CUSTOMER CARE REGISTRY

TEAM ID: PNT2022TMID03025

TEAM MEMBERS:

SWASTHIKPRIYA S (TEAM LEAD)
DEVIDHARSSHINI E
PRIYADHARSINI S
SUKITHA S B

1.INTRODUCTION

1.1 PROJECT OVERVIEW

Customer care and customer service together help create a positive customer experience, or the overall impression a person has when interacting with your company. Both are vital, but there are subtle differences in how they are implemented. High-quality customer care is proactive. The needs of customers throughout the buyer's journey are anticipated, making customers feel supported. That, in turn, helps create an emotional connection between the customer and the company. Customer service is reactive. Here, the focus is on helping customers solve problems or answer questions before purchase, either in a self-serve fashion or via the customer support team. Customer care is more than just providing great customer service. It's a proactive approach to providing information, tools, and services to customers at each point they interact with a brand. If a company neglects customer care, it can negatively impact the customer service experience. For example, when a website chatbot can't provide key information about a product, customers are more likely to get frustrated and reach out to a customer service agent for help. Consumer expectations are extremely high, putting increased pressure on companies to improve their customer relationships. This can lead to lost information when the same person reaches out via multiple channels. When a customer service agent doesn't know the whole story and the customer has to repeatedly share the problem, it leaves both people frustrated. They can register for an account. After the login, they can create a complaint with a description of the problem they are facing. Each user will be assigned an agent. They can view the status of their complaint.

- Customers get the insights they need to make an informed purchase.
- Customer satisfaction can increase and customer loyalty can improve.
- Customer service agents spend less time on routine tasks and answering commonly asked questions, enabling agents to do more meaningful task.

1.2 PURPOSE

There are two sides to customer service objectives. First, there are the goals and KPIs customer service teams attempt to achieve. Then, there's customer service resume objectives. It's important to understand the connection between the two: Writing a strong customer service resume objective starts with understanding the objectives of the field and its depth and possibilities. To provide insight into both levels of customer service objectives. The prime objective of customer service is to answer customer questions quickly and effectively, resolve issues with empathy and care, document pain points to share with internal teams, nurture relationships, and improve brand credibility. Great customer service can make people loyal to your brand, products, and services for years to come.

A strong customer service resume objective underscores your skills and experiences in contributing to customer service's overall goals and objectives. Meeting key customer service KPIs doesn't just involve answering phones and emails. It's a whole world of solutions development, intuition, empathy, brand management, time management-and the soft skills that help connect people and create trust. I guide my team toward giving the best service possible. Sometimes, we're not delivering good news. But the objective is to do that with compassion and empathy and in a way that we give the customer constructive next steps to move forward. We also know that as a newer, younger brand, customers may be wary of our credibility. It usually takes a few consistently excellent customer experiences to feel connected and loyal to the brand. That awesome experience starts from the very first touchpoint, whether it be web, email, brick and mortar, or Instagram, and carries through to when they're wearing our product

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

A strong customer problem statement should provide a detailed description of your customer's current situation. Consider how they feel, the financial and emotional impact of their current situation, and any other important details about their thoughts or feelings.

Customer Satisfaction is an attitude that is decided based on the experience obtained. Satisfaction is an assessment of the characteristics or privileges of a product or service, or the product itself, that provides a level of consumer pleasure with regard to meeting consumer consumption needs.

Customer Satisfaction is the customer's response to the evaluation of perception of differences in initial expectations prior to purchase (or other performance standards) and the actual performance of the product as perceived after wearing or consuming the product in question.

The level of complaint is how high the complaint or delivery of dissatisfaction, discomfort, irritation, and anger over the service of the service or product. The dimension or indicator of complaint level is the high level of complaint.

Product Quality affects Customer Satisfaction, where the dimensions or indicators of Product Quality are quality products, in accordance with the price offered, and ease of use affects the dimensions or indicators of Customer Satisfaction in relation to subscription decisions.

2.2 REFERENCES

1]   Uddi Executive Overview: Enabling Service Oriented Architecture, 2004 Oct.
[2]   "Web Services Architecture", http://www.w3.org/TR/ws-arch/. Date Accessed: 11/02/2004.
[3]   UDDI V3 Specification. http://uddi.org/pubs/uddi-v3.00-published-20020719.html. 19/07/2002.
[4]   Christensen E, Curbera F, Meredith G, Weerawarana S, Web Services Description Language (WSDL) 1.1, W3C Note, 2001
[5]   Ali A S, Rana O F, Ali R A, Walker D W, UDDIe: an extended registry for Web services, SAINT-w'03 Proceedings of the 2003
Symposium on Applications and the Internet Workshops, 2003 Jan, pp .85-89.
[6]   Tsai W T, Paul R, Cao Z, Yu L, Saimi A, Xiao B, Verification of Web Services Using an Enhanced UDDI Server ,Proceedings of The
Eighth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems ,2003 Jan, pp 131-38.
[7]   Liu J, Gu N, Zong Y Ding Z, Zhang S, Zhang Q Service Registration and Discovery in a Domain-Oriented UDDI Registry ,

Proceedings of the 2005 The Fifth International Conference on Computer and Information Technology (CIT'05),2005, pp .276-83.

[8]   Jian W, Zhaohui W, Similarity-based Web Service Matchmaking Proceedings of the 2005 IEEE International Conference on Services

Computing (SCC'05), 2005.

[9]   Tretola G, Zimeo E , Structure Matching for Enhancing UDDI Queries Results , IEEE International Conference on Service-Oriented

Computing and Applications(SOCA'07),2007.

[10]   Ayorak E, Bener a, Super Peer Web Service Discovery Architecture, IEEE Proceedings - International Conference on Data

Engineering, 2007 pp .287-94.

