A PROJECT REPORT ON

CUSTOMER CARE REGISTRY

Domain : Cloud App Development

Team ID : PNT2022TMID29528

College Name : Arunai Engineering College

ILAKKIYA.S(510419106010)

Department Of Electronics and Communication Engineering

NIHA AMREEN.M(510419106018)

Department Of Electronics and Communication Engineering

PREETHA.T(510419106020)

Department Of Electronics and Communication Engineering

PRAVEENA.A(510419106019)

Department Of Electronics and Communication Engineering

1. **INTRODUCTION**
    a. Project Overview
    b. Purpose
2. **LITERATURE SURVEY**
    a. Existing problem
    b. References
    c. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    a. Empathy Map Canvas
    b. Ideation & Brainstorming
    c. Proposed Solution
    d. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    a. Functional requirement
    b. Non-Functional requirements
5. **PROJECT DESIGN**
    a. Data Flow Diagrams
    b. Solution & Technical Architecture
    c. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    a. Sprint Planning & Estimation
    b. Sprint Delivery Schedule
    c. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    a. Feature 1
    b. Feature 2
    c. Database Schema (if Applicable)
8. **TESTING**
    a. Test Cases
    b. User Acceptance Testing
9. **RESULTS**
    a. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code
GitHub & Project Demo Link

1.INTRODUCTION

1.a Project Overview

The Customer Service Desk is a web based project..With a platform of a typical “service center”, this system provides online technical services to its customers on a 24×7 basis .Its significance varies by product, industry and domain. In many cases customer services is more important if the information relates to a service as opposed to a customer. Customer Service may be provided by a service representatives customer service is normally an integral part of a company's customer value proposition. Developing a cloud application not only for solving customer complaints but also gives satisfaction to the customer to use the respective business product. This Application helps a customer to raise complaints for the issue they are facing in the products. The Customer needs to give the detailed description and the priority level of the issues that they are facing. After the complaint reviewed by the admin, then the agents assigned to the complaints raised by the customer. The respective customer of the complaints gets the email notification of the process. And additionally, they can able to see the status of the complaints**.**

1.b Purpose

An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through and e-mail. The system should have capability to integrate with any Service Provider from any domain or industry like Banking, Telecom Insurance etc. This Application mainly developed to help the customer in processing their complaints and issues. It is a process of examining customer tickets, which should be carried out in a systematic and orderly manner. This practice is primarily aimed at minimizing consumer dissatisfaction with the purchased products, increasing service satisfaction, and ensuring quality. It allows companies to respond to customer inquiries, provides support, and improves the handling of tickets at the appointed time. The main objective of this online customer care and service center software is to develop an information system to store, maintain, update and process data relating to the stop. It will prepare various reports to aid in smooth and speedy functioning of ‘service center activities. The main purpose of customer care is to provide the proper service to the customer and satisfy the needs of the customer.

2.LITERATURE SURVEY

2.a Existing problem

The existing system is a semi-automated at where the information is stored in the form of excel sheets in disk drives. The information sharing to the Volunteers, Group members, etc. is through mailing feature only. The information storage and maintenance is more critical in this system. Tracking the member's activities and progress of the work is a tedious job here. This system cannot provide the information sharing by 24x7 days. When the company pushes the wrong product or service to customer this can severely impact to company's profit, growth and brand reputation. The customer cannot track the status of the Queries that are posted by them. Some queries will be left Unanswered. To overcome this issues a good customer care should be provided to solve the customer's queries.

2.b References

PAPER 1: Customer Centric Cloud Service Model

Published year: September 2009

Author: Hong Cai , Ke Zhang

Journal name: IEEE International Conference on Cloud Computing

Methodology used: Cloud App Development

Summary: This paper proposes a cloud service model that centres around customer business requirements. The model covers the customers' subscribed services and the service providers' offered cloud services. It covers the relationship among offering versus subscription, cloud infrastructure service versus cloud application service, configuration versus customization of cloud services, etc. It then depicts a customer centric cloud service model which models the artifacts of enterprises owned by public serving cloud services. It's an extension of enterprise services to open service ecosystem leveraging the latest technical innovation of cloud computing. Later, this paper carries out a case study on an IBM software group ongoing project, commerce as a service, which aims to provide e-commerce functions as services over a cloud infrastructure.

