Project Report

Team ID	PNT2022TMID39363
Project Name	**Customer Care Registry**

1. INTRODUCTION

1.1 Project Overview:

the Customer Service Desk is a web-based project. Customer Service also known as Client Service is the provision of service to customers Its significance varies by product, industry and domain. In many cases customer services is more important if the information relates to a service as opposed to a customer. Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition. 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 arefacing. 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.

1.2 Purpose:

An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through and e-mail. the system should have capability to integrate with any Service Provider from any domain or industry like Banking, telecom Insurance etc. It is also known as Client Service is the provision of service to customers Its significance varies by product industry and domain. In many cases customer services is more important if the information relates to a service as opposed to as Customer. Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition. this Application mainly developed to help the customer in processing their complaints and issues. It is a process of examining customer tickets, which should be carried out in a systematic and orderly manner. this practice is primarily aimed at minimizing consumer dissatisfaction with the purchased products, increasing service satisfaction, and ensuring quality. It allows companies to respond to customer inquiries, providessupport, and improves the handling of tickets at the appointed time.

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. When the company pushes the wrong product or service to customer this can severely impact to company’s profit, growth and brand reputation. the customer cannot track the status of the Queries that are posted by them. Some queries will be left Unanswered. to overcome this issues a good customer care should be provided to solve the customer’s queries.

2.2 References
:

title: Automated ticket Routing System **Author NAME:**

Muhammad Zikri bin Zulkifli**Publication YEAR:** 2011

Description:

In the existing helpdesk system, the tickets were created and assigned to the end user manually. When the ticket is created, it is assigned to the agent manually before they attend that specific ticket. this manual process of ticket creation needs more manpower and takes more time. Instead of putting the effort and time into this task, the ticket creation and assigning can be done automatically when we create an Automated ticket Routing system. the automated ticket creation and assignment process reduce the time and then the manpower can be used for other purposes. then, by using the manual ticket creation and assignment process, the distribution of good skill sets, and workload balancing will be missed out. Finding a good skill set and assigning the tickets to the specific skilled agent automatically is considered a good job distribution. Here, the wrong agent represents the sense that the agent doesn’t know well about that particular problem or issue. If the tickets are mistakenly routed, then the resources may get wasted and a lotof time will be spent unnecessarily. Using the location, skill sets, work schedule, and workload balancing, the tickets can be routed automatically to that particular agent perfectly. We can execute the above process perfectly by categorizing the tickets based on the issues.

title: Knowledge-Based Helpdesk System

Author NAME: Mohamad Safran Bin Sulaiman, Abdul Muin AbdulRahman, Norvaline Bt. Nasiruddin

Publicaton YEAR: 2012

Descripton:

A knowledge-Based helpdesk system is a web-based system that is used to provide technical support to an organization or to management. then, it acts as a Service Provider to that particular organization. the main objective of this Knowledge based system is to provide technical support to the end users of a particular organization. Using this Knowledge-based Helpdesk system, an organization can improve their end user's performance and make their end users technically well educated. Once the Knowledge- based helpdesk system is designed, it is tested on the Information technology (I t) center, Engineering Division (BKJ), etc. to havea better support solution for management, the Knowledge-based system is introduced.Usually, the Knowledge-based system consists of questions that are frequently raised by the end users. All the frequent questions are combined into categories and then, it is provided as a solution. the end users can solve their problems manually by themselves just by reading and implementing the solution that is provided. Also, the solutions that are provided by the helpdesk team can be used on future problems too. Hence, it is called a continuity and contingency process.

TITLE: Smart Help Desk Automated ticketing System

AUTHOR NAME: Dhiraj temkar, Sheetal Singh, Leema Bari, Prof.SnigdhaBanga

PUBLICATION YEAR: 2021

Description:

Automated technical queries help desk is proposed to possess instant real-time quick solutions and ticket categorization. Incorrect routing of tickets to the incorrect resolver group causes delays in resolving the matter. It also causes unnecessary resource utilization, and customer dissatisfaction and affects the business. to beat these problems, the proposed "Smart Automated ticketing System" supports supervised machine learning techniques that automatically predict the category of the ticket using the natural language ticket description entered by the user through a chat interface. It also helps in faster resolution of customer issuesandsends them an email about the status of the ticket. this process assures customer satisfaction and also keeps the customers within the loop.

Title: Theory And Practice Of Customer-Related Improvements

Author NAME: Daniel Gyllenhammar, Et Al**Publication**

YEAR: 2022 **Description:**

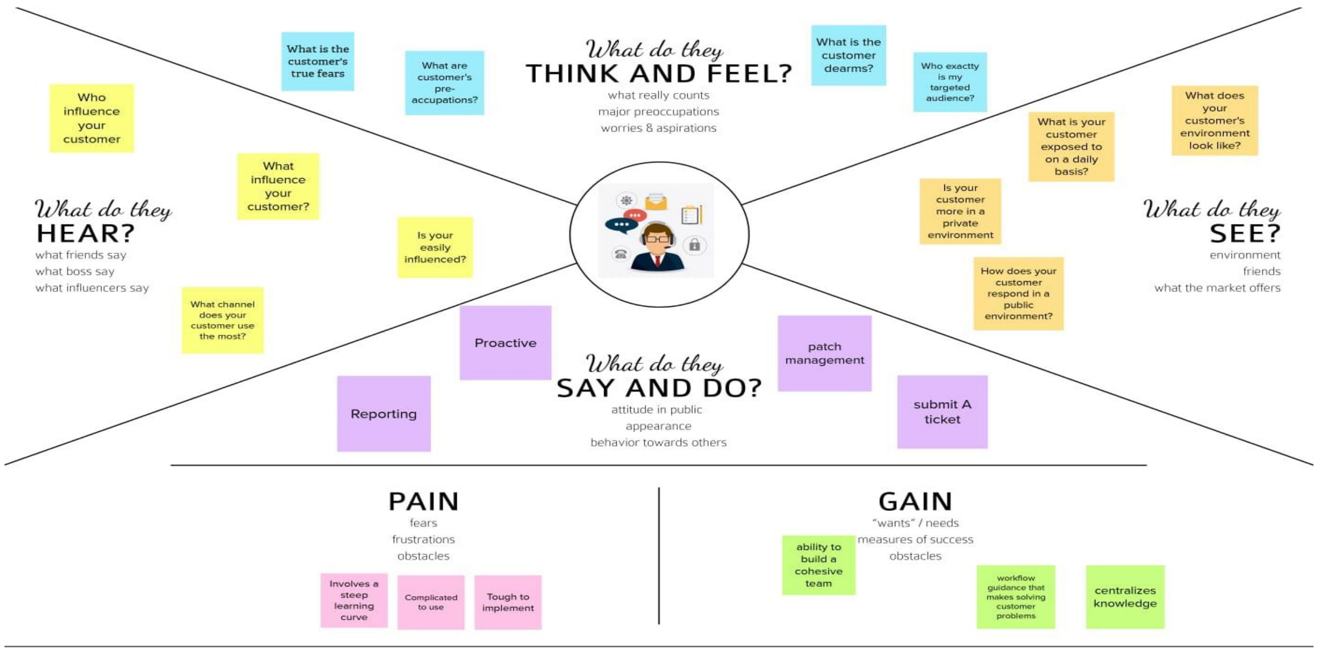
In An Organization, The Information Technology (I t) Support Help Desk Operation Is An Important Unit That Handles The I t Services Of A Business. Many Large-Scale Organizations Handle Engagement And Requests With Employees On A 24×7 Basis. As With Any Routine Tasks, Most Processes Of The Support Help Desk Unit Are Considered Repetitive In Nature Repetitive Tasks Such As Entering Information Into An Application, Resetting Passwords, Unlocking Applications, And Credentials Errors. The Industry Has Now Come To Realize That Many Repetitive Business Processes And Tasks Can Be Automated By Using Robotic Process Automation (RPA) Bots Or Robotic Processes Automotive Software Bots. The Idea Is To Take The Repetitive Workload And Hand It Over To The RPA Bots So That The Employees Could Focus On More Value-Adding Tasks And Decision-Making For The Organization. The RPA Bot Would Also Help To Reduce Human Errors And Make Processes More Efficient, Which Would Finally Result In Cost Savings And Productivity Increase.

2.3 Problem Statement Definition:

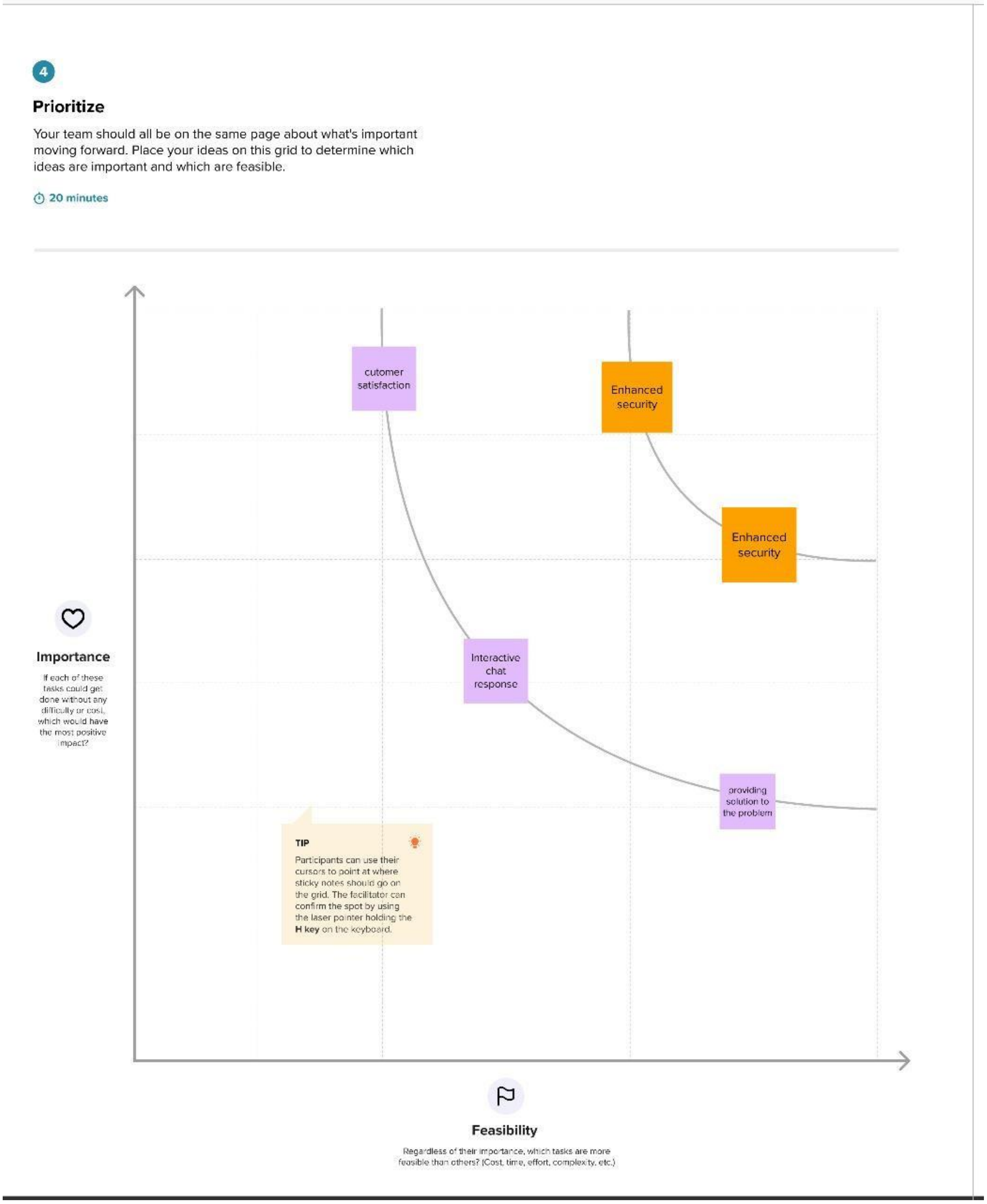
Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	User	Ticket Booking	Time Delay	Agent Not Responding	Sad
PS-2	User (Agent)	Solve Problem	Customer Not Responding	Customer Unavailable	Frustrated
PS-3	User (Admin)	Backup Data	Data Loss	System Failure	Anxiety
PS-4	User	Looking for Status	Status Unavailable	Agent Not Updated	Stressed

