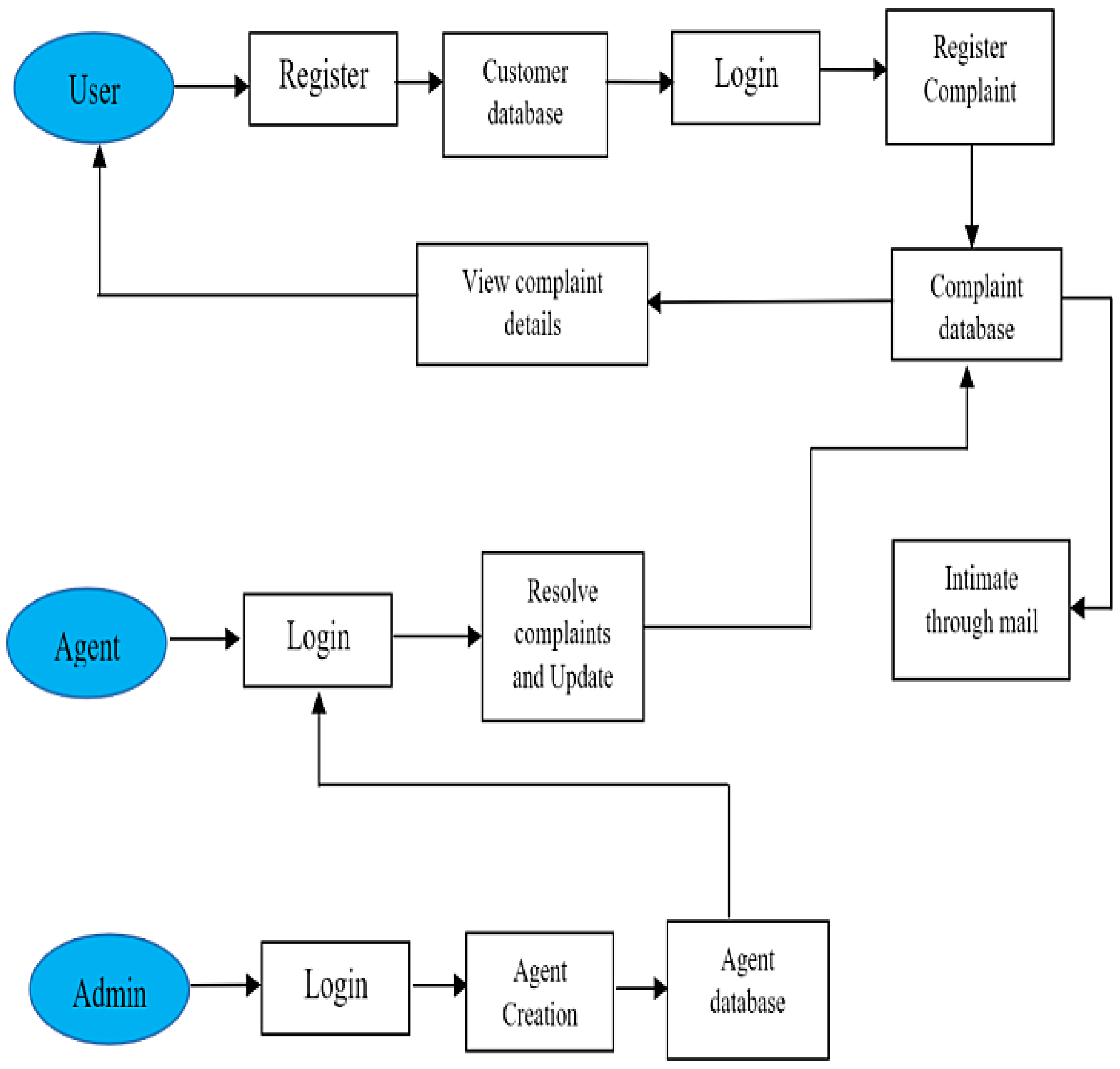
- Alerts

4.2 NON FUNCTIONAL REQUIREMENT

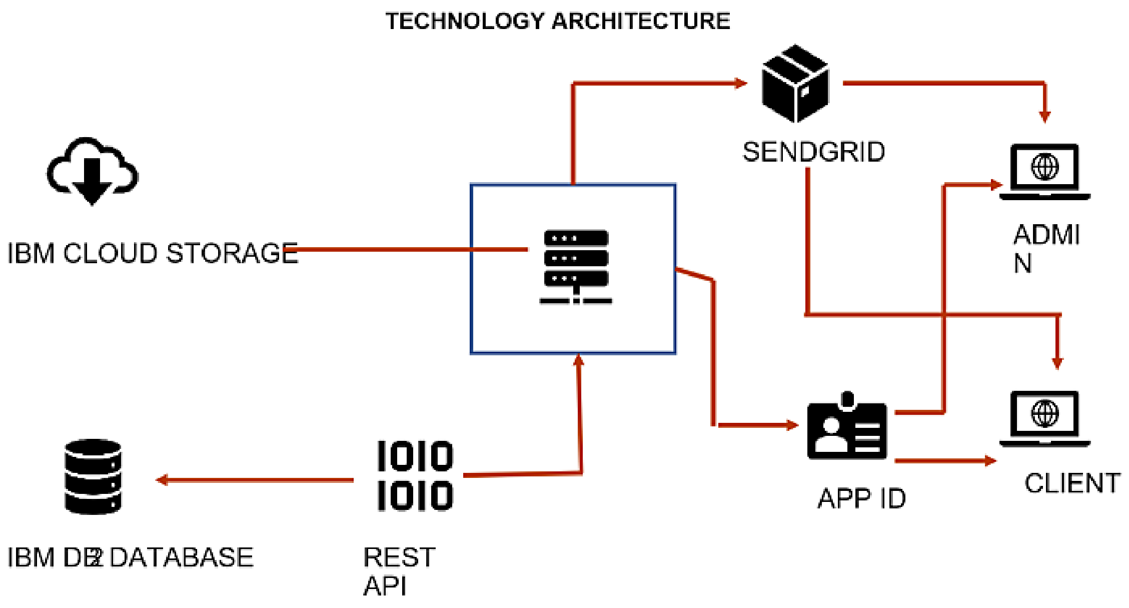
Customer is also a client; buyer or purchaser is the buyer or user of the paid products of an individual or an organization, mostly called the supplier or seller. This is typically through purchasing or renting goods or services. A group of services are provided to customer like banking, telecom, railway etc. Customer need to select help from any of these sectors. A customer complaint with voice is recorded in this system, and process the complaint within a certain amount of time.

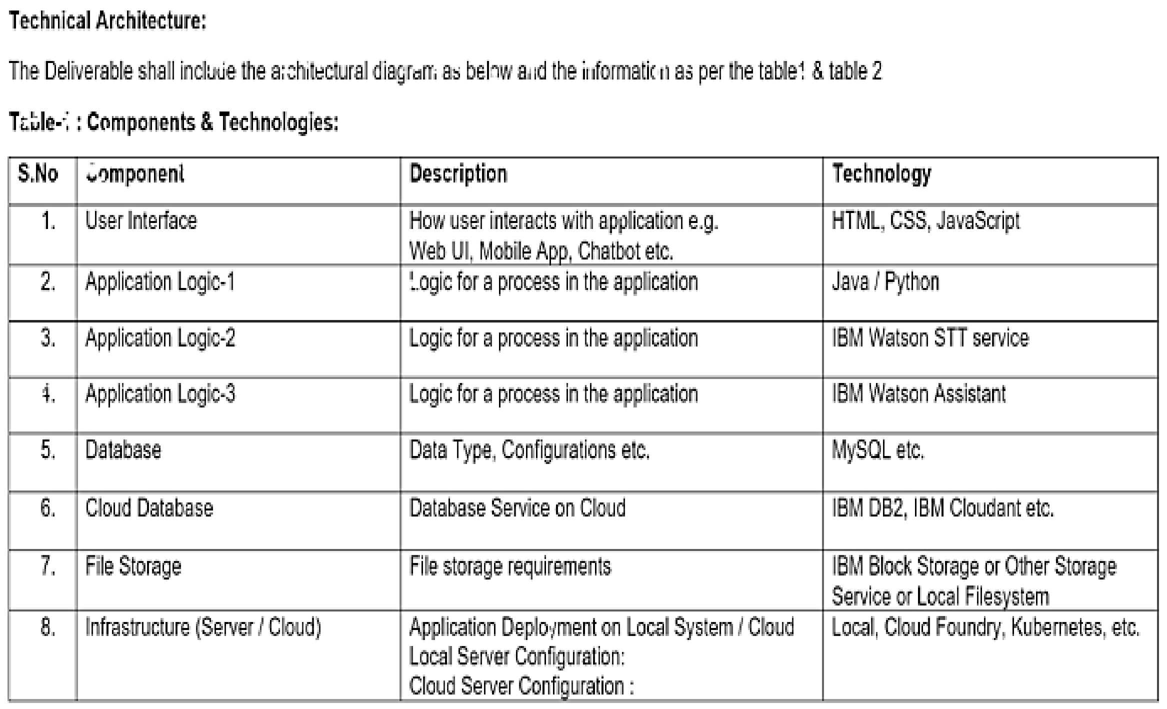
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS


User
Register
Customer database
Login
Register Complaint
Complaint database
View complaint details
Intimate through mail
Agent
Login
Resolve complaints and Update
Admin
Login
Agent Creation
Agent database