PAPER 2: Customer Care in Service Organisations

Published year: March 1998
Author: Hong Cai , Ke Zhang
Journal name: IEEE International Conference on Cloud Computing
Methodology used: Cloud App Development

Summary: The quality of service and customer care in the context of the marketing of services are considered. The focus is on distinguishing characteristics of services, definitions of service quality and the use of consumer research to assess expectations of and satisfactions with service quality — providing examples from a variety of organisations. Particular attention is given to the interpersonal interactions between contact personnel in service companies and customers, and the need for internal marketing, a consumer orientation, and the consequent provision of customer care, with reference to a number of examples in the tourism and financial service sectors

PAPER 3: Application of data mining techniques in customer relationship management

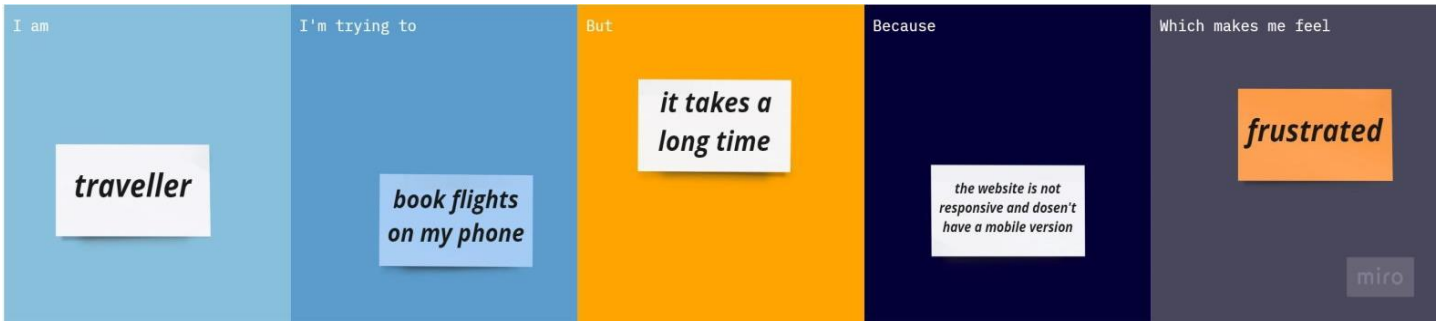
Published year: March 2009
Author: Eric W.T. Ngai, Li Xiu
Journal name: Experts Systems with Applications
Methodology used: IOT Methodology

Summary*:* Despite the importance of data mining techniques to customer relationship management (CRM), there is a lack of a comprehensive literature review and a classification scheme for it. This is the first identifiable academic literature review of the application of data mining techniques to CRM. It provides an academic database of literature between the period of 2000–2006 covering 24 journals and proposes a classification scheme to classify the articles. Nine hundred articles were identified and reviewed for their direct relevance to applying data mining techniques to CRM. Eighty-seven articles were subsequently selected, reviewed and classified. Each of the 87 selected papers was categorized on four CRM dimensions (Customer Identification, Customer Attraction, Customer Retention and Customer Development) and seven data mining functions (Association, Classification, Clustering, Forecasting, Regression, Sequence Discovery and Visualization). Papers were further classified into nine sub-categories of CRM elements under different data mining techniques based on the major focus of each paper. The review and classification process were independently verified. Findings of this paper indicate that the

research area of customer retention received most research attention. Of these, most are related to one-to-one marketing and loyalty programs respectively. On the other hand, classification and association models are the two commonly used models for data mining in CRM. Our analysis provides a roadmap to guide future research and facilitate knowledge accumulation and creation concerning the application of data mining techniques in CRM.

