CUSTOMER CARE REGISTRY

RETAILS AND E-COMMERCE

TEAM MEMBERS:

DHEVA M (TEAM LEAD)
HEMANTH E C
HENDRY CHARLES G
SUDEEP P

1.1INTRODUCTION:

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

ORGANIZATION PROFILE

SOFTWARE SOLUTION

Software Solutions is an IT solution provider for a dynamic environment where business and technology strategies converge. Their approach focuses on new ways of business combining It innovation and adoption while also leveraging an organization's current IT assets. Their work with large global corporations and new products or services and to implement prudent business and technology strategies in today's environment.

RANGE OF EXPERTISE INCLUDES:

- Software Development Services
- Engineering Services
- Systems Integration
- Customer Relationship Management
- Product Development
- Electronic Commerce
- Consulting
- IT Outsourcing

We apply technology with innovation and responsibility to achieve two broadobjectives:

- Effectively address the business issues our customers face today.

THIS APPROACH RESTS ON:

- A strategy where we architect, integrate and manage technology services

and solutions - we call it AIM for success.

- A robust offshore development methodology and reduced demand on customer resources.
- A focus on the use of reusable frameworks to provide cost and times benefits.

They combine the best people, processes and technology to achieve excellent results - consistency. We offer customers the advantages of SPEED

SPEED:

They understand the importance of timing, of getting there before the competition. A rich portfolio of reusable, modular frameworks helps jump-start projects. Tried and tested methodology ensures that we follow a predictable, low - risk path to achieve results. Our track record is testimony to complex projects delivered within and events before schedule

SERVICES:

Our Service is providing it's services to companies which are in the field of production, quality control etc. With their rich expertise and experience and information technology they are in best position to provide software solutions to distinct business requirements.

1.2 PURPOSEOF THE PROJECT

An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through and e-mail. The system should have capability to integrate with any Service Provider from any domain or industry like selling product and give offers insight on customer data ,sell quickly, low cost etc.

Customer Service also known as Client Service is the provision of service to customers Its significance varies by product, industry and domain. In many cases customer services is more important if the information relates to a service as opposed to a Customer.

Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer.

2.1 PROBLEMS IN EXISTING SYSTEM

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

2.2 REFERENCE

TITLE: Retails and E-commerce

AUTHOR NAME: Mar Novita

YEAR: 2020

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 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; 5) Customer Satisfaction affects complaint level. Keywords: Customer Satisfaction, Complaint Level, Product Quality and Service Quality

LITERATURE REVIEW

Customer Satisfaction:

Customer Satisfaction is a feeling of pleasure or disappointment of someone who appears after comparing the performance (results) of the product thought against the expected performance results (Kotler 2006:177, 2019)). The dimension or indicator of Customer Satisfaction is if the performance is below the expectations of eating dissatisfied customers, if the performance meets expectations then the customer is satisfied, if the performance exceeds expectations then the customer is very satisfied or happy (Kotler 2006:177, 2019) .

Customer Satisfaction is an attitude that is decided based on the experience obtained. Satisfaction is an assessment of the characteristicsor privileges of a product or service, or the product itself, that provides a levelof consumer pleasure with regard to meeting consumer consumption needs(Sugeng, 2016).Dimensions or indicators of Customer Satisfaction can be created through quality, service, and value. The key to generating customer loyalty is to provide high customer value. (Sugeng, 2016)

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. (Tjiptono, 2012)

Customer Satisfaction has been researched a lot by previous researchers including (Afriliana et al., 2018; Librianty & Yuliarto, 2019; Purwanti et al., 2014; Rahayu & Setyawarti, 2018; Rangkuti, 2003; Risdah, 2019; SiahaanSodiq & Wijaksana, 2014; Supardiasa et al., 2018; Wahyuddinet al., n.d.; Wijayanti, 2019; YUNIATI, 2016; Zahratul Aini, 2019)

Complaint Level

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 (Tjiptono, 2007) of complaint level is the high level of complaint. (Tjiptono, 2007) This level of complaint hasbeen researched by many previous researchers, among which are , .(Rizqi et al., 2020),(Setiadi & Wahyudi, 2020)

Product Quality

Product Quality is the ability of a product to perform its functions, including the overall product, reliability, accuracy, ease of operation, repair, and other attributes. Dimensions or indicators of Product Quality is that customers can get all the benefits of the products offered to him.(Novia et al., 2020)

Product Quality is a dynamic condition that is interconnected although it can have different definitions but in essence has a specification that can cause a sense of satisfaction that exceeds expectations for customers who use it.(Rahman et al., 2018).

Product Quality has been researched by many previous researchers, including (Irma Ike Saputri, 2017; Novia et al., 2020; Rahman etal., 2018)

Quality of Service

Service Quality is a way of companies that try to make continuous quality improvements to the processes, products, and services produced by the company Dimensions or indicators of Service Quality is the more quality of service provided by the company then the satisfaction felt by customers will be higher, and vice versa.(Marnovita, 2020).

Quality of Service is good and or bad or satisfied or not customers are satisfied with the service provided. Dimensions or indicators of Quality of Service is the level ofsatisfaction measured through questionnairesor questionnaires in assessing the quality of a service.(Risdah, 2019)

The quality of service has been researched by many previous researchers, including, (Mulyadi, 2020),(Purwanti et al., 2014) , (Kuswatiningsih, 2010), (Supardiasa et al., 2018)

S NO	Author (year)	Previous research results	Security with this article	Difference with this article
1	(Novia et al., 2020)	product quality, service quality have a positive and significant effect on customer satisfaction and complaint level	quality of service affects customer satisfaction & complaint level	product quality affects customer satisfaction & complaint level
2	(Rahman et al., 2018)	Product quality has a positive and significanteffect on customer satisfaction and complaint levels	quality of service affects the level of complaints	Product quality affects customer satisfaction
3	(Purwanti et al., 2014)	product quality, service quality and x3 are positive and significant towards customer satisfaction and complaint level	Product quality affects customer satisfaction	quality of service affects the levelof complaints
4	(SiahaanSodiq & Wijaksana, 2014)	product quality, service quality and x3 are positive and significant towards customer satisfaction and complaint level	quality of service affects customer satisfaction & complaint level	product quality affects customer satisfaction & complaint level
5	(Librianty & Yuliarto, 2019)	product quality & x3 positive and significant impact on customer satisfaction and complaint level	quality of service affects the level of complaints	Product quality affects customer satisfaction
6	(Supardiasa et al., 2018)	product quality, service quality and x3 are positive and significant towards customer satisfaction and complaint level	Product quality affects customer satisfaction	quality of service affects the levelof complaints
7	Zahratul Aini, 2019)	product quality, quality of service is positive and significant to customer satisfaction and complaint level	quality of service affects customer satisfaction & complaint level	product quality affects customer satisfaction & complaint level
8	(Rangkuti, 2003)	product quality is positive and significant to customer satisfaction and complaint level	quality of service affects the level of complaints	Product quality affects customer satisfaction
9	(Rahayu & Setyawarti, 2018)	product quality, quality of service is positive and significant to customer satisfaction and complaint level	Product quality Affects customer satisfaction	Product quality affects customer satisfaction
10	(Hidayati, 2020)	product quality, quality of service is positive and significant to customer satisfaction and complaint level	quality of service affects customer satisfaction & complaint level	product quality affects customer satisfaction & complaint level

2.3 PROBLEM STATEMENT DEFINITION

The development of this new system objective is to provide the solution to the problems of existing system. By using this new system, we can fully automate the entire process of the current system. The new system would like to make as web-enabled so that the information can be shared between the members at any time using the respective credentials. To track the status of an individual process, the status update can be centralized using the new system. Being a web-enabled system, the process can be accessed across the world over internet.

This system also providing the features like Chatting, Mailing between the members; Images Upload – Download via the web site; updating the process status in centralized location; generated reports can also be exporting to the applications like MS-Excel, PDF format, etc. In this new system, the members like Donors can give their valuable feedback to the Volunteers so that the Volunteers can check their progress of the tasks.

The entire process categorized as different modules like Admin module, Volunteer module, etc. at where we can classify the functionality as an individual process. Using the new system entering into Admin module we can perform…. In this new system using the Volunteer module we can do….

Customer care is a way of dealing with customers when they interact with your brand, products, or services. 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 the customer, they will be notified with an email alert. Customers can view the status of the ticket till the service is provided. Customer 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.

The main roles and responsibilities of the admin is to take care of the whole process. Starting from Admin login followed by the agent creation and assigning the customers complaints. Finally, he will be able to track the work assigned to the agent and notification will be sent to the customer.

The main use of this project is to help the customer in processing their complaints. The customers can raise the customer care of their issues and the problem will be solved by the organization.
Developing a cloud application to help the customer in processing their complaints. In this application, the customer can raise an issue or a ticket with a detailed description, then the admin review and acknowledge the ticket by assigning an agent to the customer care.

An email notification triggered to the customer and the customer can track the status of the resolving process. The customer and agent can communicate with each other by calls or live chat.

This enables the agent to understand the issue and to solve the issue quickly. The customer can use channels or forums or FAQs to know more about the issues before raising a customer care in E-commerce

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP

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.

Creating an 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's perspective along with his or her goals and challenge

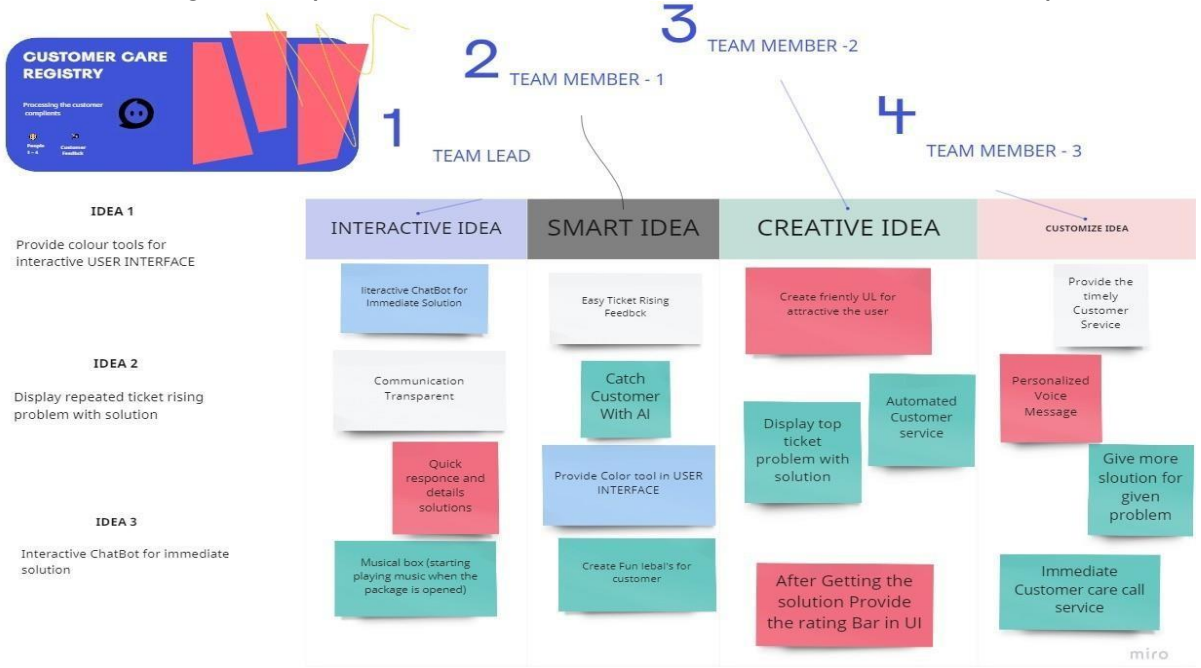
3.2 IDEATION AND BRAINSTORMING

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.

Ideation is the process where you generate ideas and solutions through session.

Ideation is the third phase of the Design thinking process and it all about generating ideas.

Brain storming, Worst possible Idea, and a wealth of other Ideation techniques.