[11]   Libing He W, Wu Y, Jianqun D A novel interoperable model of distributed UDDI - proceedings of the 2008 IEEE International

Conference on Networking, Architecture, and Storage - IEEE NAS 2008 Jun, pp .153-54.

[12]   Nawaz F, Qadir K, Ahmad H F, and SEMREG-Pro: A Semantic based Registry for Proactive Web Service Discovery using Publish

Subscribe Model, Fourth International Conference on Semantics, Knowledge and Grid, IEEE Xplore, 2008 Dec, pp .301-08.

[13]   LIANG1 Q, CHUNG2 J A Federated UDDI System for Concurrent Access to Service Data, IEEE International Conference on e-

Business Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, and DKEEE'08, 2008 Oct, pp .71-

78.

[14]   Vandan T, Nirmal D, InderjeetGarg S, Garg N, Soni P. An Improved Discovery Engine for Efficient and Intelligent discovery of Web

Service with publication facility, SERVICES 2009 - 5th 2009 World Congress on Services, 2009, pp .63-70.

[15]   Ma C, Song M, Xu K, Zhang X.Web Service Discovery Research and Implementation Based on Semantic Search Engine, IEEE 2nd

Symposium on Web Society, Beijing. 2010 Aug 16-17, pp .672-77.

[16]   Rajendran T.Balasubramanie P, Flexible and Intelligent Architecture for Quality Based Web Service Discovery with an Agent- Based

Approach, International Conference on Communication and Computational Intelligence (INCOCCI), Erode. 2010 Dec 27-29, pp .617-

22.

[17]   Johnsen F T, Hafsøe T, Eggen A, Griwodz C, Halvorsen P, Web Services Discovery across Heterogeneous Military Networks, IEEE

Communications Magazine, 2010 Oct, 48(10), pp. 84-90.

[18]   Ourania H, Georgios B, Mara N, Dimosthenis A, "A Specialized Search Engine for Web Service Discovery" IEEE 19th International

Conference on Web Services,USA. 2012 June 24-29, pp. 448-55.

[19]   Raj R J R, Sasipraba T., Web Service Recommendation Framework Using Qos Based Discovery and Ranking Process 3rd

International Conference on Advanced Computing, ICoAC. Chennai. 2011Dec 14-16, pp .371-77.
[20]   Ren X, Hou R, Extend UDDI Using Ontology for Automated Service Composition 2nd International Conferences on Mechanic
Automation and Control Engineering, MACE, 2011 July, pp .298-301.

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 your customers. It can also help you understand a new audience when creating a new product or service.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

A Customer Problem Statement is a detailed description of an issue that needs to be addressed. This document thoroughly elaborates on the problem that your product or your service solves for your particular customers. It takes into consideration your customer's unique pain points and how your product goals about solving their situation. A customer problem statement helps you and your team understand the detailed experience you are attempting to transform by analyzing and empathizing with your customers.

The customer problem statement is a critical component of a project. It benefits everyone involved with the project because it helps people understand why they're working on the project, providing clarity on the reasons behind the product or service. Team members will consider how your customers will be impacted by your project, what their thoughts and needs are, and thus come up with truly effective and valuable ways to improve their experience.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS
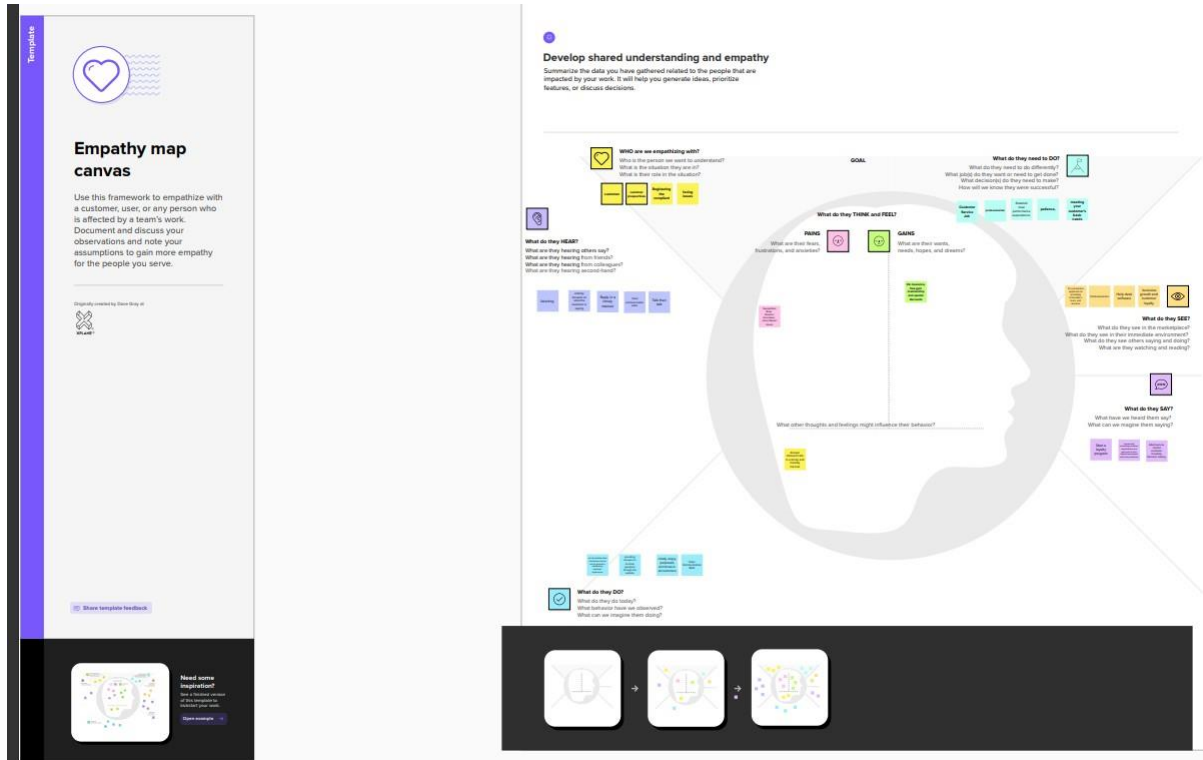
3.2 IDEATION & BRAINSTORMING