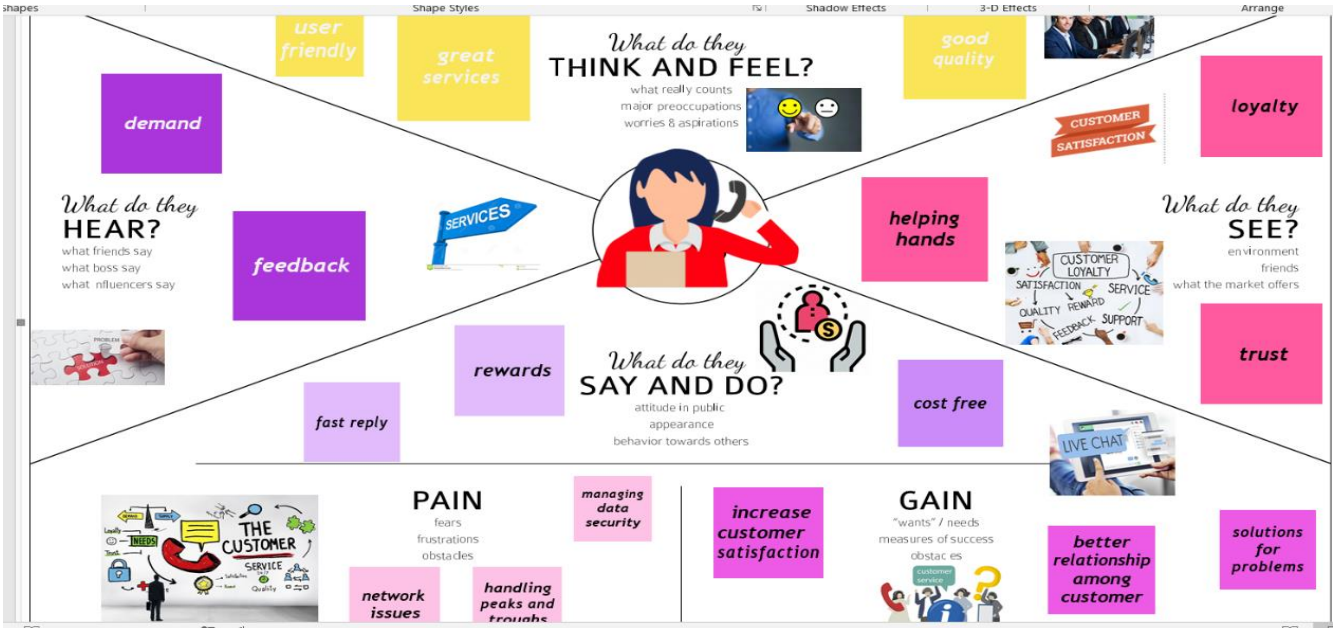
2.c Problem Statement Definition

A problem statement is a concise description of the problem or issues a project seeks to address. The problem statement identifies the current state, the desired future state and any gaps between the two. A problem statement is an important communication tool that can help ensure everyone working on a project knows what the problem they need to address is and why the project is important.



3.IDEATION & PROPOSED SOLUTION

3.a Empathy Map Canvas



3.b Ideation & Brainstorming

1

Define your problem statement

Problems on customer care registry

5 minutes

Customers have been decreased significantly over the past few years .

dealing with angry customer

transferring customer calls.

customer complains are increased

2

Brainstorm

Ideas to overcome the problem.

10 minutes

ILAKKIYA

- track complaints using help desk software
- leverage the latest technology
- builds customer confidence
- meeting customer expectation
- direct the customer to right path
- keeps crisis management in mind

NIHA AMREEN

- promise only what you can deliver
- efficiently handle all aspects
- promote customer satisfaction
- customer engagement software
- layout a plan to address the situation
- build a customer journey map

PREETHA

- deliver omni channel support
- acknowledge the customer's questions
- explain the problem in simple terms
- chats to be answered
- detailing every touchpoint
- allow access to chatbot.livechat