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas


What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
What is the customer's true fears
What are customer's pre-accupations?
What is the customer dearms?
Who exactty is my targeted audience?
What do they
HEAR?
what friends say
what boss say
what influencers say
Who influence your customer
What influence your customer?
Is your easily influenced?
What channel does your customer use the most?
What do they
SEE?
environment
friends
what the market offers
What does your customer's environment look like?
What is your customer exposed to on a daily basis?
Is your customer more in a private environment
How does your customer respond in a public environment?
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
Proactive
Reporting
patch management
submit A ticket
PAIN
fears
frustrations
obstacles
Involves a steep learning curve
Complicated to use
Tough to implement
GAIN
"wants" / needs
measures of success
obstacles
ability to build a cohesive team
workflow guidance that makes solving customer problems
centralizes knowledge

3.2 Ideation & Brainstorming



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

Importance

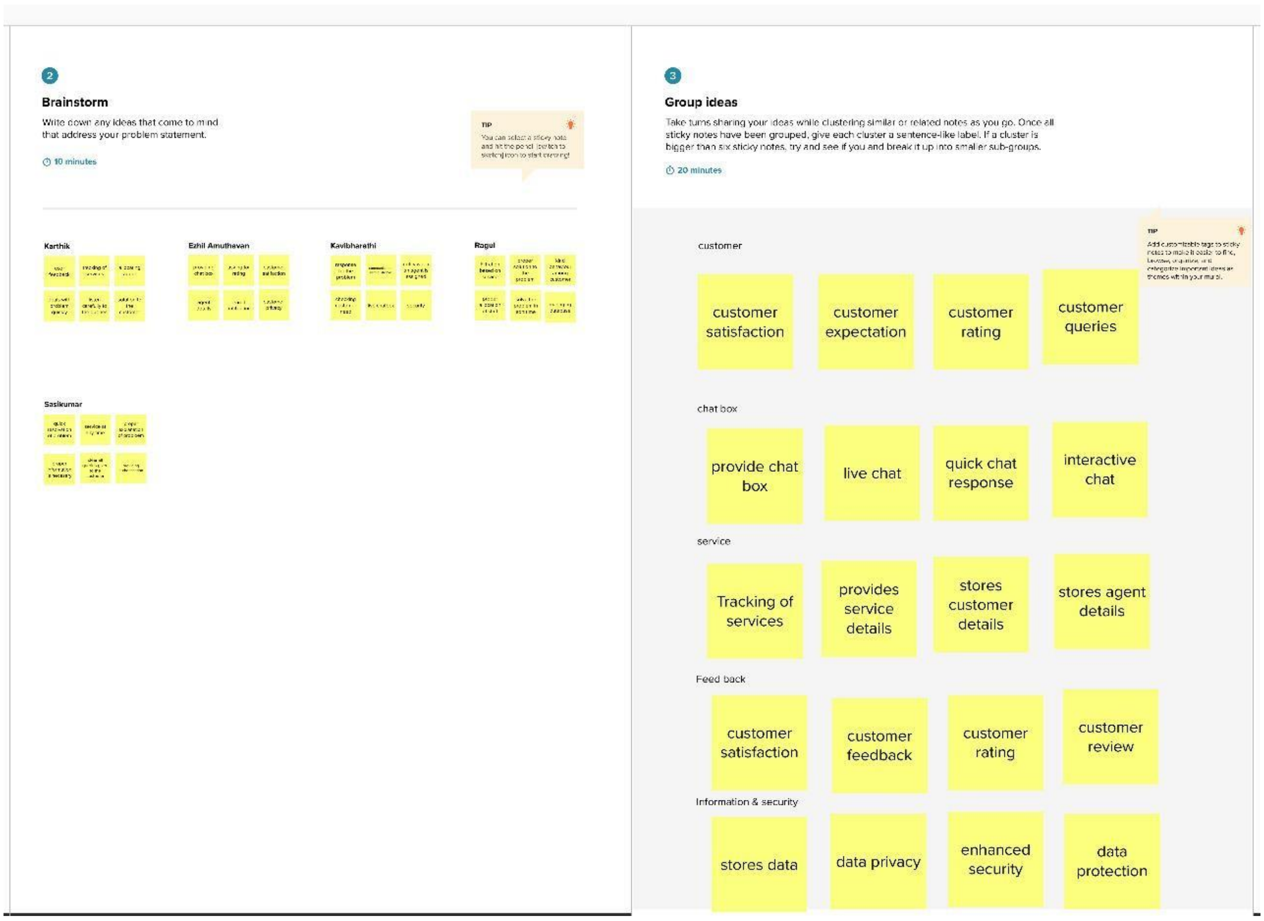
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Karthik

- user feedback
- tracking of services
- [illegible]
- [illegible] problem quickly
- [illegible] carefully to the customer
- solution to the customer

Ezhil Amuthevan

- providing chat box
- asking for rating
- customer satisfaction
- agent details
- [illegible]
- customer privacy

Kavibharathi

- response for the problem
- [illegible]
- [illegible] agent is assigned
- checking [illegible] need
- [illegible]
- security

Rogul

- [illegible] based on [illegible]
- proper solutions for problem
- kind [illegible] customer
- [illegible]
- [illegible]
- [illegible]

Sasikumar

- quick resolution of problem
- service at any time
- proper explanation of problem
- proper information is necessary
- [illegible]
- [illegible]

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

customer

- customer satisfaction
- customer expectation
- customer rating
- customer queries

chat box

- provide chat box
- live chat
- quick chat response
- interactive chat

service

- Tracking of services
- provides service details
- stores customer details
- stores agent details

Feed back

- customer satisfaction
- customer feedback
- customer rating
- customer review

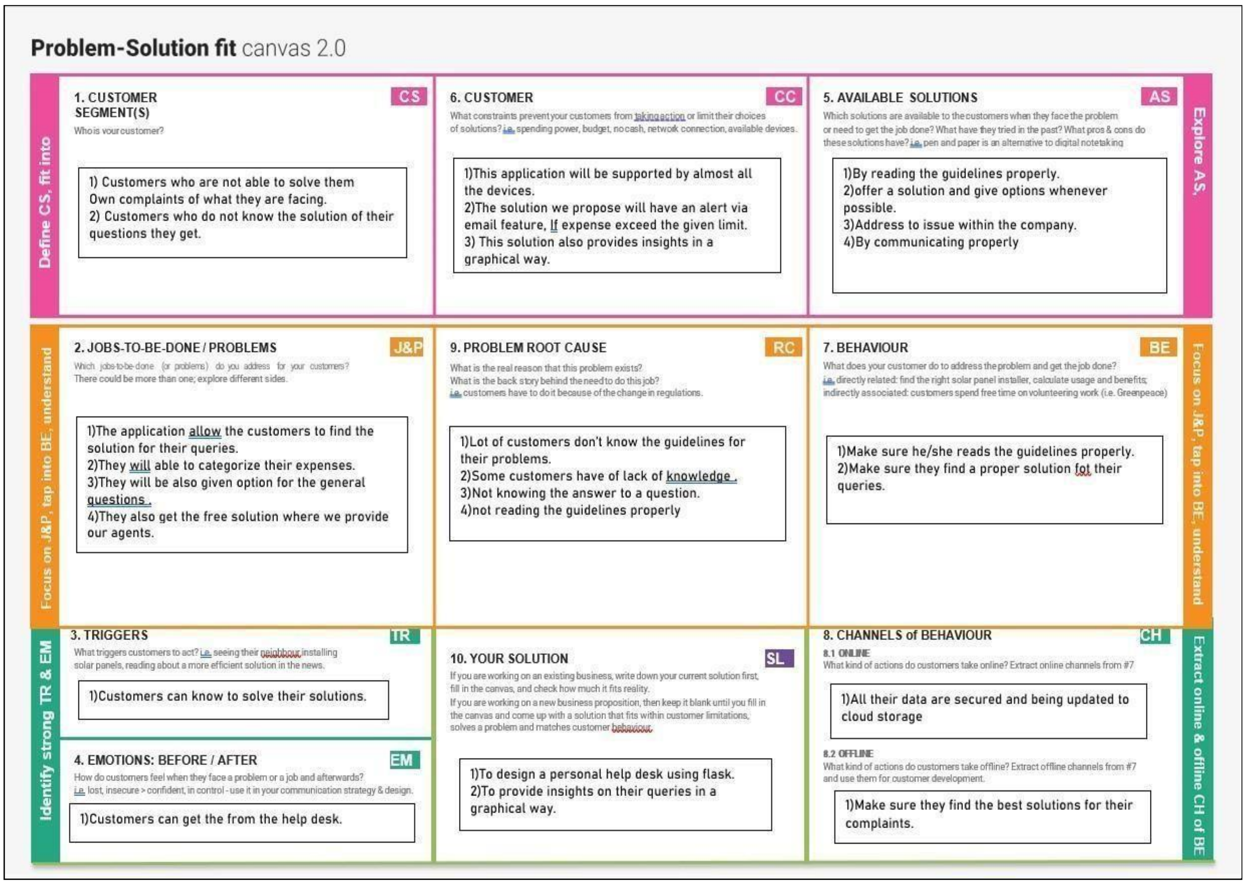
Information & security

- stores data
- data privacy
- enhanced security
- data protection

3.3 Proposed Solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Your customer service problem- solving starts by diving due importance to listening. this is often overlooked, which may result in catching the customer service agent off guard with questions to which youmay not have the appropriate answer.
2.	Idea / Solution description	Implement a website-based chat bot system Okay, we aren’t bring something totally innovative to the conversation here...but if you don’t have a live chat with a degree of automation,. To learn more about the business and how they can help you.
3.	Novelty / Uniqueness	One of the most important aspects of call center customer service is maintaining a **professional tone of voice**. Communicating over the phone poses challenges and risks miscommunication
4.	Social Impact / Customer Satisfaction	Meeting Customer Expectation Level Three: Delighting your Customers on theany situation Level Four: Amazing your Customers
5.	Business Model (Revenue Model)	Create a customer service strategy Setting the customer service goals Assess and build a customer service team Create a customer journey and service design map
6.	Scalability of the Solution	Offer Multi-Channel Customer Service Find the Perfect Help Desk Platform Train Your Support Agents Early and Often. Keep up With (and Analyze) Reports

3.4 Problem Solution fit



Problem-Solution fit canvas 2.0

Define CS, fit into

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

1) Customers who are not able to solve them
Own complaints of what they are facing.
2) Customers who do not know the solution of their questions they get.

6. CUSTOMER CC

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

1)This application will be supported by almost all the devices.
2)The solution we propose will have an alert via email feature, If expense exceed the given limit.
3) This solution also provides insights in a graphical way.

5. AVAILABLE SOLUTIONS AS

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

1)By reading the guidelines properly.
2)offer a solution and give options whenever possible.
3)Address to issue within the company.
4)By communicating properly

Explore AS,

Focus on J&P, tap into BE, understand

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

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

1)The application allow the customers to find the solution for their queries.
2)They will able to categorize their expenses.
3)They will be also given option for the general questions .
4)They also get the free solution where we provide our agents.