3.3 PROPOSED SOLUTION

S.NO.	PARAMETER	DESCRIPTION
01	Problem Statement (Problem to be solved)	To solve customer issues using Cloud Application Development.
02	Idea / Solution description	Assigned Agent routing can be solved by directly routing to the specific agent about theissue using the specific Email. Automated Ticket closure by using daily sync of the daily database. Status Shown to the Customer can display the status of the ticket to the customer. Regular data retrieval in the form of retrieving lost data.
03	Novelty / Uniqueness	Assigned Agent Routing, Automated Ticket Closure, Status Shown to the Customer, and Backup data in case of failures.

S.NO.	PARAMETER	DESCRIPTION
04	Social Impact / Customer Satisfaction	Customer Satisfaction, Customer can track their status and Easy agent communication.
05	Business Model (Revenue Model)	● Key Partners are Third-party applications, agents, and customers. ● Activities held as Customer Service, System Maintenance. ● Key Resources support Engineers, Multi-channel. ● Customer Relationship have 24/7 Email Support, Knowledge-based channel. ● Cost Structure expresses Cloud Platform, Offices

S.NO.	PARAMETER	DESCRIPTION
06	Scalability of the Solution	The real goal of scaling customer service is providing an environment that will allow your customer service specialists to be as efficient as possible. An environment where they will be able to spend less time on gruntwork and more time on actually resolving critical customer issues

3.4 PROBLEM SOLUTION FIT

PROJECT DESIGN PHASE –I (PROBLEM-SOLUTION FIT)

Problem-Solution fit canvas 2.0

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

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.

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.

Problem-Solution Fit

4

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

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

4.2 NON-FUNCTIONAL REQUIREMENT

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

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS
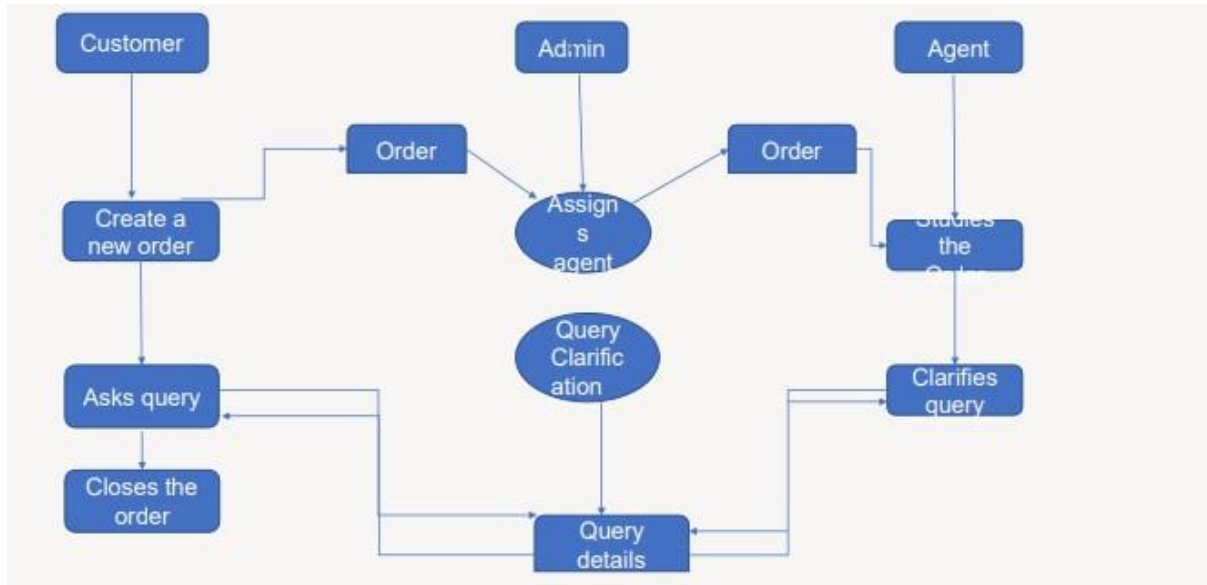
5.2 SOLUTION AND TECHNICAL ARCHITECTURE
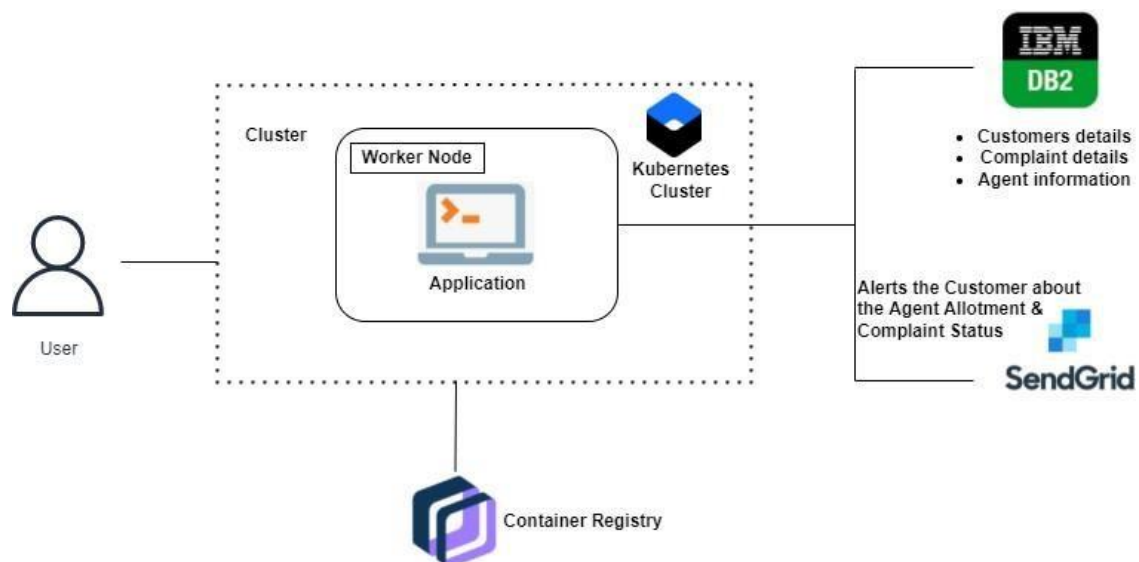
5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	login	USN-2	As a customer, I can login to the application by entering correct email and password.	I can access my account/dashboard.	High	Sprint-1
	Dashboard	USN-3	As a customer, I can see all the orders raised by me.	I get all the info needed in my dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can place my order with the detailed description of my query	I can ask my query	Medium	Sprint-2
	Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	My queries are clarified.	High	Sprint-3
	Forgot password	USN-6	As a customer, I can reset my password by this option incase I forgot my old password.	I get access to my account again	Medium	Sprint-4
	Order details	USN-7	As a Customer, I can see the current stats of order.	I get a better 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 & SCHEDULE