5.2 SOLUTION AND TECHNICAL ARCHITECTURE





Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

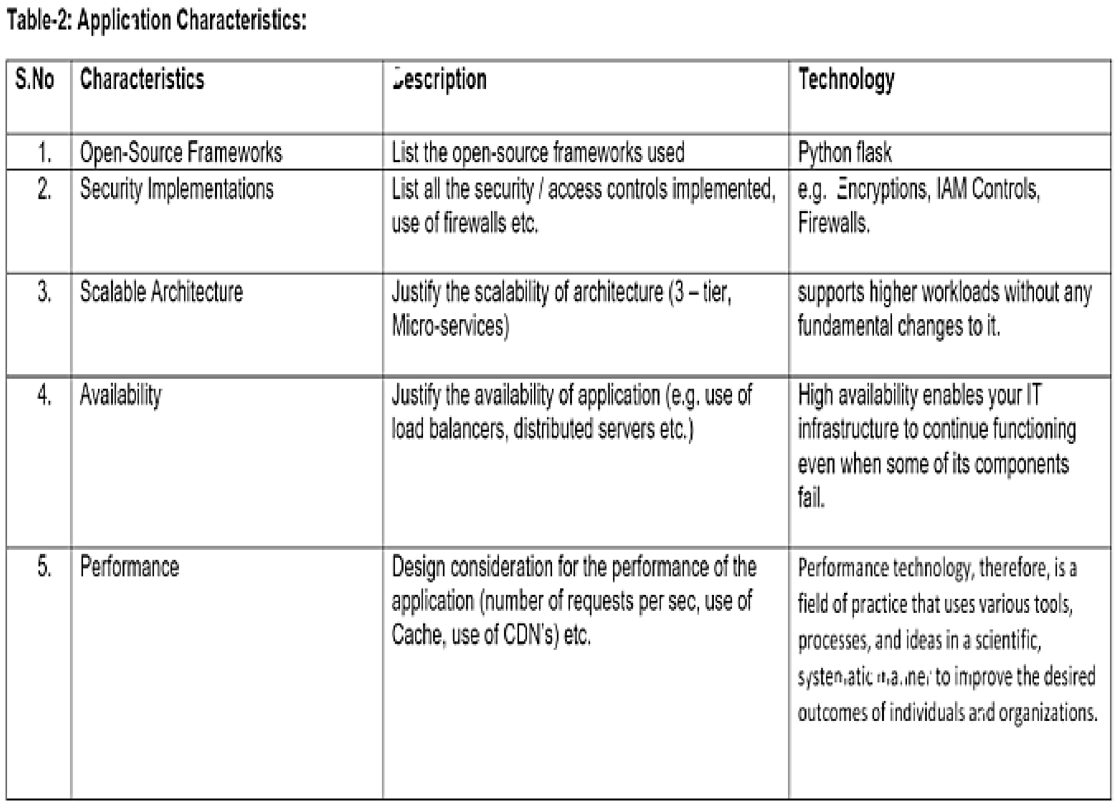
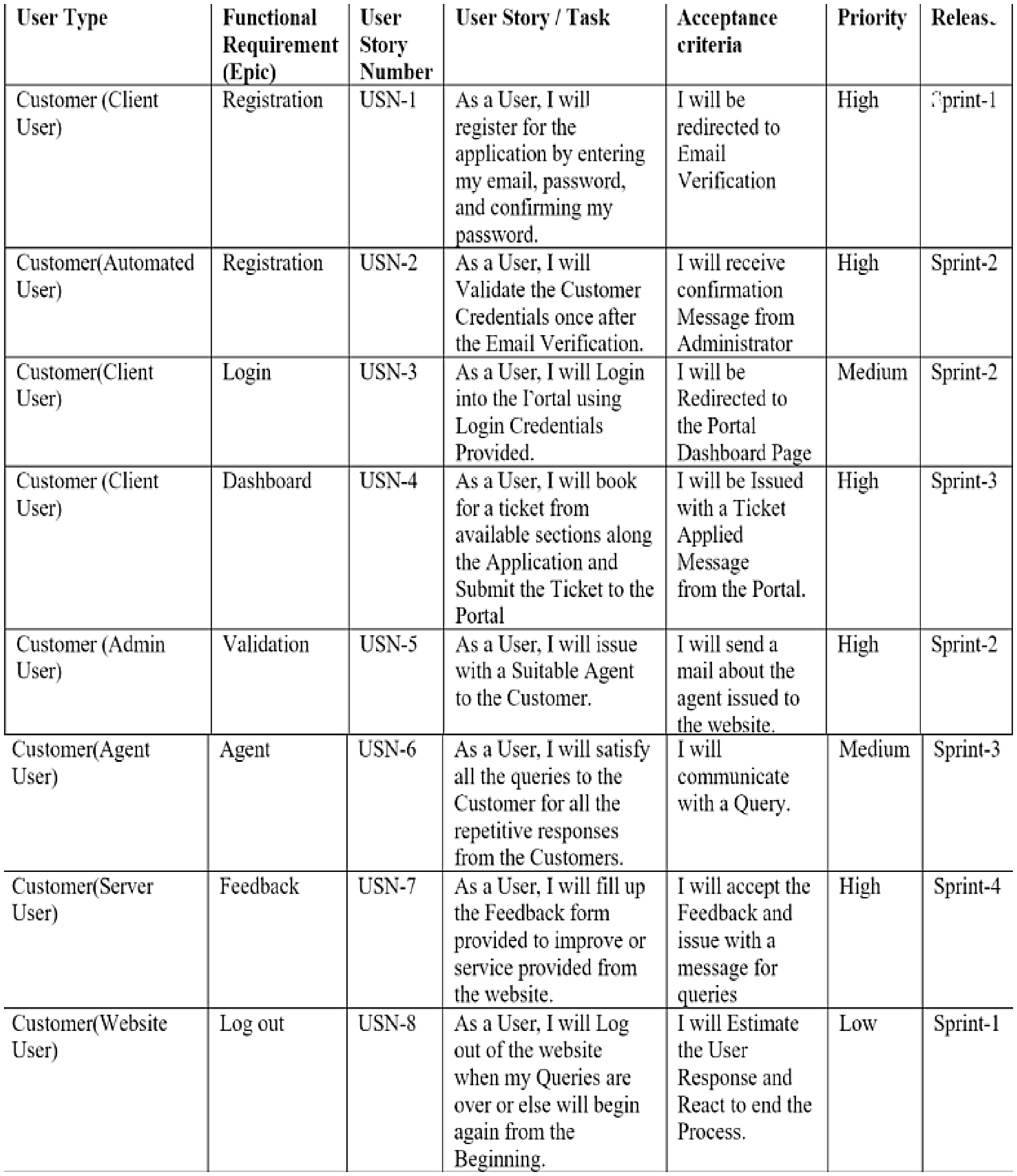


Table-2: Application Characteristics:

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. Encryptions, IAM Controls, 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 to continue 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 of practice that uses various tools, processes, and ideas in a scientific, systematic manner to improve the desired outcomes of individuals and organizations.

5.3 USER STORIES

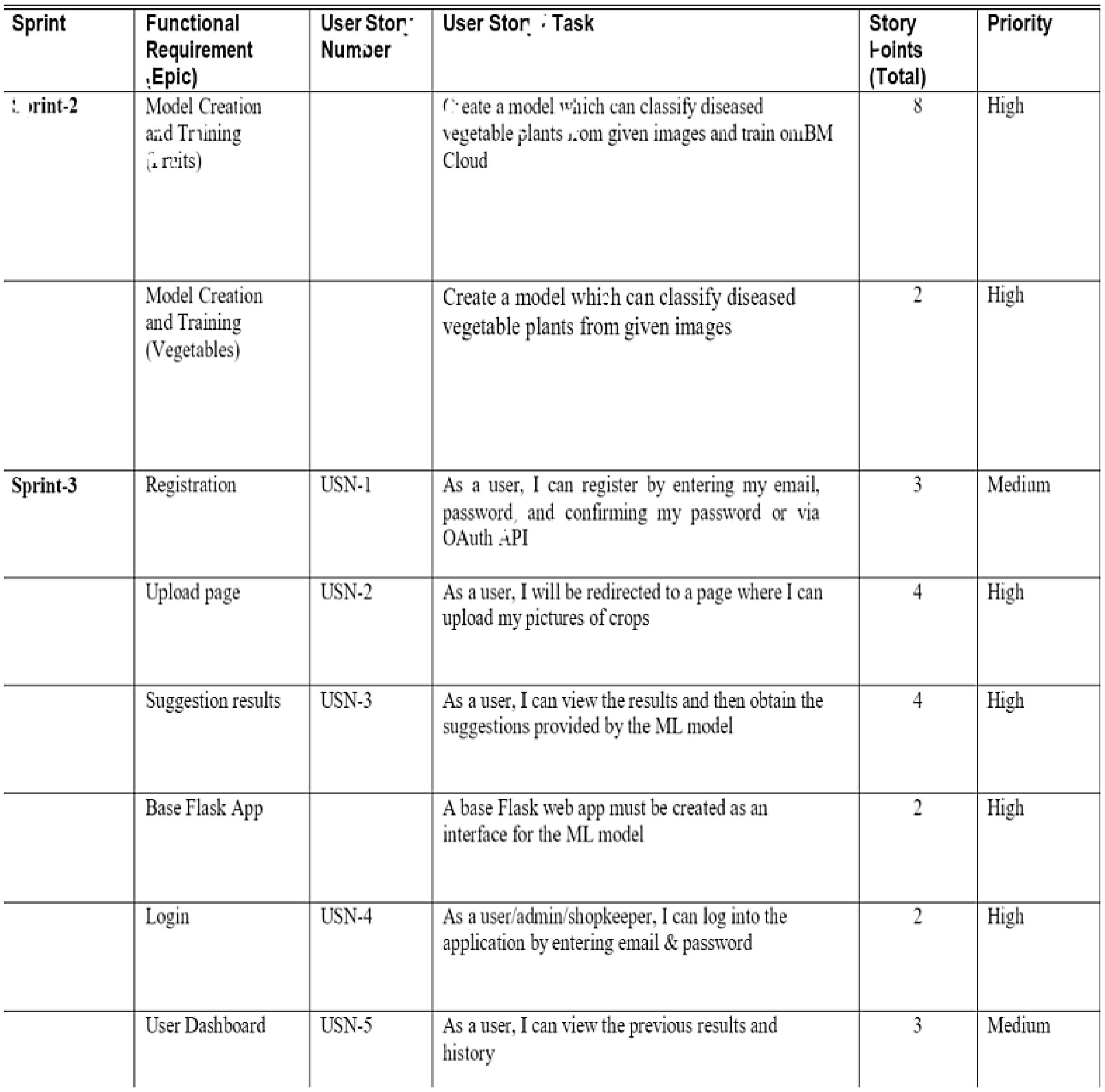
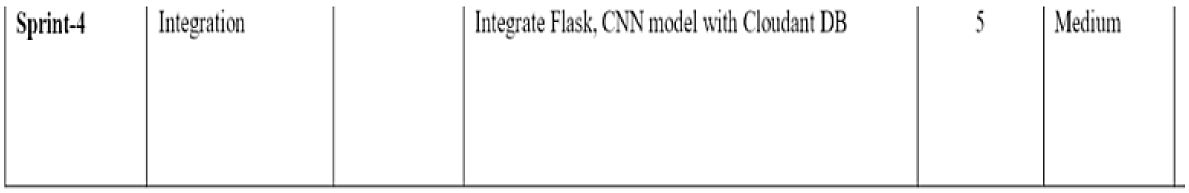
These end-users don't necessarily have to be external customers; they can also be stakeholders inside the company using the software or even colleagues and team members. The development and product management teams will write these user stories on index cards or sticky notes and use them as part of the discussions that take place during the development phase. These discussions help the team focus on the user — who is at the center of the conversation — to create a better product and user experience.



