PROJECT REPORT

TITLE	**CUSTOMER CARE REGISTRY**
TEAM ID	PNT2022TMID33777

TEAM LEADER	**JESINTHA A**
TEAM MEMBER 1	**BISMI MAJIDHA S**
TEAM MEMBER 2	**MELCIYA JAFFRIN A**
TEAM MEMBER 3	**MOHAMED ARSHAD M**

TABLE OF CONTENT

- **PROJECT DESIGN**
  - Data Flow Diagrams
  - Solution & Technical Architecture
  - User Stories
- **PROJECT PLANNING & SCHEDULING**
  - Sprint Planning & Estimation
  - Sprint Delivery Schedule
  - Reports from JIRA
- **CODING & SOLUTION (Explain the features added in the project along with code)**
  - Feature 1
  - Feature 2
  - Database Schema (if Applicable)
- **TESTING**
  - Test Cases
  - User Acceptance Testing
- **RESULTS**
  - Performance Metrics
- **ADVANTAGES & DISADVANTAGES**
- **CONCLUSION**
- **FUTURE SCOPE**
- **APPENDIX**

Source Code
GitHub & Project Demo Link

CHAPTER 1

INTRODUCTION

• PROJECT OVERVIEW

Customer care is more than just providing great customer service. It's proactive approach to providing information, tools, and services to customers at each point they interact with a brand. This Application has been developed to help the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to a customer they will be notified with an email alert. Customers can view the status of the ticket till the service is provided. The main role and responsibility of the admin are to take care of the

whole process. Starting from Admin login followed by the agent creation and assigning the customer's complaints. Finally, He will be able to track the work assigned to the agent and a notification will be sent to the customer. User can register for an account. After the login, they can create the complaint with a description of the problem they are facing. Each user will be assigned with an agent. They can view the status of their complaint.

- **PURPOSE**

This Application has been developed to help the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to a customer they will be notified with an email alert. Customers can view the status of the ticket till the service is provided. When customers are happy with the service they receive, they are more likely to trust and be loyal to that company. Good customer service creates a positive experience for customers, which can result in repeat business and referrals. Good customer service is the lifeblood of any business. You can offer promotions and slash prices to bring in as many new customers as you want, but unless you can get some of those customers to come back, your business won't be profitable for long. Good customer service is all about bringing customers back. Good customer service makes it easy for customers to do business with you. When customers have a positive experience with your company, they are more likely to come back and do business with you again. Good customer service also makes it easy for customers to recommend yourcompany to their friends and family.

CHAPTER 2

LITERATURE SURVEY

1. EXISTING PROBLEM

- **TITLE-** CUSTOMER CARE REGISTRY **DESCRIPTION**

Previous research or relevant research is very important in a scientific research or article. Previous research or relevant research serves to strengthen the theory and influence of relationships or influences between variables. Article in the review customer satisfaction determination and complaint level: Product Quality and Service Quality Analysis, A Study of Marketing Management Literature. The purpose of writing this article is to build a hypothesis of influence between variables to be used in future research. The result of this research library is that: 1) Product Quality affects Customer Satisfaction; 2) Service Quality affects Customer Satisfaction; 3) Product Quality affects complaint level; 4) Service Quality affects complaint level; and 5) Customer Satisfaction affects complaint level.

- **TITLE-** CUSTOMER CARE REGISTRY

DESCRIPTION

Customer satisfaction is decisive for construction field and firms relying on customer's relationship. Measuring the customer satisfaction has several benefits such as for improving communication between parties, evaluation of progress towards goals and enabling of mutual agreement and monitoring results. This paper focuses on analyzing the satisfaction factors of customers including all aspects of products and services in the construction projects. In this study factors for customer satisfaction in construction industry are taken from the past literature review. The literature reviews are summarized and various factors related to customer satisfaction in construction industry based on literature review summary.

- **TITLE-** CUSTOMER CARE REGISTRY

DESCRIPTION

Customer Satisfaction Study Of The Mumbai Metro Service". In this study they

investigated about the service quality of the metro service based on the performance leading to customer satisfaction. The survey was conducted and analyzed with SPSS tool. This survey is based

on Gap 5 SERVQUAL model and identified the level of satisfaction with their parameter

2.1.4 TITLE- CUSTOMER CARE REGISTRY

DESCRIPTION

"Study of Factors Affecting Customer Satisfaction for Residential Flats in Surat and Ahmedabad city in Gujarat Region of India". In this paper, factors affecting the customer satisfaction among the residential flats are analyzed in the region. They find the satisfaction and dissatisfaction factors from flat owners .They find out the factor for customer service satisfaction and dissatisfaction factor such as Builder reputation, Materials & Method Used In Construction, Location Of The Building, Aesthetic Appearance Of The Building, Security Provisions, Fire Safety and Protection, Size and space of rooms, Drawing Or Living Room, Bathroom, Area Calculation, Ventilation, Water supply, Parking, Recreational Facilities and Interiors of building

- **REFERENCES**

- Lucas, Robert (2015). Customer Service Skills For Success. New York: McGraw-Hill. ISBN HYPERLINK "https://en.wikipedia.org/wiki/ISBN_(identifier)" 978-0-07-354546-2.

1) Buchanan, Leigh (1 March 2011). " HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"A HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html" HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"C HYPERLINK

"http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"u HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"s HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"t HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"o HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"m HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"e HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"r S HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"e HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"r HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"v HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"i HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"c HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"e M HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"a HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"k HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"e HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"o HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"v HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"e HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"r HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html"" HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html". HYPERLINK "http://www.inc.com/magazine/20110301/a-customer-service-makeover_pagen_2.html" HYPERLINK

"https://en.wikipedia.org/wiki/Inc._(magazine)"I HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)"n HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)"c HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)". m HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)"a HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)"g HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)"a HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)"z HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)"i HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)"ne HYPERLINK
"https://en.wikipedia.org/wiki/Inc._(magazine)".Retrieved 29 Oct 2012.

1) Teresa Swartz, Dawn Iacobucci. Handbook of Services Marketing and Management. Thousand
Oaks, CA: Sage

1) Bordoloi, Sanjeev (2019). Service Management Operations, Strategy, Information Technology.New York: McGraw-Hill. ISBN HYPERLINK "https://en.wikipedia.org/wiki/ISBN_(identifier)" 978-1-260-09242-4.

2. PROBLEM STATEMENT DEFINITION

An effective complaints management system is integral to providing quality customer service. It helps to measure customer satisfaction and is auseful source of information and feedback for improving services . Often customers are the first to identify when things are not working properly .Customer can track their state of the issue in real time.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

- **EMPATHY MAP CANVAS**

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Creatingan effective solution requires understanding the true problem and the person who is experiencing it.

The exercise of creating the map helps participants consider things from the user'sperspective along with his or her goals and challenges.

Figure 3.1.1 Empathy Map Canvas

- **IDEATION AND BRAINSTORMING**

Brainstorm & Idea Prioritization Template: Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Step-1: Team Gathering, Collaboration and Select the Problem Statement

A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a groupactivity.