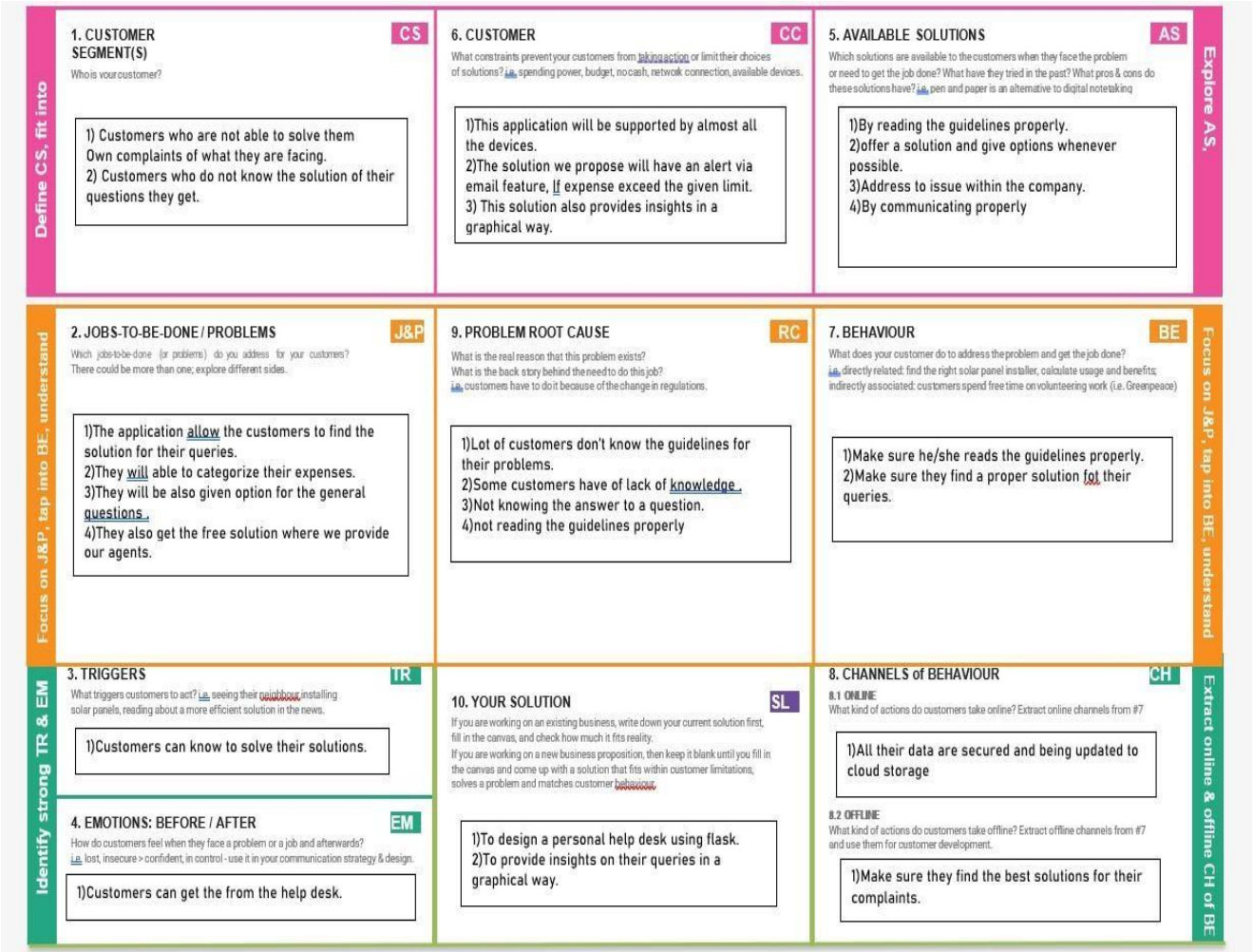


3.3 PROPOSED SOLUTIONS

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

		Closure, status Shown to the Customer, and Backup data in case of failures
04	Social Impact / Customer Satisfaction	Customer Satisfaction, Customer can track their status and Easy agent communication
05	Business Model (Revenue Model)	Key Partners are Third-party applications, agents, and customers. ● Activities held as Customer Service, System Maintenance. ● Key Resources support Engineers, Multi-channel. ● Customer Relationship have 24/7 Email Support, Knowledge-based channel. Cost Structure expresses Cloud Platform, Offices
06	Scalability of the Solution	The real goal of scaling customer service is providing an environment that will allow your customer service specialists to be as efficient as possible. An environment where they will be able to spend less time on grunt work and more time on actually resolving critical customer issues

3.4 PROBLEM SOLUTION FIT



4.1FUNCTIONAL REQUIREMNETS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
1	User Registration	Registration through Form Registration through Gmail Registration through Linked IN
2	User Confirmation	Confirmation via Email Confirmation via OTP
3	Defining problem	Type what is the problem.
4	Allocating agents	According to the problem agent will be allocated.
5	Analyzing problem	Problem and its requirements are analyzed by the agents.
6	Tracking problem solution	Track what is the condition of the problem solution through credentials.
7	Solving problem	Agents solve the problem and inform to user through mail.
8	Customer feedback	User can send feedback through credentials.

4.1 NON FUNCTIONAL REQUIREMNETS

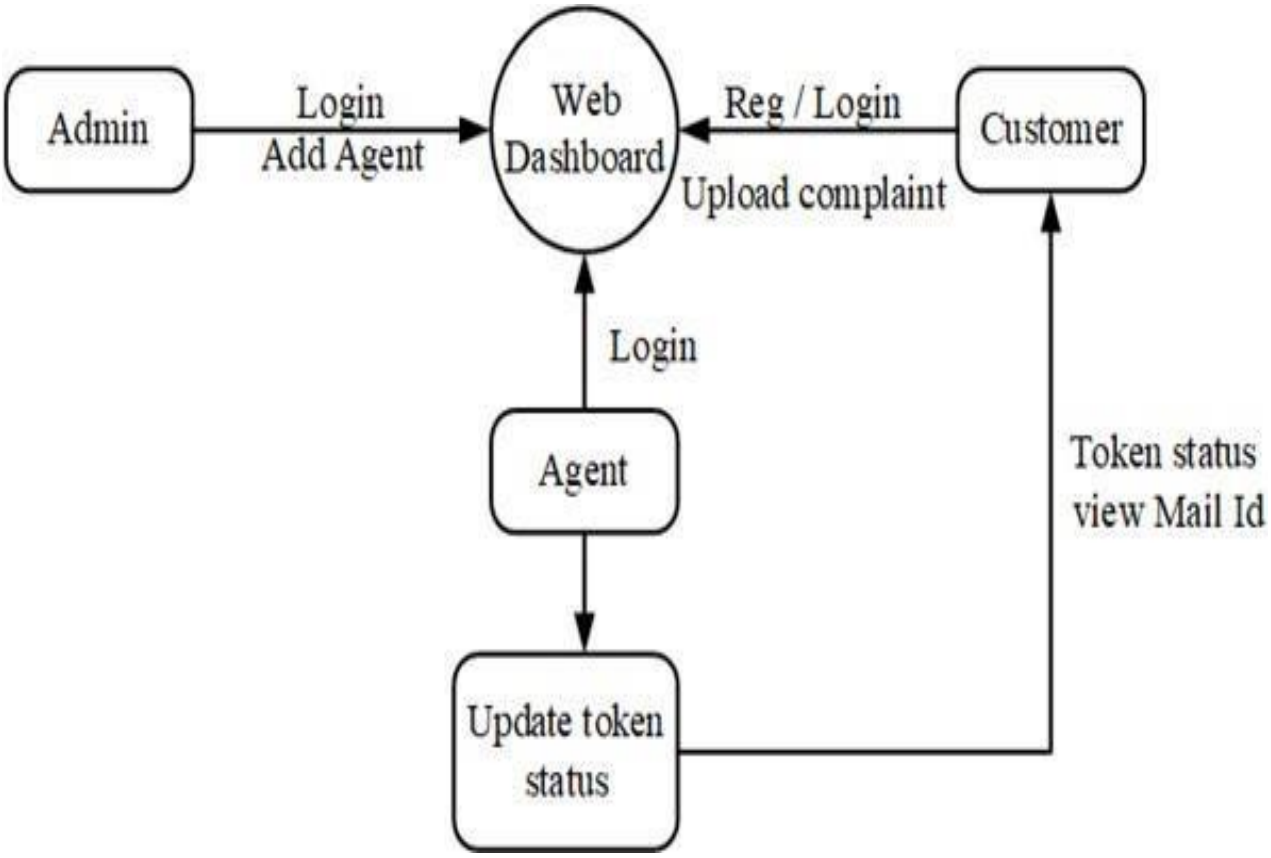
FR No.	Non-Functional Requirement	Description
1	**Usability**	The error rate of users submitting theirproblem details at the ticket mustn't exceed 10 percent.
2	**Security**	Assures All the data inside the system or inthe part will be protected against the malware attack or unauthorized access.
3	**Reliability**	The system must perform without failure in95percent of use cases during a month.
4	**Performance**	The landing page supporting 3,000 users per hour must provide 5 second or less response time in a Chrome desktop browser, including the rendering of text and images.
5	**Availability**	This must be available to US users99.98percent of the time every month during business hours IST.
6	**Scalability**	The system must be scalable enough tosupport1,00,000 visits at the same timewhile maintaining optimal performance.

PROJECT DESIGN

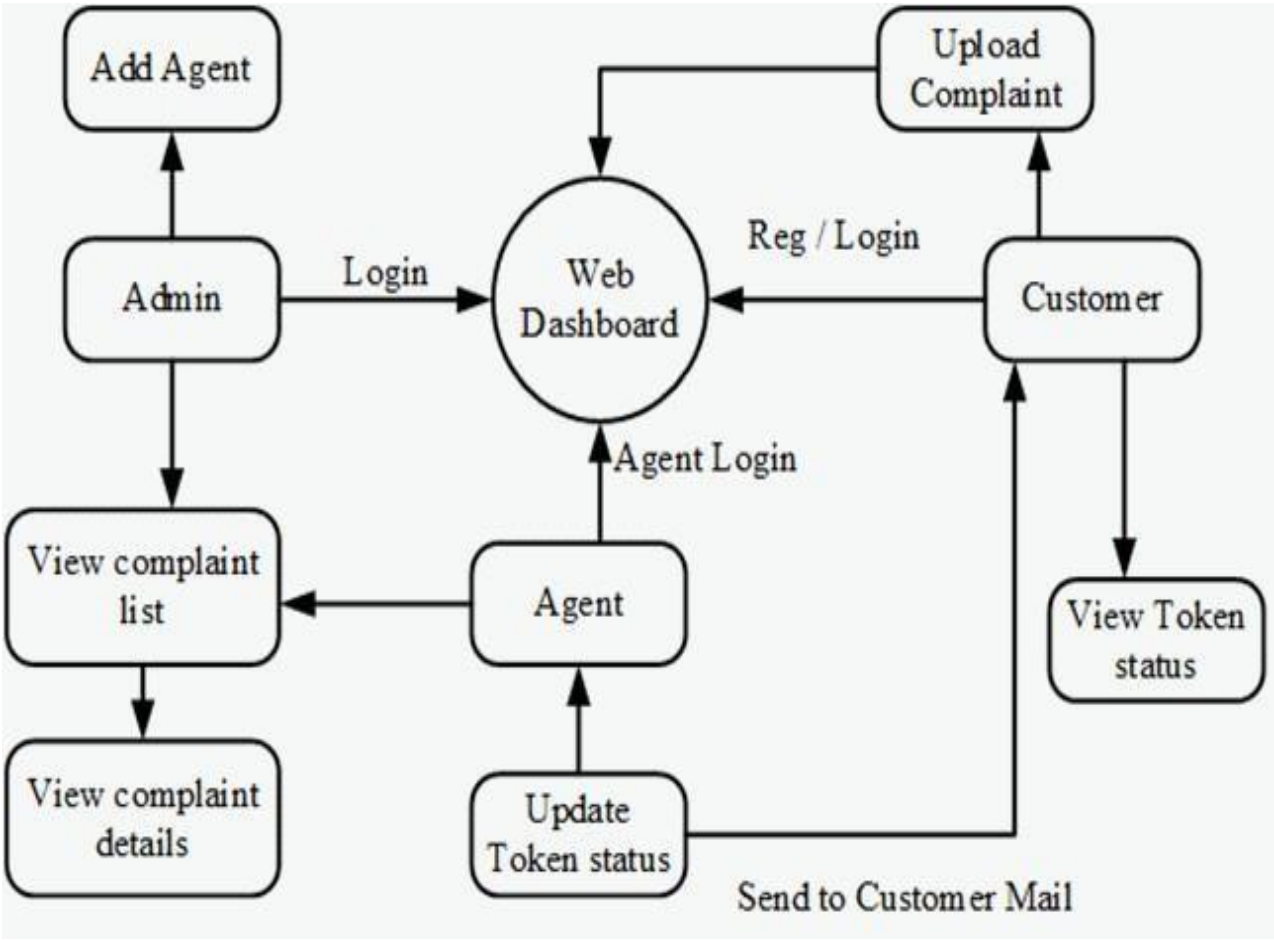
5.1 DATAFLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