9. PROBLEM ROOT CAUSE RC

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

1)Lot of customers don't know the guidelines for their problems.
2)Some customers have of lack of knowledge .
3)Not knowing the answer to a question.
4)not reading the guidelines properly

7. BEHAVIOUR BE

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

1)Make sure he/she reads the guidelines properly.
2)Make sure they find a proper solution fot their queries.

Focus on J&P, tap into BE, understand

Identify strong TR & EM

3. TRIGGERS TR

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

1)Customers can know to solve their solutions.

4. EMOTIONS: BEFORE / AFTER EM

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

1)Customers can get the from the help desk.

10. YOUR SOLUTION SL

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

1)To design a personal help desk using flask.
2)To provide insights on their queries in a graphical way.

8. CHANNELS of BEHAVIOUR CH

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

1)All their data are secured and being updated to cloud storage

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

1)Make sure they find the best solutions for their complaints.

Extract online & offline CH of BE

4.Requirement ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn Register with valid mobile number
FR-2	User Confirmation	Confirmation via Email Confirmationvia OTP Two step verification for new device login.
FR-3	Agent Registration	Registration through Form Registration through Gmail Registration through LinkedIn Register with valid mobile number
FR-4	Agent Confirmation	Confirmation via Email Confirmationvia OTP Two step verification for new device login.
FR-5	Admin	Admin have both user details and agent detail. Admin maintain agent allotment to the user based on problem's category.

4.2 Non-Functional requirements

Following are the non-functional requirements of the proposed solution.

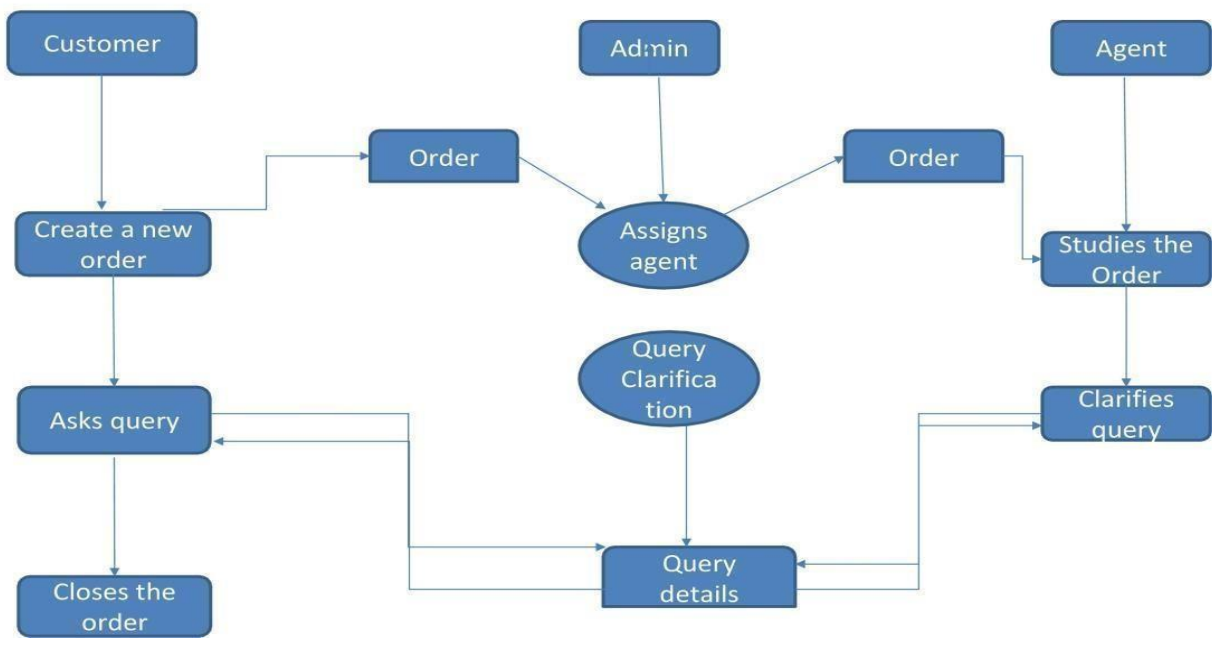
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	To provide optimal usability for our proposed solution we have mainly concentrated on easier navigation throughout our website. For user, they can easily login with their credentials and also they can register by themselves either with unique valid email id or with their mobile number if they don't have any prior account. After good navigation we have concentrated on visual clarity and developed web application which looks pleasant and simple thus making easier accessible to any aged person. For the first time users, Guide tour will also be available in order to provide better user satisfaction. Also, made our web application flexible to all type of devices such as android, mac and desktops.
NFR-2	**Security**	Before any user trying to login their account to any new device ,verification code will be sent either to their registered email id or to their registered mobile number. Only after entering their code, they will be allowed to login. That code will also made expire within particular

		time limit. Also notification will be sent for each and every user activity. Thus everyone will have a secured account and also their details will be maintained securely in the admin side.
NFR-3	**Reliability**	Since we had split the agents into categories, system’s response time for each and every individual will be lesser. Thus making our web application more reliable.
NFR-4	**Performance**	In order to bring best performance, we have concentrated on overload of user requests. To minimize the overloads and to minimize the system’s response time we have created more agents service. Agents will be separated and categorized according to the user’s needs. For example to resolve product missing category some agents will be assigned and to resolve damaged products category some agents will be assigned. so every individual user will be allotted with individual agents.
NFR-5	**Availability**	Customer care registry will be made available even in the weekends and our agents will also be allotted at anytime to any individual user. User can interact with their respective agents 24*7 by following proper user-agent guidelines.
NFR-6	**Scalability**	With respect to increase in user’s requests ,allotment will be increased. Data storage will increase accordingly. Rescaling is always adaptable.

5. PROJECT DESIGN

5.1 Dataflow Diagrams

Data flow diagram for Customer care Registry



5.2 Solution & technical Architecture:

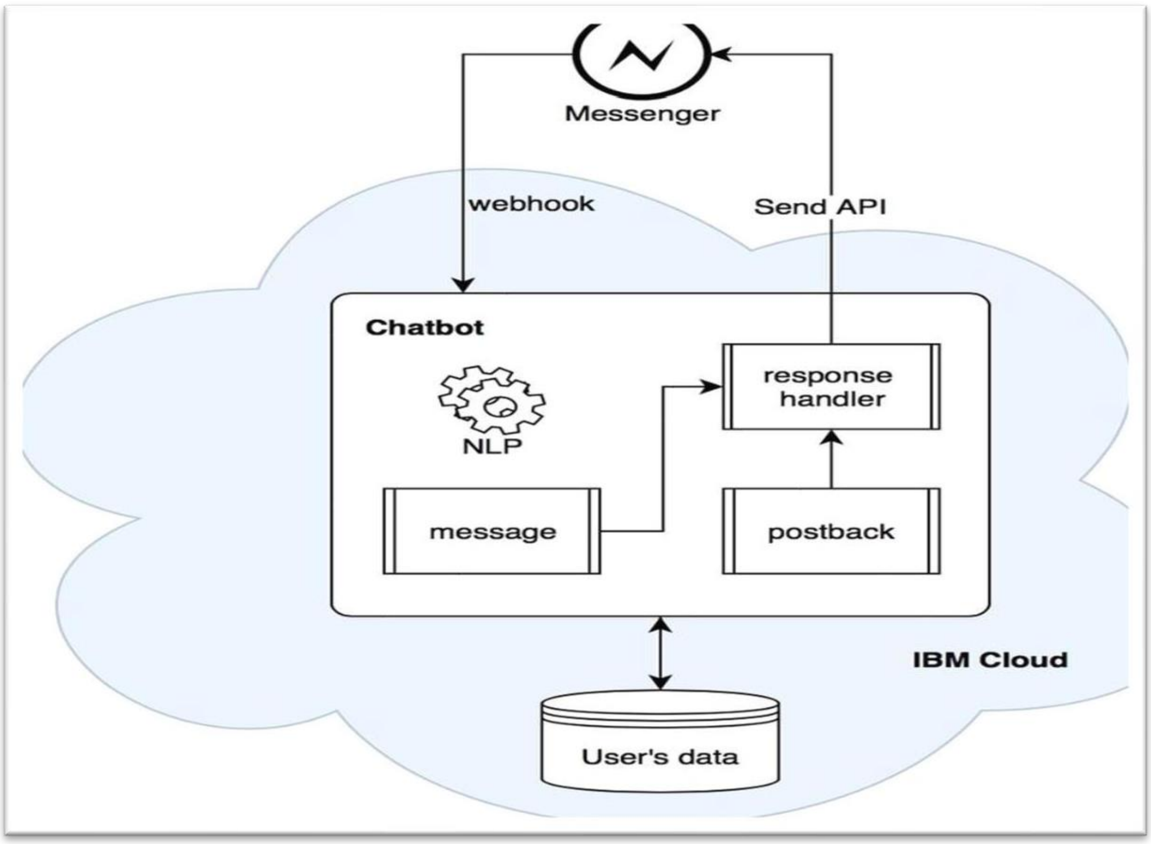


table-1: Components & technologies:

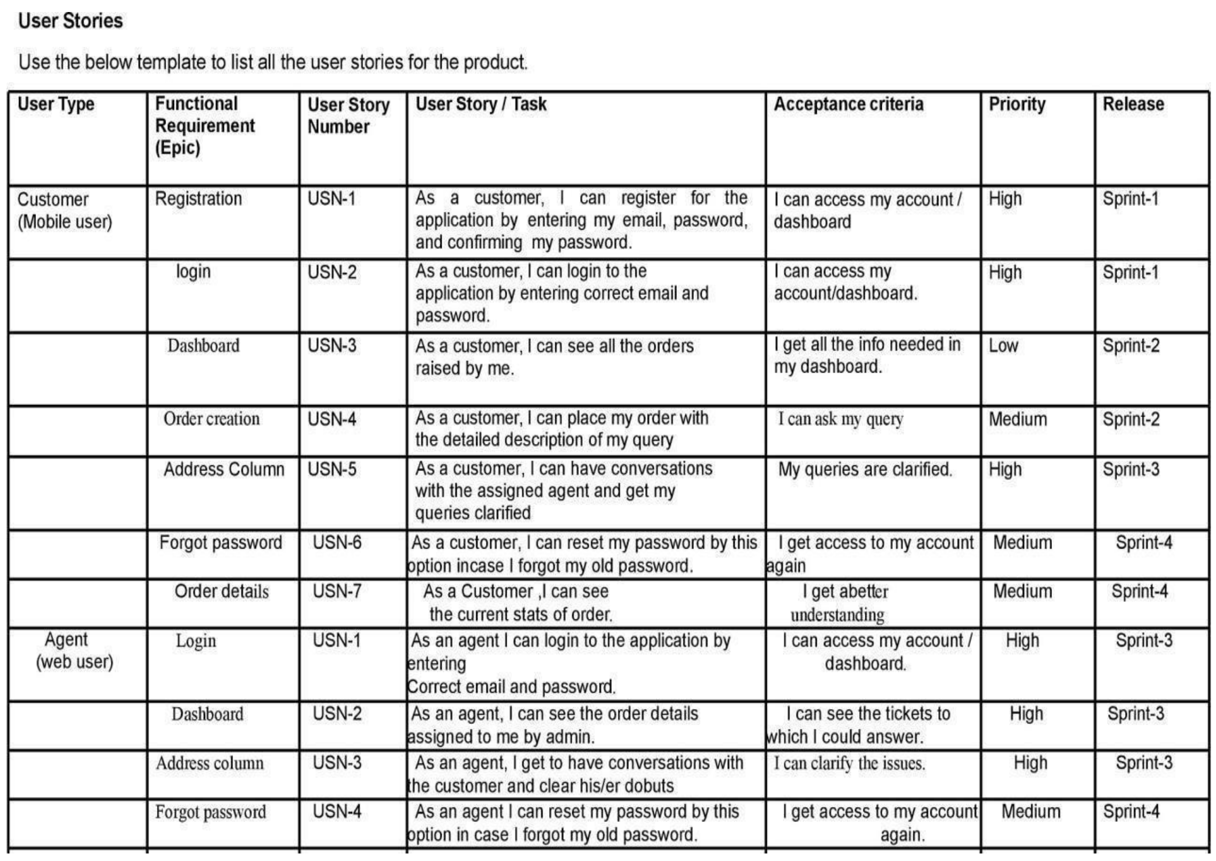
S.NO	COMPONENT	DESCRIPTION	TECHNOLOGY
1.	User interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript Angular js/node js
2.	Application Logic-1	Logic for a process in the application	Python
3.	application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	database	Data Type, Configurations etc.	My SQL
6.	Cloud database	Database Service on Cloud	IBM DB2, IBM Cloud nt etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other storage service orLocal Filesystem
11.	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	Flask-Python Framework	Flask -Python
2.	Security Implementations	Digital Certificate SSL Secuirty	IBM Cloud and IBM DB2
3.	Scalable Architecture	Sendgrid API and Json Server	IBM Object Storage
4.	Availability	Large Number of Customer Utilize	IBM Kubernates
5.	Performance	Fast Recovering Data From IBMDB2 and flexiblerequestand response from Cloud	IBM Cloud