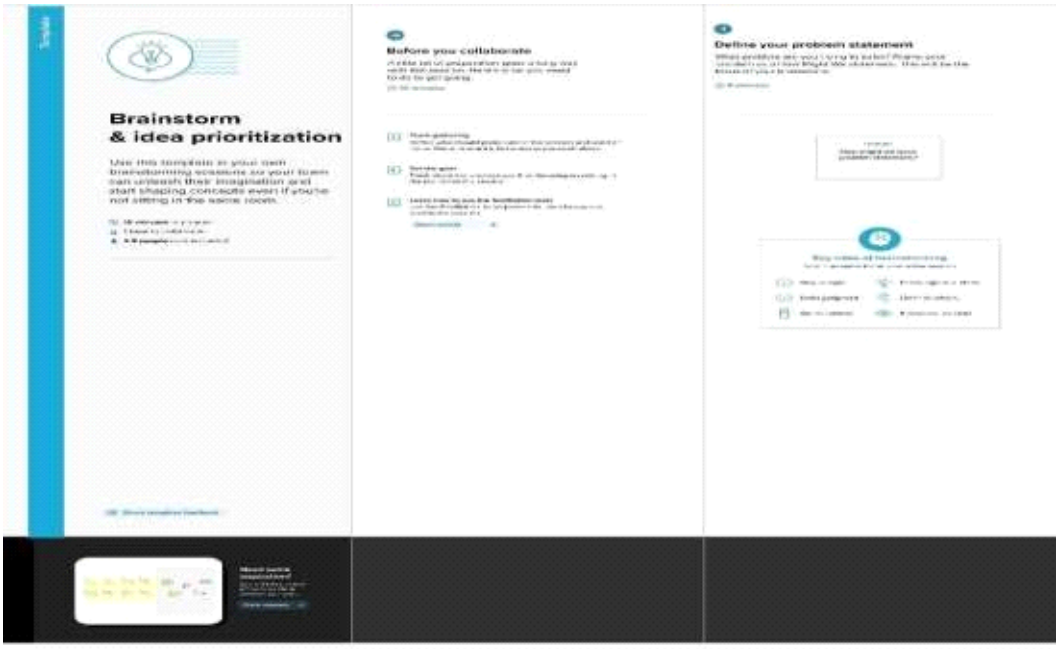


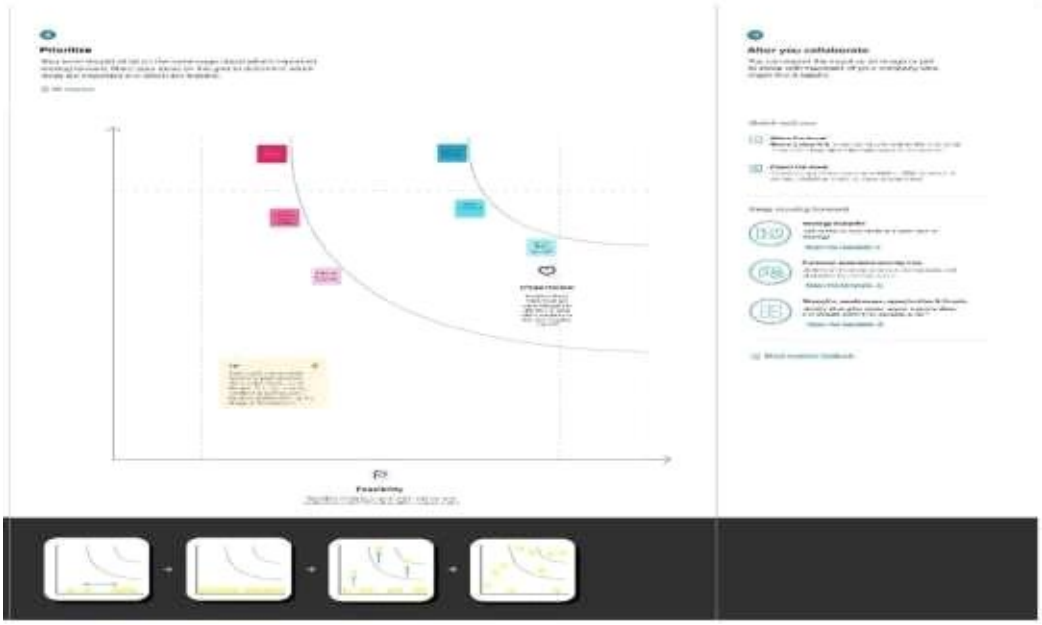
Figure 3.2.1 Team Gathering, Collaboration and Select the Problem Statement

Step-2: Brainstorm, Idea Listing and Grouping

The idea listing and grouping is used to organize and analyse large numbers of ideas by categorizing them. By organizing and reorganizing ideas, students gain a better appreciation of, and dialogue about, their ideas. As students create idea clusters, new contexts and connections amongthemes emerge.

Figure 3.2.2 Brainstorm, Idea Listing and Grouping

Step-3: Idea Prioritization



Idea prioritization is just a part of the idea management process. Having a structured idea management process and a systematic way of gathering, evaluating and prioritizing new ideas takes time. To make it work, the entire idea management process should be integrated to the everyday ways of working.

Figure 3.2.1 Idea Prioritization

- **PROPOSED SOLUTION**

S. NO	PARAMETER	DESCRIPTION
01	Problem Statement (Problem to be solved)	Companies today are modernizing customer care, using advanced methodologies to ensure a positive customer experience starting from the first interaction and throughout the buyer's journey. A 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. Hence ,an application is needed for processing the complaints raised by the customers.

02	Idea / Solution description	A customer care Registry not only boosts customer satisfaction but also helps in improving customer loyalty. If a company neglects customer care, it can negatively impact the customer service experience. Hence, an application needs to be developed to help the customer in processing their complaints where the customers will be able to raise a ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. The admin has the main responsibility to take care of the whole process. He will be able to track the work assigned to the agent. Whenever the agent is assigned to a customer, they will be notified with an email alert. Customers can view the status of the ticket till the service is provided. The agent will quickly address the customer's issue and mitigate any effects of the negative experience. Therefore, this application adds up satisfied customers and brings in more customers to an organization.
03	Novelty / Uniqueness	With an integrative approach, the project aims in establishing an end-to-end connection between the customer and the service agent through a chat service. These chatbots can transfer customers to service agents whenever human touch is required. This can help businesses speed up response times and also answer routine questions.
04	Social Impact / Customer Satisfaction	Customer satisfaction is based on understanding, defining, assessing and managing customer needs so that their expectations are met. This project ensures that the policies, objectives and responsibilities of the project will satisfy the customer needs where customer service agents spend less time on routine tasks and answering commonly asked questions.
05	Business Model (Revenue Model)	This model helps in improving the efficiency and productivity of the organization as the use of chatbots can save up to 30% in customer support cost and can help businesses save on customer service costs by speeding up response times and answering up to 80% of routine questions.
06		This project aims at solving all the complaints

	Scalability of the Solution	faced by the customers which sequentially ensures rapid business growth. It helps in enhancing the flexibility to deliver new features faster. It helps in maintaining long-term relationships with the customers which in turn helps in increasing the operational efficiency among the organization.

- **PROBLEM SOLUTION FIT**

Figure 3.4.1 Problem Solution Fit

CHAPTER 4
REQUIREMENT ANALYSIS

- **FUNCTIONAL REQUIREMENT**

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login via Google with Email Id or Username and password
FR-4	Admin Login	Login with google API or other social media API
FR-5	User Ticket Raising	Raise the ticket with detailed description for query
FR-6	Notification	Notification Via Email to user
FR-7	Feedback Customer	Feedback through the application

Table 4.1.1 Functional Requirement

- **NON-FUNCTIONAL REQUIREMENT**

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 ofdevices such as android, mac and desktops.
NFR-2	Security	Before any user trying to login their account to any newdevice ,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 particulartime limit. Also notification will be sent for each and every customer
NFR-3	Reliability	Providing Quality Content
NFR-4	Availability	24*7 Support
NFR-5	Scalability	Good performance for large Customers and workload