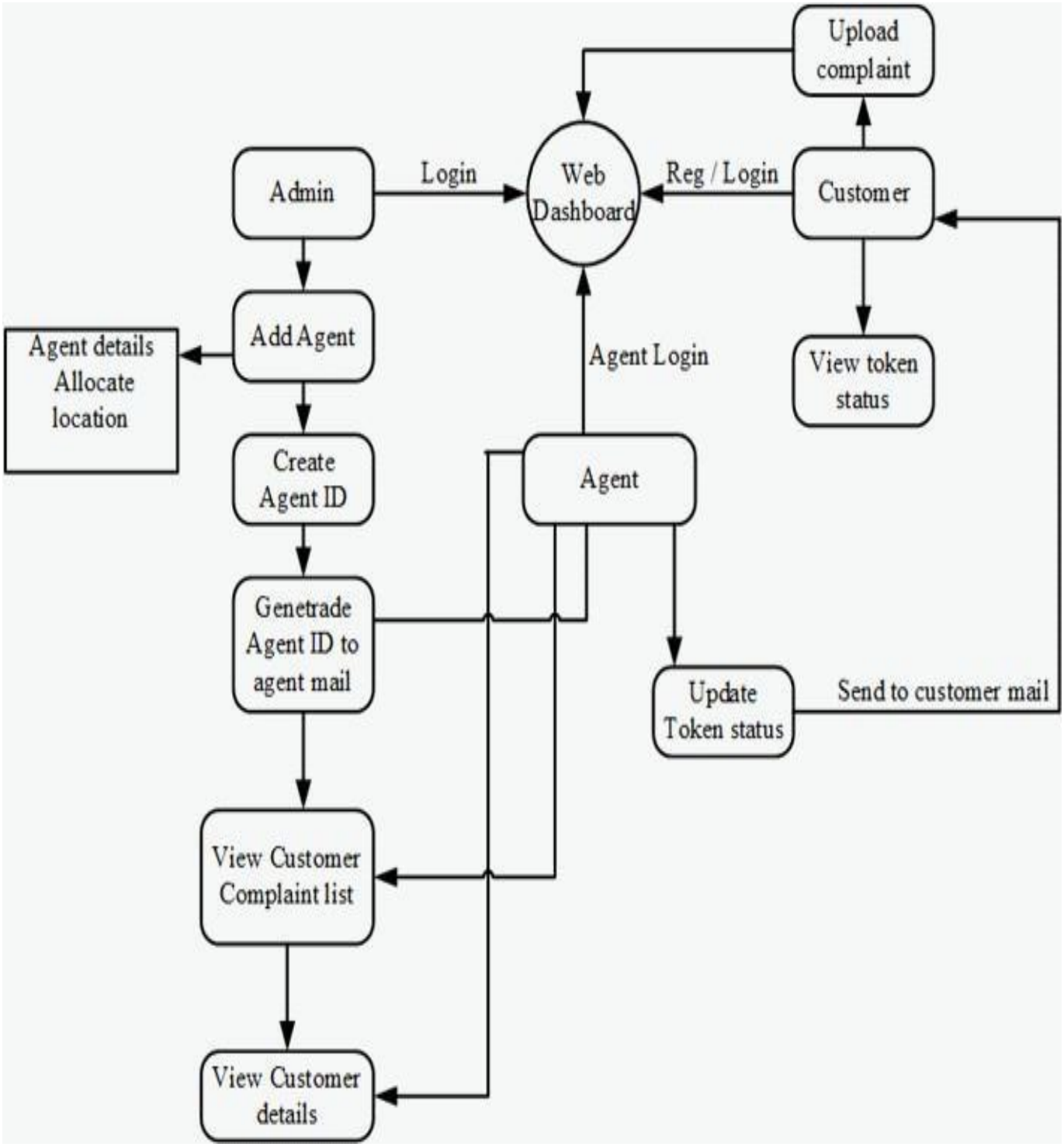
Data flow Diagram Level: 0



Data Flow Diagram Level: 1



Data flow Diagram Level: 2



5.2 SOLUTION AND TECHNICAL ARCHITECTURE

Technical Architecture:

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

Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)
6. Hire the Right Employees.
7. Set Goals for Customer Service.
8. Train on Service Skills.
9. Hold People Accountable.
10. Reward and Recognize Good Service

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service
8	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the application	Aadhar API, etc.

10	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
11	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Technology used
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

5.3USER STORIES

ype	Functional Requirements(EPIC)	User Story Number	User Story/Task	Acceptance criteria	Priority	Release
ner	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 tothe 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 placemy ordered 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 resetmy password by this Option in case I forgot my old password.	I get access to my accountagain	Medium	Sprint-4
	Order details	USN-7	As a Customer, I can see the current stats of order.	I get a better understandi ng	Medium	Sprint-4
Veb	Login	USN-1	As an agent I can login tothe application by Entering Correct email and password.	I can access my account /Dashboard.	High	Sprint-3

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

PROJECT PLANNING (SPIRINT SCHEDULE):

Use the below template to create product backlog and sprint schedule

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 intothe website and go through the services available on the webpage	20	High	SUDEEP P DHEVA M HEMANT H E C HENDRY CHARLES G
Sprint 2	Admin panel	USN-2	The role of the admin is tocheck out the database about the availability and have a track of all the things that the users are going to service	20	High	DHEVA M HENDRY CHARLES G HEMANTH E C SUDEEP P
Sprint 3	Chat Bot	USN-3	The user can directly talkto Chatbot regarding the services. Get the recommendations basedon information provided by the user.	20	High	HEMANT H E C SUDEEP P HENDRY CHARLES G DHEVA M
Sprint4	Final delivery	USN-4	Container of applications using docker Kubernetes and deployment the application. Create the documentation and final submit the application	20	High	HENDRY CHARLES G DHEVA M HEMANTH E C SUDEEP P

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart:

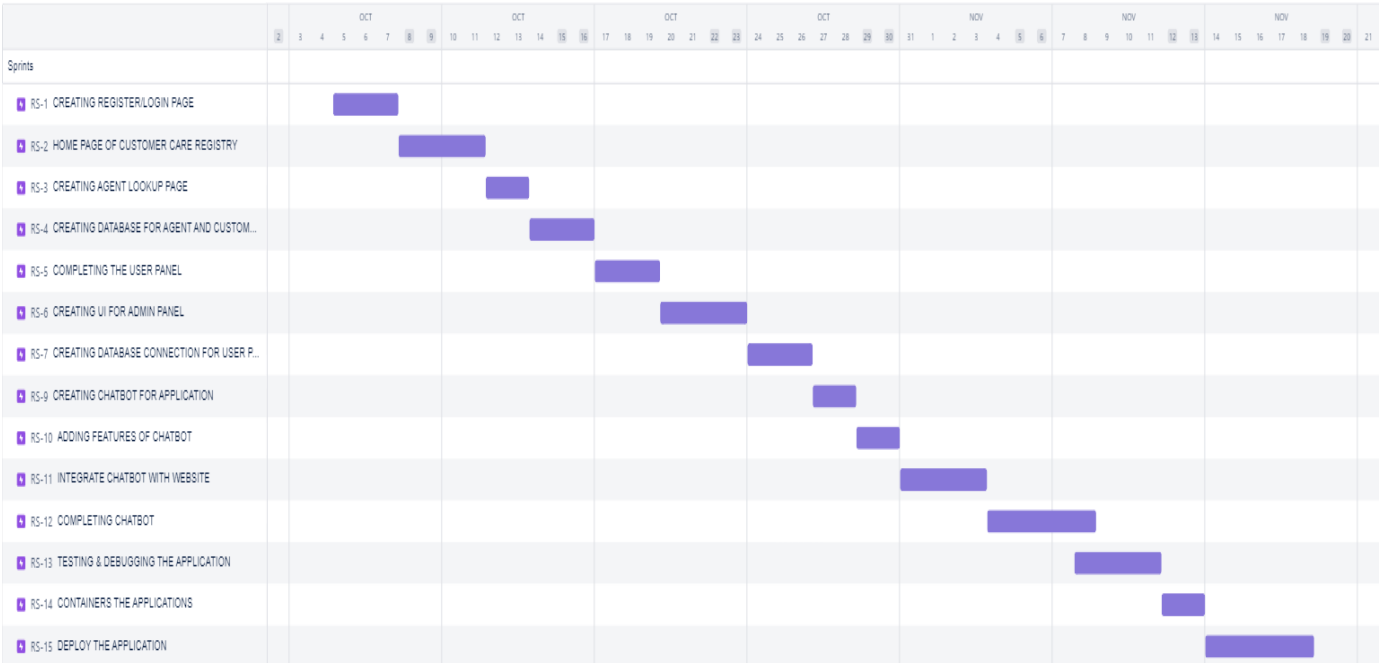
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	25 OCT 2022	30 OCT 2022	20	30 OCT 2022
Sprint-2	20	6 Days	31 OCT 2022	5 NOV 2022	20	5 NOV 2022
Sprint-3	20	6 Days	07 NOV 2022	13 NOV 2022	20	13 NOV 2022
Sprint-4	20	6 Days	11 NOV 2022	17 NOV 2022	20	11 NOV 2022

Velocity:

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

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

BURNDOWN CHART:



Reports from JIRA

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

CODING & SOLUTIONING

7.1 Feature -1

CSS

```
*{
        margin: 0;
        padding: 0;
        font-family: 'Roboto', sans-serif;
```

```
}


:root{
    --nav-color: #212121;
    --body-bg-color: white;
    --online-green-color: #00FF00;
    --white-color: white;
    --black-color: black;
    --link-color: blue;
    --line-grey-color: #D3D3D3;
    --input-bg-color: #efefef;
}


body{
    background-color: var(--body-bg-color);
    min-height: 100%;
    max-width: 100%;
    width: 100%;
}


/* ********************************************************************************** */
.login-section{
    width: 40%;
    margin: 100px auto;
}


.login-div{
    width: 70%;
    margin: 0 auto;
    padding: 15px 0;
    border: 1px solid var(--line-grey-color);
    border-radius: 5%;
}


.login-header{
    text-align: center;
}


.login-header h2{
    font-weight: 400;
}


.login-header p{
```

```
    margin: 10px 0;
    font-size: 16px;
}


.login-img{
    width: 100px;
    text-align: center;
}


.login-form{
    width: 70%;
    margin: 10px auto;
}


label{
    display: block;
    margin-top: 15px;
}


.login-form input{
    width: 100%;
    height: 50px;
    border: none;
    background-color: var(--input-bg-color);
    border-radius: 3px;
    font-size: 15px;
    text-indent: 10px;
}


.role-div{
    align-items: center;
    display: flex;
    margin-top: 10px;
}


.role-div > div{
    display: flex;
    margin-left: 10px;
    align-items: center;
    align-self: center;
}


.role-div > div > div{
```

```
    display: flex;
    align-items: center;
    align-self: center;
    margin-right: 15px;
    margin-top: -8px;
}


.role-div > div > div > input{
    width: 15px;
    height: 15px;
    margin-right: 8px;
    cursor: pointer;
}


.submit-btn{
    width: 100%;
    height: fit-content;
    padding: 15px;
    margin-top: 15px;
    background-color: var(--nav-color);
    color: var(--white-color);
    font-size: 16px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
}


.submit-btn:hover{
    background-color: var(--online-green-color);
    color: var(--black-color);
}


.login-form > div{
    margin-top: 15px;
}


.login-form > div > div{
    margin-top: 10px;
}


.links{
    color: var(--link-color);
    font-size: 15px;
}
```

```
/* ********************************************************************************** */
.register-section{
    width: 50%;
    margin: 120px auto;
    display: flex;
    padding: 15px;
    border: 1px solid var(--line-grey-color);
    border-radius: 10px;
}


.register-left{
    flex-basis: 60%;
    padding: 15px;
}


.register-left h1{
    font-size: 25px;
    font-weight: 400;
}


.reg-left-input-div{
    display: flex;
    width: 100%;
    margin-top: 10px;
}


.reg-left-input-div > div{
    height: 40px;
    flex-basis: 50%;
    height: fit-content;
}


.reg-left-input-div > div > input, .email-reg > input{
    width: 80%;
    height: 45px;
    margin-top: 5px;
    text-indent: 8px;
    font-size: 15px;
    background-color: var(--input-bg-color);
    border: none;
}
```

```
.register-left > .email-reg{
    margin: 25px 0;
}


.email-reg > p, .register-left > p{
    margin: 5px 0;
    font-size: 13px;
    color: var(--nav-color);
}




.show-pass-div{
    display: flex;
    margin: 10px 0;
}


.show-pass-div > input{
    width: 30px;
    margin-right: 5px;
    cursor: pointer;
}


.show-pass-div > p{
    font-size: 16px;
}


.sign-instead-div{
    width: 95%;
    margin: 10px auto;
    display: flex;
    justify-content: space-between;
}


.submit-1{
    width: 100px;
}


.register-right{
    flex-basis: 40%;
    align-self: center;
    text-align: center;
}
```

```
.register-right > img{
    width: 250px;
    margin-top: -20px;
}


.register-right > h1{
    font-size: 17px;
    font-weight: 400;
    margin-top: -20px;
}


.cust-link{
    color: var(--link-color);
    text-decoration: none;
    font-size: 16px;
    font-style: italic;
    align-self: center;
}


/* ********************************************************************************** */
.dashboard-div{
    width: 65%;
    margin: 0 auto;
    min-height: 100vh;
    position: relative;
}


nav{
    width: 100%;
    background-color: var(--nav-color);
}


.dash-nav{
    background-color: var(--nav-color);
    padding: 20px 40px;
    display: flex;
    justify-content: space-between;
    cursor: pointer;
    height: 50px;
}


.img-in-nav{
```

```
    width: 50px;
}


.dash-img-text{
    display: flex;
    justify-content: space-between;
    align-items: center;
}


.dash-img-text > h3{
    color: white;
    font-size: 20px;
    margin-left: 8px;
    font-weight: 400;
}


.online-div{
    position: relative;
}


.settings-menu{
    position: absolute;
    width: 200px;
    height: fit-content;
    background-color: #fff;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.4);
    border-radius: 4px;
    top: 100px;
    right: 0%;
    padding: 15px;
    z-index: 3;
}


.logout-btn{
    width: 100%;
    margin: 10px auto;
    background-color: var(--nav-color);
    color: white;
    font-size: 17px;
    padding: 10px 20px;
    border-radius: 8px;
    cursor: pointer;
    border: none;
}
```

```
.logout-btn:hover{
    background-color: red;
}


.online-div::after{
    content: '';
    width: 10px;
    height: 10px;
    border-radius: 50%;
    background: var(--online-green-color);
    border: 2px solid var(--white-color);
    position: absolute;
    top: 0;
    right: 0;
}


.dash-body{
    width: 100%;
    display: flex;
}


.dash-left{
    flex-basis: 30%;
    min-height: calc(100vh - 90px);
}


.dash-right{
    flex-basis: 70%;
    min-height: calc(100vh - 90px);
}


.links-left-div{
    margin: 20px;
    cursor: pointer;
}


.links-left-div > ul{
    list-style: none;
}


.links-left-div > ul li{
    padding: 15px 0;
```

```
    color: black;
}


.links-left-div > ul li:hover{
    background-color: #f2f2f2;
    border-top-right-radius: 12px;
    border-bottom-right-radius: 12px;
    border-top-left-radius: 5px;
    border-bottom-left-radius: 5px;
}


.active{
    background-color: var(--line-grey-color);
    border-top-right-radius: 12px;
    border-bottom-right-radius: 12px;
    border-top-left-radius: 5px;
    border-bottom-left-radius: 5px;
}


.links-left-div > ul > li > a{
    text-decoration: none;
    color: black;
}


.link-items{
    display: flex;
    justify-content: flex-start;
    align-items: center;
}


.link-items > div{
    flex-basis: 13%;
    text-align: center;
    margin-right: 10px;
}


.link-items > div > i{
    font-size: 25px;
}


.link-items > h3{
    font-size: 18px;
    font-weight: 300;
```

```
}


.profile-div{
    margin: 50px;
}


.prof-table {
    width: 70%;
}


.prof-table, .prof-table td{
    margin-top: 20px;
    border: 1px solid black;
    padding: 15px;
    border-collapse: collapse;
    font-size: 18px;
}


.prof-table tr:nth-child(odd){
    background-color: #f2f2f2;
}
```