6.1 SPRINT PLANNING &ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and go through the services available on the webpage.	20	Medium	Shyam R Sujindhar C
Sprint-2	Agent Panel	USN-2	The role of the agent is to check out the complaint tickets and to contact the user and solve the complaint they raise.	20	High	Shankar P Jayasurya C
Sprint-3	Admin Panel	USN-3	The role of the admin is to check out the database about the availability and have a track of all the things that the users are going to experience and manage the agent and complaint tickets.	20	High	Shankar P
Sprint-4	Chat Bot	USN-4	The user can directly talk to Chatbot regarding the services. Get the recommendations based on information provided by the user.	20	High	Jayasurya S
Sprint-5	Final Delivery	USN-5	Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application	20	High	Sujindhar C Shyam R Shankar P Jayasurya S

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022
Sprint-2	6	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022
Sprint-3	6	3 Days	07 Nov 2022	09 Nov 2022	6	09 Nov 2022
Sprint-4	5	3 Days	09 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-5	8	6 Days	13 Nov 2022	19 Nov 2022	8	19 Nov 2022

6.3 REPORTS FROM JIRA

🔖	CCR-3 The user will login into the website and go throug...	DONE ⌄	😀
🔖	CCR-4 The role of the agent is to check out the complaint...	DONE ⌄	😀
🔖	CCR-5 The role of the admin is to check out the database...	DONE ⌄	😀
🔖	CCR-6 he user can directly talk to Chatbot regarding the ...	DONE ⌄	😀
🔖	CCR-7 Container of applications using docker kubernetes...	DONE ⌄	😀
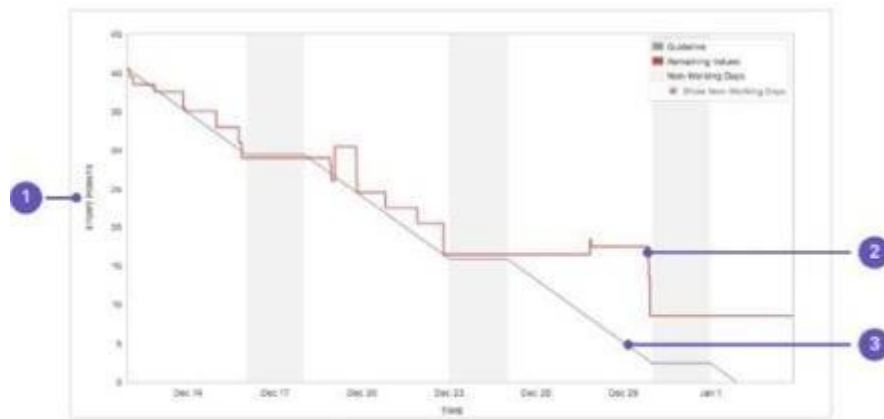
Burndown Graph

7. CODING & SOLUTIONING

7.1 FEATURE 1

Raise you complaint in a minute.

Login

Email

Password

Login

For User registration →

For Admin Login →

Raise you complaint in a minute.

User Name

Email

Password

Confirm Password

Register

For User login →

For Admin Login →

Raise you complaint in a minute.