5.3 User Stories

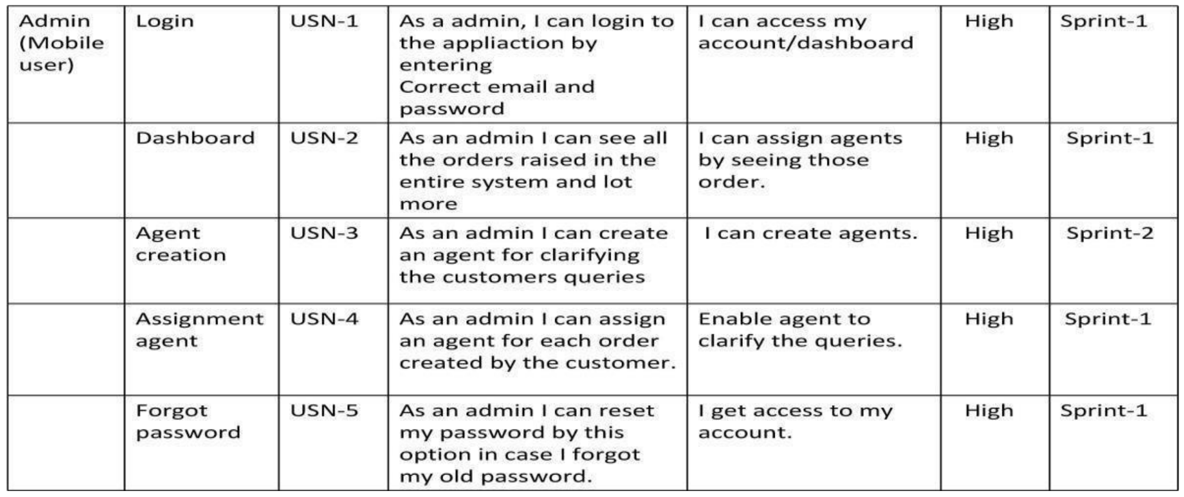
Use the below template to list all the user stories for the product.



User Stories

Use the below template to list all the user stories for the product.

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

6. PROJECT PLANNING & SCHEDULING

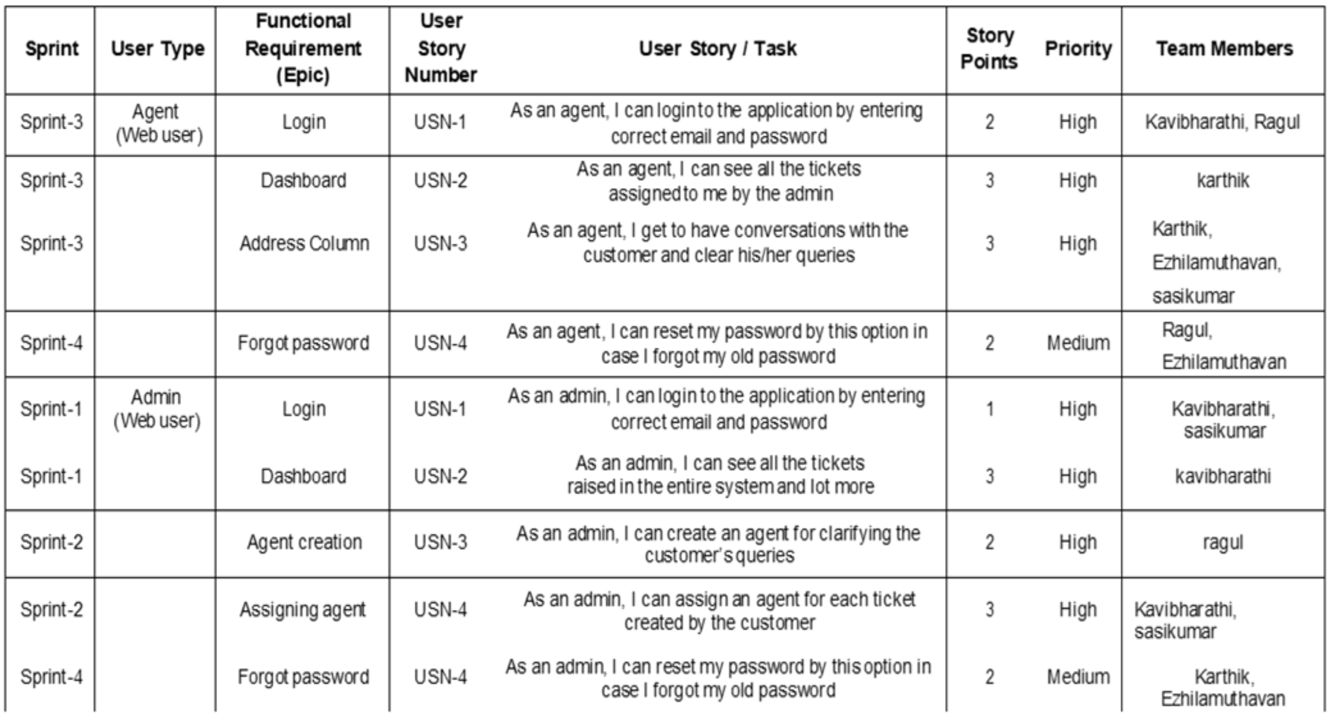
6.1 Sprint Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information byreferring to technical papers research publications etc.	09 SEPTEMBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	10 SEPTEMBER 2022
Ideation	List them by organizing the brainstorming session and prioritize the top 3 ideas based on feasibility & importance.	12 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	03 OCTOBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	05 OCTOBER 2022
Solution Architecture	Prepare a solution architecture document.	07 OCTOBER 2022

Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	15 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	15 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	19 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram.	14 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	24 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	20 NOVEMBER 2022 (PLANNED)

Product Backlog, Sprint Schedule, and Estimation

Sprint	User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Customer (Web User)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	2	High	Kavibharathi, Ragul
Sprint-1		Login	USN-2	As a customer, I can login to the application by entering correct email and password	1	High	Karthik, Ezhilamuthavan
Sprint-1		Dashboard	USN-3	As a customer, I can see all the tickets raised by me and lot more	3	High	Sasikumar
Sprint-2		Ticket creation	USN-4	As a customer, I can create a new ticket with the detailed description of my query	2	High	Kavibharathi, Karthik
Sprint-3		Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	3	High	Ragul, Sasikumar, Ezhilamuthavan
Sprint-4		Forgot password	USN-6	As a customer, I can reset my password by this option in case I forgot my old password	2	Medium	Kavibharathi,ragul, sasikumar
Sprint-4		Ticket details	USN-7	As a customer, I can see the current status of my tickets	2	Medium	Karthik, Ezhilamuthavan



Sprint	User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Agent (Web user)	Login	USN-1	As an agent, I can login to the application by entering correct email and password	2	High	Kavibharathi, Ragul
Sprint-3		Dashboard	USN-2	As an agent, I can see all the tickets assigned to me by the admin	3	High	karthik
Sprint-3		Address Column	USN-3	As an agent, I get to have conversations with the customer and clear his/her queries	3	High	Karthik, Ezhilamuthavan, sasikumar
Sprint-4		Forgot password	USN-4	As an agent, I can reset my password by this option in case I forgot my old password	2	Medium	Ragul, Ezhilamuthavan
Sprint-1	Admin (Web user)	Login	USN-1	As an admin, I can login to the application by entering correct email and password	1	High	Kavibharathi, sasikumar
Sprint-1		Dashboard	USN-2	As an admin, I can see all the tickets raised in the entire system and lot more	3	High	kavibharathi
Sprint-2		Agent creation	USN-3	As an admin, I can create an agent for clarifying the customer's queries	2	High	ragul
Sprint-2		Assigning agent	USN-4	As an admin, I can assign an agent for each ticket created by the customer	3	High	Kavibharathi, sasikumar
Sprint-4		Forgot password	USN-4	As an admin, I can reset my password by this option in case I forgot my old password	2	Medium	Karthik, Ezhilamuthavan

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 Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022
Sprint-2	7	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022
Sprint-3	11	4 Days	06 Nov 2022	11 Nov 2022	11	09 Nov 2022
Sprint-4	8	4 Days	10 Nov 2022	15 Nov 2022	8	13 Nov 2022

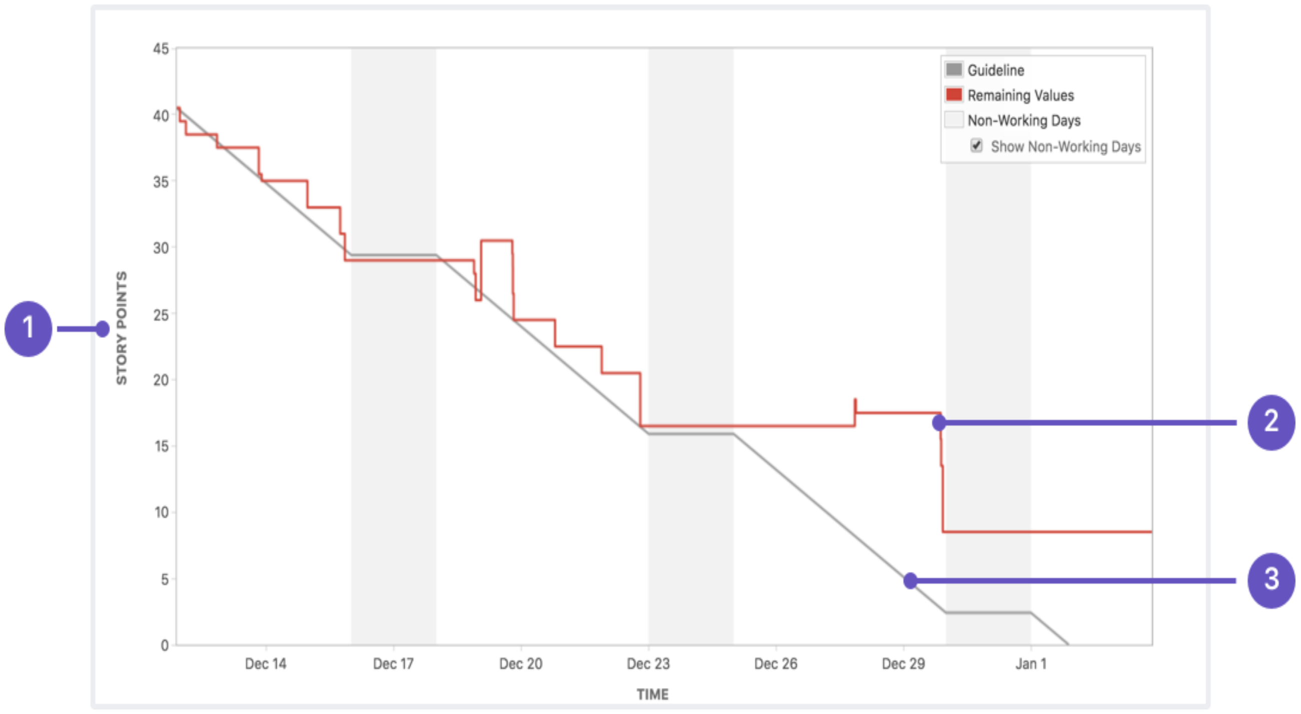
VELOCITY;