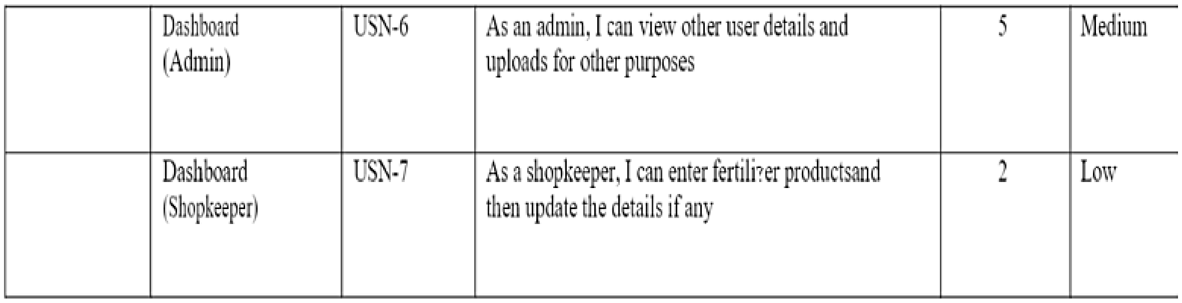
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Releas.
Customer (Client User)	Registration	USN-1	As a User, I will register for the application by entering my email, password, and confirming my password.	I will be redirected to Email Verification	High	Sprint-1
Customer(Automated User)	Registration	USN-2	As a User, I will Validate the Customer Credentials once after the Email Verification.	I will receive confirmation Message from Administrator	High	Sprint-2
Customer(Client User)	Login	USN-3	As a User, I will Login into the Portal using Login Credentials Provided.	I will be Redirected to the Portal Dashboard Page	Medium	Sprint-2
Customer (Client User)	Dashboard	USN-4	As a User, I will book for a ticket from available sections along the Application and Submit the Ticket to the Portal	I will be Issued with a Ticket Applied Message from the Portal.	High	Sprint-3
Customer (Admin User)	Validation	USN-5	As a User, I will issue with a Suitable Agent to the Customer.	I will send a mail about the agent issued to the website.	High	Sprint-2
Customer(Agent User)	Agent	USN-6	As a User, I will satisfy all the queries to the Customer for all the repetitive responses from the Customers.	I will communicate with a Query.	Medium	Sprint-3
Customer(Server User)	Feedback	USN-7	As a User, I will fill up the Feedback form provided to improve or service provided from the website.	I will accept the Feedback and issue with a message for queries	High	Sprint-4
Customer(Website User)	Log out	USN-8	As a User, I will Log out of the website when my Queries are over or else will begin again from the Beginning.	I will Estimate the User Response and React to end the Process.	Low	Sprint-1

6.PROJECT PLANING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

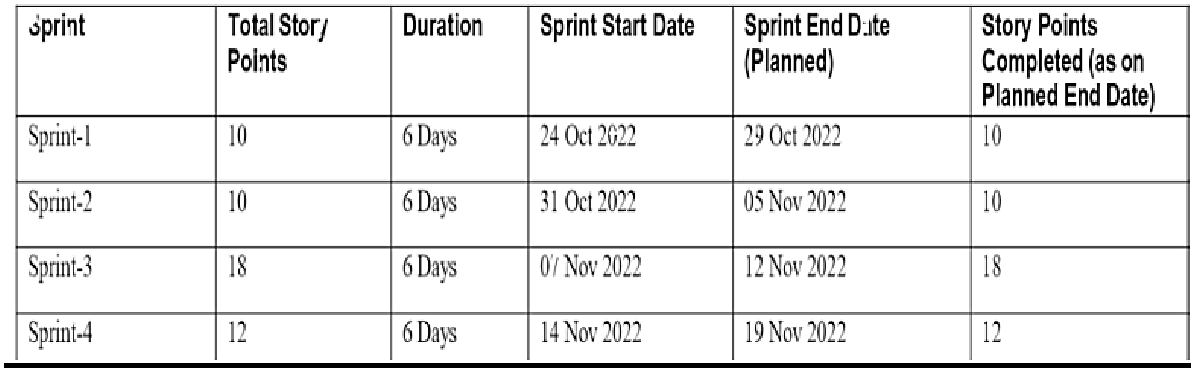



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points (Total)	Priority
Sprint-2	Model Creation and Training (Fruits)		Create a model which can classify diseased vegetable plants from given images and train on IBM Cloud	8	High
	Model Creation and Training (Vegetables)		Create a model which can classify diseased vegetable plants from given images	2	High
Sprint-3	Registration	USN-1	As a user, I can register by entering my email, password, and confirming my password or via OAuth API	3	Medium
	Upload page	USN-2	As a user, I will be redirected to a page where I can upload my pictures of crops	4	High
	Suggestion results	USN-3	As a user, I can view the results and then obtain the suggestions provided by the ML model	4	High
	Base Flask App		A base Flask web app must be created as an interface for the ML model	2	High
	Login	USN-4	As a user/admin/shopkeeper, I can log into the application by entering email & password	2	High
	User Dashboard	USN-5	As a user, I can view the previous results and history	3	Medium
Sprint-4	Integration		Integrate Flask, CNN model with Cloudant DB	5	Medium



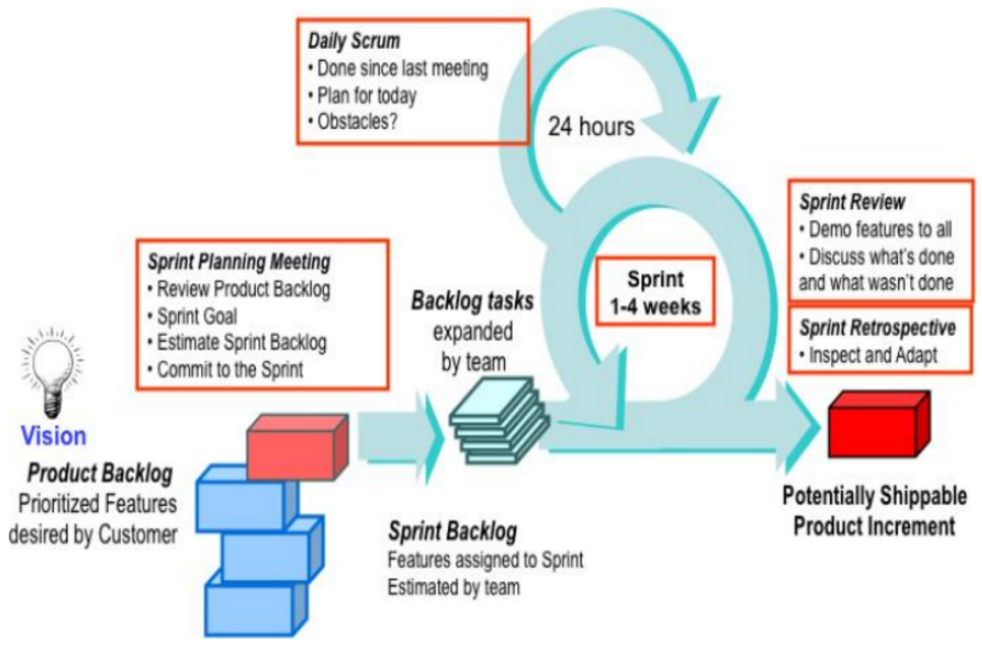
	Dashboard (Admin)	USN-6	As an admin, I can view other user details and uploads for other purposes	5	Medium
	Dashboard (Shopkeeper)	USN-7	As a shopkeeper, I can enter fertilizer productsand then update the details if any	2	Low

Project Tracker, Velocity & Burndown Chart: (4 Marks)



Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10
Sprint-2	10	6 Days	31 Oct 2022	05 Nov 2022	10
Sprint-3	18	6 Days	07 Nov 2022	12 Nov 2022	18
Sprint-4	12	6 Days	14 Nov 2022	19 Nov 2022	12

6.2 SPRINT DEVILERY SCHEDULE



The sprint-planning meeting is held on the first day of every sprint. The ScrumMaster, Product Owner, and Team are all in attendance. The Product Owner presents the set of features he or she would like to see completed in the sprint (the "what") then the team determines the tasks needed to implement these features (the "how"). Work estimates are reviewed to see if the team has the time to complete all the features requested in the sprint. If so, the team commits to the sprint. If not, the lower priority features go back into the product backlog, until the workload for the sprint is small enough to obtain the team's commitment.