Table 4.2.1 Non-Functional Requirement

CHAPTER 5

PROJECT DESIGN

- **DATA FLOW DIAGRAMS**

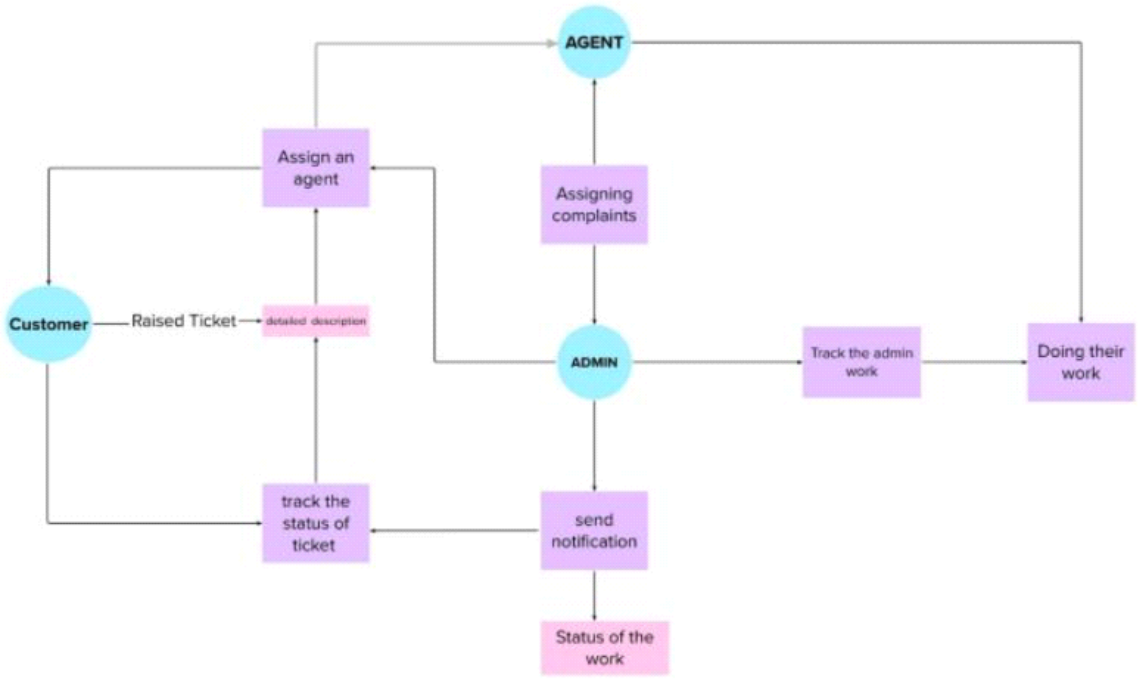
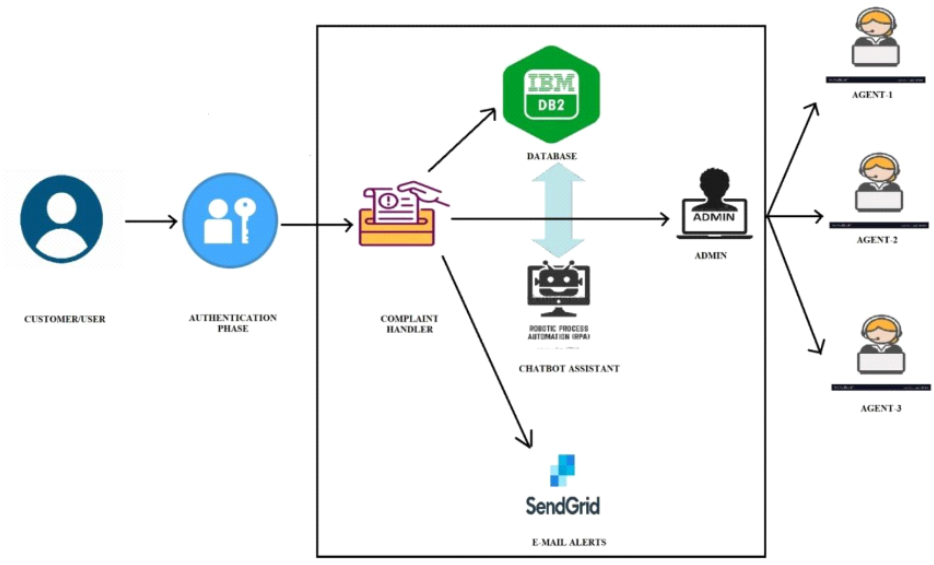


Figure 5.1.1 Data Flow of Customer care registry

- **SOLUTION AND TECHNICALARCHITECTURE**



USER STORIES:

User Type	Functional Requirement(Epic)	User Story Number	User Story Task	Acceptance Criteria	priority	Release
Customer	Registration	USN-1	As a user, I can register for the application by entering my email, password,and confirming my password.	I can access My account	High	Sprint -1
	Login	USN-2	As a user I can login the application by entering registered email and password	I can login to my account and can access the dash board	High	Sprint -1
	Raising Tickets	USN-3	As	I can raise	High	Sprint -1

			a user, I can raise a ticket regarding my problem	the ticket		
	Track Tickets	USN-4	As a user i can track my ticket status	I can view the statu sof my raised problem	Medium	

	Log out	USN-5	As a user, I can logout from my account	I can log out from my account	High	Sprint-2

Admin	Login	ASN-1	As an admin I can login the application by entering registered email and password	I can access my account	High	Sprint-2

	Assign an agent	ASN-2	As an admin, I can track the statusAssign an agent to the user complaints	I can assign an agent	High	Sprint-2

	Track the admin work	ASN-3	As an admin, I can track the status of the user raised queriesand the admin work	I can track the admin work	High	Sprint-2
	Send notification to customer	ASN-4	As an admin i can send the notification to the customer regarding raised ticket status	I can send notification to the customer	High	Sprint-2
	Ban doubtful user logins	ASN-5	As a admin, I can ban the suspicious accountsor users	I can ban the customer	Medium	
AGENT	Resolve queries	AGSN-1	As an Agent, I can resolve the customerqueries	I can resolve customer problem	High	Sprint-3
	Connecting with related problem	AGSN-2	As an Agent, I can conne	I can connect related	Medium	

s		ctto the related problems	problems or queries		
Flag the Tickets	AGSN-3	As a Agent, I can flag the status of the raised ticket	I can flagthe ticket	Low	Sprint-4

Figure 5.2.1 Solution Architecture

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	User Type	Functional Requirements	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	Selva Muthu Kannan, Rudresh
Sprint-1		Login	USN-2	As a customer, Ican login to the application by entering correct email and password	1	High	Rudresh
Sprint-1		Dashboard	USN-3	As a customer, I tickets raised by meant lot more	3	High	RohithSai
Sprint-2		Ticket creation	USN-4	As a customer I can create anew ticket with the detailed description of my query	2	High	Vishal Prasanth
Sprint-3		Address column	USN-5	As a customer, I can have conversations	3	High	Rudresh, Vishal Prasanth

				with the			
				assigned agent			
				16and get my			
				queries			
				clarified			
Spri		Forget	USN-6	As a	2	Medi u	Rohith

				mytickets			
Spri nt-	Agent (Web	Login	USN- 1	As an agent, I can login to	2	High	Rohith Sai
3	User)			the			
				application by			
				enteringcorre			
				ct			
				email and			
				password			
Spri nt-		Dashboard	USN- 2	As an agent, I can see all the	3	High	Vishal Prasanth
3				ticketsassigne			
				d to me by			
				the			
				admin			
Spri nt- 3		Address Column	USN- 3	As an agent, I get to have conversations with the	3	High	Rudresh, Vishal Prasanth
				customer and			
				clear his/her			
				queries			
Spri nt-		Forget Password	USN- 4	As an agent, I can reset my	2	Mediu m	Rohith Sai,
4				password by			Vishal
				this option in			Prasanth
				case I forgot			
				my old			
				password			
Spri nt- 1	Admin (Web User)	Login	USN- 1	As an admin, I	1	High	Selva Muthu Kannan,

				can login to the			Rohith
				application by			Sai
				enteringcorre			
				ct email and			
				password			
Spri nt-		Dashboard	USN-2	As an admin, I	3	High	Rohith Sai,
1				can see all the			Vishal
				tickets raised in the entire			Prasanth
				system and lot			
				more			
Spri nt- 2		Agent Creation	USN-3	As an admin, I can create an agent for clarifying the customer's queries	2	High	Rohith Sai, Rudresh
Spri		Assigning	USN-4	As an admin,	3	High	Rudresh,
Spri nt- 4		Forget password	USN-4	As an admin,I can reset my password by this option in case I forgot password	2	Mediu m	Vishal Prasanth, Selva Muthu Kannan