7.2 Feature-2

```
from flask
import Blueprint,
render_template
```

```
admin = Blueprint("admin", __name__)


@admin.route('/admin/tickets')
def tickets():
    return render_template('admin tickets.html', id = 0)


@admin.route('/admin/agents')
def agents():
    return render_template('admin agents.html', id = 1)


@admin.route('/admin/accept')
def accept():
    return render_template('admin acc agent.html', id = 2)


@admin.route('/admin/about')
def about():
    return render_template('admin about.html', id = 3)


@admin.route('/admin/support')
def support():
    return render_template('admin support.html', id = 4)
```

```
from flask import
Blueprint,
render_template,
session
```

```
from flask_login import login_required


cust = Blueprint("customer", __name__)


@cust.route('/customer/')
@login_required
def profile():
    from .views import customer


    return render_template('cust profile.html', customer = customer, id = 0)


@cust.route('/customer/tickets')
@login_required
def tickets():
    return render_template('cust tickets.html', id = 1)


@cust.route('/customer/new')
@login_required
def new():
    return render_template('cust new ticket.html', id = 2)


@cust.route('/customer/change')
@login_required
def change():
    return render_template('cust change.html', id = 3)
```

```
@cust.route('/customer/about')
@login_required
def about():
    return render_template('cust about.html', id = 4)


@cust.route('/customer/support')
@login_required
def support():
    return render_template('cust support.html', id = 5)
```

```
from flask import Flask, session
                                    from flask_login import LoginManager


                                    def create_app():
                                        app = Flask(__name__)
                                        app.config['SECRET_KEY'] = "PHqtYfAN2v"


                                        # registering the blue prints with the app
                                        from .views import views
                                        app.register_blueprint(views, appendix='/')


                                        from .cust import cust
                                        app.register_blueprint(cust, appendix='/customer/')


                                        from .admin import admin
```

```
app.register_blueprint(admin, appendix='/admin/')


# setting up the login manager
login_manager = LoginManager()
login_manager.login_view = "blue_print.login"
login_manager.init_app(app)


@login_manager.user_loader
def load_user(id):
    if session.get('LOGGED_IN_AS') is not None:
        if session['LOGGED_IN_AS'] == "CUSTOMER":
            from .views import customer
            return customer


    else:
        return None


return
```

```
from flask_login import
UserMixin


class Customer(UserMixin):

    def set(self, uuid, first_name, last_name, email,
password, date):

        self.uuid = uuid

        self.first_name = first_name

        self.last_name = last_name
```

```
        self.email = email
        self.password = password
        self.date = date

    def get_id(self):
        return (self.uuid)

import hashlib
import re
from flask_login import login_required, login_user, logout_user
import ibm_db
import uuid
from datetime import date
from .model import Customer

views = Blueprint("blue_print", __name__)
email_regex = r"\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b"
pass_regex = r"^[A-Za-z0-9_-]*$"

customer = Customer()

conn = ibm_db.connect('DATABASE=bludb;HOSTNAME=2f3279a5-73d1-4859-88f0-
a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=30756;SECURITY=
SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=tdn81266;PWD=7LY8okjAouJf3LoO',
'', '')
```

```
@views.route('/logout')
@login_required
def logout():
    session.pop('LOGGED_IN_AS')
    logout_user()

    return redirect(url_for('blue_print.login'))

@views.route('/', methods = ['GET', 'POST'])
def login():
    # if method is POST
    if request.method == 'POST':
        # getting the data entered by the user
        email = request.form.get('email')
        password = request.form.get('password')
        role = request.form.get('role-check')

        msg = ""
        to_show = False

        # validating the inputs entered by the user
        if(not (re.fullmatch(email_regex, email))):
            msg = "Enter a valid email"
            to_show = True
```

```
elif (len(password) < 8):
    msg = "Password must be atleast 8 characters long!"
    to_show = True

# Admin login
if email == "admin.ccr@gmail.com":
    if password == "admin.ccr@2022":
        return redirect('/admin/tickets')

    else:
        to_show = True
        password = ""
        msg = "Invalid password!"

# Customer or Agent
else:
    if to_show:
        # there is something fishy with the user's inputs
        password = ""

    elif (not to_show):
        # the user's inputs are valid
        # checking if the login credentials are valid
        if role == "Customer":
            # checking if the entry of the mail entered is present in the database
```

```
mail_check_query = "SELECT * FROM customer WHERE email = ?"
stmt = ibm_db.prepare(conn, mail_check_query)
ibm_db.bind_param(stmt, 1, email)
ibm_db.execute(stmt)

account = ibm_db.fetch_assoc(stmt)

if account:
    # valid customer
    # i.e, mail is present in the database

    # checking if the customer entered a valid password now
    # encrypting the entered password
    passcode = str(hashlib.sha256(password.encode()).hexdigest())

    # now checking if the encrypted string is same as that of the one in database
    if (account['PASSCODE'] == passcode):
        msg = "Valid Login"
        to_show = True

        # creating a customer object
        customer.set(
            account['CUST_ID'],
            account['FIRST_NAME'],
            account['LAST_NAME'],
```

```
            account['EMAIL'],
            account['PASSCODE'],
            account['DATE_JOINED']
        )

        session.permanent = False
        session['LOGGED_IN_AS'] = "CUSTOMER"
        login_user(customer, remember=True)

        return redirect('/customer/')

    else:
        # customer entered invalid password
        msg = "Invalid password"
        password = ""
        to_show = True

else:
    # invalid customer
    # i.e, entered mail is not present in the database
    msg = "User does not exist"
    email = ""
    password = ""
    to_show = True
```

```
            else:
                # user is an Agent
                print("hello")

        return render_template(
            'login.html',
            to_show = to_show,
            message = msg,
            email = email,
            password = password
        )

    return render_template('login.html')

@views.route('/register', methods = ['GET', 'POST'])
def register():
    # if method is POST
    if request.method == 'POST':
        # getting all the data entered by the user
        first_name = request.form.get('first_name')
        last_name = request.form.get('last_name')
        email = request.form.get('email')
        password = request.form.get('password')
        confirm_password = request.form.get('confirm_password')
        role = request.form.get('role-check')
```

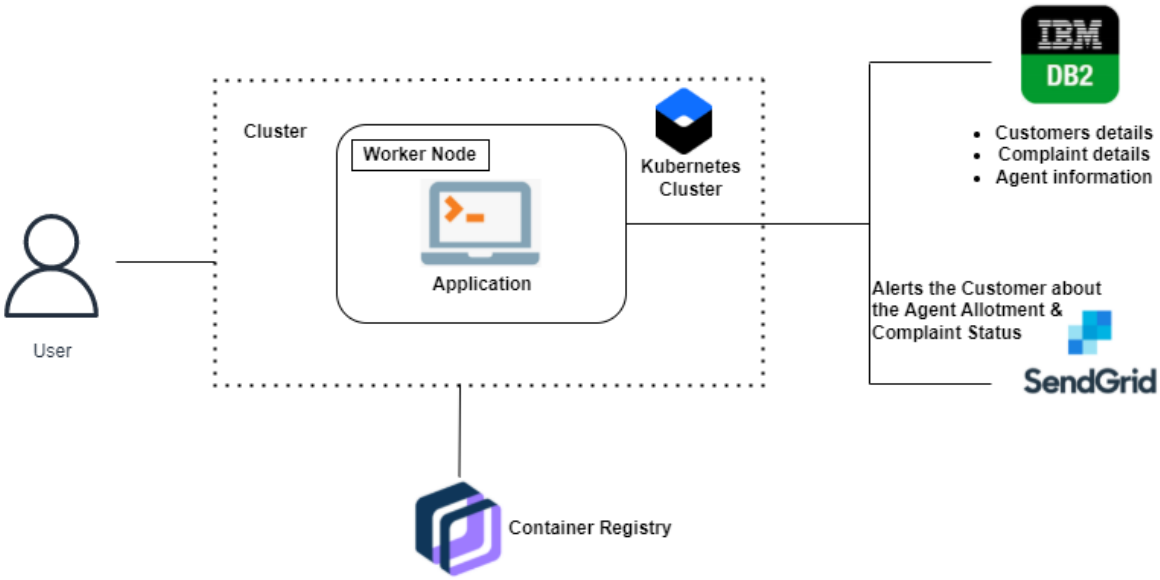
```
msg = ""

to_show = False


# validating the inputs

if len(first_name) < 2:

    msg = "First Name must be atleast 5 characters long!"

    to_show = True


elif len(last_name) < 2:

    msg = "Last Name must be atleast 5 characters long!"

    to_show = True


elif(not (re.fullmatch(email_regex, email))):

    msg = "Please enter valid email"

    to_show = True


elif((len(password) < 8) or (len(confirm_password) < 8)):

    msg = "Password must be atleast 8 characters long!"

    to_show = True


elif (password != confirm_password):

    msg = "Passwords do not match"

    to_show = True
```