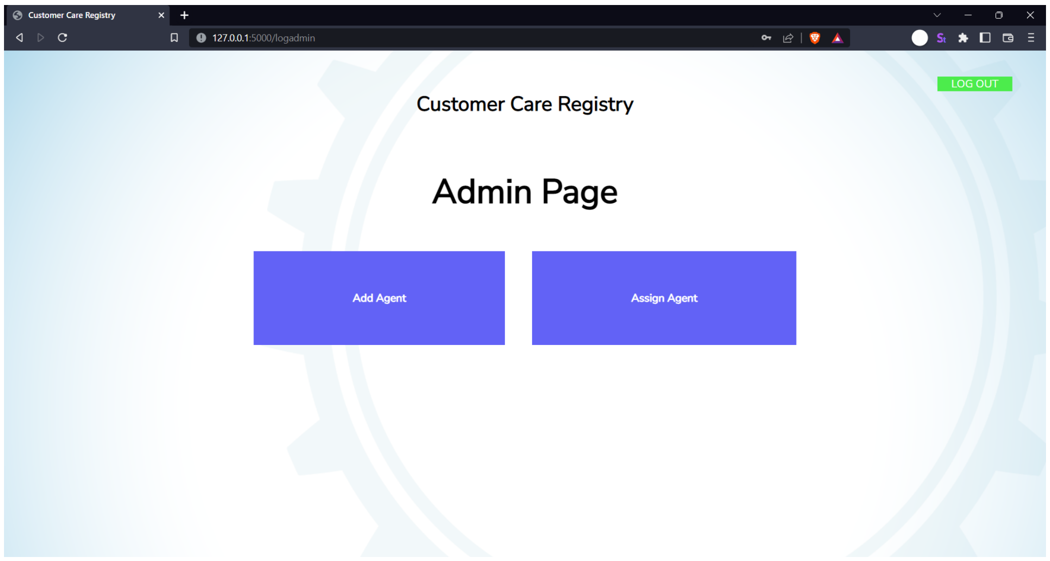
7.CODING & SOLUTIONING

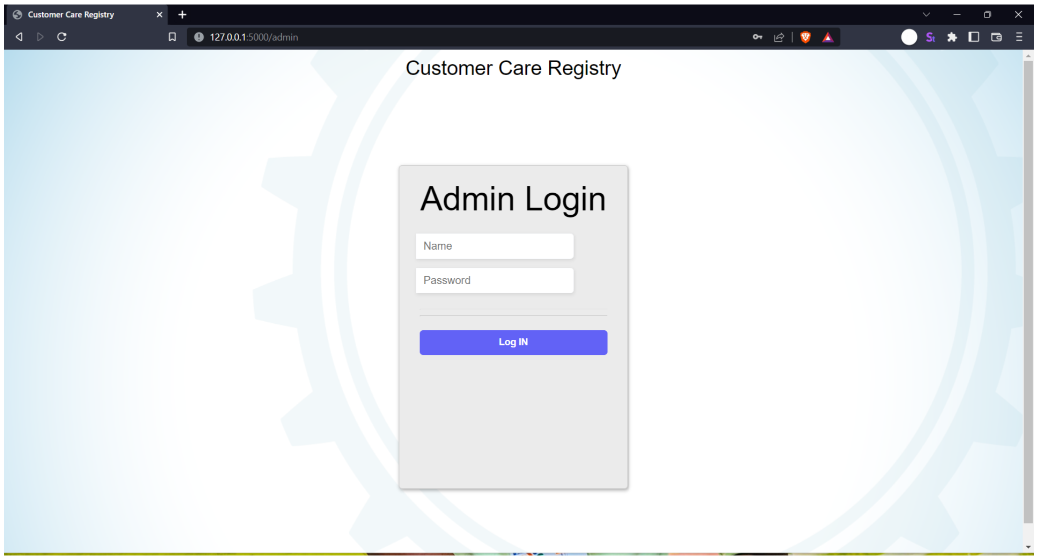
This Application has been developed to help the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to a customer they will be notified with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

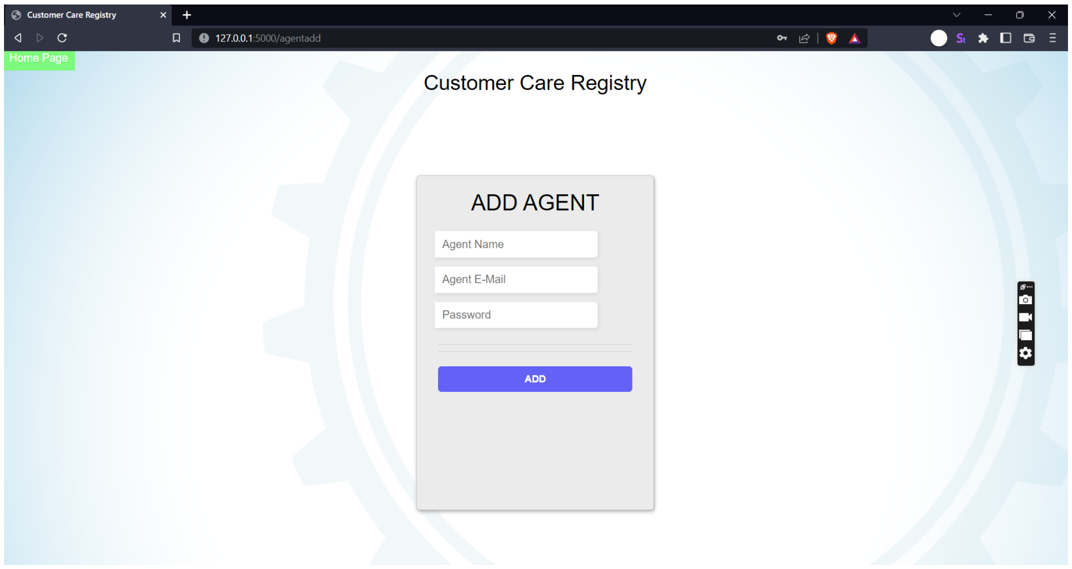
7.1 ADMIN PANEL

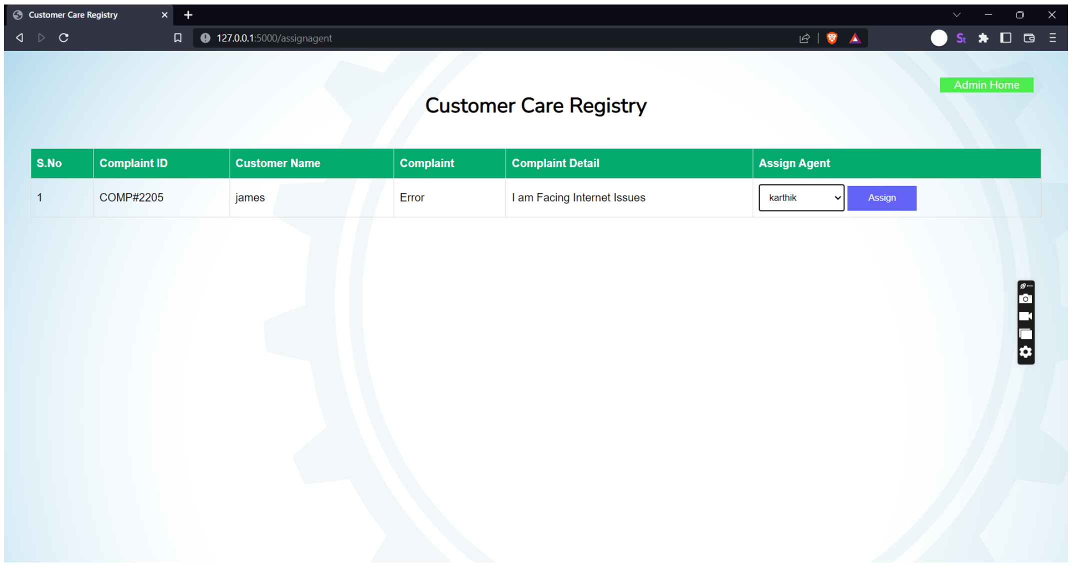
The main role and responsibility of the admin are to take care of the whole process. Starting from Admin login followed by the agent creation and assigning the customer's complaints. Finally, He will be able to track the work assigned to the agent and a notification will be sent to the customer.