Table 6.1.1 Sprint Planning & Estimation

- **SPRINT DELIVERY SCHEDULE**

Sprint	Total story points	Duration	Sprint start date	Sprint end date	Story points completed	Sprint releasedate

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 **Velocity :**	11	6 Days	07 Nov 2022	12 Nov 2022	11	12 Nov 2022
I Sprint-4 hmagine we	8a 10-day av s e	6 Days pri ion, nt dura	14 Nov and ocity 2022 ve	19 Nov o am is 2022 te	8 20 (points per	19 Nov sp 2022).

Table 6.2.2 Sprint Delivery Schedule

Let's calculate the team's average velocity (AV) per iteration unit (storypoints per day)

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

- **REPORTS FROM JIRA**

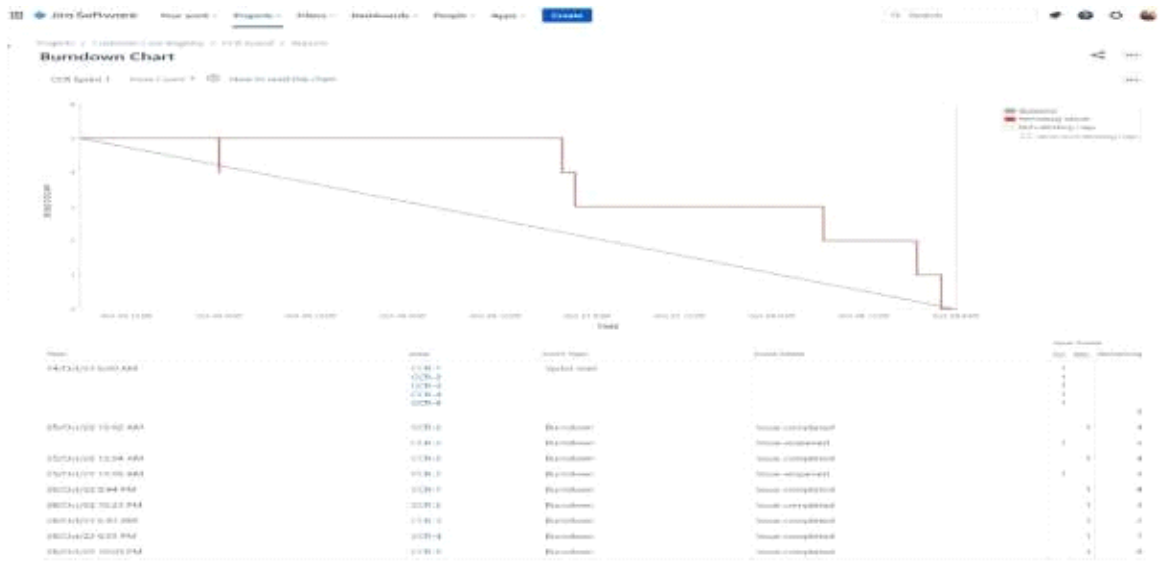


Figure 6.3.1 Reports from JIRA

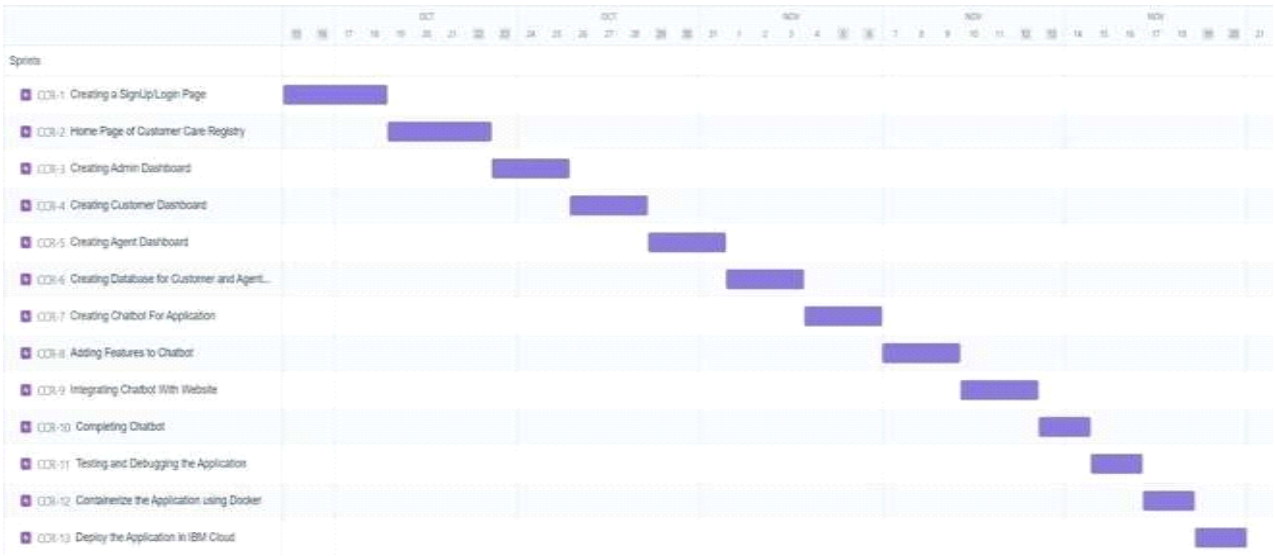


Figure 6.3.2 Reports from JIRA

CHAPTER 7

CODING AND SOLUTION

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 programme at no cost to the graduates. We provide a pool of varied, well-trained, techs-savvy individuals that wants to launch and advance their career in Alabama. The mission of veteran- and woman-owned Coding Solutions is to mobilize the next generation of IT 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 your Alabama team. Our applicants are swiftly hired at the top of the list by growing businesses for lucrative, long-term positions.

- **Features 1**

7 Main types of customer needs

□User-friendly

□ Empathy

- Fairness
- Control
- Alternatives
- Information

- **Features 2**

Complaint Tracking

- Email Alert
- 24/7 Monitoring

CHAPTER 8

TESTING

- **TEST CASES**

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on "HOW" to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

• S.no	Scenario	Input	Excepte doutput	• Actual output
• 1	Admin LoginForm	User name and password	Login	• Login success.
• 2	Employee Login Form	User name and password	Login	• Login success.
• 3	User RegistrationForm	User basic details • Table 8.1.1 Test Cas	Registered successfully • es	User basic details are stored in the database.
• 4	User Login Form	User name and password	Login	• Login success.

•

• USER ACCEPTANCE TESTING

This is a type of testing done by users, customers, or other authorized entities to determine application/software needs and business processes. Acceptance testing is the most important phase of testing as this decides whether the client approves the application/software or not. It may involve functionality, usability, performance, and UI of the application. It is also known as user acceptance testing (UAT), operational acceptance testing (OAT), and end-user testing.