Admin Login

Email

Password

Login

For User registration →

For User Login →

Complaint   Home   Raise Complaint   Feedback   Logout

Complaint Detail

User Name

First Name

Last Name

Email

Phone Number

Description

Save

Home   Raise Complaint   Feedback

Logout

Feedback

Email

priya@gmail.com

Feedback

Save

Complaint

Dashboard   Complaints   Feedback

Logout

Complaint No	First Name	Last Name	Description	Status	
1	Rajee	M	Kind request to add hot water in canteen	Complete	
1	Priyadharsini	S	My Fridge is not working	Complete	
1	Swasthik	Sivakumar	My lap is not working	Complete	
1	priyadharsini	s	we could not access the website	Pending	Make Complete
1	Vijayakumar	KTM	Complaint about hostel	Complete	
1	Raj	S	sadqfd	Pending	Make Complete

Complaint

Dashboard   Complaints   Feedback

Logout

Complaint Count

7

Feedback Count

2

Feedback No	Name	Email	Description
1	Vijay	mathivijay18@gmail.com	Thanks for feedbacks
1	swasthik	19euit166@skcet.ac.in	app is nice
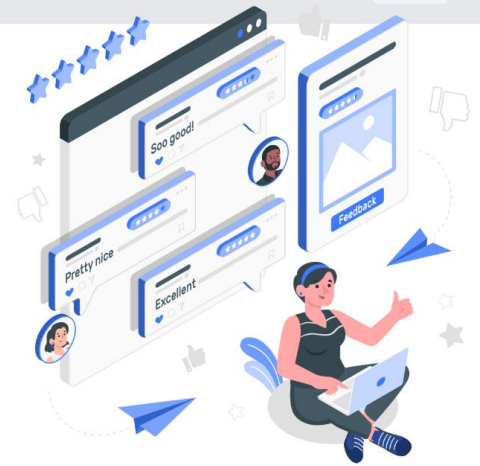
8. TESTING

8.1 TEST CASES

8.1.1 FUNCTIONAL TESTING

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

8.1.2 WHITE BOX TESTING:

Testing based on an analysis of internal workings and structure of a piece of software. This testing can be done sing the percentage value of load and energy. The tester should know what exactly is done in the internal program. Includes techniques such as Branch Testing and Path Testing. Also known as Structural Testing and Glass Box Testing.

8.1.3 BLACK BOX TESTING:

Testing without knowledge of the internal workings of the item being tested. Tests are usually functional. This testing can be done by the user who has no knowledge of how the shortest path is found.

8.2 USER ACCEPTANCE TESTING

Acceptance testing can be defined in many ways, but a simple definition is the succeeds when the software functions in a manner that can be reasonable expected by the customer. After the acceptance test has been conducted, one of the two possible conditions exists. This is to fine whether the inputs are accepted by the database or other validations. For example accept only numbers in the numeric field, date format data in the date field. Also the null check for the not null fields. If any error occurs then show the error messages. The function of performance characteristics to specification and is accepted. A deviation from specification is uncovered and a deficiency list is created. User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

8.3 TEST RESULTS

|	A	B	C	D	E	F	G	H	I
	Sprint 1 UI/UX								
	Testcase	Type	Component	Scenario	Step to execute	Expected result	Actual result	Status	Executed by
1	UI		login/signup page	clicking on site link	click in site link	login/signup page loads	page load	PASS	Shyam
2	Functional		login/signup page	login in to user acco	enter credentials	login to home page	home page loads	PASS	
3	Functional		login/signup page	signup a user	enter user details	login to home page	home page loads	PASS	
4	Functional		home page	logout of the home p	logout the user	back to login page	login page loads	PASS	
5	Functional		login/signup page	login with unregister	redirect to signup pa	back to signup page	signup page loads	PASS	
6	Functional		login/signup page	signup a existing em	use a existing user e	back to sign up page	signup page loads	PASS	
7	Functional		wrong password	sign in with wrong pa	login with wrong pas	back to sign up page	signup page loads	PASS	
8	Functional		wrong email	signin with wring em	login with wrong em	back to signup page	signup page loads	PASS	
	Spirnt 2 db2								
1	Functional		complaint page	display registerd cor	click on complaint ta	complaint list down	complaint list dow	PASS	
2	Functional		complaint page	clicking on solve	click on solve button	solve the complaint	complete the com	PASS	
3	Functional		complaint page	click on the dismiss	click dismiss button	delete the complaint	deletes the compl	PASS	
4	Functional		complaint page	fill up the complaint 1	click submit	create the complaint	create a new com	PASS	

9. RESULTS

9.1 PERFORMANCE METRICES

Login Page

Admin Page

Complaint page

Response Time Ranges

Number of requests

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

Run Information
Date: 2022-11-13 05:09:19 GMT
Duration: 40s
Description: —

Number of Requests (y-axis: 0, 5, 10, 15, 20)
t < 800 ms | t ≥ 800 ms t < 1200 ms | t ≥ 1200 ms | failed

■ OK ■ KO

Expand all groups   Collapse all groups

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	16	16	0	0%	0.38	117	480	608	880	1027	1059	528	262
request_0	1	1	0	0%	0.024	360	360	360	360	360	360	360	0
request_6	1	1	0	0%	0.024	1000	1000	1000	1000	1000	1000	1000	0
request_2	1	1	0	0%	0.024	570	570	570	570	570	570	570	0
request_3	1	1	0	0%	0.024	448	448	448	448	448	448	448	0
request_4	1	1	0	0%	0.024	454	454	454	454	454	454	454	0
request_5	1	1	0	0%	0.024	607	607	607	607	607	607	607	0
request_1	1	1	0	0%	0.024	790	790	790	790	790	790	790	0
request_7	1	1	0	0%	0.024	405	405	405	405	405	405	405	0
request_13	1	1	0	0%	0.024	482	482	482	482	482	482	482	0
request_13 Redirect 1	1	1	0	0%	0.024	812	812	812	812	812	812	812	0
request_15	1	1	0	0%	0.024	411	411	411	411	411	411	411	0
request_28	1	1	0	0%	0.024	607	607	607	607	607	607	607	0
request_28 Redirect 1	1	1	0	0%	0.024	408	408	408	408	408	408	408	0
request_31	1	1	0	0%	0.024	922	922	922	922	922	922	922	0
request_42	1	1	0	0%	0.024	606	606	606	606	606	606	606	0
request_41	1	1	0	0%	0.024	117	117	117	117	117	117	117	0