Imagine we have 10 days sprint duration and the velocity of the team is 20(point's per sprint)let's calculate the team average velocity(AV) per iteration unit (story point per day)

Sprint	Total Story Points	Duration	Average Velocity
Sprint 1	10	6 Days	10/6 = 1.66
Sprint 2	7	6 Days	7/6 = 1.16
Sprint 3	11	4 Days	11/4 = 2.75
Sprint 4	8	4 Days	8/4 = 2
Total	36	20 Days	36 / 20 = 1.8

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

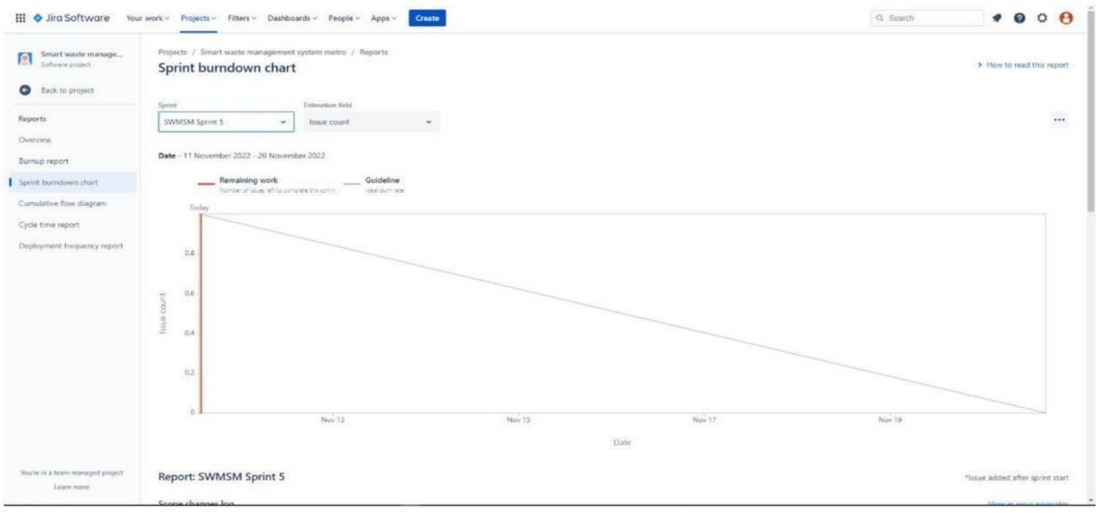


6.2. Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint StartDate	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint ReleaseDate (Actual)
Sprint-1	20	7 Days	24 Oct 2022	30 Oct 2022		30 Oct 2022
Sprint-2	20	7 Days	31 Oct 2022	06 Nov 2022		06 Nov 2022
Sprint-3	20	8 Days	07 Nov 2022	14 Nov 2022		14 Nov 2022
Sprint-4	20	7 Days	14 Nov 2022	21 Nov 2022		21 Nov 2022

6.3 Reports from JIRA

BURNDOWN CHART


Projects / Smart waste management system metro / Reports

Sprint burndown chart

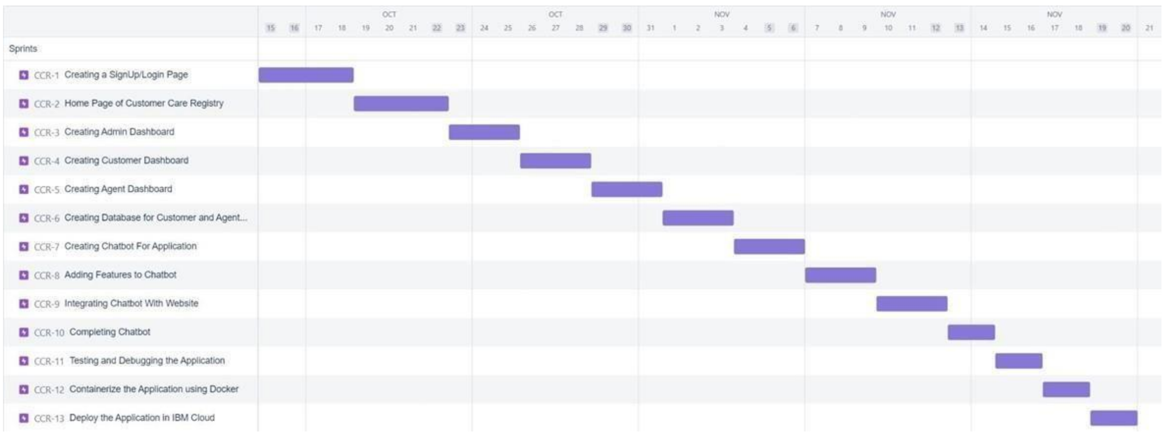
Sprint: SWMSM Sprint 5

Estimation field: Issue count

Date - 11 November 2022 - 20 November 2022

Report: SWMSM Sprint 5

*Issue added after sprint start


Sprints

- CCR-1 Creating a SignUp/Login Page
- CCR-2 Home Page of Customer Care Registry
- CCR-3 Creating Admin Dashboard
- CCR-4 Creating Customer Dashboard
- CCR-5 Creating Agent Dashboard
- CCR-6 Creating Database for Customer and Agent...
- CCR-7 Creating Chatbot For Application
- CCR-8 Adding Features to Chatbot
- CCR-9 Integrating Chatbot With Website
- CCR-10 Completing Chatbot
- CCR-11 Testing and Debugging the Application
- CCR-12 Containerize the Application using Docker
- CCR-13 Deploy the Application in IBM Cloud

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

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

the mission of veteran- and woman-owned Coding Solutions is to mobile the next generation of I t talent and provide them the tools and resources they require to make your business successful. Innovative talent is necessary for innovative technologies. We wish to provide Coding Solutions prospects to assist you expand our Alabama team.

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

7.1Feature 1

7 Main types of customer needs:

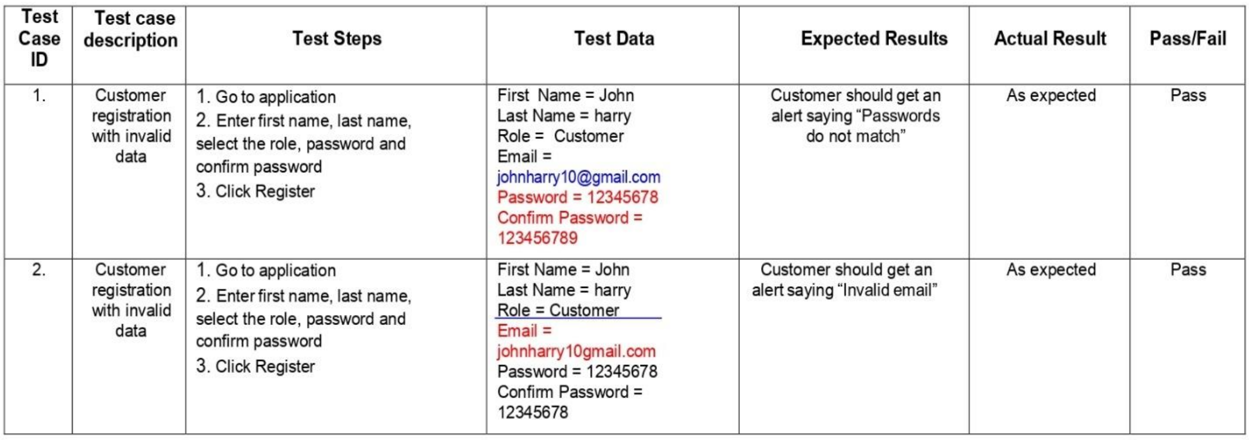
- Friend lines
- Empathy
- Fairness
- Control
- Alternatives
- Information

7.2 Feature

- Complaint tracking
- Email Alert
- 24/7 Monitoring

8. TESTING

8.1 test Cases



Test Case ID	Test case description	Test Steps	Test Data	Expected Results	Actual Result	Pass/Fail
1.	Customer registration with invalid data	1. Go to application 2. Enter first name, last name, select the role, password and confirm password 3. Click Register	First Name = John Last Name = harry Role = Customer Email = johnharry10@gmail.com Password = 12345678 Confirm Password = 123456789	Customer should get an alert saying “Passwords do not match”	As expected	Pass
2.	Customer registration with invalid data	1. Go to application 2. Enter first name, last name, select the role, password and confirm password 3. Click Register	First Name = John Last Name = harry Role = Customer Email = johnharry10gmail.com Password = 12345678 Confirm Password = 12345678	Customer should get an alert saying “Invalid email”	As expected	Pass

3.	Customer registration with invalid data	1. Go to application 2. Enter first name, last name, select the role, password and confirm password 3. Click Register	First Name = Jo Last Name = harry Role = Customer Email = johnharry@10gmail.com Password = 12345678 Confirm Password = 12345678	Customer should get an alert saying "First name should be at least 3 characters long!"	As expected	Pass
4.	Customer registration with invalid data	1. Go to application 2. Enter first name, last name, select the role, password and confirm password 3. Click Register	First Name = John Last Name = harry Role = Customer Email = johnharry@10gmail.com Password = 1234 Confirm Password = 1234	Customer should get an alert saying "Passwords must be at least 8 characters long!"	As expected	Pass
5.	Customer registration with valid data	1. Go to application 2. Enter first name, last name, select the role, password and confirm password 3. Click Register	First Name = John Last Name = harry Role = Customer Email = johnharry@10gmail.com Password = 12345678 Confirm Password = 12345678	Customer should be able to register to the application	As expected	Pass
6.	Customer login using the invalid data	1. Go to application 2. Enter email, password 3. Click Login	Email = johnharry10gmail.com Password = 12345678	Customer should get an alert saying "Invalid email"	As expected	Pass
7.	Customer login using the invalid data	1. Go to application 2. Enter email, password 3. Click Login	Email = johnharry@gmail.com Password = 12345678	Customer should get an alert saying "User does not exist"	As expected	Pass
8.	Customer login using invalid data	1. Go to the application 2. Enter email, password 3. Click Login	Email = johnharry10@gmail.c om Password = 12345678999	Customer should get an alert saying "Wrong Password!"	As expected	Pass
9.	Customer Login using Valid data	1. Go to application 2. Enter email, password 3. Click Login	Email = johnharry@10gmail.c om Password = 12345678	Customer should login to the application	As expected	Pass
10.	Admin login using invalid data	1. Go to application 2. Enter email, password 3. Click Login	Email = admin.ccr@gmail.com Password = admin.ccr	Admin should get an alert saying 'Invalid Password!'	As expected	Pass
11.	Admin login using invalid data	1. Go to application 2. Enter email, password 3. Click Login	Email = admin@gmail.com Password = admin.ccr@2022	Admin should get an alert saying 'Invalid Email!'	As expected	Pass
12.	Admin login using valid data	1. Go to application 2. Enter email, password 3. Click Login	Email = admin.ccr@gmail.com Password = admin.ccr@2022	Admin should login to the application	As expected	Pass
13.	Customer logging out	1. Go to the Nav Bar 2. Click on the right-side circular image 3. Click Logout	-	Customer should be able to log out from the application	As expected	Pass
14.	Customer logging out	1. Go to the Nav Bar 2. Click on the right-side circular image 3. Click Logout	-	Admin should be able to log out from the application	As expected	Pass