• DEFECT ANALYSIS

This report shows the number of resolved or closed bugs at each severity level, and howthey were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
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**

Table 8.2.1. 1 Defect Analysis

-

- **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

Table 8.2.2.1 Testcase analysis

CHAPTER 9

RESULTS

- **PERFORMANCE METRICS**

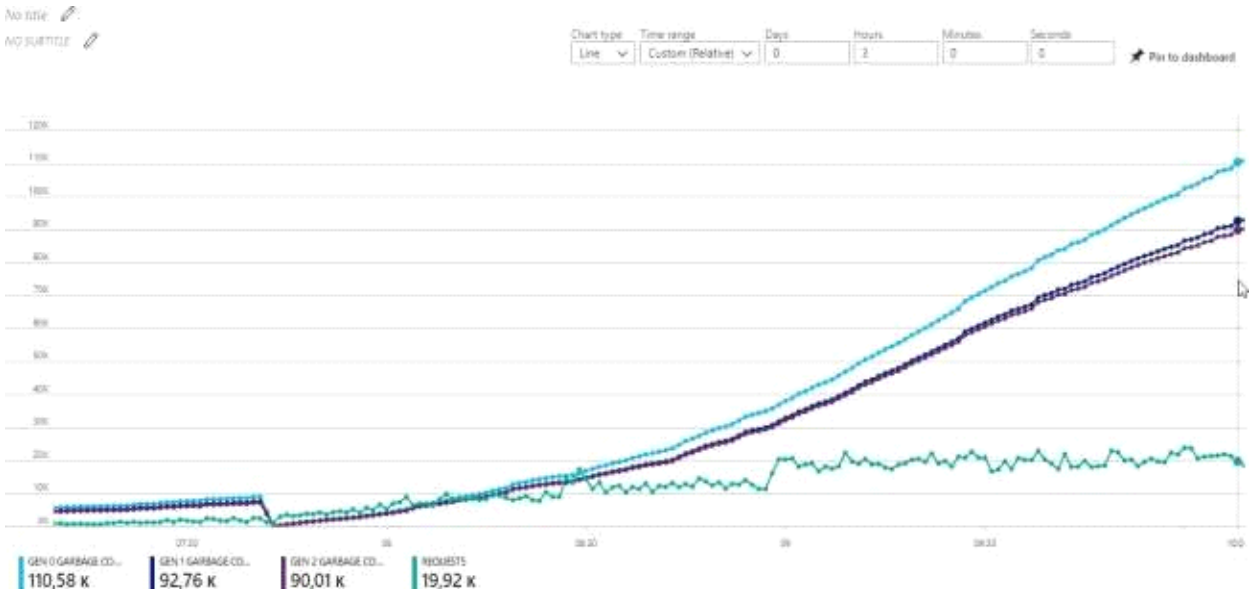


Figure 9.1.1 Performance Metrics

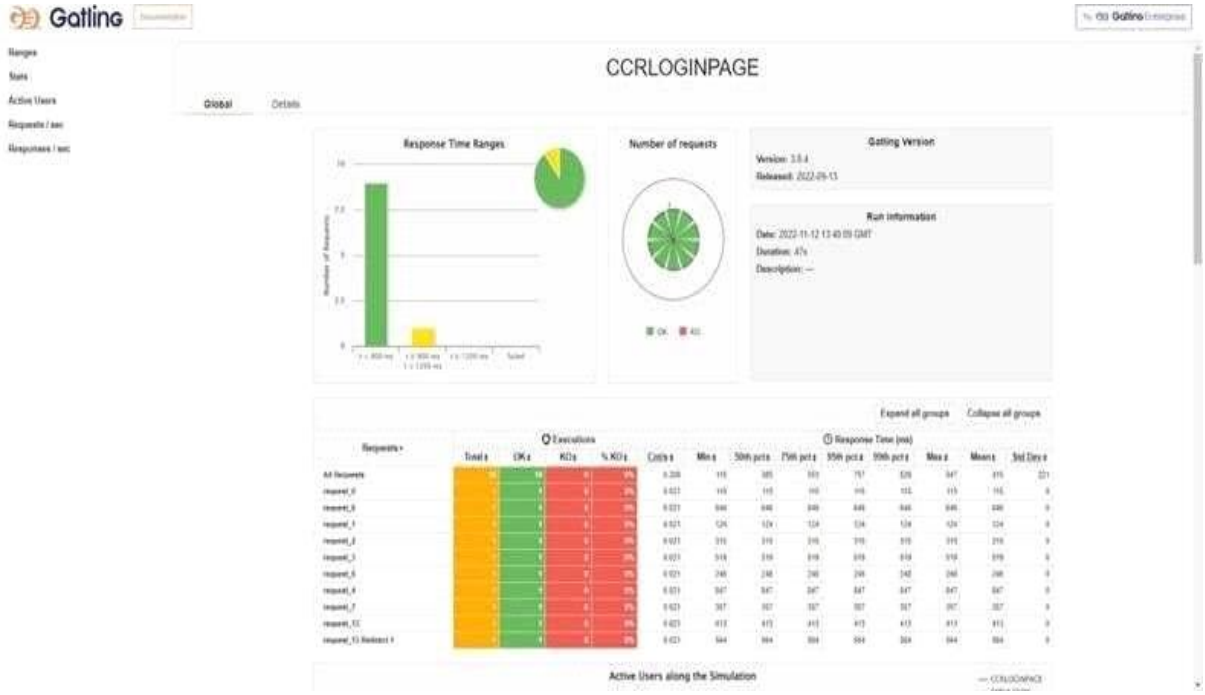


Figure 9.1.2 Performance Metrics

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- System is easy to understand and user friendly.
- The system is purely based on prediction which predicts an internet plan for the
- customer.
- Admin can easily view employee report based on the resolution provided on the
- complaint.
- Handle large number of contextual information.
- User friendly and time consuming process.
- Using this project, the user can know about status of complaint through website.
- Keep track of daily information exchange at the server by the administrator.
- Increase in processing and transfer speeds of information over the network.

DISADVANTAGES

- Requires an active internet connection.
- System may provide inaccurate results if the data entered incorrectly.
- Difficult to provide proper intimation system
- Current system is manual process
- Cannot always taking a call
- Tower problem during call conversation

CHAPTER 11

CONCLUSION

Application software has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly, and has required option, which can be utilized by the user to perform the desired operations. Application meets the information requirements specified to a great extent. The system has been designed keeping in view the present and future requirements in mind and made very flexible. The goals that are achieved by the software are Instant access,

improved productivity, Optimum utilization of resources, Efficient management of records, Simplifications of the operations, Less processing time and getting required information, User friendly, Portable and flexible for further enhancement. The system has the benefits of easy access because it is be developed as a platform independent web application, so the admin can maintain a proper contact with their users, which may be access anywhere. All communications between the police and administrator has done through the online, so this communication cost also is reduced.

CHAPTER 12

FUTURE

S

C

O

P

E

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.