Complaint form

Response Time Ranges

Number of requests

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

Run Information
Date: 2022-11-13 05:14:30 GMT
Duration: 1m 6s
Description: —

Number of Requests (y-axis: 0, 2.5, 5, 7.5, 10, 12.5)
t < 800 ms | t ≥ 800 ms t < 1200 ms | t ≥ 1200 ms | failed

■ OK ■ KO

Expand all groups   Collapse all groups

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	17	17	0	0%	0.254	270	828	1092	1121	1128	1127	779	276
request_0	1	1	0	0%	0.015	270	270	270	270	270	270	270	0
request_6	1	1	0	0%	0.015	1109	1109	1109	1109	1109	1109	1109	0
request_1	1	1	0	0%	0.015	827	827	827	827	827	827	827	0
request_2	1	1	0	0%	0.015	828	828	828	828	828	828	828	0
request_3	1	1	0	0%	0%	0.015	829	829	829	829	829	829	829	0
request_5	1	1	0	0%	0.015	831	831	831	831	831	831	831	0
request_4	1	1	0	0%	0.015	844	844	844	844	844	844	844	0
request_7	1	1	0	0%	0.015	407	407	407	407	407	407	407	0
request_13	1	1	0	0%	0.015	403	403	403	403	403	403	403	0
request_13 Redirect 1	1	1	0	0%	0.015	1127	1127	1127	1127	1127	1127	1127	0
request_18	1	1	0	0%	0.015	1120	1120	1120	1120	1120	1120	1120	0
request_19	1	1	0	0%	0.015	610	610	610	610	610	610	610	0
request_25	1	1	0	0%	0.015	593	593	593	593	593	593	593	0
request_25 Redirect 1	1	1	0	0%	0.015	1082	1082	1082	1082	1082	1082	1082	0
request_30	1	1	0	0%	0.015	1081	1081	1081	1081	1081	1081	1081	0
request_37	1	1	0	0%	0.015	604	604	604	604	604	604	604	0
request_35	1	1	0	0%	0.015	547	547	547	547	547	547	547	0

10. ADVANTAGES &DISADVANTAGES

Advantage

- Flow sheet 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

- Creating loyal customers through good customer service can provide businesses with lucrative long-term relationships.

- Customer loyalty. Loyal customers have many benefits for businesses

Disadvantage

- 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.
- Experience burnout and stress. Working as a customer service representative requires you to maintain a friendly demeanour at all times, regardless of how customers act or how you personally feel

11. CONCLUSION

Companies today are modernizing customer care, using advanced AI to ensure a positive customer experience starting from the first interaction and throughout the buyer's journey. To properly manage customer care, companies must understand how they are succeeding and what needs improvement. This requires establishing key performance indicators (KPIs) for customer service and creating a system of gathering metrics across channels. In conclusion, customer care, involves the use of basic ethics and any company who wants to have success and grow, needs to remember, that in order to do so, it must begin with establishing a code of ethics in regards to how each employee is to handle the dealing with customers. Customers are at the heart of the company and its growth or decline. Customer care involves, the treatment, care, loyalty, trust the employee should extend to the consumer, as well in life. This concept can be applied to so much more than just customer care. People need to treat others with respect and kindness; people should try to take others into consideration when making any decision. If more people were to practice this policy, chances are the world would be a better, more understanding place for all to exist. Thereby, the customer care registry would be far helpful and approachable. It offers easy tracking, recording and notification than any other means.

12. FUTURE SCOPE

The current state of customer care registery, in so many companies, looks something like this:

- Customer acquisition is prioritised over retention

- Customer service investment projects are sidelined.

- Departmental efficiency is of highest priority.

- Businesses see employees in the customer service department as short-term and disposable. They are there to fulfil a specific, repetitive, purpose.

- Employees are considered unskilled and leaders hire accordingly.

- New agents view customer service as a 'last resort' or 'short term' job. People often see careers in customer support as unambitious.

- Agent training rarely goes beyond product and people skills.

In the next 3-5 years, we expect to see these **future customer care registry trends**:

- The shift from a primarily 'cost centre' to primarily 'growth centre' worldview.

- The job desk for a customer care registry 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 care registry 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.

Modern and developing technology enables this future to exist. With new technology, administrative tasks will tend toward zero.

- The sole purpose of the customer service is to meet the expectations of the customers so that they are satisfied with the outcome. These services are also available to understand the queries of the customers and ensure that they enjoy a cost-effective experience after purchasing any product from the respective company.