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Pass / Fail
15.	Customer creating a new ticket with empty query	1. Go to site 2. Customer login using email and password 3. Click "New Ticket" option in the Dashboard 4. Clicking the "New Ticket" button without typing any query in the given text area	Query = NULL	Customer should get an alert saying "Query cannot be empty!"	As expected	Pass
16.	Customer creating a new ticket with a valid query	1. Go to site 2. Customer login using email and password 3. Click "New Ticket" option in the Dashboard 4. Typing the query in the given text area 5. Clicking the "New Ticket" button	Query = "Hi. My I Phone 14 pro max is not turning on. It is a new unit I bought it just 2 days back. I don't know what happened. Can you help me please?"	The ticket gets inserted in the database. After that customer gets an alert saying 'Ticket created'	As expected	Pass
17.	Customer seeing all the tickets raised by him/her	1. Go to site 2. Customer login using email and password 3. Click "Tickets" option in the Dashboard	Tickets created by the customer which are already being inserted in the database	Customer should see the list of all the tickets raised by him/her	As expected	Pass
18.	Customer seeing all the tickets raised by him/her	1. Go to site 2. Customer login using email and password 3. Click "Tickets" option in the Dashboard	-	Customer should see a message "You are yet to raise a ticket"	As expected	Pass
19.	Customer seeing the query of a ticket	1. Go to site 2. Customer login using email and password 3. Click "Tickets" option in the Dashboard 4. Click "View" option in a ticket from the list of tickets	Tickets created by the customer which are already being inserted in the database	An alert should be shown having the actual query posted by the customer	As expected	Pass
20.	Customer seeing the assigned agent for a ticket	1. Go to site 2. Customer login using email and password 3. Click "Tickets" option in the Dashboard	• Tickets created by the customer which are already being inserted in the database • Admin assigned the agent for the ticket	Customer should be able to see the first name of the agent assigned	As expected	Pass
21.	Customer seeing the assigned agent for a ticket	1. Go to site 2. Customer login using email and password 3. Click "Tickets" option in the Dashboard	• Tickets created by the customer which are already being inserted in the database • Admin is yet to assign the agent	Customer should be able to see the "N/A" message displayed	As expected	Pass

22.	Admin seeing all the unassigned tickets	1. Go to site 2. Admin login using email and password 3. Click "Tickets" option in the Dashboard	• Tickets created by the customers which are already being inserted in the database • Admin did not assign agent for the tickets	Showing the tickets that are yet to be assigned an agent by the admin	As expected	Pass
23.	Admin seeing all the unassigned tickets	1. Go to site 2. Admin login using email and password 3. Click "Tickets" option in the Dashboard	• Tickets created by the customers which are already being inserted in the database • Admin assigned agents for all the tickets	Admin should just see the message "There is nothing left to assign"	As expected	Pass
24.	Admin assigning an agent for a ticket	1. Go to site 2. Admin login using email and password 3. Click "Tickets" option in the Dashboard 4. Select an agent from the dropdown given	• Tickets created by the customers which are already being inserted in the database • Admin did not assign the agent yet	Admin should get an alert saying "Do you really want to assign the agent for this ticket?". If admin clicks OK, then the agent is assigned for the ticket. The list gets updated	As expected	Pass
25.	Admin seeing the requests section	1. Go to site 2. Admin login using email and password 3. Click "Requests" option in the Dashboard	• Agent details in the database • Admin is yet to accept the agent	Admin should be able to see the list of all the requests made by the agents to the admin	As expected	Pass

26.	Admin seeing the requests section	1. Go to site 2. Admin login using email and password 3. Click "Requests" option in the Dashboard	• Agent details in the database • Admin accepted all the agents	Admin should just see the message "There are no pending requests"	As expected	Pass
27.	Admin accepting an agent from the request section	1. Go to site 2. Admin login using email and password 3. Click "Requests" option in the Dashboard 4. Click "Tick" mark that is against the agent details	• Agent details in the database • Admin is yet to accept the agent	The agent gets accepted and the same is updated in the database. The list gets updated	As expected	Pass
28.	Agent registration using invalid data	1. Go to site 2. Click on "Don't have an account yet? Register" option 3. Fill the form	First Name = Agent 1 Last Name = NULL Email = agent1@gmail.com Password = 12345678 Confirm password = 12345678	Agent should get an alert saying "Last Name must be at least 1 character long!"	As expected	Pass
29.	Agent registration using invalid data	1. Go to site 2. Click on "Don't have an account yet? Register" option 3. Fill the form	First Name = Agent 1 Last Name = Agent Email = agent1gmail.com Password = 12345678 Confirm password = 12345678	Agent should get an alert saying "Invalid Email"	As expected	Pass
30.	Agent registration using invalid data	1. Go to site 2. Click on "Don't have an account yet? Register" option 3. Fill the form	First Name = Agent 1 Last Name = Agent Email = agent1@gmail.com Password = 123456789 Confirm password = 12345678	Agent should get an alert saying "Passwords do not match!"	As expected	Pass

31.	Agent registration using invalid data	1. Go to site 2. Click on "Don't have an account yet? Register" option 3. Fill the form	First Name = Agent 1 Last Name = Agent Email = agent1@gmail.com Password = 123456789 Confirm password = 12345678	Agent should get an alert saying "Passwords do not match!"	As expected	Pass
32.	Agent registration using invalid data	1. Go to site 2. Click on "Don't have an account yet? Register" option 3. Fill the form	First Name = Agent 1 Last Name = Agent Email = agent1@gmail.com Password = 1234 Confirm password = 1234	Agent should get an alert saying "Passwords must be at least 8 characters long!"	As expected	Pass
33.	Agent registration using valid data	1. Go to site 2. Click on "Don't have an account yet? Register" option 3. Fill the form	First Name = Agent 1 Last Name = Agent Email = agent1@gmail.com Password = 12345678 Confirm password = 12345678	Agent details gets updated in the database. Then an alert "Account created. Login!" is shown	As expected	Pass
34.	Agent login using invalid data	1. Go to site 2. Fill out the login form 3. Enter email and password	Email = agent1@gmail Password = 12345678	Agent should get an alert "Invalid email"	As expected	Pass
35.	Agent login using invalid data	1. Go to site 2. Fill out the login form Enter email and password	Email = agent@gmail.com Password = 12345678	Agent should get an alert "Agent does not exist"	As expected	Pass
36.	Agent login using valid data	1. Go to site 2. Fill out the login form Enter email and password	• Email = agent1@gmail.com Password = 12345678 • Admin did not accept the agent yet	Agent should be redirected to a page, that has the status of the confirmation	As expected	Pass

Along with these test cases, the test cases performed during Sprint 1 were also done.

8.2 User Acceptance testing

Purpose of Document

the purpose of this document is to briefly explain the test coverage and open issues of the [CUSTOMER CARE REGISTRY] project at the time of the release to User Acceptance testing (UA t).

Defect Analysis

this report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity1	Severity2	Severity3	Severity4	Sub total
By Design	10	3	1	2	17
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	40
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	13	12	25	78

test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	1	0	0	1
Outsource Shipping	3	0	0	3
Exception Reporting	8	0	0	8
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULT

9.1 Performance Metrics

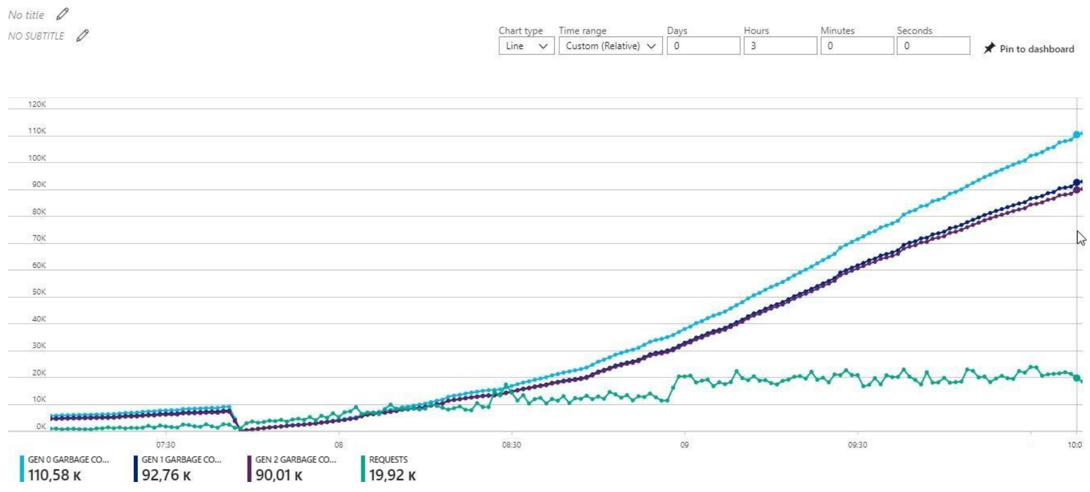
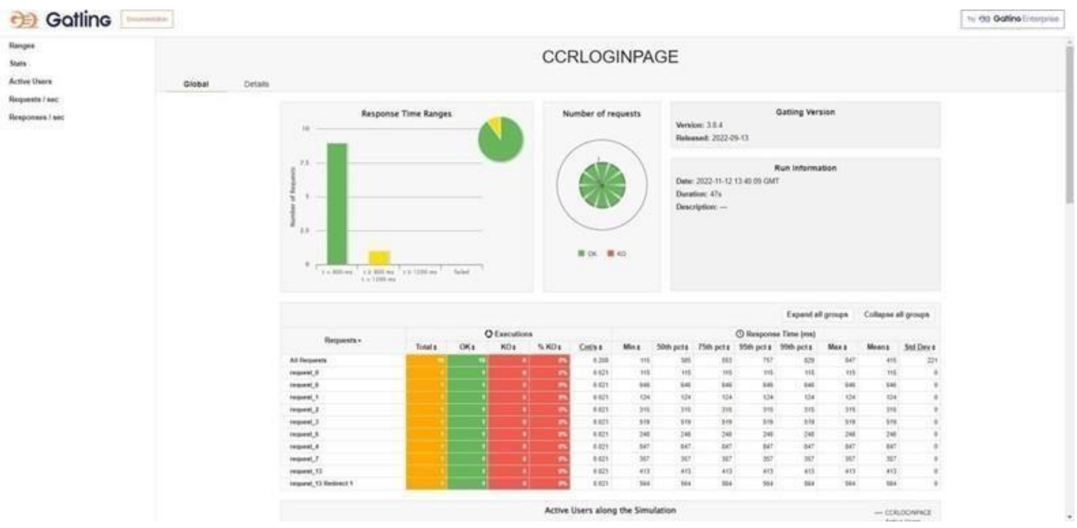


Chart type: Line
Time range: Custom (Relative)
Days: 0
Hours: 3
Minutes: 0
Seconds: 0