PRAVEENA

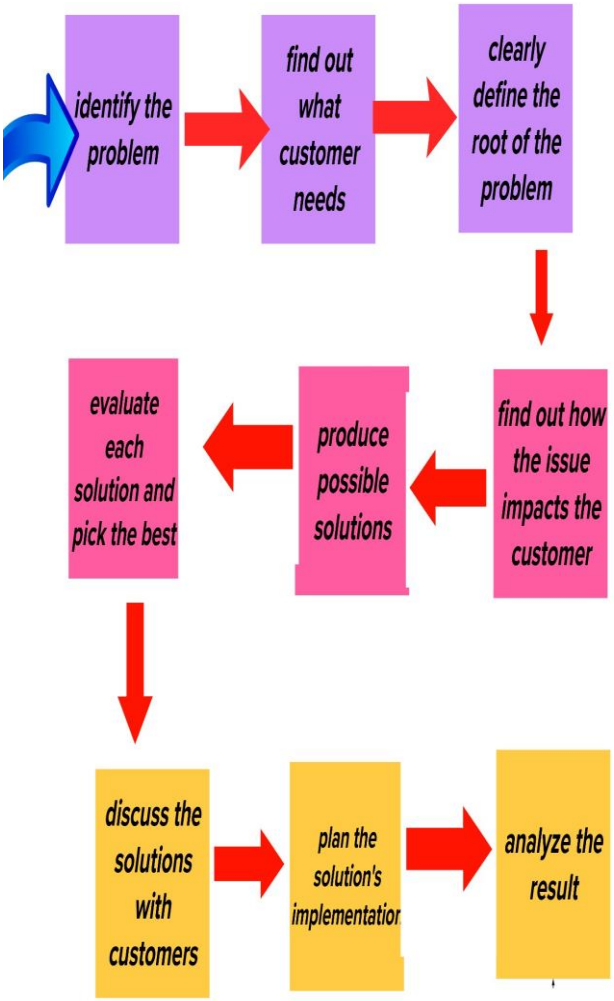
- HEARD technique
- handle crises and escalation
- slove customer problem as fast as
- build workflow
- focus on the small things
- collect customer feedback