CHAPTER 13 APPENDIX

SOURCE CODE

base.html

```
<!DOCTYPE html>
<head>
<link rel="stylesheet" href="static/css/main.css"/>
{% block head %}
{% endblock %}
</head>
<body>
{% block body %}
{% endblock %}
<script>
```

```
var coll = document.getElementsByClassName("collapsible");
var i;
for (i = 0; i < coll.length; i++) {
coll[i].addEventListener("click", function () {
this.classList.toggle("active");
var content = this.nextElementSibling;
if (content.style.display === "block") {
content.style.display = "none";
} else {
content.style.display = "block";
}
});
}
</script>
<footer style="text-align: right ;">
<a href="/about">Wanna know more about us? Click here</a>
</footer>
</body>
</html>
```

login.html

```
{% extends 'base.html' %}
{% block head %}
<title>
Login
</title>
{% endblock %}
{% block body %}
```

```
<div class="forpadding">
<!-- for box of the signup form -->
<div class="sign">
<div>
<p class="fortitle">
Sign In
</p>
<hr>
<form action="/login" method="post">
<div class="forform">
<div class="textinformleft">
Username
</div>
<div class="textinformright">
<input type="name" name="username">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Password
</div>
<div class="textinformright">
<input type="password" name="pass">
</div>
</div>
<br>
<div>
```

```
<button class="forbutton" type="submit"> Sign In
>></button>
</div>
</form>
<br>
<div>
New user? <a href="/signup">Sign up</a>
</div>
<br>
</div>
</div>
</div>
{% endblock %}
```

signup.html

```
{% extends 'base.html' %}
{% block head %}
<title>
Sign Up
</title>
{% endblock %}
{% block body %}
<div class="forpadding">
<!-- for box of the signup form -->
<div class="sign">
<div>
<p class="fortitle">
Register Now!!
```

```
</p>
<hr>
<form action="/signup" method="post">
<div class="forform">
<div class="textinformleft">
Username
</div>
<div class="textinformright">
<input type="name" name="username">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Name
</div>
<div class="textinformright">
<input type="name" name="name">
</div>
</div>
<div class="forform">
<div class="textinformleft">
E - mail
</div>
<div class="textinformright">
<input type="name" name="email">
</div>
</div>
```

```
<div class="forform">
<div class="textinformleft">
Phone Number
</div>
<div class="textinformright">
<input type="name" name="phn">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Password
</div>
<div class="textinformright">
<input type="password" name="pass">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Re - enter Password
</div>
<div class="textinformright">
<input type="password" name="repass">
</div>
</div>
<br>
<div>
<button class="forbutton" type="submit"> Sign up
```

>></button>

</form>

{{msg}}

Already have an account? Sign in

{% endblock %}

dashboard.html

{% extends 'base.html' %}

{% block head %}

<title>

Dashboard

</title>

{% endblock %}

{% block body %}


```
Welcome {{ name }},
</div>
<div class="fordashboardtopelements2">
<a href="/login"><button class="forbutton">Sign out</button></a>
</div>
</div>
<br>
<div class="outerofdashdetails">
<div class="fordashboarddetails">
<br>
<!-- table of customers complaints -->
<table class="fortable">
<thead>
<th>Complaint ID</th>
<th class="pad">Complaint Detail</th>
<th>Assigned Agent</th>
<th>Status</th>
<th>Solution</th>
</thead>
<tbody>
{% for i in complaints %}
<tr>
<td>
{{ i['C_ID'] }}
</td>
<td class="pad">
{{ i['TITLE'] }}
```

```
</td>
<td>
{{ i['ASSIGNED_AGENT'] }}
</td>
<td>
{% if i['STATUS'] == 1 %}
Completed
{% elif i['STATUS'] == 0 %}
Not completed
{% else %}
In progress
{% endif %}
</td>
<td>
{{ i['SOLUTION'] }}
</td>
</tr>
{% endfor %}
</tbody>
</table>
<br>
<center>
<div class="fordashboarddetails">
<button type="button" class="collapsible">Add new complaint
+</button>
<div class="content">
<br>
```

```
<form action="/addnew" method="post">
<div class="forform">
<div class="textinformleft">
Title
</div>
<div class="textinformright">
<input type="name" name="title">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Complaint
</div>
<div class="textinformright">
<textarea name="des"
style="border-radius: 1rem;width:
90%;height: 150%;background-color: black;color: white;"></textarea>
</div>
</div>
<br>
<br>
<div>
<button class="forbutton" type="submit"> Submit
</button>
</div>
</form>
<br>
```

```
</div>
</div>
</center>
</div>
</div>
{% endblock %}
```

admin.html

```
{% extends 'base.html' %}
{% block head %}
<title>
Admin Dashboard
</title>
{% endblock %}
{% block body %}
<br>
<div class="fordashboardtop">
<div class="fordashboardtopelements1">
Welcome Admin,
</div>
<div class="fordashboardtopelements2">
<a href="/login"><button class="forbutton">Sign out</button></a>
</div>
</div>
<br>
<div class="outerofdashdetails">
<div class="fordashboarddetails">
<br>
```

```
<!-- table of customers complaints -->
<table class="fortable">
<thead>
</thead>
<tbody>
<tr>
<td class="pad">
<a href="/agents">Agent Details</a>
</td>
<td class="pad">
<a href="/tickets">Customer Ticket Details</a>
</td>
</tr>
</tbody>
</table>
<br>
</div>
</div>
{% endblock %}
```

agents.html

```
{% extends 'base.html' %}
{% block head %}
<title>
Dashboard
</title>
{% endblock %}
{% block body %}
```


Welcome Admin,

<button class="forbutton">Sign out</button>

<!-- table of customers complaints -->

<table class="fortable">

<thead>

<th class="pad">Name</th>

<th>Username</th>

<th>Email</th>

<th>Phone</th>

<th>Domain</th>

<th>Status</th>

</thead>

<tbody>

{% for i in agents %}

<tr>

<td class="pad">

```
{{ i['NAME'] }}
</td>
<td class="pad">
{{ i['USERNAME'] }}
</td>
<td>
{{ i['EMAIL'] }}
</td>
<td>
{{ i['PHN'] }}
</td>
<td>
{{ i['DOMAIN'] }}
</td>
<td>
{% if i['STATUS'] == 1 %}
Assigned to job
{% elif i['STATUS'] == 0 %}
not Available
{% else %}
Available
{% endif %}
</td>
</tr>
{% endfor %}
</tbody>
</table>
```


<button type="button" class="collapsible">Add new agent

+</button>

<form action="/addnewagent" method="post">

Username

<input type="name" name="username">

Name

<input type="name" name="name">

Email

```
</div>
<div class="textinformright">
<input type="name" name="email">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Phone
</div>
<div class="textinformright">
<input type="name" name="phone">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Domain
</div>
<div class="textinformright">
<input type="name" name="domain">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Password
</div>
<div class="textinformright">
<input type="password" name="password">
```

```
</div>
</div>
<br>
<br>
<div>
<button class="forbutton" type="submit"> Submit
</button>
</div>
</form>
<br>
</div>
</div>
</center>
</div>
</div>
{% endblock %}
```

agentdash.html

```
{% extends 'base.html' %}
{% block head %}
<title>
Agent Dashboard
</title>
{% endblock %}
{% block body %}
<br>
<div class="fordashboardtop">
<div class="fordashboardtopelements1">
```

```
Welcome {{ name }},
</div>
<div class="fordashboardtopelements2">
<a href="/login"><button class="forbutton">Sign out</button></a>
</div>
</div>
<br>
<div class="outerofdashdetails">
<div class="fordashboarddetails">
<br>
<!-- table of customers complaints -->
<table class="fortable">
<thead>
<th>Complaint ID</th>
<th class="pad">Username</th>
<th>Title</th>
<th>Complaint</th>
<th>Solution</th>
<th>Status</th>
</thead>
<tbody>
{% for i in complaints %}
<tr>
<td class="pad">
{{ i['C_ID'] }}
</td>
<td class="pad">
```

```
{{ i['USERNAME'] }}
</td>
<td>
{{ i['TITLE'] }}
</td>
<td>
{{ i['COMPLAINT'] }}
</td>
<td>
{{ i['SOLUTION'] }}
</td>
<td>
{% if i['STATUS'] == 1 %}
Completed
{% else %}
Not Completed
{% endif %}
</td>
</tr>
{% endfor %}
</tbody>
</table>
<br>
<center>
<div class="fordashboarddetails">
<button type="button" class="collapsible">Solve an Issue
⚡</button>
```

```
<div class="content">
<br>
<form action="/updatecomplaint" method="post">
<div class="forform">
<div class="textinformleft">
Complaint ID
</div>
<div class="textinformright">
<input type="name" name="cid">
</div>
</div>
<div class="forform">
<div class="textinformleft">
Solution
</div>
<div class="textinformright">
<input type="text" name="solution">
</div>
</div>
<br>
<br>
<div>
<button class="forbutton" type="submit"> Submit
</button>
</div>
</form>
<br>
```

```
</div>
</div>
</center>
</div>
</div>
{% endblock %}
```