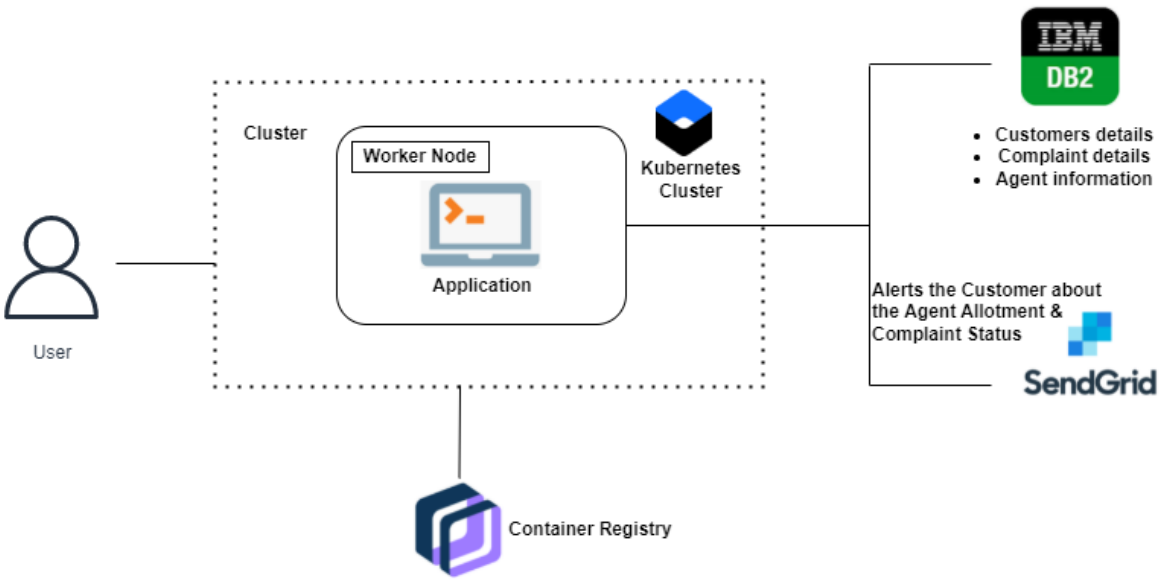
```
elif (not (re.fullmatch(pass_regex, password))):
    msg = "Enter valid password"
    to_show = True

if to_show:
    # there is something fishy with the inputs
    password = confirm_password = ""

# by here the inputs are validated, because to_show is False
# registering the user / agent with the database
elif (not to_show):
    if role == "Customer":
        # the user is a Customer
        # checking whether the user with the same email already there
        check_mail_query = "SELECT * FROM customer WHERE email = ?"
        stmt = ibm_db.prepare(conn, check_mail_query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        if account:
            # user already exists
            msg = "Email already exists!"
            to_show = True
```

```
else:
    # new customer
    # adding the customer details to the detabase
    user_insert_query ="'INSERT INTO customer
            (cust_id, first_name, last_name, email, passcode, date_joined)
            VALUES (?, ?, ?, ?, ?, ?)"'

    # creating a UUID for the customer
    user_uuid = str(uuid.uuid4())

    # encrypting the customer's password using SHA-256
    passcode = str(hashlib.sha256(password.encode()).hexdigest())
    date_joined = date.today()

    stmt = ibm_db.prepare(conn, user_insert_query)
    ibm_db.bind_param(stmt, 1, user_uuid)
    ibm_db.bind_param(stmt, 2, first_name)
    ibm_db.bind_param(stmt, 3, last_name)
    ibm_db.bind_param(stmt, 4, email)
    ibm_db.bind_param(stmt, 5, passcode)
    ibm_db.bind_param(stmt, 6, date_joined)

    ibm_db.execute(stmt)
```

```
                # redirecting the customer to the login page
                msg = "Account created. Please Login!"
                to_show = True

                return render_template('login.html', message = msg, to_show = to_show)

        else:
            # the role is Agent
            # can be done in Sprint 2/3
            print("Sprint 2/3")

    return render_template(
        'register.html',
        to_show = to_show,
        message = msg,
        first_name = first_name,
        last_name = last_name,
        email = email,
        password = password,
        confirm_password = confirm_password,
        role = role
    )

return render_template('register.html')
```