SCREEN SHOTS:




Customer Care Registry
127.0.0.1:5000/admin
Customer Care Registry
Admin Login
Name
Password
Log IN


Customer Care Registry
127.0.0.1:5000/agentadd
Home Page
Customer Care Registry
ADD AGENT
Agent Name
Agent E-Mail
Password
ADD



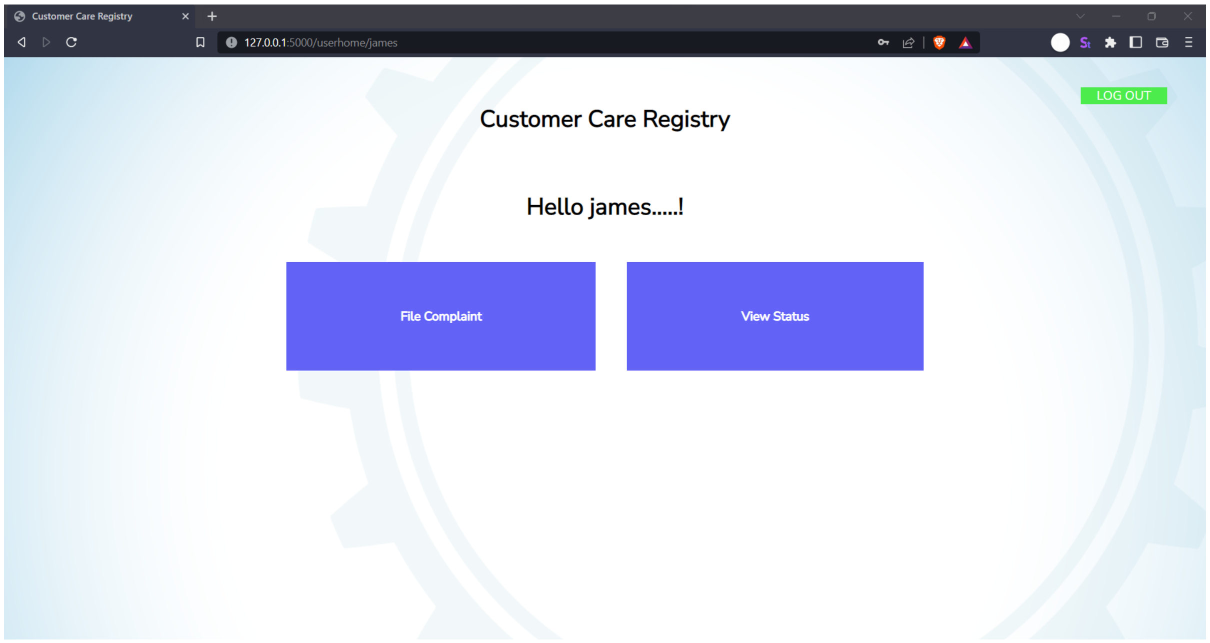
Customer Care Registry

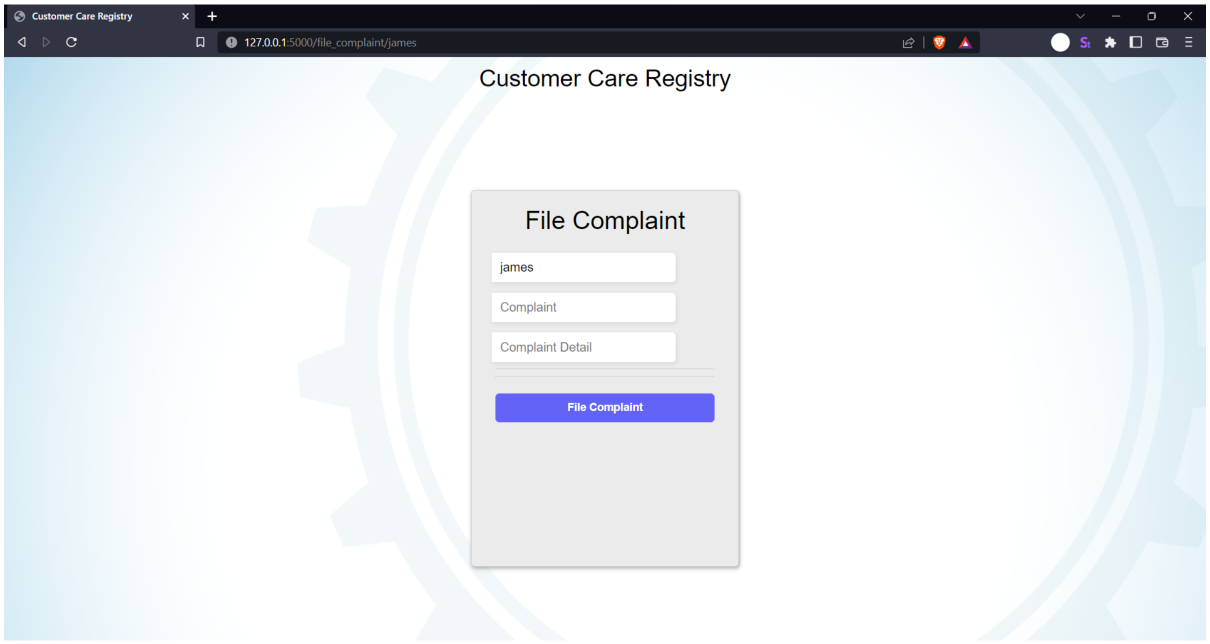
S.No	Complaint ID	Customer Name	Complaint	Complaint Detail	Assign Agent
1	COMP#2205	james	Error	I am Facing Internet Issues	karthik Assign

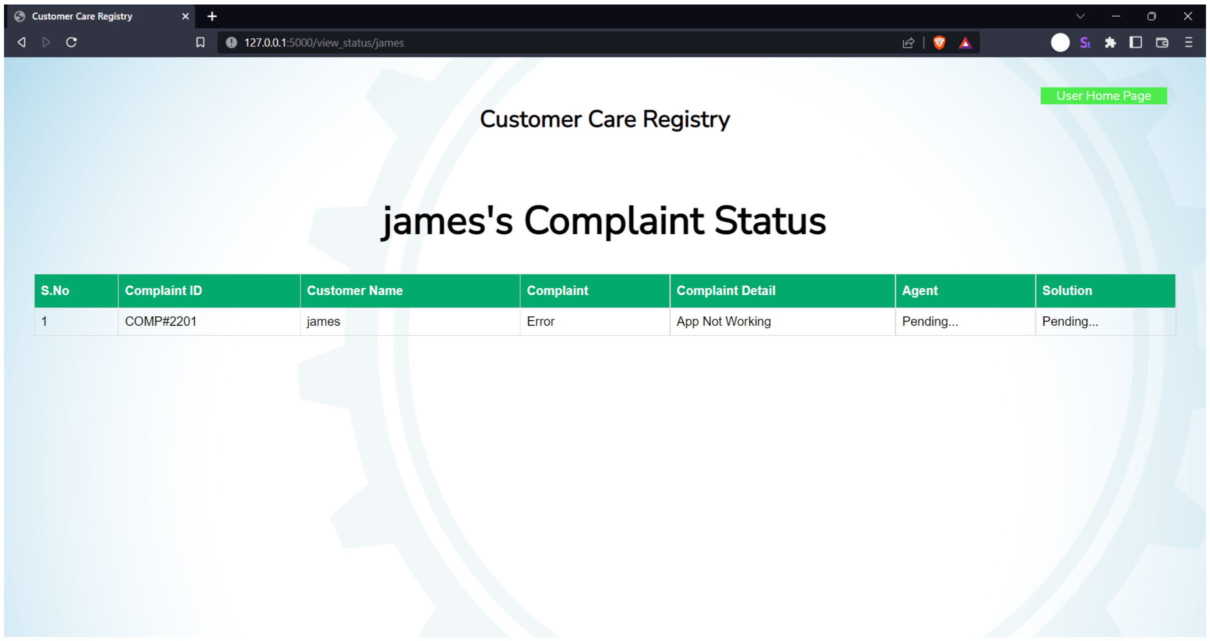
7.2 USER

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

Screenshots


Customer Care Registry
127.0.0.1:5000/userhome/james
LOG OUT
Customer Care Registry
Hello james.....!
File Complaint
View Status


Customer Care Registry
127.0.0.1:5000/file_complaint/james
Customer Care Registry
File Complaint
james
Complaint
Complaint Detail
File Complaint