GEN 0 GARBAGE CO...	GEN 1 GARBAGE CO...	GEN 2 GARBAGE CO...	REQUESTS
110,58 K	92,76 K	90,01 K	19,92 K



CCRLOGINPAGE

Gatling Version
Version: 3.8.4
Released: 2022-09-13

Run Information
Date: 2022-11-12 13:40:09 GMT
Duration: 47s
Description: —

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
All Requests	10	10	0	0%	0.208	115	385	553	757	829	847	415	221
request_0	1	1	0	0%	0.021	115	115	115	115	115	115	115	0
request_6	1	1	0	0%	0.021	646	646	646	646	646	646	646	0
request_1	1	1	0	0%	0.021	124	124	124	124	124	124	124	0
request_2	1	1	0	0%	0.021	316	316	316	316	316	316	316	0
request_3	1	1	0	0%	0.021	519	519	519	519	519	519	519	0
request_5	1	1	0	0%	0.021	248	248	248	248	248	248	248	0
request_4	1	1	0	0%	0.021	847	847	847	847	847	847	847	0
request_7	1	1	0	0%	0.021	357	357	357	357	357	357	357	0
request_13	1	1	0	0%	0.021	413	413	413	413	413	413	413	0
request_13 Redirect 1	1	1	0	0%	0.021	564	564	564	564	564	564	564	0

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- It retains the customer
- Gets you more references
- Increases profitability
- Gives you and your employees confidence
- Creates a holistic marketing scenario
- Competitive advantage
- Boost Customer Loyalty
- Enhance Brand Reputation
- Improve Products, Services, Procedures and Staff

DISADVANTAGES:

- Higher staff wages from hiring employees who are experts in customer service.
- Paying for staff training
- the extra services offered, such as refreshments
- Higher wage costs from the extra time staff take to provide post-sales service.
- It can be particularly difficult for small businesses to cope with these costs

11. CONCLUSION

In conclusion, customer care, involves the use of basic ethics and any company who wants to have success and grow, needs to remember, that in order to do so, it must begin with establishing a code of ethics in regards to how each employee is to handle the dealing with customers. Customers are at the heart of the company and its growth or decline. Customer care involves, the treatment, care, loyalty, trust the employee should extend to the consumer, as well in life.

11.FUTURE SCOPE

Machine learning (ML), emerging customer service trends 2022 can help businesses in improving overall CX. Chat applications powered by AI are trending. Large companies, as well as startups, are leveraging this to reduce costs and improve service for customers.

Predictive analytics has particularly proved to be very useful. through this, quarries that will result in a call for assistance can be predicted easily. Implementing ML in customer service trends will give you a significant difference in business growth.