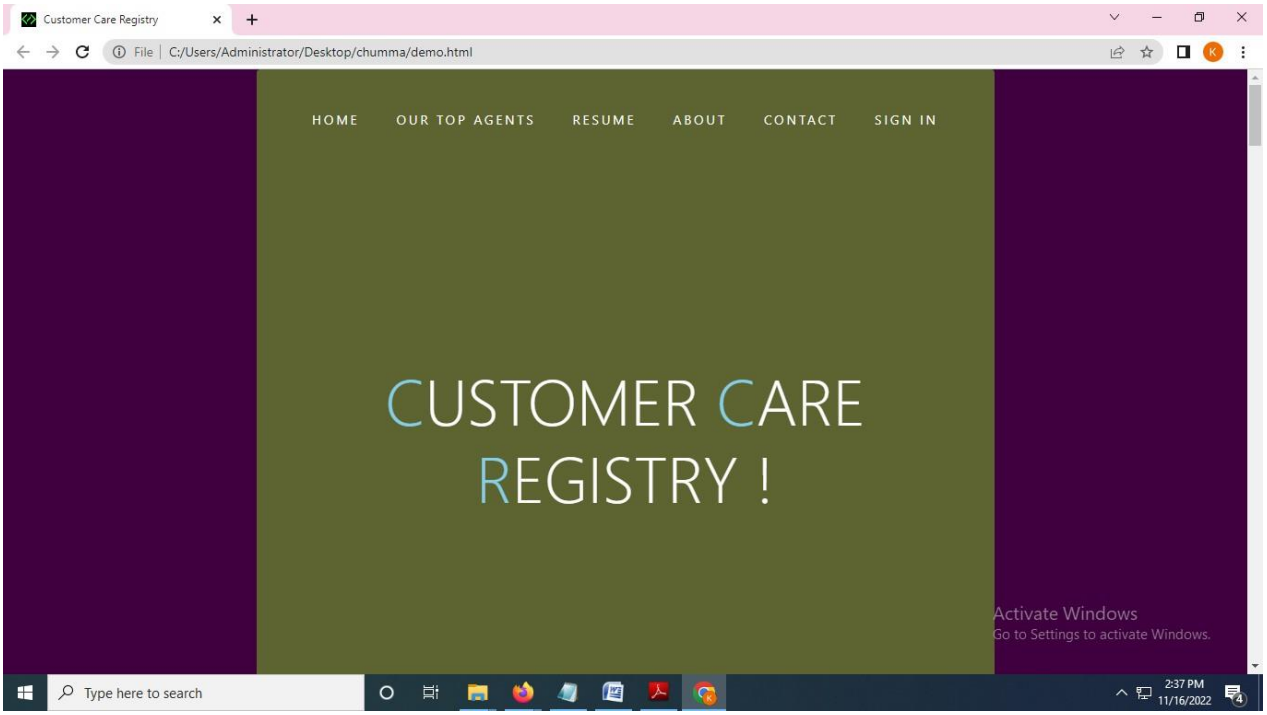
7.3 Database Schema

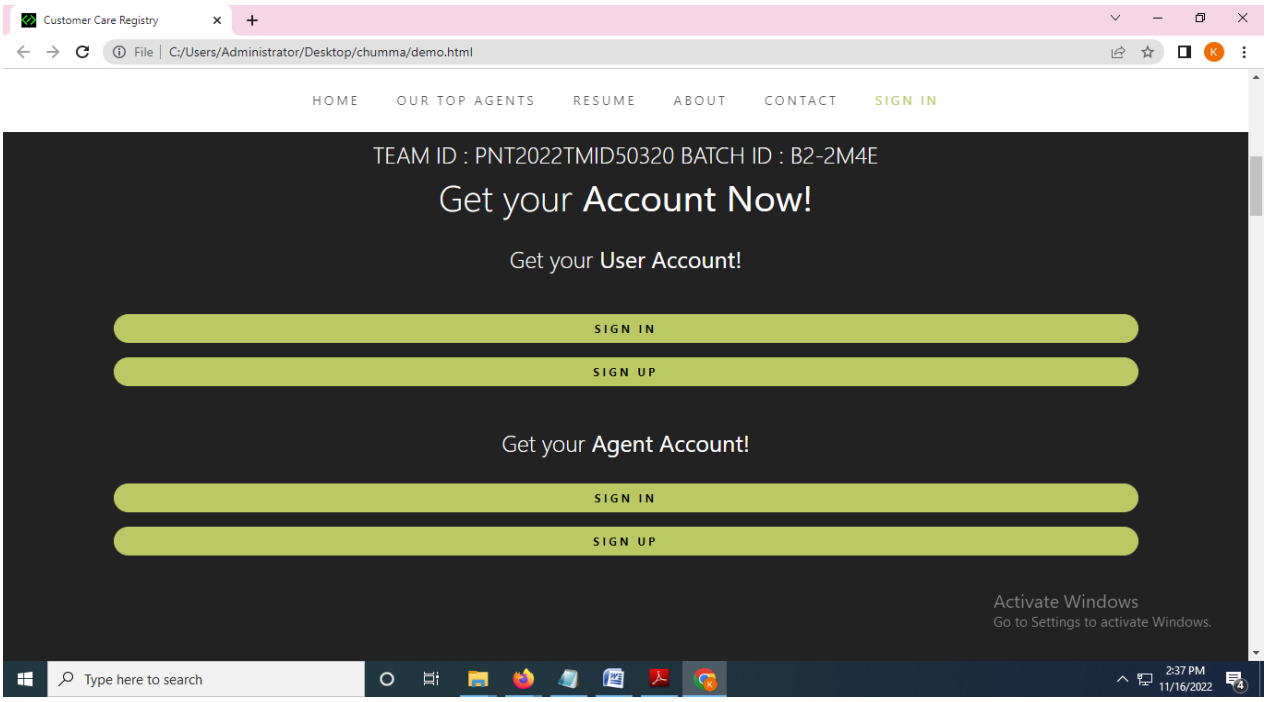


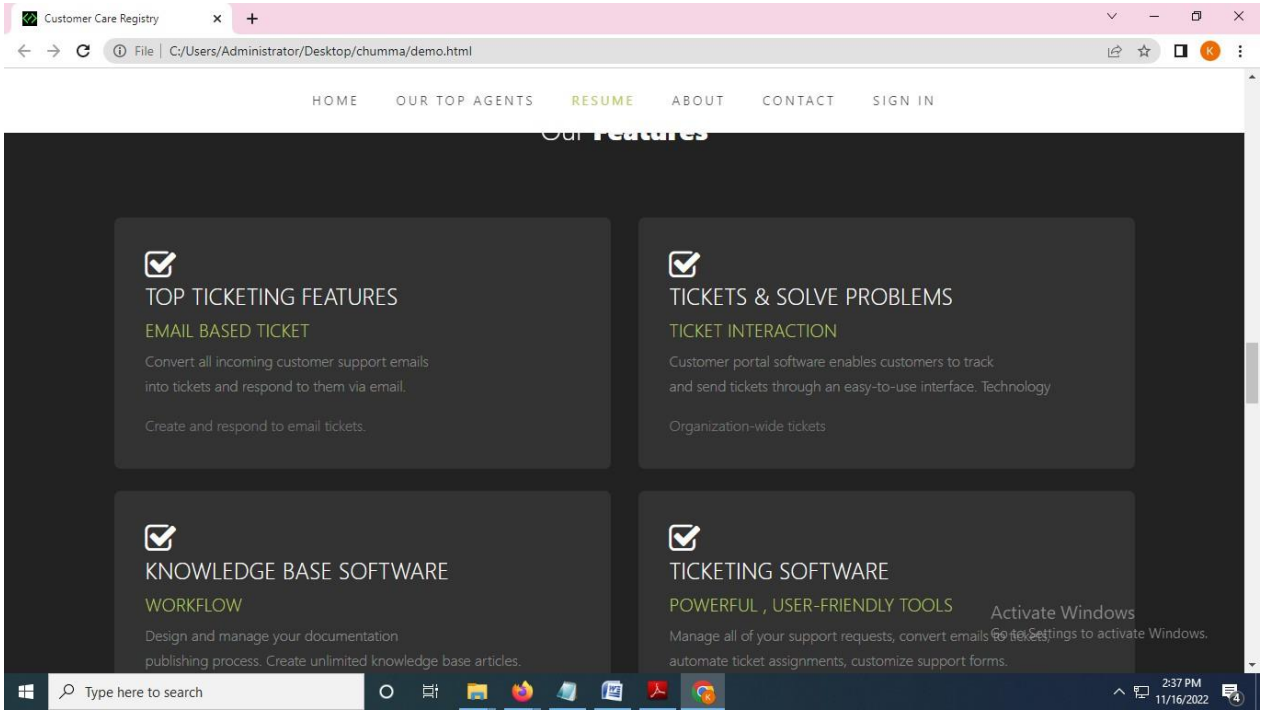


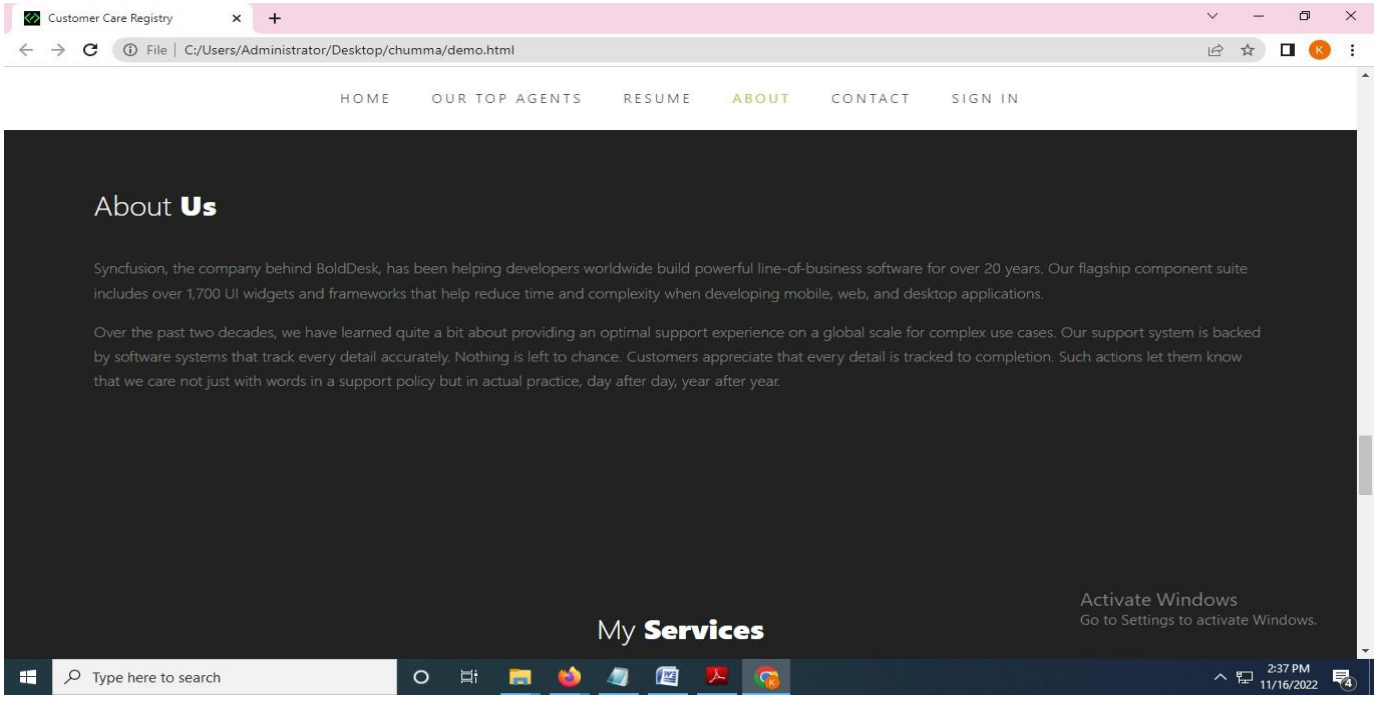
TESTING

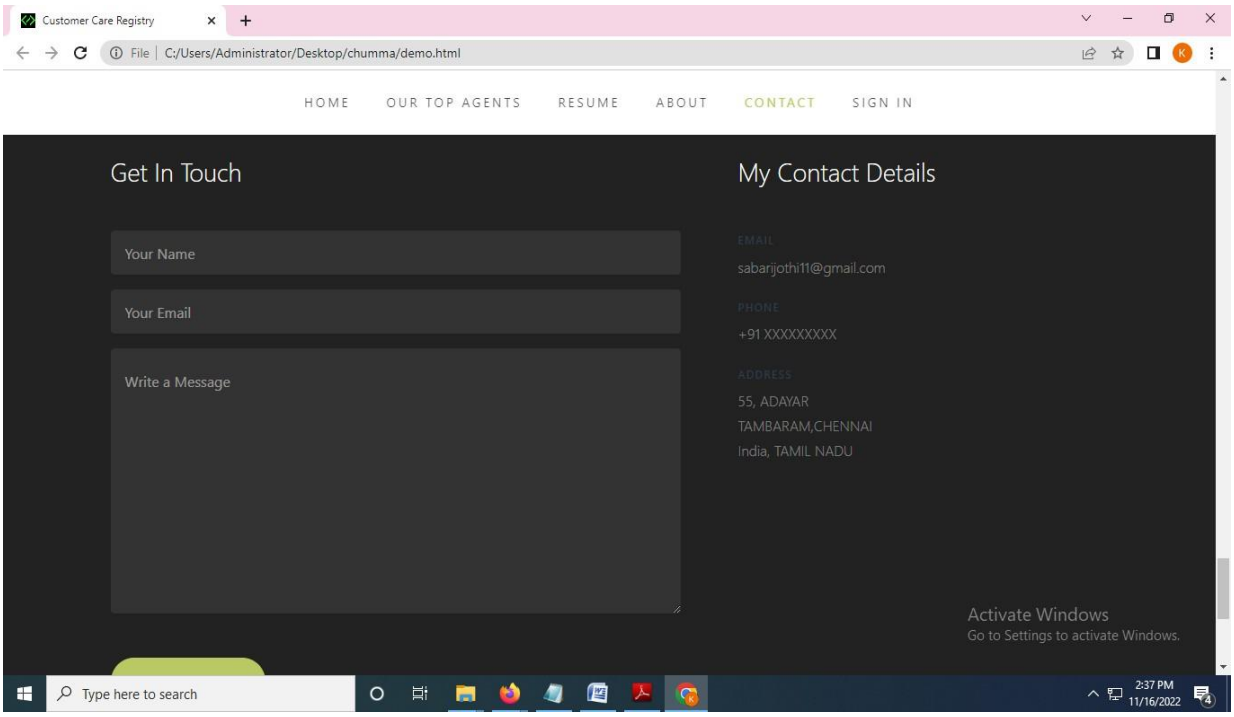
8.1 TEST CASES

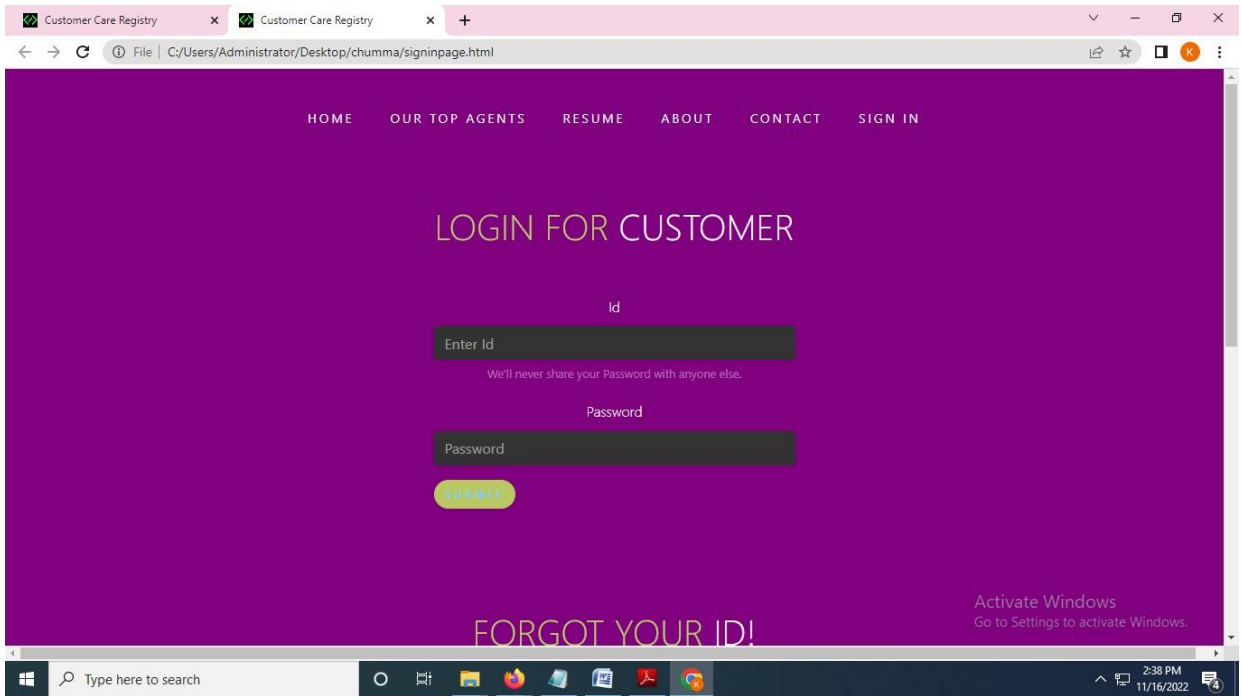


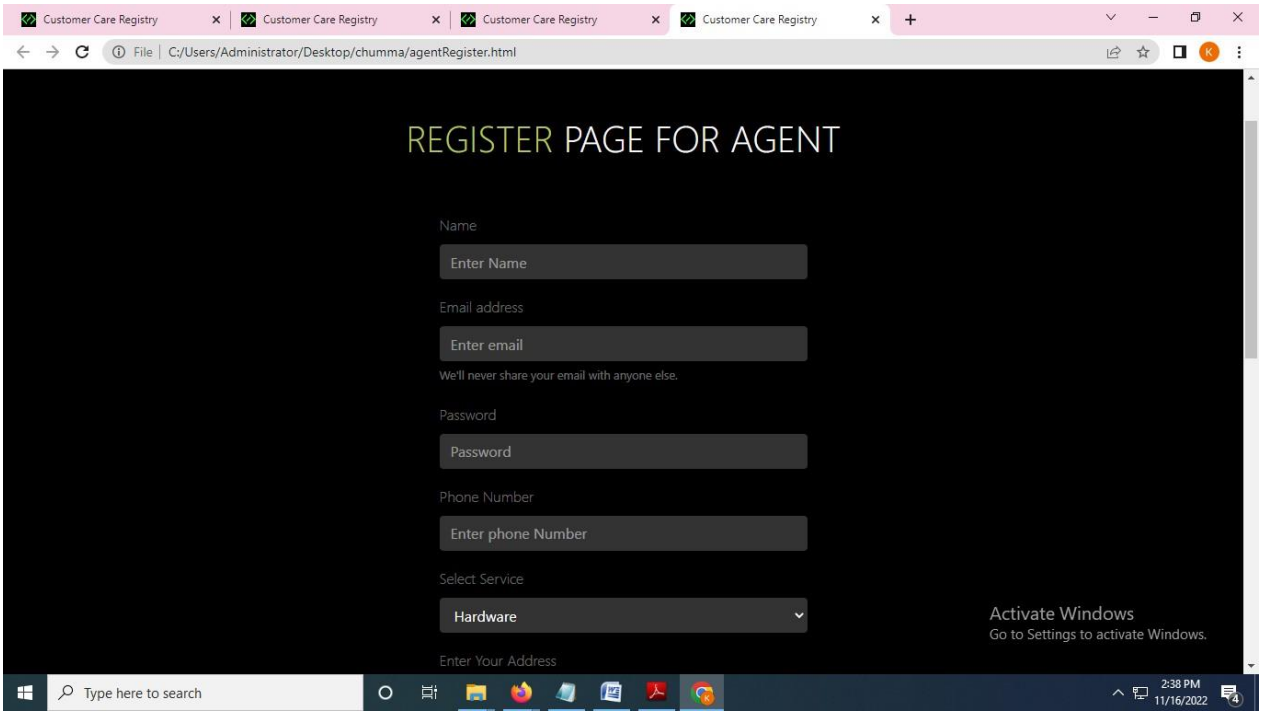


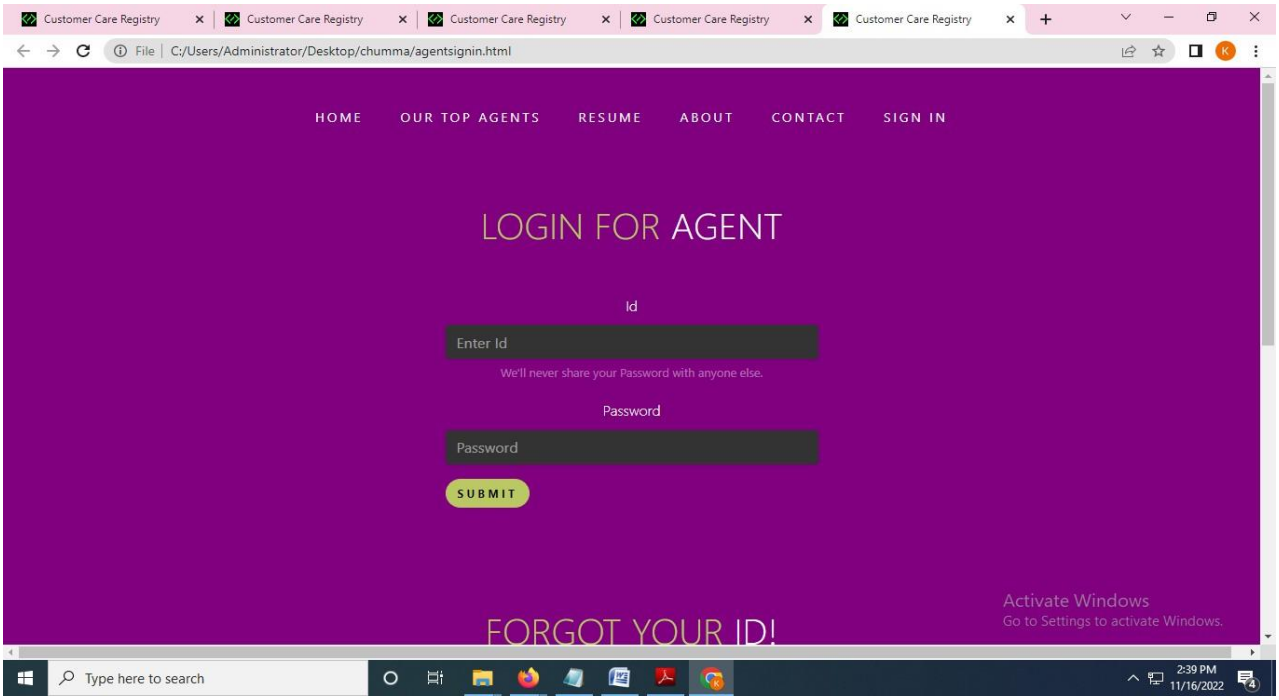

Customer Care Registry
File | C:/Users/Administrator/Desktop/chumma/demo.html
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
TOP TICKETING FEATURES
EMAIL BASED TICKET
Convert all incoming customer support emails into tickets and respond to them via email.
Create and respond to email tickets.
TICKETS & SOLVE PROBLEMS
TICKET INTERACTION
Customer portal software enables customers to track and send tickets through an easy-to-use interface. Technology
Organization-wide tickets
KNOWLEDGE BASE SOFTWARE
WORKFLOW
Design and manage your documentation publishing process. Create unlimited knowledge base articles.
TICKETING SOFTWARE
POWERFUL , USER-FRIENDLY TOOLS
Manage all of your support requests, convert emails to tickets, automate ticket assignments, customize support forms.
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:37 PM
11/16/2022