Customer Care Registry

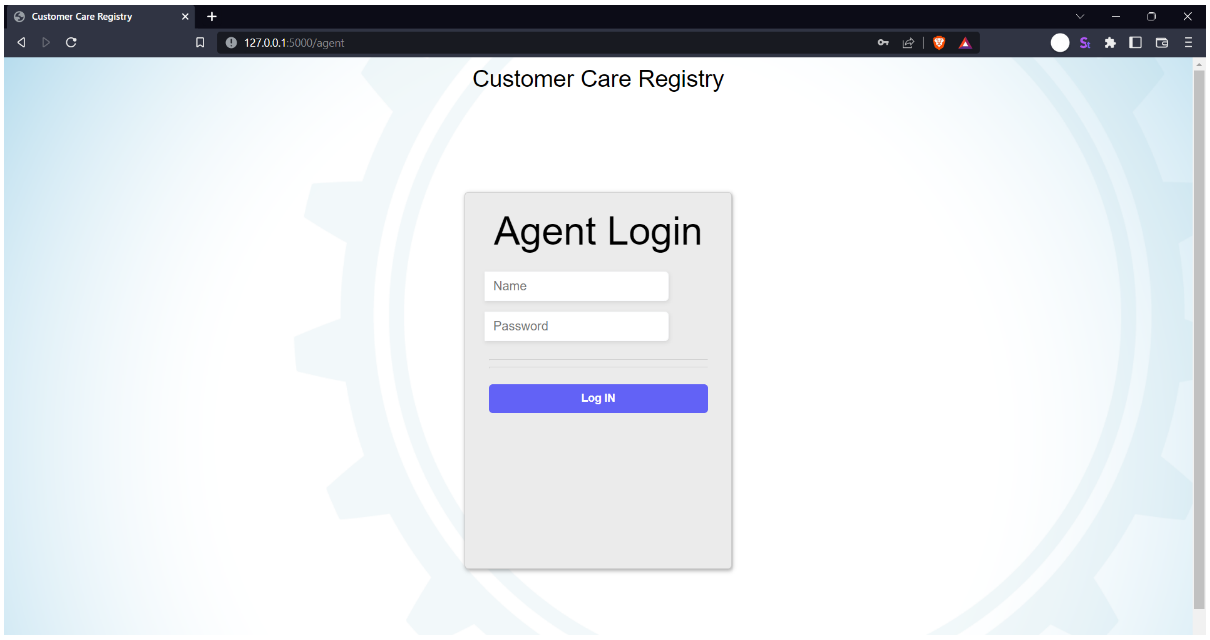
james's Complaint Status

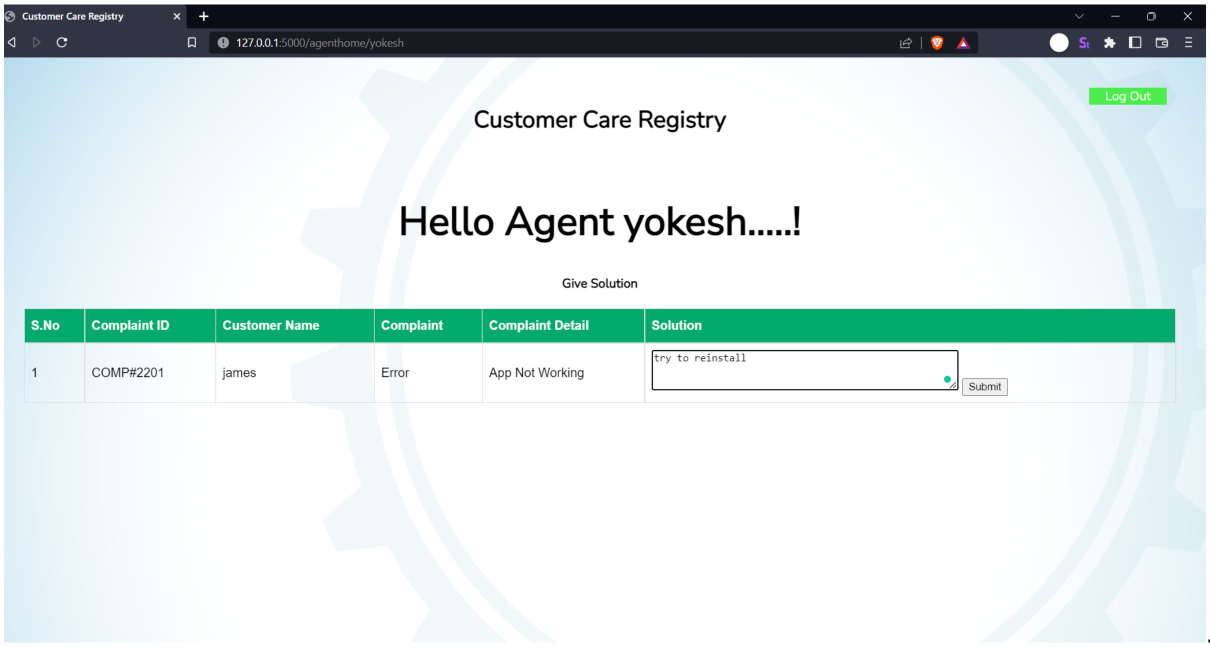
S.No	Complaint ID	Customer Name	Complaint	Complaint Detail	Agent	Solution
1	COMP#2201	james	Error	App Not Working	Pending...	Pending...

7.3 Agent

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

Screenshots:





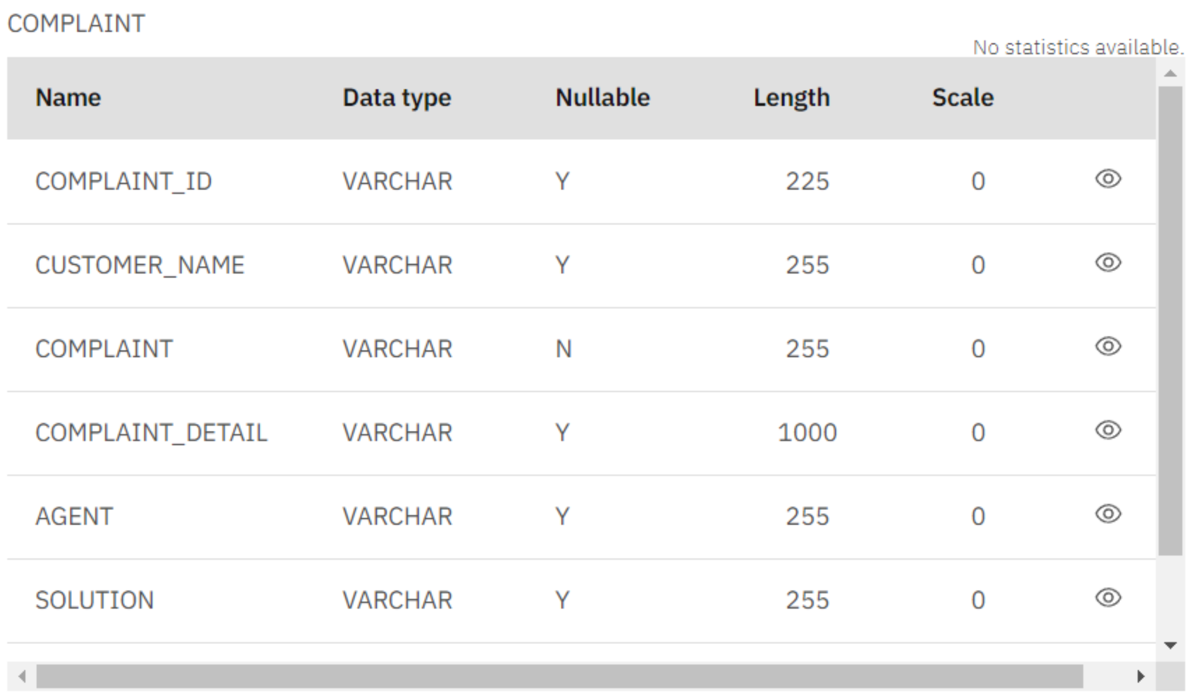
Hello Agent yokesh.....!

Give Solution

S.No	Complaint ID	Customer Name	Complaint	Complaint Detail	Solution
1	COMP#2201	james	Error	App Not Working	try to reinstall Submit

7.4 DATABASE SCHEMA:

COMPLAINT TABLE:

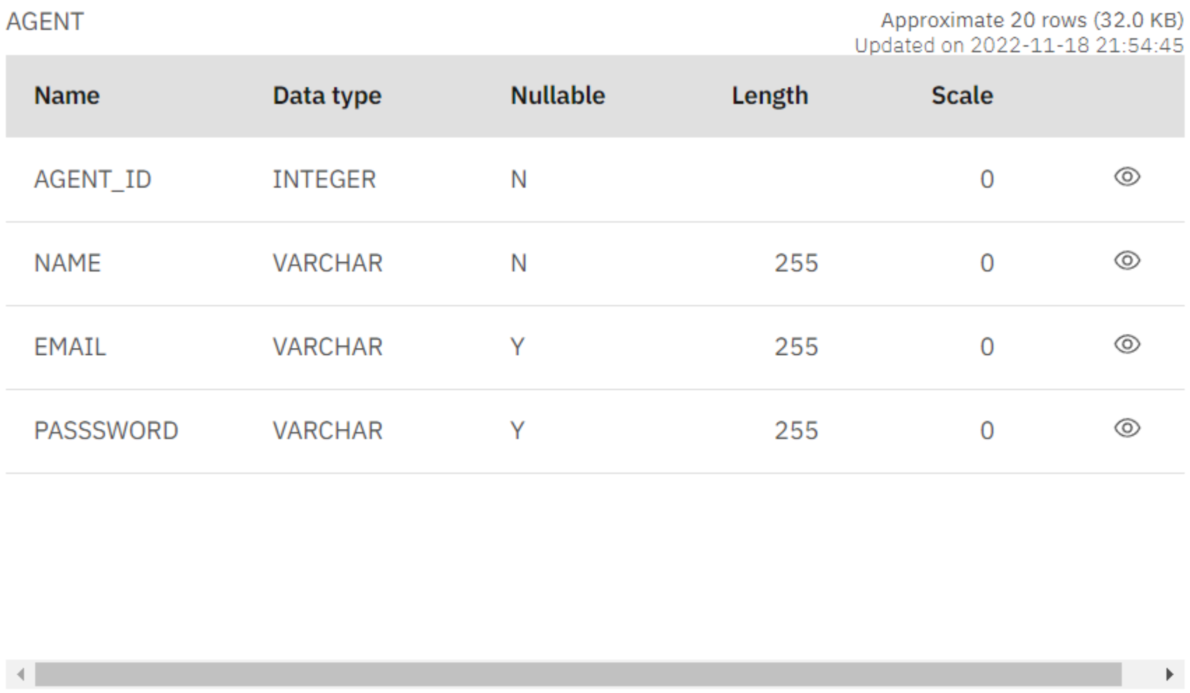


COMPLAINT

No statistics available.