13. APPENDIX

SOURCE CODE

```python
from flask import Flask,render_template,request,url_for,session,redirect

from flask_mysqldb import MySQL

from sendmail import sendemail,forget_password_mail,updated_password_mail,solve_mail

import json

import ibm_db

import re

from random import randint

from datetime import date


app = Flask(__name__)
# http://remotemysql.com/


# dsn_hostname = "b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud"

# dsn_uid = "dmt13873"

# dsn_pwd = "740yZ1Yq8Uj2E4qm"

# dsn_database = 'bludb'

# dsn_port = 31249

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=SSL;SSLServerCertificate=src/DigiCertGlobalRootCA.crt;UID=dmt13873;PWD=740yZ1Yq8Uj2E4qm","","")  # type: ignore

print(conn)

print("connection successful...")
```

```python
# database configuration

# app.config['MYSQL_HOST'] = 'sql12.freesqldatabase.com'

# app.config['MYSQL_USER'] = 'sql12552843'

# app.config['MYSQL_PASSWORD'] = 'zWIzHmXNi8'

# app.config['MYSQL_DB'] = 'sql12552843'

app.secret_key = "super secret key"

# mysql = MySQL(app)


@app.route('/')
def home():
    today = date.today()
    current_date = today.strftime('%d/%m/%Y')
    if "google_token" in session:
        session["current_date"] = current_date
        return render_template('home.html')
    if "username" in session:
        session["current_date"] = current_date
        return render_template('home.html')
    return render_template('index.html')


# manually registration
@app.route('/register',methods=["POST"])
def register():
    if request.method == 'POST':
        name = request.form['uname']
```

```python
mail = request.form['mail']

pwd = request.form['pwd']

cpwd = request.form['confirmpwd']

if not re.match(r'[^@]+@[^@]+\.[^@]+', mail):

    msg = 'Invalid email address !'

    return render_template('index.html',signupmsg=msg)

if pwd != cpwd:

    msg = 'Please enter correct confirm password'

    return render_template('index.html',signupmsg=msg)

# check account is exists or not

# cursor = mysql.connection.cursor()

rCheckQuery = ''

result = ibm_db.exec_immediate(conn,f"SELECT * FROM customerdeatils
WHERE email LIKE '{mail}'")

# cursor.execute('SELECT * FROM customerdeatils WHERE email LIKE
% s',[mail])

# existing_user = cursor.fetchone()

# cursor.close()

existing_user = ibm_db.fetch_row(result)

 #exits

if existing_user:

    msg = 'Account already exists please login.'

    return render_template('index.html',signupmsg = msg)

# not exists


# cursor = mysql.connection.cursor()
```

```python
    # cursor.execute('INSERT INTO customerdeatils VALUES(null,% s,% s,% s)',(name,mail,pwd))

    # mysql.connection.commit()

    # cursor.close()

    regInsertQuery = f"INSERT INTO customerdeatils (username,email,passwrd) VALUES('{name}','{mail}','{pwd}')"

    insertflag = ibm_db.exec_immediate(conn,regInsertQuery)

    msg = 'Your registration successfully completed.'

    # send mail

    sendemail(mail,'Account_creation')


    return render_template('index.html',signupmsg = msg)
# admin page
@app.route('/admin/<which>')
def admin(which):
    if which == 'customers':

        # cursor = mysql.connection.cursor()

        result = ibm_db.exec_immediate(conn,'SELECT * FROM customerdeatils')

        data = []

        while ibm_db.fetch_row(result):

            temp = [ibm_db.result(result,0),ibm_db.result(result,1),ibm_db.result(result,2),ibm_db.result(result,3)]

            data.append(temp)

        return render_template('admin.html',customers=data,complaints=None)

    if which == 'complaints':

        # cursor = mysql.connection.cursor()
```

```python
    result = ibm_db.exec_immediate(conn,'SELECT * FROM complaints')

    data = []

    while ibm_db.fetch_row(result):

        temp =
[ibm_db.result(result,0),ibm_db.result(result,1),ibm_db.result(result,2),ibm_db.re
sult(result,3),ibm_db.result(result,4),ibm_db.result(result,5)]

        data.append(temp)

    return render_template('admin.html',customers=None,complaints=data)
# admin delete

@app.route('/Delete/<type>/<id>')

def Delete(type,id):

    if type == 'customers':

        # cursor = mysql.connection.cursor()

        result = ibm_db.exec(conn,f'DELETE FROM customerdeatils WHERE id =
"{id}"')

        # mysql.connection.commit()

        # cursor.close()

        return redirect(url_for('admin',which='customers'))

    if type == 'complaints':

        # cursor = mysql.connection.cursor()

        result = ibm_db.exec_immediate(conn,f'DELETE FROM complaints
WHERE id = {id}')

        # mysql.connection.commit()

        # cursor.close()

        return redirect(url_for('admin',which='complaints'))
# manually login

@app.route('/login',methods=['POST','GET'])
```

```python
def login():
    if request.method == 'POST':
        mail = request.form['mail1']
        password = request.form['pwd1']
        # login is admin or not
        if mail == "admin" and password == 'admin@1810':
            return redirect(url_for('admin',which='customers'))
        # check account is exists or not
        # cursor = mysql.connection.cursor()
        query = "SELECT * FROM customerdeatils WHERE email=? AND passwrd=?"
        stmt = ibm_db.prepare(conn, query) # type:ignore
        ibm_db.bind_param(stmt,1,mail) # type:ignore
        ibm_db.bind_param(stmt,2,password) # type:ignore
        ibm_db.execute(stmt) # type:ignore
        user = ibm_db.fetch_assoc(stmt) # type:ignore
        print(user,password)
        #exists
        if user:
            session["username"] = user['USERNAME']
            session['mail'] = mail
            return render_template('home.html',username=session["username"],mail=session["mail"])
        else:
            msg = 'mail or password is not valid.'
            return render_template('index.html',signinmsg=msg)
```

```python
    if request.method == "GET":

        return redirect(url_for('home'))


# logout method
@app.route('/logout')
def logout():
    if "username" in session:

        session.pop("username")

    if "google_token" in session:

        session.pop("google_token")

        session.pop("mail")

    if "mail" in session:

        session.pop("mail")