Customer Care Registry
File | C:/Users/Administrator/Desktop/chumma/demo.html
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
About Us
Syncfusion, the company behind BoldDesk, has been helping developers worldwide build powerful line-of-business software for over 20 years. Our flagship component suite includes over 1,700 UI widgets and frameworks that help reduce time and complexity when developing mobile, web, and desktop applications.
Over the past two decades, we have learned quite a bit about providing an optimal support experience on a global scale for complex use cases. Our support system is backed by software systems that track every detail accurately. Nothing is left to chance. Customers appreciate that every detail is tracked to completion. Such actions let them know that we care not just with words in a support policy but in actual practice, day after day, year after year.
My Services
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:37 PM
11/16/2022


Customer Care Registry
File | C:/Users/Administrator/Desktop/chumma/demo.html
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
Get In Touch
Your Name
Your Email
Write a Message
My Contact Details
EMAIL
sabarijothi11@gmail.com
PHONE
+91 XXXXXXXXX
ADDRESS
55, ADAYAR
TAMBARAM,CHENNAI
India, TAMIL NADU
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:37 PM
11/16/2022


Customer Care Registry
Customer Care Registry
File | C:/Users/Administrator/Desktop/chumma/signinpage.html
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
LOGIN FOR CUSTOMER
Id
Enter Id
We'll never share your Password with anyone else.
Password
Password
SUBMIT
FORGOT YOUR ID!
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:38 PM
11/16/2022


Customer Care Registry
File | C:/Users/Administrator/Desktop/chumma/agentRegister.html
REGISTER PAGE FOR AGENT
Name
Enter Name
Email address
Enter email
We'll never share your email with anyone else.
Password
Password
Phone Number
Enter phone Number
Select Service
Hardware
Enter Your Address
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:38 PM
11/16/2022


Customer Care Registry
File | C:/Users/Administrator/Desktop/chumma/agentsignin.html
HOME
OUR TOP AGENTS
RESUME
ABOUT
CONTACT
SIGN IN
LOGIN FOR AGENT
Id
Enter Id
We'll never share your Password with anyone else.
Password
Password
SUBMIT
FORGOT YOUR ID!
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:39 PM
11/16/2022

RESULT
S

9.1 PERFORMANCE METRICS

The Customer Feedback metric:

The most important metric for your contact center and the broader business is customer feedback. From the business perspective, you want to know if your customersare going to stay or leave and if they will recommend your business to their friends and colleagues. If you are the customer you want the right or incorrect processes improved so that you get the best experience. From the perspective of the employee, you want processes that will help deliver good service to customers.

The Service Efficiency Metric:

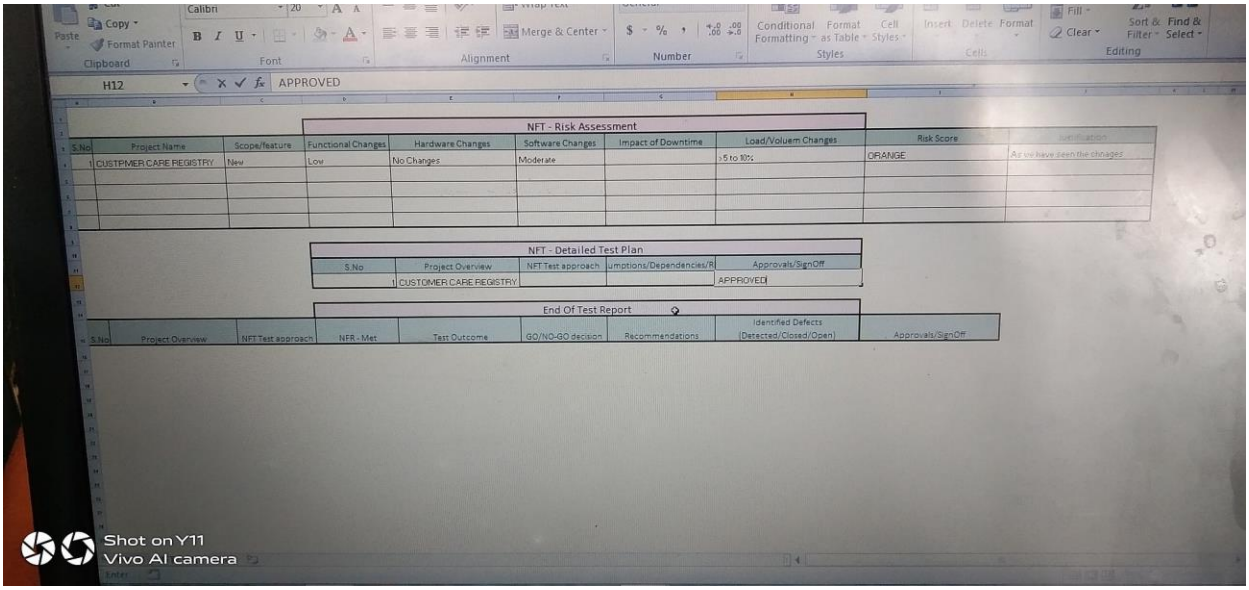
The next thing is service efficiency. The main thing is to measure service efficiency at a macro level rather than at an individual level to avoid bad behaviors. The most common story is the service agent that passes customers to other agents to improve their average handling time. As a service organization, you want to give your employees the tools that they need to provide the best experience for customers.

Quality, Consistency and Compliance:

The next of the 4 key metrics for customer service focuses on the effective running of a contact center or customer service organization is quality, consistency and compliance.If you're a customer you want to ensure that you get the same experience each time they come into contact with the service team. As a customer in addition to great service, you want to ensure that processes are in place to protect and safeguard sensitive information.

Employee Engagement:

The final metric that we believe to be important is employee engagement. How engaged is the team that is interacting with customers? We have found that engaged employees provide better service and help deliver increased customer satisfaction. As a business, you want to ensure that employees are being managed properly and that employees.


Paste
Copy
Format Painter
Clipboard
Calibri
20
Font
Merge & Center
Alignment
Number
Conditional Formatting
Format as Table
Cell Styles
Styles
Insert
Delete
Format
Cells
Fill
Clear
Sort & Filter
Find & Select
Editing
H12
APPROVED
NFT - Risk Assessment
S.No
Project Name
Scope/feature
Functional Changes
Hardware Changes
Software Changes
Impact of Downtime
Load/Voluem Changes
Risk Score
Justification
1
CUSTPMER CARE REGISTRY
New
Low
No Changes
Moderate
>5 to 10%
ORANGE
As we have seen the chnages
NFT - Detailed Test Plan
S.No
Project Overview
NFT Test approach
umptions/Dependencies/R
Approvals/SignOff
1
CUSTOMER CARE REGISTRY
APPROVED!
End Of Test Report
S.No
Project Overview
NFT Test approach
NFR - Met
Test Outcome
GO/NO-GO decision
Recommendations
Identified Defects (Detected/Closed/Open)
Approvals/SignOff
Shot on Y11
Vivo AI camera

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

1. Customer loyalty

Loyal customers have many benefits for businesses. 91% of customers say a positive customer service experience makes them more likely to make a further purchase. Also, investing in new customers is five times more expensive than retaining existing ones (source: Invest). Creating loyal customers through good customer service can therefore provide businesses with lucrative long-term relationships.

2. Increase profits

These long-term customer relationships established through customer service can help businesses become more profitable. Businesses can grow revenues between 4% and 8% above their market when they priorities better customer service experiences (source:Bain & Company). Creating a better customer service experience than those offered by competitors can help businesses to standout in their market place, and in turn make more sales.

3. Customer recommendations

Providing good customer service can create satisfied customers, who are then more likely to recommend the business to others. 94% of customers will recommend a company whose service they rate as “very good” (source: Qualtrics XM Institute). This isuseful, as 90% of customers are influenced by positive reviews when buying a product (source: Zendesk). Customers recommending a company through word of mouth or online reviews can improve the credibility of the business.

4. Increase conversion

Good customer service can help businesses turn leads into sales. 78% of customers say they have backed out of a purchase due to a poor customer experience (source: Glance). It is therefore safe to assume that providing good customer service will help toincrease customer confidence and in turn increase conversion.

5.Improve public image

Customer service can help businesses to improve the public perception of the brand, which can then provide protection if there is a slip up. 78% of customers will forgive a company for a mistake after receiving excellent service (source: Salesforce Research). Meanwhile, almost 90% of customers report trusting a company whose service they rate as “very good.” On the other hand, only 16% of those who give a “very poor” rating trust companies to the same degree(source: Qualtrics XM Institute). Creating positive customer experiences is vital in gaining customer trust and creating a strong public image.

CONCLUSION

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

FUTURE

SCOPE

The future of customer service increasingly will be driven by technology innovations. Ideally, these new technologies will improve customer and agent experiences, along with business metrics like revenue, operational costs and customer ratings. But businesses often miss the mark when they try to move too quickly with too much technology, ultimately resulting in consumer dissatisfaction rather than elation.

Customer expectations for what defines a good experience stay fairly consistent over time, but the approach to providing that experience changes. Meanwhile, advanced technologies, largely driven by artificial intelligence, analytics and automation, arm companies with new techniques for driving customer satisfaction and loyalty.

How are customer expectations changing?

Over the years, customer expectations generally haven't changed. Customers want to be served quickly and completely on the first try. If they're speaking to a human agent, they want a friendly, knowledgeable interaction -- the goal being to resolve the customer's problem or answer their question quickly and easily.
Drilling down, however, customer expectations are influenced by the changes in technology. Just five years ago, for example, few customers would have expected to communicate with businesses over SMS or messaging services from their mobile phone. Now, it's common because consumers use those applications in other areas of their lives.

Perhaps the biggest area of change is the interaction channels used to communicate with businesses. Today, 58% of customers interact with digital channels, and 50% of all transactions start digitally, according to Metering's research. Consumers now expect to have several options for communication, including messaging apps like Facebook Messenger, WeChat, WhatsApp and Apple Business Chat, along with web chat, SMS, screen-sharing, video, self-service knowledge bases and FAQs, and chatbots.

Consumers also are more open to proactive outreach -- whether the customer service team is inviting them to a customer loyalty program or reminding them of an appointment -- so long as those reminders, confirmations and invitations arrive at their preferred application.

How is technology influencing the future of customer service?

Businesses can provide quick, contextual customer service with tools like analytics, agent assist and workforce optimization (WFO) for agents in the contact center, as well as customer-facing tools such as self-service, chatbots and personalization.

At the core of most new technologies are the three As -- artificial intelligence, automation and analytics. Working together, these technologies can provide organizations with advice, context, results and metrics for improvement, but it's imperative to roll out deployments cautiously instead of trying to boil the ocean. Businesses will then be able to identify how well these customer service tools address specific problems or opportunities and evaluate their performance through analytics.