Name	Data type	Nullable	Length	Scale
COMPLAINT_ID	VARCHAR	Y	225	0
CUSTOMER_NAME	VARCHAR	Y	255	0
COMPLAINT	VARCHAR	N	255	0
COMPLAINT_DETAIL	VARCHAR	Y	1000	0
AGENT	VARCHAR	Y	255	0
SOLUTION	VARCHAR	Y	255	0

AGENT TABLE :

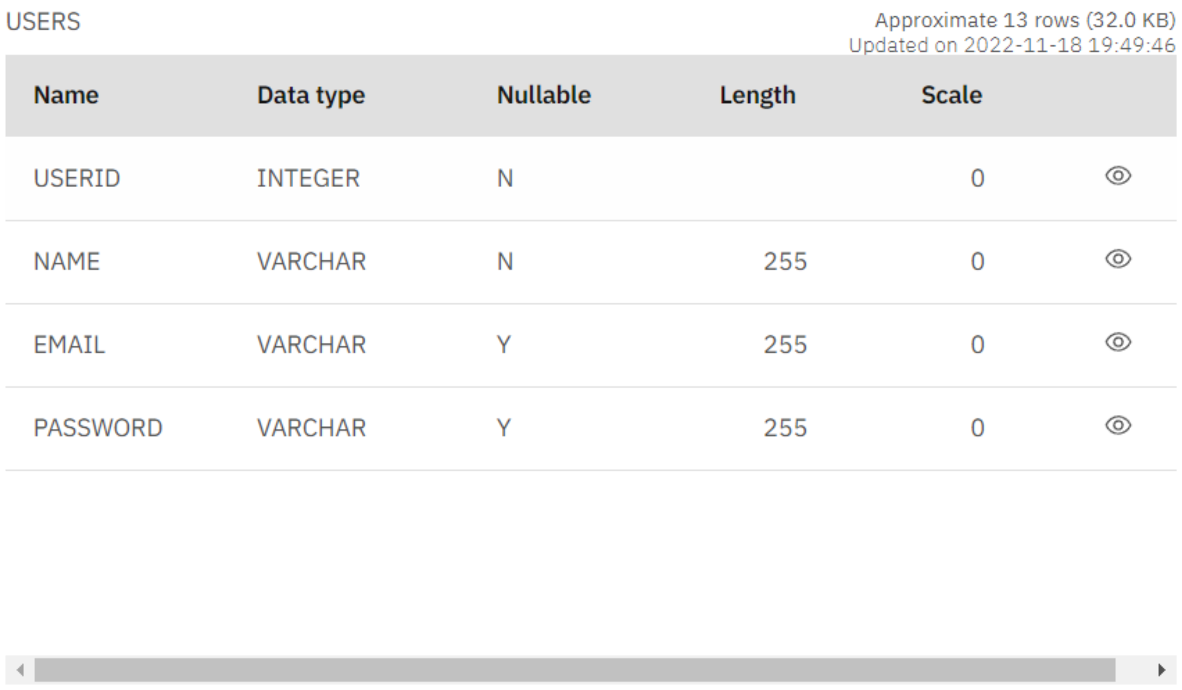


AGENT

Approximate 20 rows (32.0 KB)
Updated on 2022-11-18 21:54:45

Name	Data type	Nullable	Length	Scale
AGENT_ID	INTEGER	N		0
NAME	VARCHAR	N	255	0
EMAIL	VARCHAR	Y	255	0
PASSSWORD	VARCHAR	Y	255	0

USER TABLE :



USERS

Approximate 13 rows (32.0 KB)
Updated on 2022-11-18 19:49:46

Name	Data type	Nullable	Length	Scale
USERID	INTEGER	N		0
NAME	VARCHAR	N	255	0
EMAIL	VARCHAR	Y	255	0
PASSWORD	VARCHAR	Y	255	0

8.TESTING

Project testing is essential for every small and large enterprise while launching new projects. Rigorous testing helps in analyzing the system workflow and different applications integrated with the system

- **Functional Testing**: Testing healthcare application against functional capabilities
- **Conformance Testing**: Conformance test Healthcare security requisites and industry frameworks
- **Platform Testing**: Testing of applications on Mobile platform and applications testing for cross-browser compatibility
- **Interoperability Testing**: Testing conformance to interoperability standards (Eg; DICOM, HL7, CCD/CDA)

Test Cases:

- 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.
- 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.

8.2 USER ACCEPTANCE TEST :

User Acceptance Testing (UAT), which is performed on most UIT projects, sometimes called beta testing or end-user testing, is a phase of software development in which the software is tested in the "real world" by the intended audience or business representative.

Test Cases Performed:

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass / Fail	Executed By
1	Customer need to login with valid user credentials	1. Go to the site 2. Login with credentials	Username – Karthik Password – Karthik546	It will display invalid credentials and login page should be reloaded	As expected	Pass	Karthikeyan
2.	Customer need to login with valid user credentials	1.Go to the site 2.Login with credentials	Username – Nikitha Password – NIKI@123	With valid login credentials, the customer lead to customer dashboard page	As expected	Pass	Nikitha
3.	Customer signup action	1.Go to the site 2.Click Signup	Username – customer45 Password – kjbnjt^&*456 Email – agent123gmail.com	Email is not in correct format. It will display as invalid email address	As expected	Pass	Arthi Devi
4.	Customer signup action	1.Go to the site 2.Click Signup	Username – Customer4563 Password – customer@123 Email - customer12@gmail.com	Customer account created	As expected	Pass	Harishankar
5.	Admin need to login with valid user credentials	1.Go to the site 2.Login with credentials	Username – Admin45 Password – admin654	It will display invalid credentials and login page should be reloaded	As expected	Pass	Karthikeyan

6.	Admin need to login with valid user credentials	1.Go to the site 2.Login with credentials	Username – Admin654 Password – admin@123	With valid login credentials, the admin lead to admin dashboard page	As expected	Pass	NIKITHA
7.	Agent need to login with valid user credentials	1.Go to the site 2.Login with credentials	Username – Agent45 Password – agent654	It will display invalid credentials and login page should be reloaded	As expected	Pass	ArthiDevi
8.	Agent need to login with valid user credentials	1.Go to the site 2.Login with credentials	Username – Agent654 Password – agent@123	With valid login credentials, the agent lead to agent dashboard page	As expected	Pass	Karthikeyan
9.	Admin Add an Agent	1.Go to the site 2.Login with credentials 3.Go to dashboard 4.click on add agent	Username – Agent1 Password – agent@123 Email – agent123@gmail.com	Agent account will be created	As expected	Pass	Harishankar

9.RESULT

9.1 Performace Metrics

Performance metrics are defined as figures and data representative of an organization's actions, abilities, and overall quality.These metrics are used to track and measure the effectiveness and profitability of various projects. Each stage of the project is tracked and measured against the goals that the project set out to achieve.The data compiled from the metrics can be used to plan future projects and gives insight on how to make projects more efficientPerformance metrics is a key tool for any business owner to finely tune their business and streamline their processes. To create an efficient and profitable business, it is key that everything is running smoothly and at an acceptable level

A metric is a meaningful measurement taken over a period of time that communicates vital information about a process or activity, leading to fact-based decisions. Metrics are usually specialized by the subject area. In business, they are sometimes referred to as key performance indicators (KPI). Performance metrics should be constructed to encourage performance improvement, effectiveness, efficiency and appropriate levels of internal controls.

10.ADVANTAGES AND DISADVANTAGES ADVANTAGES

Flowsheet is a powerful tool to monitor clinical data and track trends

- Provides a dashboard of who needs what • Provides total population data reporting with no chart abstraction
- Generates revenue (it shows when services are needed) • Provides outreach information at fingertips
- Improves team-based care
- Smaller software package than EHRs DISADVANTAGES Disease-specific, not longitudinal
- Does not include information necessary for billing
- Requires hardware, software, and maintenance
- Requires data entry and data maintenance
- Parallel documentation system (i.e., some information has to be entered in two systems)
    - Can’t stand alone, must have an additional documentation system.

11.CONCLUSION

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

12.FUTURE SCOPE

- The job desc for a customer service director will focus more on leadership, innovation, and ability to drive company-wide improvement.
- Customer service will shift to become a strategic partner of marketing, sales, and product development. CS will help with direction, project prioritisation, and impact.
- A need for customer service leaders to take a highly strategic seat at the table. They'll need to argue for investment in talent, technology, and innovation.
- A shift in performance metrics. Forget # of resolved tickets. In the future, we'll measure performance based on # of customers saved from the precipice of churn.
- A career in customer service will not be a last resort. Top graduates will prioritise getting an education in strategic customer interaction.
- Focus on ticket deflection will reduce because brands will view each customer interaction as an opportunity to learn, build a relationship, and grow profits. They deserve a well-trained, human touch

13.APPENDIX

APP.PY

```
from flask import Flask, render_template, request, redirect, url_for,
session, flash
import ibm_db
from markupsafe import escape
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
```

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


conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f5-
23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32716;SEC
URITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=
rdb98826;PWD=8RbXIwMNNwOtDRaA;", "", "")
print(conn)
print("connection successful...")
@app.route('/agent',methods=['POST', 'GET'])
def agent():
      return render_template('agentlogin.html')

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



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

              sql = f"select * from agent where name='{escape(id)}' and
passsword='{escape(password)}'"
              stmt = ibm_db.exec_immediate(conn, sql)
              data = ibm_db.fetch_both(stmt)
              print(data)
              if data:
                session["name"] = escape(id)
                session["password"] = escape(password)
                return redirect(url_for("agenthome",name=id))

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

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

            return render_template("agentlogin.html")


@app.route('/agenthome/<name>',methods=['POST', 'GET'])
def agenthome(name):
      com = []
      sql =f"SELECT complaint_id,customer_name,complaint,complaint_detail
FROM complaint where status='no'and agent='{escape(name)}'"
      stmt = ibm_db.exec_immediate(conn, sql)
      dictionary = ibm_db.fetch_both(stmt)
      while dictionary != False:
            com.append(dictionary)
            dictionary = ibm_db.fetch_both(stmt)
      return render_template('agenthome.html',comp=com , name=name)
@app.route('/addsoluin',methods=['POST', 'GET'])
def addsoluin():
        if request.method =='POST':
             try:
               nam=request.form['gname']
               solu = request.form['solution']
               cid = request.form['cid']
               sql=f"UPDATE COMPLAINT SET
solution='{escape(solu)}',status='yes' WHERE
COMPLAINT_ID='{escape(cid)}';"
               ibm_db.exec_immediate(conn, sql)
             except:
               flash("Error in Insertion operation", "danger")
             finally:
               flash('Solution Send Successfully')
               return redirect(url_for('agenthome',name=nam))