tickets.html

```
{% extends 'base.html' %}
{% block head %}
<title>
Agent Dashboard
</title>
{% endblock %}
{% block body %}
<br>
<div class="fordashboardtop">
<div class="fordashboardtopelements1">
Welcome Admin,
</div>
<div class="fordashboardtopelements2">
<a href="/login"><button class="forbutton">Sign out</button></a>
</div>
</div>
<br>
<div class="outerofdashdetails">
<div class="fordashboarddetails">
<br>
```

```
<!-- table of customers complaints -->
<table class="fortable">
<thead>
<th>Complaint ID</th>
<th class="pad">Username</th>
<th>Title</th>
<th>Complaint</th>
<th>Solution</th>
<th>Status</th>
</thead>
<tbody>
{% for i in complaints %}
<tr>
<td>{{ i['C_ID'] }}</td>
<td class="pad">
{{ i['USERNAME'] }}
</td>
<td>
{{ i['TITLE'] }}
</td>
<td>
{{ i['COMPLAINT'] }}
</td>
<td>
{{ i['SOLUTION'] }}
</td>
<td>
```

{% if i['STATUS'] == 1 %}

Completed

{% else %}

Not Completed

{% endif %}

</td>

</tr>

{% endfor %}

</tbody>

</table>

<button type="button" class="collapsible">Assign an agent

⚡</button>

<form action="/assignagent" method="post">

Complaint ID

<input type="name" name="ccid">

```
<div class="textinformleft">
<label for="agent">Choose an agent:</label>
</div>
<div class="textinformright">
<select name="agent" id="agent">
{% for i in freeagents %}
<option value={{ i['USERNAME'] }}>{{
i['USERNAME'] }}</option>
{% endfor %}
</select>
</div>
</div>
<br>
<br>
<div>
<button class="forbutton" type="submit"> Submit
</button>
</div>
</form>
<br>
</div>
</div>
</center>
</div>
</div>
{% endblock %}
```

main.css

```
.sign {
border-radius: 1rem;
background-color: rgba(255, 185, 46, 0.976);
text-align: center;
padding: 1%;
}
.fortitle {
font-size: medium;
font-weight: 500;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
padding: 5px;
}
.forp {
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
.textinformleft {
text-align: left;
padding-left: 5%;
width: 50%;
border-radius: 1rem;
font-size: medium;
font-weight: 500;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
.textinformright {
width: 50%;
padding-right: 10px;
```

```
border-radius: 1rem;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
.textinformright2 {
width: 100%;
text-align: center;
padding-right: 10px;
border-radius: 1rem;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
input {
border-radius: 1rem;
color: white;
background-color: black;
padding-left: 15px;
}
input:focus {
border-color: yellow;
}
.forform {
display: flex;
padding: 15px;
border-radius: 1rem;
}
.forpadding {
padding-top: 5%;
padding-left: 25%;
```

```
padding-right: 25%;
}
body {
background-image: url('bg9.jpg');
background-repeat: no-repeat;
/* background-color: black; */
/* background-image: url('F:\Own\IBM project\Sample2\static\css\bg.png');
*/
}
.forbutton {
background-color: black;
color: white;
border-radius: 1rem;
padding: 7px;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
button:hover {
background-color: white;
color: black;
box-shadow: white;
cursor: pointer;
}
/* for dashboard */
.fordashboardtop {
border-radius: 1rem;
display: flex;
background-color: rgba(255, 185, 46, 0.976);
```

```
}
.fordashboardtopelements1 {
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
width: 90%;
font-size: large;
padding: 2%;
}
.fordashboardtopelements2 {
width: 10%;
padding-top: 1%;
padding-bottom: 1%;
}
.fordashboarddetails {
padding: 2%;
border-radius: 1rem;
background-color: rgba(255, 185, 46, 0.976);
}
.outerofdashdetails {
/* padding-top: 2%; */
padding-left: 5%;
padding-right: 5%;
}
.fortable {
width: 100%;
padding: 1%;
text-align: center;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
```

```
}
.pad {
padding: 7px;
}
.forbutton2 {
background-color: black;
color: white;
border-radius: 1rem;
padding: 7px;
width: 200%;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
.foraddbutton{
/* width: 30%; */
background-color: black;
color: white;
border-radius: 1rem;
padding: 7px;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
.collapsible {
background-color: black;
color: white;
border-radius: 1rem;
padding: 7px;
width: 30%;
font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
```

```
/* background-color: #777; */
/* color: white; */
cursor: pointer;
/* padding: 18px; */
/* width: 100%; */
/* border: none;
text-align: left; */
/* outline: none;
font-size: 15px; */
}
.collapsible:hover {
background-color: white;
}
.content {
/* padding: 0 18px; */
display: none;
border-radius: 1rem;
background-color: rgba(255, 185, 46, 0.976);
width: 50%;
/* overflow: hidden; */
/* background-color: #f1f1f1; */
}
```