12) APPENDIX

Source Code

index.py

```
from flask import Flask, render_template, request, redirect, url_for, session, flash, jsonify
from flask_mysqldb import MySQL
import MySQLdb.cursors
import ibm_db
import re, random, smtplib, os, time, datetime
from flask_mail import Mail, Message


app  =   Flask(__name__)

app.secret_key = '12345'

conn            =            ibm_db.connect("DA    tABASE=bludb;HOS    tNAME=21fecfd8-47b7-4937-
840d-   d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;POR    t=31864;SECURI
tY=SSL;      SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=rtp84701;PWD=DJ4gX1wChd
tCGZPz",",")

mail= Mail(app)

app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_POR t'] = 465
app.config['MAIL_USERNAME']='customercareregistry22@gmail.co
m' app.config['MAIL_PASSWORD'] = 'vxzttcjvdvrqeeve'
app.config['MAIL_USE_   tLS'] = False
app.config['MAIL_USE_SSL'] = true
mail = Mail(app)



@app.route('/', methods =['GE t', 'POS t'])
def index():
  if request.method == 'POS t' and 'email' in request.form:
    email = request.form['email']
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('SELEC t * FROM subscriptions WHERE email = % s', (email, ))
    subscriptions = cursor.fetchone()
    if subscriptions:
      flash(' this Email Is Already Subscribed')
    else:
      ts = time.time()
```

```
        timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('INSER t IN tO subscriptions VALUES (%s, % s, % s)', (None, email,
timestamp, ))
        mysql.connection.commit()
        flash('You have successfully Subscribed')
   return render_template('index.html')



@app.route('/customerlogin', methods =['GE t', 'POS
t'])def customerlogin():
   msgdecline = ''
   if request.method == 'POS t' and 'cemail' in request.form and 'cpassword' in request.form:cemail
      = request.form['cemail']
      cpassword = request.form['cpassword']
      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      cursor.execute('SELEC t * FROM customers_details WHERE customer_email = % s AND
customer_password = % s', (cemail, cpassword, ))
      customers_details = cursor.fetchone()if
      customers_details:
         session['loggedin'] = true
         session['cemail'] = customers_details['customer_email']
         msgsuccess = 'Logged in successfully !'
         return
      redirect(url_for('welcome'))else:
         msgdecline = 'Incorrect Email / Password !'
   return render_template('customerlogin.html', msgdecline = msgdecline)



@app.route('/agentlogin', methods =['GE t', 'POS
t'])def agentlogin():
   msgdecline = ''
   if request.method == 'POS t' and 'aemail' in request.form and 'apassword' in request.form:aemail
      = request.form['aemail']
      apassword = request.form['apassword']
      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      cursor.execute('SELEC t * FROM agent_information WHERE agent_email = % s AND
agent_password = %s', (aemail, apassword,))
      agent_information = cursor.fetchone()
      if agent_information:
         session['loggedin'] = true
         session['aemail'] = agent_information['agent_email']
         msgsuccess = 'Logged in successfully !'
         return redirect(url_for('agentdashboard'))
      else:
        msgdecline = 'Incorrect Email / Password !'
    return render_template('agentlogin.html', msgdecline = msgdecline)



@app.route('/adminlogin', methods =['GE t', 'POS
t'])def adminlogin():
    msgdecline = ''
    if request.method == 'POS t' and 'adminusername' in request.form and 'adminpassword' in
request.form:
      adminusername = request.form['adminusername']
      adminpassword = request.form['adminpassword']
      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      cursor.execute('SELEC t * FROM admin_details WHERE admin_username = % s AND
admin_password = % s', (adminusername, adminpassword, ))
      admin = cursor.fetchone()
      if admin:
        session['loggedin'] = true
        session['adminusername'] = admin['admin_username']
        msgsuccess = 'Logged in successfully !'
        return redirect(url_for('admindashboard'))
      else:
        msgdecline = 'Incorrect Username / Password !'
    return render_template('adminlogin.html', msgdecline = msgdecline)



@app.route('/customerregister', methods =['GE t', 'POS t'])
def customerregister():
    msgdecline = ''
    if request.method == 'POS t' and 'cname' in request.form and 'cemail' in request.form and
'cpassword' in request.form and 'cconfirmpassword' in request.form :
      cname = request.form['cname']
      cemail = request.form['cemail']
      cpassword = request.form['cpassword']
      cconfirmpassword = request.form['cconfirmpassword']
      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      cursor.execute('SELEC t * FROM customers_details WHERE customer_email = % s', (cemail,
      ))user_registration = cursor.fetchone()
      if user_registration:
        msgdecline = 'Account already exists ! try Login'
      elif cpassword != cconfirmpassword:
        msgdecline = 'Password did not match !'
      else:
        ts = time.time()
        timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
        cursor.execute('INSER t IN tO customers_details VALUES (%s, % s, % s, % s, % s)',
        (None,
cname, cemail, cpassword, timestamp, ))
        mysql.connection.commit()
        flash('You have successfully registered ! try Login')
        try:
          mailmsg = Message('Customer Care Registry', sender = 'Registration Successful', recipients
= ['{}', cemail])
          mailmsg.body = "Hello {},\nYou have successfully registered on Customer Care
Registry".format(cname)
          mail.send(mailmsg)
        except:
          pass
        return
  redirect(url_for('customerlogin'))elif
  request.method == 'POS t':
    msgdecline = 'Please fill out the form !'
  return render_template('customerregister.html', msgdecline = msgdecline)



@app.route('/agentregister', methods =['GE t', 'POS
t']) def agentregister():
  if not session.get("adminusername"):
    return
  redirect("/adminlogin")else:
    msgdecline = ''
    if request.method == 'POS t' and 'aname' in request.form and 'aemail' in request.form and
'ausername' in request.form and 'apassword' in request.form and 'aconfirmpassword'  in request.form
:
        aname = request.form['aname']
        aemail = request.form['aemail']
        ausername = request.form['ausername']
        apassword = request.form['apassword']
        aconfirmpassword = request.form['aconfirmpassword']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELEC t * FROM agent_information WHERE agent_email = % s', (aemail, ))
        agent_information = cursor.fetchone()
        if agent_information:
          msgdecline = 'Account already exists ! try
        Login'else:
          ts = time.time()
          timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
          cursor.execute('INSER t IN tO agent_information VALUES (%s, % s, % s, % s, % s, % s)',
          (None,
aname, aemail, ausername, apassword, timestamp,))
          mysql.connection.commit()
          flash('Agent Has been successfully registered !')
```

```
            try:
              mailmsg = Message('Customer Care Registry', sender = 'Registration Successful',
    recipients = ['{}', aemail])
              mailmsg.body = "Hello, You have been Successfully Registered as Agent"
              mail.send(mailmsg)
            except:
              pass
return redirect(url_for('agentlogin'))elif request.method == 'POS t':
          msg = 'Please fill out the form !'
      return render_template('agentregister.html', msgdecline = msgdecline)



    @app.route('/welcome', methods =['GE t', 'POS t'])
    def welcome():
      if not session.get("cemail"):
        return redirect("/customerlogin")
      else:
        msgsuccess = ''msgdecline
        = ''
        cmail = session['cemail']
        mycursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        mycursor.execute('SELEC t * FROM complaint_details WHERE customer_email = %s
        ORDER BY
    timestamp DESC', (cmail,))
        data = mycursor.fetchall()
        mycursor.execute('SELEC t customer_name FROM customers_details WHERE customer_email
    = %s', (cmail,))
        cname = mycursor.fetchone()
        if request.method == 'POS t' and 'name' in request.form and 'email' in request.form and 'category'
    in request.form and 'subject' in request.form and 'description' in request.form :
          name = request.form['name']
          email = request.form['email']
          category = request.form['category']
          subject = request.form['subject']
          description = request.form['description']
          ticketno = random.randint(100000, 999999)
          ts = time.time()
          timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
          cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
          cursor.execute('INSER t IN tO complaint_details VALUES (%s, % s, % s, % s, % s, % s, %
          s, % s,
    % s )', (ticketno, name, email, category, subject, description, timestamp , "pending" , "pending" , ))
          mysql.connection.commit()
          try:
            mailmsg = Message('Customer Care Registry', sender = 'Request Received', recipients =
    ['{}',email])
```

```
        mailmsg.body = "Hello {},\n\n thanks for contacting Customer Care Registry\nWe have
received your complain\nYour   ticket Number: {}\nCategory: {}\nSubject: {}\nDescription:
{}\n\nWe strive to provide excellent service, and will respond to your request as soon as
possible.".format(name, ticketno, category, subject, description)
        mail.send(mailmsg)
      except:
        pass
      flash ('Your complaint is successfully submitted !')
      return redirect(url_for('welcome'))
    elif request.method == 'POS t':
      msgdecline = 'Please fill out the form !'
  return render_template('welcome.html', msgsuccess = msgsuccess, data=data, cname=cname)



@app.route('/agentdashboard', methods =['GE t', 'POS t'])
def agentdashboard():
  if not session.get("aemail"):
    return redirect("/agentlogin")
  else:
    msg = ''
    aemail = session['aemail']
    mycursor1 = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    mycursor1.execute('SELEC t agent_name FROM agent_information WHERE agent_email =
    %s',
(aemail, ))
    agent = mycursor1.fetchone()

    for x in agent:
      agent_name = agent[x]

    mycursor2 = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    mycursor2.execute('SELEC t * FROM complaint_details WHERE agent_name = %s
    ORDER BY
timestamp DESC', (agent_name, ))
    data = mycursor2.fetchall()
    if request.method == 'POS t' and 'status' in request.form :
      status = request.form['status']
      ticketno = request.form['ticketno']
      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      cursor.execute('UPDA tE complaint_details SE t status = %s WHERE ticket_no = %s',
(status,ticketno,) )
      mysql.connection.commit()
      msg = 'Your complaint is successfully solved !'

      mailcursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      mailcursor.execute('SELEC t customer_email FROM complaint_details WHERE ticket_no =
      %s',
(ticketno,) )
```

```
        customer_mail = mailcursor.fetchone()

        for x in customer_mail:
          cemail = customer_mail[x]

        try:
          mailmsg = Message('Customer Care Registry', sender = 'Your ticket Status', recipients = ['{}',
cemail])
          mailmsg.body = "Hello, \nYour complaint has been successfully solved\nYour ticket
Number: {}".format(ticketno)
          mail.send(mailmsg)
        except:
          pass
        return redirect(url_for('agentdashboard'))
      elif request.method == 'POS t':
        msg = 'Please fill out the form !'
    return render_template('agentdashboard.html', msg = msg, data=data, agent_name=agent_name)



@app.route('/admindashboard', methods =['GE t', 'POS t'])
def admindashboard():
  if not session.get("adminusername"):
    return redirect("/adminlogin")
  else:
    msg = ''
    mycursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    mycursor.execute('SELEC t * FROM complaint_details ORDER BY timestamp
    DESC')data = mycursor.fetchall()
    mycursor.execute('SELEC t * FROM
    agent_information')agent = mycursor.fetchall()
    mycursor.execute('SELEC t COUN t(status) AS pending FROM complaint_details WHERE
    status
= %s', ("pending",))
    pending = mycursor.fetchall()
    mycursor.execute('SELEC t COUN t(status) AS assigned FROM complaint_details
WHEREstatus = %s', ("Agent Assigned",))
    assigned = mycursor.fetchall()
    mycursor.execute('SELEC t COUN t(status) AS completed FROM complaint_details WHERE
status = %s', ("Closed",))
    completed = mycursor.fetchall()
    if request.method == 'POS t' and 'agentassign' in request.form :
      agentassign = request.form['agentassign']
      adminusername = request.form['adminusername']
      ticketno = request.form['ticketno']
      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
```

```
        cursor.execute('UPDA tE complaint_details SE t agent_name = %s WHERE ticket_no =
%s',(agentassign, ticketno,) )
        cursor.execute('UPDA tE complaint_details SE t status = %s WHERE ticket_no = %s',
("AgentAssigned", ticketno,) )
        mysql.connection.commit()
        msg = 'Your complaint is Assigned to Agent !'

        mailcursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        mailcursor.execute('SELEC t customer_email FROM complaint_details WHERE ticket_no =
        %s',
(ticketno,) )
        customer_mail = mailcursor.fetchone()

        for x in customer_mail:
          cemail = customer_mail[x]

        try:
          mailmsg = Message('Customer Care Registry', sender = 'Agent Assigned', recipients = ['{}',
cemail])
          mailmsg.body = "Hello,\nWe have received your complaint and agent {} has been
Successfully Assigned\nYour ticket Number: {}\n\nYou will be notified when your complain will
besolved.".format(agentassign, ticketno)
          mail.send(mailmsg)
        except:
          pass
        return redirect(url_for('admindashboard'))
      elif request.method == 'POS t':
        msg = 'Please fill out the form !'
  return render_template('admindashboard.html', msg = msg, data=data, agent=agent,
pending=pending, assigned=assigned, completed=completed)



@app.route('/adminanalytics')
def adminanalytics():
  mycursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
  mycursor.execute('SELEC t COUN t(agent_name) AS Jen tile FROM complaint_details
WHEREagent_name = %s', ("Jen  tile",))
  Jen tile = mycursor.fetchall()
  mycursor.execute('SELEC t COUN t(agent_name) AS AllieGrater FROM complaint_details
WHERE agent_name = %s', ("Allie Grater",))
  AllieGrater = mycursor.fetchall()
  mycursor.execute('SELEC t COUN  t(agent_name) AS RaySin FROM complaint_details
WHERE agent_name = %s', ("Ray Sin",))
  RaySin = mycursor.fetchall()
  mycursor.execute('SELEC t COUN  t(category) AS Category1 FROM complaint_details
WHERE category = %s', ("Product Exchange or Return",))
```

```
  category1 = mycursor.fetchall()
  mycursor.execute('SELEC t COUN  t(categ ory)  AS  Categoíy2  ÏRO  complaint_details  WHERE
                                                  M
category = %s', ("Product Out of Stock",))
  category2 = mycursor.fetchall()
  mycursor.execute('SELEC t COUN                  AS  Categoíy3  ÏRO  complaint_details  WHERE
  t(category)                                                    M
category = %s', ("Payments & transactions",))
  category3 = mycursor.fetchall()                 AS  Categoíy4  ÏRO  complaint_details  WHERE
  mycursor.execute('SELEC  t COUN  t(categ ory)                  M
category = %s', ("Product
  Delivery",))category4 =                         AS  Categoíy5  ÏRO  complaint_details  WHERE
  mycursor.fetchall()
  mycursor.execute('SELEC  t COUN  t(category)
category = %s', ("Other",))
  category5 = mycursor.fetchall()
  print(category1)
  return  render_template('adminanalytics.html',  Jen  tile=Jen  tile,  AllieGrater=AllieGrater,
RaySin=RaySin,  category1=category1,  category2=category2,  category3=category3,
category4=category4, category5=category5)



@app.route('/customerforgotpassword', methods =['GE t', 'POS t'])
def customerforgotpassword():
    msgdecline = ''
    if request.method == 'POS  t' and 'customerforgotemail' in request.form :
      forgotemail = request.form['customerforgotemail']
      cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      cursor.execute('SELEC t * FROM customers_details WHERE customer_email = % s',
(forgotemail, ))
      customers_details = cursor.fetchone()if
      customers_details:
        session['customerforgotemail'] = forgotemail
        otp = random.randint(1000, 9999)
        session['otp'] = otp
        try:
          mailmsg = Message('Customer Care Registry', sender = 'Forgot Password', recipients = ['{}',
forgotemail])
          mailmsg.body = "Hello, \nYour O tP is: {}\nDo not share this O  tP to anyone \nUse this O
tP to reset your password.".format(otp)
          mailmsg.subject = 'Forgot Passowrd'
          mail.send(mailmsg)
          flash('O tP has been sent to your email')
          return redirect(url_for('enterotp'))
        except:
          msgdecline = 'Oops! Something went wrong! Email not sent'else:
        msgdecline = ' this email is not registered!'
    return render_template('customerforgotpassword.html', msgdecline = msgdecline)



@app.route('/agentforgotpassword', methods =['GE t', 'POS t'])
def agentforgotpassword():
    msgdecline = ''
    if request.method == 'POS t' and 'agentforgotemail' in request.form :
        forgotemail = request.form['agentforgotemail']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELEC t * FROM agent_information WHERE agent_email = % s',
        (forgotemail,
))
        agent_information = cursor.fetchone()
        if agent_information:
          session['agentforgotemail'] = forgotemail
          otp = random.randint(1000, 9999)
          session['otp'] = otp
          try:
            mailmsg = Message('Customer Care Registry', sender = 'Forgot Password', recipients = ['{}',
forgotemail])
            mailmsg.body = "Hello, \nYour O tP is: {}\nDo not share this O  tP to anyone \nUse this O
tP to reset your password.".format(otp)
            mail.send(mailmsg)
            flash('O tP has been sent to your email')
            return redirect(url_for('enterotp'))
          except:
            msgdecline = 'Oops! Something went wrong! Email not sent'
        else:
          msgdecline = ' this email is not registered!'
    return render_template('agentforgotpassword.html', msgdecline = msgdecline)



@app.route('/adminforgotpassword', methods =['GE t', 'POS t'])
def adminforgotpassword():
    msgdecline = ''
    if request.method == 'POS t' and 'adminforgotemail' in request.form :
        forgotemail = request.form['adminforgotemail']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELEC t * FROM admin_details WHERE admin_email = % s', (forgotemail, ))
        admin_details = cursor.fetchone()
        if admin_details:
          session['adminforgotemail'] = forgotemail
          otp = random.randint(1000, 9999)
          session['otp'] = otp
          try:
            mailmsg = Message('Customer Care Registry', sender = 'Forgot Password', recipients = ['{}',
forgotemail])
            mailmsg.body = "Hello, \nYour O tP is: {}\nDo not share this O  tP to anyone \nUse this O
tP to reset your password.".format(otp)
            mail.send(mailmsg)
            flash('O tP has been sent to your email')
            return redirect(url_for('enterotp'))
          except:
            msgdecline = 'Oops! Something went wrong! Email not sent'
        else:
          msgdecline = ' this email is not registered!'
      return render_template('adminforgotpassword.html', msgdecline = msgdecline)



@app.route('/enterotp', methods =['GE t', 'POS
t'])def enterotp():
      msgdecline = ''
      if request.method == 'POS  t' and 'otp' in request.form :
        otp =
        int(request.form['otp'])if
        int(session['otp']) == otp:
          msgsuccess = 'success'
          return redirect(url_for('changepassword'))
        else:
          msgdecline = 'You have entered wrong O tP'
      elif request.method == 'POS t':
        msg = 'Please fill out the form !'
      return render_template('enterotp.html', msgdecline = msgdecline)

@app.route('/changepassword', methods =['GE t', 'POS t'])
def changepassword():
      msgdecline = ''
      if request.method == 'POS t' and 'newpassword' in request.form and 'confirmnewpassword' in
request.form:
        newpassword = request.form['newpassword']
        confirmnewpassword = request.form['confirmnewpassword']
        if newpassword == confirmnewpassword:
          cursor =
          mysql.connection.cursor(MySQLdb.cursors.DictCursor)if
          session.get("customerforgotemail"):
            cursor.execute('UPDA tE  customers_details  SE  t  customer_password  =  %s  WHERE
customer_email = %s', (newpassword, session['customerforgotemail'],) )
            mysql.connection.commit()
            flash('Your password changed Successful! try Login')
            return redirect(url_for('customerlogin'))
          elif session.get("agentforgotemail"):
            cursor.execute('UPDA tE   agent_information    SE t    agent_password     =    %s
                                          WHEREagent_email = %s', (newpassword,
session['agentforgotemail'],) )
            mysql.connection.commit()
            flash('Your password changed Successful! try Login')
            return redirect(url_for('agentlogin'))
          elif session.get("adminforgotemail"):
            cursor.execute('UPDA tE admin_details SE t admin_password = %s WHERE admin_email
            =
%s', (newpassword, 'admin@xyz',) )
            mysql.connection.commit()
            flash('Password changed Successful!  try Login')
            return redirect(url_for('adminlogin'))
          else:
            msgdecline = 'Incorrect details'
        else:
          msgdecline = 'Password Did Not Match!'
      elif request.method == 'POS t':
        msgdecline = 'Please fill out the form !'
      return render_template('changepassword.html', msgdecline = msgdecline)



@app.route('/logout')
def logout():
  session.pop('loggedin', None)
  session.pop('cemail', None)
  session.pop('aemail', None)
  session.pop('adminusername', None)
  return redirect(url_for('index'))


@app.route('/offline.html')
def offline():
   return app.send_static_file('offline.html')

@app.route('/service-worker.js')
def sw():
   return app.send_static_file('service-worker.js')

@app.errorhandler(404)
def invalid_route(e):
  return render_template('404.html')



if_name_== ' _main _':
   app.run(host='0.0.0.0', debug = true,port = 8080)
```