@app.route('/userhome/<name>')
def userhome(name):
       return render_template('userhome.html',name=name)

@app.route('/file_complaint/<name>',methods=['POST', 'GET'])
def file_complaint(name):
       return render_template('filecomplaint.html',nam=name)
@app.route('/view_status/<name>',methods=['POST', 'GET'])
def view_status(name):
      com = []
      sql =f"SELECT
complaint_id,customer_name,complaint,complaint_detail,agent,solution FROM
complaint where customer_name='{escape(name)}'"
      stmt = ibm_db.exec_immediate(conn, sql)
      dictionary = ibm_db.fetch_both(stmt)
      while dictionary != False:
            com.append(dictionary)
            dictionary = ibm_db.fetch_both(stmt)
      return render_template('view_status.html',comp=com,nam=name)



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

@app.route('/agentadd')
```

```
def agentadd():
       return render_template('agentadd.html')
@app.route('/addagent',methods=['POST', 'GET'])
def addagent():
       if request.method =='POST':
             try:
               agent= request.form['agentname']
               mail= request.form['agentmail']
               password=request.form['password']
               sql=f"INSERT INTO agent (name,email, passsword )VALUES
('{escape(agent)}', '{escape(mail)}', '{escape(password)}');"
               ibm_db.exec_immediate(conn, sql)
             except:
               flash("Error in Insertion operation", "danger")
             finally:
               flash('Agent added successfully')
               return redirect(url_for('admin'))
@app.route('/assignagent')
def assignagent():
        com = []
        sql =f"SELECT
complaint_id,customer_name,complaint,complaint_detail FROM complaint where
agent='Pending...'"
        stmt = ibm_db.exec_immediate(conn, sql)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
             com.append(dictionary)
             dictionary = ibm_db.fetch_both(stmt)

        agent=[]
        sql1 = f"SELECT agent_id,name FROM agent"
        stmt1 = ibm_db.exec_immediate(conn, sql1)
        dictionary1 = ibm_db.fetch_both(stmt1)
        print(dictionary1)
        while dictionary1 != False:
             agent.append(dictionary1)
             dictionary1= ibm_db.fetch_both(stmt1)
        return render_template('assignagent.html',data=com,a=agent)
@app.route('/addagentin',methods=['POST', 'GET'])
def addagentin():

      if request.method =='POST':
             try:
               aid = request.form['agent']
               cid = request.form['cid']
               sql=f"UPDATE COMPLAINT SET AGENT='{escape(aid)}'WHERE
COMPLAINT_ID='{escape(cid)}';"
               ibm_db.exec_immediate(conn, sql)
             except:
               flash("Error in Insertion operation", "danger")
             finally:
               flash("Agent Assigned Successfully")
               return redirect(url_for('assignagent'))


@app.route('/logadmin',methods=['POST', 'GET'])
def logadmin():
      if request.method == 'POST':
             try:
              id = request.form['name']
              password = request.form['password']
              if id==admin and password=="admin":
                    return render_template('adminhome.html')
             except:
              flash("Error in Insertion operation", "danger")
             finally:
              flash("Admin login Successfully")
              return render_template('adminhome.html')
```

```
@app.route('/admin')
def admin():
       return render_template('adminlog.html')


@app.route('/userlogin')
def userlogin():

      return render_template('userlogin1.html')


@app.route('/addcomplaint', methods=['POST', 'GET'])
def addcomplaint():
       if request.method =='POST':
             try:
               cus= request.form['customer']
               complaint= request.form['complaint']
               detail= request.form['detail']
               sql1=f"SELECT COUNT(complaint)FROM complaint";
               f=ibm_db.exec_immediate(conn, sql1)
               data = ibm_db.fetch_both(f)
               s="COMP#220"+str(data[0]+1)
               sql=f"INSERT INTO complaint
(complaint_id,customer_name,complaint,complaint_detail,agent,solution,stat
us)VALUES ('{escape(s)}','{escape(cus)}','{escape(complaint)}',
'{escape(detail)}', 'Pending...','Pending...','no');"
               ibm_db.exec_immediate(conn, sql)
             except:
               flash("Error in Insertion operation", "danger")
             finally:
               flash('Complaint successfully registered')
               return redirect(url_for('userhome',name=cus))
```

```
@app.route('/adduser', methods=['POST', 'GET'])
def adduser():
       if request.method =='POST':
             try:
               name= request.form['name']
               mail= request.form['mail']
               password= request.form['password']
               sql=f"INSERT INTO users (name,email, password)VALUES
('{escape(name)}', '{escape(mail)}', '{escape(password)}');"
               ibm_db.exec_immediate(conn, sql)

             except:
               flash("Error in Insertion operation", "danger")
             finally:
               flash('User Added Successfully')
               return redirect(url_for('userhome',name=name))

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

              sql = f"select * from users where name='{escape(id)}' and
password='{escape(password)}'"
              stmt = ibm_db.exec_immediate(conn, sql)
              data = ibm_db.fetch_both(stmt)
              print(data)
              if data:
                session["name"] = escape(id)
                session["password"] = escape(password)
                flash('You were successfully logged in')
                return redirect(url_for("userhome",name=id))

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

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

            return render_template("userlogin1.html")



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

HTML FILES :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Care Registry</title>
     <link rel="stylesheet" href="../static/index.css">
     <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></s
cript>
</head>
<body><div class="homebutton"><a href="{{url_for('userlogin')}}">LOG
OUT</a></div>

        <h1 style="font-size: 30px;">Customer Care Registry</h1>
        <br>
        <div class="flash" id="para">
        {% with messages = get_flashed_messages() %}
        {% if messages %}
           <ul>
              {% for message in messages %}
              <h4>{{ message }}</h4>
              {% endfor %}
           </ul>
        {% endif %}
     {% endwith %}
    </div>
        <h3 style="text-align: center;">Hello {{name}}.....!</h3>
        <div class="box">
            <a href="{{ url_for('file_complaint',name=name) }}">File
Complaint</a>
            <a href="{{ url_for('view_status',name=name) }}">View
Status</a>

        </div>



    </body>

</html>
<script>
    $(document).ready(function(){
      $(".flash").hide(3200);
    });
  </script>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=<device-width>, initial-
scale=1.0">
```

```
    <title>Customer Care Registry</title>
    <link rel="stylesheet" href="../static/style.css">

</head>
<body>
  <style type="text/css">
    .homebutton{

    padding: 10px;
    background-color:rgb(159, 249, 159);
    color:rgb(1, 65, 8);

}
  </style>

  <h1 style="font-size: 30px;">Customer Care Registry</h1>
  <br>

    <div class="main-block">

        <h1>File Complaint</h1>
        <form action="{{ url_for('addcomplaint') }}" method="POST">

          <input type="text" name="customer" id="detail" value="{{nam}}"
required/>
          <input type="text" name="complaint" id="detail"
placeholder="Complaint" required/>

          <input type="text" name="detail" id="detail"
placeholder="Complaint Detail" required/>
          <hr>

          <hr>
          <div class="btn-block">
            <button type="submit" href="/">File Complaint</button>
          </div>
        </form>
      </div>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Customer Care Registry</title>
    <link rel="stylesheet"
href="{{url_for('static',filename='css/bootstrap.min.css')}}">
    <link rel="stylesheet" href="../static/table.css">
    <link rel="stylesheet" href="../static/index.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></s
cript>
</head>
<body>
    <div class="homebutton"><a href="{{url_for('agent')}}">Log
Out</a></div>

    <h1 style="font-size: 30px;">Customer Care Registry</h1>
    <br>
    <h1>Hello Agent {{name}}.....!</h1>
    <h4 style="text-align: center;">Give Solution</h4>
    <div class="flash" id="para">
        {% with messages = get_flashed_messages() %}
        {% if messages %}
           <ul>
              {% for message in messages %}
              <h4>{{ message }}</h4>
              {% endfor %}
           </ul>
        {% endif %}
```

```
     {% endwith %}
    </div>

            <table id="customers" >
                <thead>
                    <tr>
                    <th>S.No</th>
                    <th>Complaint ID</th>
                    <th>Customer Name</th>
                    <th>Complaint</th>
                    <th>Complaint Detail</th>
                    <th>Solution</th>

                </tr>
                </thead>
                <tbody>
                    {% if comp | length == 0  %}
                    <td>1.</td>
                    <td>No Record</td>
                    <td>No Record</td>
                    <td>No Record</td>
                    <td>No Record</td>
                    <td>No Record</td>
                    {% else %}
                    {% for res in comp %}
                        {% set i=loop.index %}
                    <tr>
                        <td>{{i}}</td>
                        <td>{{res['COMPLAINT_ID']}}</td>
                        <td>{{res['CUSTOMER_NAME']}}</td>
                        <td>{{res['COMPLAINT']}}</td>
                        <td>{{res['COMPLAINT_DETAIL']}}</td>
                        <td >
                            <form action="{{ url_for('addsoluin') }}"
method="POST">
```

```
                                            <textarea name="solution" rows="3"
cols="50">

                                                </textarea>
                                        <input type="hidden"  name="gname"
value="{{name}}">
                                        <input type="hidden"  name="cid"
value="{{res['COMPLAINT_ID']}}">
                                        <input type="submit" >
                                    </form></td>
                                </tr>
                                {% endfor %}
                                {% endif %}
                            </tbody>
                        </table>

</body>
</html>
<script>
    $(document).ready(function(){
      $(".flash").hide(3000);
    });
  </script>
```