APPENDIX

SOURCE CODE

```
from__future__import print_function
from audioop import add
import datetime
from unicodedata import name
from pprint import pprint
from flask import Flask, render_template, request, redirect, url_for, session, flash
from markupsafe import escape
from flask import *
import ibm_db
import datetime
conn =
ibm_db.connect("DATABASE=;HOSTNAME=;PORT=;SECURITY;=SSL;SSLServerCertificat
e=;UID=;PWD=", '', '')
print(conn)
print("connection successful...")
app = Flask(_name_)
app.secret_key = 'your secret key'
@app.route('/')
def home():
    message = "TEAM ID : PNT2022TMID18844” +" "+ "BATCH ID : B1-1M3E "
    return render_template('index.html',mes=message)
@app.route('/home', methods=['POST', 'GET'])
def index():
    return render_template('index.html')
@app.route('/signinpage', methods=['POST', 'GET'])
def signinpage():
```

```
    return render_template('signinpage.html')
@app.route('/agentsignin', methods=['POST', 'GET'])
def agentsignin():
  return render_template('signinpageagent.html')
@app.route('/signuppage', methods=['POST', 'GET'])
def signuppage():
    return render_template('signuppage.html')
@app.route('/agentRegister', methods=['POST', 'GET'])
def agentRegister():
    return render_template('agentregister.html')
@app.route('/forgotpass', methods=['POST', 'GET'])
def forgotpass():
    return render_template('forgot.html')
@app.route('/newissue/<name>', methods=['POST', 'GET'])
def newissue(name):
    name = name
    return render_template('complaint.html',msg=name)
@app.route('/forgot', methods=['POST', 'GET'])
def forgot():
try:
        global randomnumber
        ida = request.form['custid']
        print(ida)
        global id
        id = ida
        sql = "SELECT EMAIL,NAME FROM Customer WHERE id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, ida)
        ibm_db.execute(stmt)
        emailf = ibm_db.fetch_both(stmt)
        while emailf != False:
            e = emailf[0]
```

```
        n = emailf[1]
        break
configuration = sib_api_v3_sdk.Configuration()
    configuration.api_key['api-key']
api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
        sib_api_v3_sdk.ApiClient(configuration))
    subject = "Verification for Password"
    html_content = "<html><body><h1>Your verification Code is : <h2>" +
        \str(randomnumber)+"</h2> </h1> </body></html>"
    sender = {"name": "IBM CUSTOMER CARE REGISTRY",
            "email": "ibmdemo6@yahoo.com"}
    to = [{"email": e, "name": n}]
    reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
    headers = {"Some-Custom-Name": "unique-id-1234"}
    params = {"parameter": "My param value",
            "subject": "Email Verification"}
    send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
        to=to, reply_to=reply_to, headers=headers, html_content=html_content,
params=params, sender=sender, subject=subject)
 api_response = api_instance.send_transac_email(send_smtp_email)
pprint(api_response)
    message = "Email send to:"+e+" for password"
    flash(message, "success")
 except ApiException as e:
    print("Exception when calling SMTPApi->send_transac_email: %s\n" % e)
    flash("Error in sending mail")
  except:
    flash("Your didn't Signin with this account")
  finally:
    return render_template('forgot.html')
@app.route('/agentforgot', methods=['POST', 'GET'])
def agentforgot():
```

```
 try:
      global randomnumber
      ida = request.form['custid']
      print(ida)
      global id
      id = ida
      sql = "SELECT EMAIL,NAME FROM AGENT WHERE id=?"
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, ida)
      ibm_db.execute(stmt)
      emailf = ibm_db.fetch_both(stmt)
      while emailf != False:
         e = emailf[0]
         n = emailf[1]
         break
configuration = sib_api_v3_sdk.Configuration()
      configuration.api_key['api-key']
api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
         sib_api_v3_sdk.ApiClient(configuration))
      subject = "Verification for Password"
      html_content = "<html><body><h1>Your verification Code is : <h2>" +
         \str(randomnumber)+"</h2> </h1> </body></html>"
      sender = {"name": "IBM CUSTOMER CARE REGISTRY",
            "email": "ibmdemo6@yahoo.com"}
      to = [{"email": e, "name": n}]
      reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
      headers = {"Some-Custom-Name": "unique-id-1234"}
      params = {"parameter": "My param value",
            "subject": "Email Verification"}
      send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
         to=to, reply_to=reply_to, headers=headers, html_content=html_content,
params=params, sender=sender, subject=subject)
```

```
api_response = api_instance.send_transac_email(send_smtp_email)
pprint(api_response)
        message = "Email send to:"+e+" for
        OTP"flash(message, "success")
 except ApiException as e:
        print("Exception when calling SMTPApi->send_transac_email: %s\n" % e)
        flash("Error in sending mail")
    except:
        flash("Your didn't Signin with this account")
    finally:
        return render_template('forgot.html')
@app.route('/admin', methods=['POST', 'GET'])
def admin():
    userdatabase = []
    sql = "SELECT * FROM customer"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        userdatabase.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if userdatabase:
        sql = "SELECT COUNT(*) FROM
        customer;"stmt =
        ibm_db.exec_immediate(conn, sql)user =
        ibm_db.fetch_both(stmt)

    users = []
    sql = "select * from ISSUE"
    stmt = ibm_db.exec_immediate(conn, sql)
    dict = ibm_db.fetch_both(stmt)
    while dict != False:
        users.append(dict)
        dict = ibm_db.fetch_both(stmt)
```

```
    if users:
        sql = "SELECT COUNT(*) FROM ISSUE;"
        stmt = ibm_db.exec_immediate(conn, sql)
        count = ibm_db.fetch_both(stmt)
agent = []
    sql = "SELECT * FROM AGENT"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        agent.append(dictionary)
        dictionary =  ibm_db.fetch_both(stmt)if
   agent:
        sql = "SELECT COUNT(*) FROM AGENT;"
        stmt = ibm_db.exec_immediate(conn, sql)
        cot = ibm_db.fetch_both(stmt)
return
render_template("admin.html",complaint=users,users=userdatabase,agents=agent,mes
sage=user[0],issue=count[0],msgagent = cot[0])
@app.route('/remove', methods=['POST', 'GET'])
def remove():
otp = request.form['otpv']
    if otp == 'C':
        try:
            insert_sql = f"delete from customer"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prep_stmt)
            flash("delected successfully the Customer", "success")
        except:
            flash("No data found in Customer", "danger")
        finally:
            return redirect(url_for('signuppage'))if
    otp == 'A':
```

```
        try:
            insert_sql = f"delete from AGENT"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prep_stmt)
            flash("delected successfully the Agents", "success")
        except:
            flash("No data found in Agents", "danger")
        finally:
          return redirect(url_for('signuppage'))if
 otp == 'C':
        try:
            insert_sql = f"delete from AGENT"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prep_stmt)
            flash("delected successfully the Complaints", "success")
        except:
            flash("No data found in Complaints", "danger")
        finally:
            return redirect(url_for('signuppage'))
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        try:
id = request.form['idn']
            global hello
            hello = id
            password = request.form['password']
            print(id, password)
            if id == '1111' and password == '1111':
                return redirect(url_for('admin'))
 sql = f"select * from customer where id='{escape(id)}' and
password='{escape(password)}'"
```

```
            stmt = ibm_db.exec_immediate(conn, sql)
            data = ibm_db.fetch_both(stmt)

            if data:
                session["name"] = escape(id)
                session["password"] = escape(password)
                return redirect(url_for("welcome"))
else:
                flash("Mismatch in credetials", "danger")
        except:
            flash("Error in Insertion operation", "danger")
return render_template('signinpage.html')
@app.route('/welcome', methods=['POST', 'GET'])
def welcome():
    try:
        id = hello
        sql = "SELECT
ID,DATE,TOPIC,SERVICE_TYPE,SERVICE_AGENT,DESCRIPTION,STATUS
FROM ISSUEWHERE CUSTOMER_ID =?"
        agent = []
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        otpf = ibm_db.fetch_both(stmt)
        while otpf != False:
            agent.append(otpf)
            otpf = ibm_db.fetch_both(stmt)
sql = "SELECT COUNT(*) FROM ISSUE WHERE CUSTOMER_ID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        t = ibm_db.fetch_both(stmt)
```

```
        return render_template("welcome.html",agent=agent,message=t[0])
    except:
        return render_template("welcome.html")
@app.route('/loginagent', methods=['GET', 'POST'])
def loginagent():
    if request.method == 'POST':
        try:
            global loginagent
            id = request.form['idn']
            loginagent = id
            password = request.form['password']
    sql = f"select * from AGENT where id='{escape(id)}' and
password='{escape(password)}'"
            stmt = ibm_db.exec_immediate(conn, sql)
            data = ibm_db.fetch_both(stmt)

            if data:
                session["name"] = escape(id)
                session["password"] = escape(password)
                return redirect(url_for("agentwelcome"))
else:
                flash("Mismatch in credetials", "danger")
        except:
            flash("Error in Insertion operation", "danger")
return render_template("signinpageagent.html")
@app.route('/delete/<ID>')
def delete(ID):
    sql = f"select * from customer where Id='{escape(ID)}'"
    print(sql)
    stmt = ibm_db.exec_immediate(conn, sql)
    student = ibm_db.fetch_row(stmt)
```

```
    if student:
        sql = f"delete from customer where id='{escape(ID)}'"
        stmt = ibm_db.exec_immediate(conn, sql)

        flash("Delected Successfully", "success")
        return redirect(url_for("admin"))
@app.route('/agentform', methods=['GET', 'POST'])
def agentform():
    if request.method == 'POST':
 try:
            x = datetime.datetime.now()
            y = x.strftime("%Y-%m-%d %H:%M:%S")
            name1 = request.form['name']
            email = request.form['email']
            password = request.form['password']
            phonenumber = request.form['phonenumber']
            service = request.form['service']
            address = request.form['address']
            city = request.form['city']
            state = request.form['state'] country
            = request.form['country']link =
            request.form['link']
 sql = "SELECT * FROM AGENT WHERE EMAIL = ?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, email)
            ibm_db.execute(stmt)
            account = ibm_db.fetch_assoc(stmt)
if account:
                flash("Record Aldready found", "success")
            else:
                print("exec")
                insert_sql = "INSERT INTO AGENT
```

```
(NAME,EMAIL,PASSWORD,PHONENUMBER,SERVICE_AGENT,ADDRESS,CITY,STATE,COU
NTRY,RESUME_LINK,DATE) VALUES(?,?,?,?,?,?,?,?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, name1)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, password)
        ibm_db.bind_param(prep_stmt, 4, phonenumber)
        ibm_db.bind_param(prep_stmt, 5, service)
        ibm_db.bind_param(prep_stmt, 6, address)
        ibm_db.bind_param(prep_stmt, 7, city)
        ibm_db.bind_param(prep_stmt, 8, state)
        ibm_db.bind_param(prep_stmt, 9, country)
        ibm_db.bind_param(prep_stmt, 10, link)
        ibm_db.bind_param(prep_stmt, 11,
        y)ibm_db.execute(prep_stmt)
        flash("Record stored Successfully", "success")
        sql = "SELECT ID FROM AGENT WHERE
        email=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        hi = ibm_db.fetch_tuple(stmt)
configuration = sib_api_v3_sdk.Configuration()
        configuration.api_key['api-key']
api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
        sib_api_v3_sdk.ApiClient(configuration))
        subject = "Registering Account in Customer Care Registry"
        html_content = " <html><body><h1>Thanks for Registering into Customer Care
Registry</h1> <h2>Your Account Id is :"+str(hi[0])+"</h2><h2>With
Regards:</h2><h3>Customer Care Registry</h3> </body></html>"
        sender = {"name": "IBM CUSTOMER CARE REGISTRY",
          "email":  "ibmdemo6@yahoo.com"}
        to = [{"email": email, "name": name1}]
```

```
            reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
            headers = {"Some-Custom-Name": "unique-id-1234"}
            params = {"parameter": "My param value",
              "subject": "Email Verification"}
            send_smtp_email  =  sib_api_v3_sdk.SendSmtpEmail(
            to=to, reply_to=reply_to, headers=headers, html_content=html_content,
params=params, sender=sender, subject=subject)
api_response = api_instance.send_transac_email(send_smtp_email)
  pprint(api_response)
     except:
         flash("Error in Insertion Operation", "danger")
       finally:
         return redirect(url_for("agentRegister"))
         con.close()
return render_template('agentregister.html')
@app.route('/completed/<DESCRIPTION>', methods=['GET', 'POST'])
def completed(DESCRIPTION):
   status ="Completed"try:
 sql = "UPDATE ISSUE SET STATUS = ? WHERE DESCRIPTION =?"
       stmt = ibm_db.prepare(conn, sql)
       ibm_db.bind_param(stmt,1,status)
       ibm_db.bind_param(stmt,2,DESCRIPTION)
       ibm_db.execute(stmt)
flash("Successful","success")
       return redirect(url_for('agentwelcome'))
   except:
       flash("No record found","danger")
       return  redirect(url_for('agentwelcome'))
@app.route('/deletecomplaint/<ID>')
def deletecomplaint(ID):
   sql = f"select * from ISSUE where ID='{escape(ID)}'
```

```
        print(sql)
    stmt =
    ibm_db.exec_immediat
    e(conn, sql)student =
    ibm_db.fetch_row(stmt)
    if student:
        sql = f"delete from ISSUE where
        ID='{escape(ID)}'"stmt =
        ibm_db.exec_immediate(conn,
        sql)
        users = []
        flash("Delected  Successfully",  "success")


        return
redirect(url_for("ad
min"))if__name
__________== '
main_':
    app.run(host='0.0.0.0',  port=5000,  debug=True)
```

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-36813-1660298081