    return redirect(url_for('home'))


# complaint register
@app.route('/complaint',methods=['POST'])
def complaint():
    if request.method == 'POST':

        complaint_name = request.form['complaint_name']

        name = request.form['name']

        mail = request.form['email']

        against_person = request.form['against_person']

        date = request.form["date"]

        des = request.form['complaint_des']

        # cursor = mysql.connection.cursor()
```

```python
    if not name == session["username"] or not mail == session["mail"]:

        msg = "please don't change username and mail."

        return render_template('home.html',msg=msg)

    result = ibm_db.exec_immediate(conn,f"INSERT INTO complaints
(username,email,against_person,des,date,solved)
VALUES('{name}','{mail}','{against_person}','{des}','{date}','{'0'}')")

    # mysql.connection.commit()

    # cursor.close()

    sendemail(mail,'complaint_creation')

    msg = 'Complaint registerd you check out complaints section.'

    return render_template('home.html',msg=msg)


# show complaints and progress
@app.route('/showcomplaints')
def showcomplaints():
    # cursor = mysql.connection.cursor()

    # cursor.execute("SELECT * FROM complaints WHERE username= % s
AND email=% s",(session["username"],session["mail"]))

    # details = cursor.fetchall()

    # cursor.close()

    query = "SELECT * FROM complaints WHERE username=? AND email=?"

    stmt = ibm_db.prepare(conn, query) # type:ignore

    ibm_db.bind_param(stmt,1,session["username"]) # type:ignore

    ibm_db.bind_param(stmt,2,session['mail']) # type:ignore

    ibm_db.execute(stmt)

    data = []

    while ibm_db.fetch_row(stmt):
```

```python
        temp =
[ibm_db.result(stmt,0),ibm_db.result(stmt,1),ibm_db.result(stmt,2),ibm_db.result
(stmt,3),ibm_db.result(stmt,4),ibm_db.result(stmt,5),ibm_db.result(stmt,6)]

        print(temp)

        data.append(temp)

    return render_template('complaints.html',complaints=data)


# update complaint

@app.route('/solve',methods=["POST"])

def solve_complaint():

    if request.method == "POST":

        c_id = request.form['c_id']


        print(c_id)

        # cursor = mysql.connection.cursor()

        # cursor.execute("UPDATE complaints SET solved = % s WHERE id = %
s",('1',c_id,))

        query = "UPDATE complaints SET solved = '1' WHERE id = ?"

        # mysql.connection.commit()

        stmt = ibm_db.prepare(conn, query) # type:ignore

        ibm_db.bind_param(stmt,1,c_id) # type:ignore

        ibm_db.execute(stmt)

        detail = ibm_db.result(stmt,0)

        print(detail)

        # cursor.execute("SELECT * FROM complaints WHERE id = % s",[c_id])

        query2 = "SELECT * FROM complaints WHERE id = ?"

        stmt1 = ibm_db.prepare(conn, query2) # type:ignore
```

```python
        ibm_db.bind_param(stmt1,1,c_id) # type:ignore

        ibm_db.execute(stmt1)

        details = ibm_db.result(stmt1,0)

        # cursor.close()

        print(details)

        # solve_mail(session['mail'],'user')

        return redirect(url_for('showcomplaints'))

    return redirect(url_for('showcomplaints'))


# # admin agent allot

# @app.route('/solve_admin',methods=["POST"])

# def solve_admin():

#    if request.method == "POST":

#        c_id = request.form['c_id']

#        # cursor = mysql.connection.cursor()

#        cursor.execute("SELECT * FROM complaints WHERE id = % s",[c_id])

#        query = "SELECT * FROM complaints WHERE id = ?"

#        details = cursor.fetchone()

#        cursor.close()

#        solve_mail(details[3],'admin')

#        return redirect(url_for('admin',which='complaints'))

#    return redirect(url_for('admin',which='complaints'))

# remove complaint

@app.route('/dismiss',methods=["POST"])

def dismiss_complaint():

    if request.method == "POST":
```

```python
        c_id = request.form["c_id"]

        # cursor = mysql.connection.cursor()

        # cursor.execute("DELETE FROM complaints WHERE id = % s",[c_id])

        # mysql.connection.commit()

        # cursor.close()

        query = "DELETE FROM complaints WHERE id = ?"

        stmt = ibm_db.prepare(conn, query)

        ibm_db.bind_param(stmt,1,c_id) # type:ignore

        ibm_db.execute(stmt)

        return redirect(url_for('showcomplaints'))

    return redirect(url_for('showcomplaints'))
# send otp in user mail id
@app.route('/send_otp',methods=["POST","GET"])
def send_otp():
    if request.method == "POST":

        mail = request.form["mail"]

        cursor = mysql.connection.cursor()

        cursor.execute("SELECT * FROM customerdeatils WHERE email = % s",[mail])

        temp = cursor.fetchone()

        cursor.close()

        if not temp:

            return render_template('forget.html',type='otp',msg1='Your account
doesn\'t exist please register')

        otp = randint(10 ** 5,10**6)

        forget_password_mail(mail,otp)

        session["otp"] = otp
```

```python
        return render_template('forget.html',type='update_password',tempmail=mail)
# forget password method
@app.route('/forgetpassword/<type>',methods=["POST","GET"])
def forgetpassword(type):
    if type == 'otp':
        return render_template('forget.html',type=type)
    if request.method == "POST":
        mail = request.form["mail"]
        otp = request.form["otp"]
        pwd = request.form["password"]
        c_pwd = request.form["con_pwd"]
        print(otp,session['otp'])
        if not pwd == c_pwd:
            msg = 'Please Enter Password properly'
            return render_template('forget.html',type='updatePassword',msg=msg)
        if not otp == str(session['otp']):
            msg = "Your OTP is Incorrect."
            return render_template('forget.html',type='updatePassword',msg=msg)
        cursor = mysql.connection.cursor()
        cursor.execute("UPDATE customerdeatils SET passwrd = % s WHERE email = % s",(pwd,mail))
        mysql.connection.commit()
        cursor.close()
        msg = 'password updated successfully'
        updated_password_mail(mail)
        return render_template('forget.html',type='updatePassword',msg=msg)
```

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