app.py

```
from flask import Flask, render_template, request, redirect, session, url_for
import ibm_db
import re
app = Flask(__name__)
```

```
# for connection

# conn= ""

app.secret_key = 'a'

print("Trying to connect...")

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=ea286ace-86c7-4d5b-8580-

3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31505;SECURIT

Y=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=zyr46226;PWD=fIKQqRn
XO

VfcA0Ht;", ", ")

print("connected..")

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

def signup():

global userid

msg = ''

if request.method == 'POST':

username = request.form['username']

name = request.form['name']

email = request.form['email']

phn = request.form['phn']

password = request.form['pass']

repass = request.form['repass']

print("inside checking")

print(name)

if len(username) == 0 or len(name) == 0 or len(email) == 0 or len(phn)

== 0 or len(password) == 0 or len(repass) == 0:

msg = "Form is not filled completely!!"

print(msg)
```

```
return render_template('signup.html', msg=msg)
elif password != repass:
msg = "Password is not matched"
print(msg)
return render_template('signup.html', msg=msg)
elif not re.match(r'[a-z]+', username):
msg = 'Username can contain only small letters and numbers'
print(msg)
return render_template('signup.html', msg=msg)
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
msg = 'Invalid email'
print(msg)
return render_template('signup.html', msg=msg)
elif not re.match(r'[A-Za-z]+', name):
msg = "Enter valid name"
print(msg)
return render_template('signup.html', msg=msg)
elif not re.match(r'[0-9]+', phn):
msg = "Enter valid phone number"
print(msg)
return render_template('signup.html', msg=msg)
sql = "select * from users where username = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
print(account)
```

```
if account:
msg = 'Acccount already exists'
else:
userid = username
insert_sql = "insert into users values(?,?,?,?,?)"
prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prep_stmt, 1, username)
ibm_db.bind_param(prep_stmt, 2, name)
ibm_db.bind_param(prep_stmt, 3, email)
ibm_db.bind_param(prep_stmt, 4, phn)
ibm_db.bind_param(prep_stmt, 5, password)
ibm_db.execute(prep_stmt)
print("successs")
msg = "succesfully signed up"
return render_template('dashboard.html', msg=msg, name=name)
else:
return render_template('signup.html')
@app.route('/dashboard')
def dashboard():
return render_template('dashboard.html')
@app.route('/')
def base():
return redirect(url_for('login'))
@app.route('/login', methods=["GET", "POST"])
def login():
global userid
msg = "
```

```
if request.method == 'POST':
username = request.form['username']
userid = username
password = request.form['pass']
if userid == 'admin' and password == 'admin':
print("its admin")
return render_template('admin.html')
else:
sql = "select * from agents where username = ? and password = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, password)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
print(account)
if account:
session['Loggedin'] = True
session['id'] = account['USERNAME']
userid = account['USERNAME']
session['username'] = account['USERNAME']
msg = 'logged in successfully'
# for getting complaints details
sql = "select * from complaints where assigned_agent = ?"
complaints = []
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)
```

```
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
complaints.append(dictionary)
dictionary = ibm_db.fetch_assoc(stmt)
print(complaints)
return render_template('agentdash.html',
name=account['USERNAME'], complaints=complaints)
sql = "select * from users where username = ? and password = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, password)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
print(account)
if account:
session['Loggedin'] = True
session['id'] = account['USERNAME']
userid = account['USERNAME']
session['username'] = account['USERNAME']
msg = 'logged in successfully'
# for getting complaints details
sql = "select * from complaints where username = ?"
complaints = []
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)
dictionary = ibm_db.fetch_assoc(stmt)
```

```
while dictionary != False:
# print "The ID is : ", dictionary["EMPNO"]
# print "The Name is : ", dictionary[1]
complaints.append(dictionary)
dictionary = ibm_db.fetch_assoc(stmt)
print(complaints)
return render_template('dashboard.html', name=account['USERNAME'],
complaints=complaints)
else:
msg = 'Incorrect user credentials'
return render_template('dashboard.html', msg=msg)
else:
return render_template('login.html')
@app.route('/addnew', methods=["GET", "POST"])
def add():
if request.method == 'POST':
title = request.form['title']
des = request.form['des']
try:
sql = "insert into complaints(username,title,complaint)
values(?,?,?)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, userid)
ibm_db.bind_param(stmt, 2, title)
ibm_db.bind_param(stmt, 3, des)
ibm_db.execute(stmt)
except:
```

```
print(userid)
print(title)
print(des)
print("cant insert")
sql = "select * from complaints where username = ?"
complaints = []
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, userid)
ibm_db.execute(stmt)
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
# print "The ID is : ", dictionary["EMPNO"]
# print "The Name is : ", dictionary[1]
complaints.append(dictionary)
dictionary = ibm_db.fetch_assoc(stmt)
print(complaints)
return render_template('dashboard.html', name=userid,
complaints=complaints)
@app.route('/agents')
def agents():
sql = "select * from agents"
agents = []
stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
agents.append(dictionary)
```

```
dictionary = ibm_db.fetch_assoc(stmt)
return render_template('agents.html', agents=agents)
@app.route('/addnewagent', methods=["GET", "POST"])
def addagent():
if request.method == 'POST':
username = request.form['username']
name = request.form['name']
email = request.form['email']
phone = request.form['phone']
domain = request.form['domain']
password = request.form['password']
try:
sql = "insert into agents values(?,?,?,?,?,?,2)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, name)
ibm_db.bind_param(stmt, 3, email)
ibm_db.bind_param(stmt, 4, phone)
ibm_db.bind_param(stmt, 5, password)
ibm_db.bind_param(stmt, 6, domain)
ibm_db.execute(stmt)
except:
print("cant insert")
sql = "select * from agents"
agents = []
stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
```

```
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
agents.append(dictionary)
dictionary = ibm_db.fetch_assoc(stmt)
return render_template('agents.html', agents=agents)
@app.route('/updatecomplaint', methods=["GET", "POST"])
def updatecomplaint():
if request.method == 'POST':
cid = request.form['cid']
solution = request.form['solution']
try:
sql = "update complaints set solution =?,status=1 where c_id = ?
and assigned_agent=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, solution)
ibm_db.bind_param(stmt, 2, cid)
ibm_db.bind_param(stmt, 3, userid)
ibm_db.execute(stmt)
sql = "update agents set status =3 where username=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, userid)
ibm_db.execute(stmt)
except:
print("cant insert")
sql = "select * from complaints where assigned_agent = ?"
complaints = []
stmt = ibm_db.prepare(conn, sql)
```

```
ibm_db.bind_param(stmt, 1, userid)
ibm_db.execute(stmt)
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
complaints.append(dictionary)
dictionary = ibm_db.fetch_assoc(stmt)
# print(complaints)
return render_template('agentdash.html', name=userid,
complaints=complaints)
@app.route('/tickets')
def tickets():
sql = "select * from complaints"
complaints = []
stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
complaints.append(dictionary)
dictionary = ibm_db.fetch_assoc(stmt)
sql = "select username from agents where status <> 1"
freeagents = []
stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
freeagents.append(dictionary)
dictionary = ibm_db.fetch_assoc(stmt)
```

```
print(freeagents)
return render_template('tickets.html', complaints=complaints,
freeagents=freeagents)
@app.route('/assignagent', methods=['GET', 'POST'])
def assignagent():
if request.method == "POST":
ccid = request.form['ccid']
agent = request.form['agent']
print(ccid)
print(agent)
try:
sql = "update complaints set assigned_agent =? where c_id = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, agent)
ibm_db.bind_param(stmt, 2, ccid)
ibm_db.execute(stmt)
sql = "update agents set status =1 where username = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, userid)
ibm_db.execute(stmt)
except:
print("cant update")
return redirect(url_for('tickets'))
if __name__ == "__main__":
app.run(debug=True)
```

Sendgrid Integration using python

CODE :

```
import smtplib

import sendgrid as sg

import os from sendgrid

import SendGridAPIClient from sendgrid.helpers.mail import Mail, Email, To, Content
SUBJECT =

"customer care registry"

s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):

from_email = Email("tour7107@gmail.com")

to_email = To(email)

subject = "Sending with SendGrid is Fun"

content = Content("text/plain",TEXT)

mail = Mail(from_email, to_email, subject, content)

try:

 sg=SendGridAPIClient('SG.3wVvuDLTQ
aoSvEgQ8xy7w.2Mp38QJmhoG_p09E3x7HA2OAGRCx9TD7QTenuE

Hfp9k')

 response = sg.send(mail)

print(response.status_code)

print(response.body)

print(response.headers)

except Exception as e:

print(e)

 # print("sorry we cant process your candidature")

 # s = smtplib.SMTP('smtp.gmail.com', 587)

 # s.starttls()

 # # s.login("il.tproduct8080@gmail.com", "oms@1Jessi")

 # s.login("tour7107@gmail.com", "1234567890123456")
```

```
 # message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
 # # s.sendmail("il.tproduct8080@gmail.com", email, message)
 # s.sendmail("tour7107@gmail.com", email, message)
 # s.quit()
# def sendgridmail(user,TEXT):

# # # from_email = Email("shridhartp24@gmail.com")
# # from_email = Email("tour7107@gmail.com")
# # to_email = To(user)
# # subject = "Sending with SendGrid is Fun"
# # content = Content("text/plain",TEXT)
# # mail = Mail(from_email, to_email, subject, content)
# # # Get a JSON-ready representation of the Mail object
# # mail_json = mail.get()
# # # Send an HTTP POST request to /mail/send
# # response = sg.client.mail.send.post(request_body=mail_json)
# # print(response.status_code)
# # print(response.headers)
# message = Mail(
# from_email='tour7107@gmail.com',
# to_emails='melciyajaffrin@gmail.com',
# subject='Sending with Twilio SendGrid is Fun',
# html_content='<strong>and easy to do anywhere, even with Python</strong>')
# try:
# sg=SendGridAPIClient('SG.3wVvuDLTQ
aoSvEgQ8xy7w.2Mp38QJmhoG_p09E3x7HA2OAGRCx9TD7QTenuE
Hfp9k')
```

```
# response = sg.send(message)
# print(response.status_code)
# print(response.body)
# print(response.headers) # except
Exception as e:
# print(e)
```