3

Group ideas

Build informal connections with peers

20 minutes

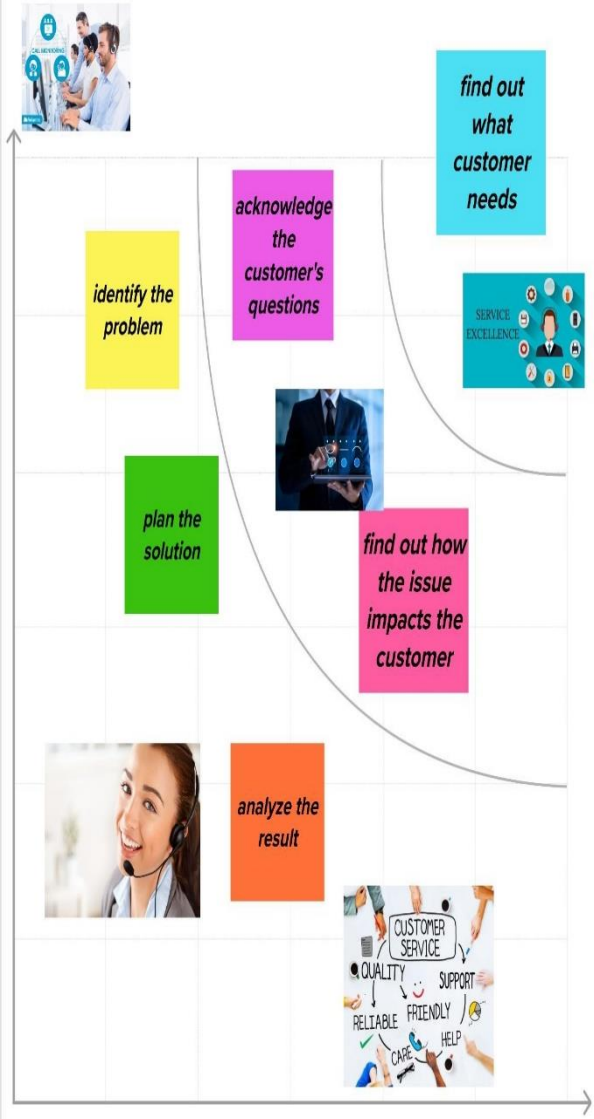


4

Prioritize

Idea priortization

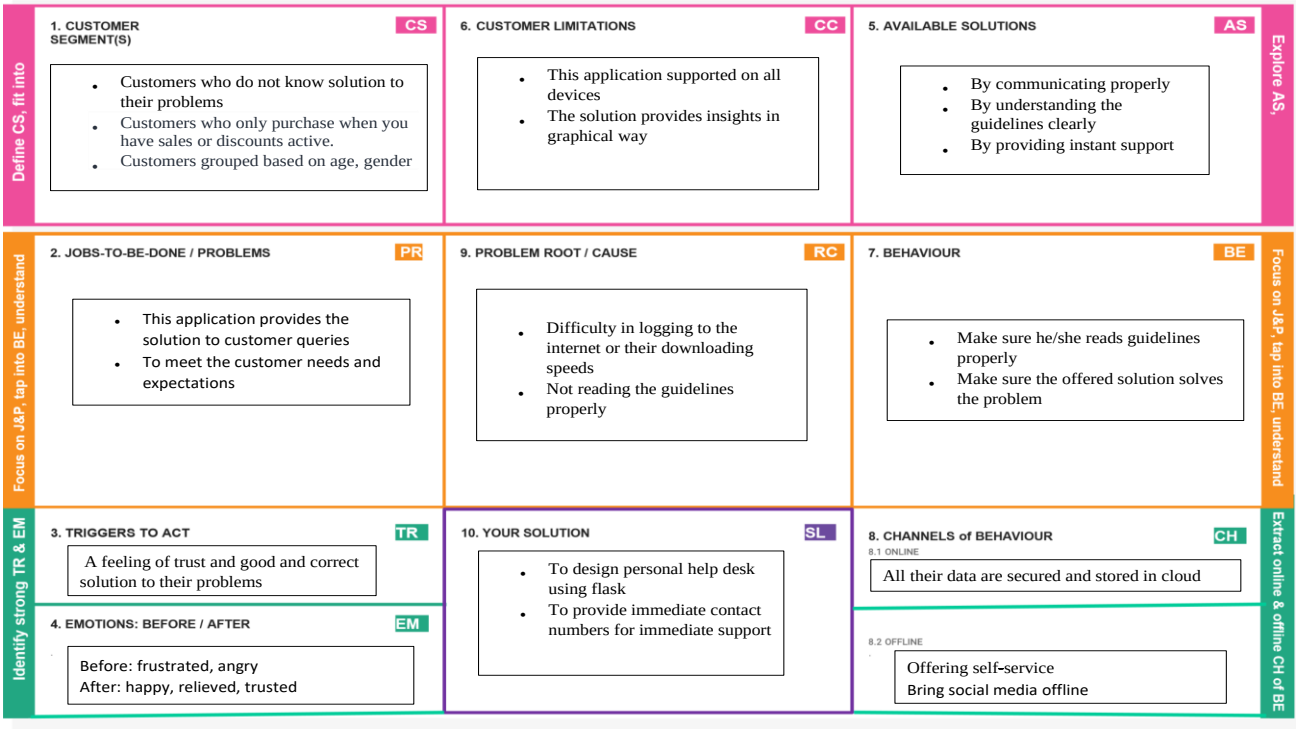
20 minutes



3.c Proposed Solution

S.No.	*Parameter*	*Description*
1.	Problem Statement (Problem to be solved)	• To solve customer issues using Cloud Application Development • To solve the long response time
2.	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. • Regular data retrieval in the form of retrieving lost data.
3.	Novelty / Uniqueness	Assigned Agent Routing, Automated Ticket Closure, Status Shown to the Customer, and Backup data in case of failures.
4.	Social Impact / Customer Satisfaction	Customer can track their status Easy agent communication.
5.	Business Model (Revenue Model)	Partners are Third-party applications, agents, of a customer. • Activities held as Customer Service, System maintenance. • Customer Relationship have 24/7Email-support, knowledge-based channel. • Cost Structure expresses Cloud Platform, Offices
6.	Scalability of the Solution	• The real goal of scaling customer 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 resolving critical customer issues. • Onboarding is also a key element in scaling

3.d Problem Solution fit



4.REQUIREMENT ANALYSIS

4.a Functional Requirements

Following are the functional requirements of the proposed solution.

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

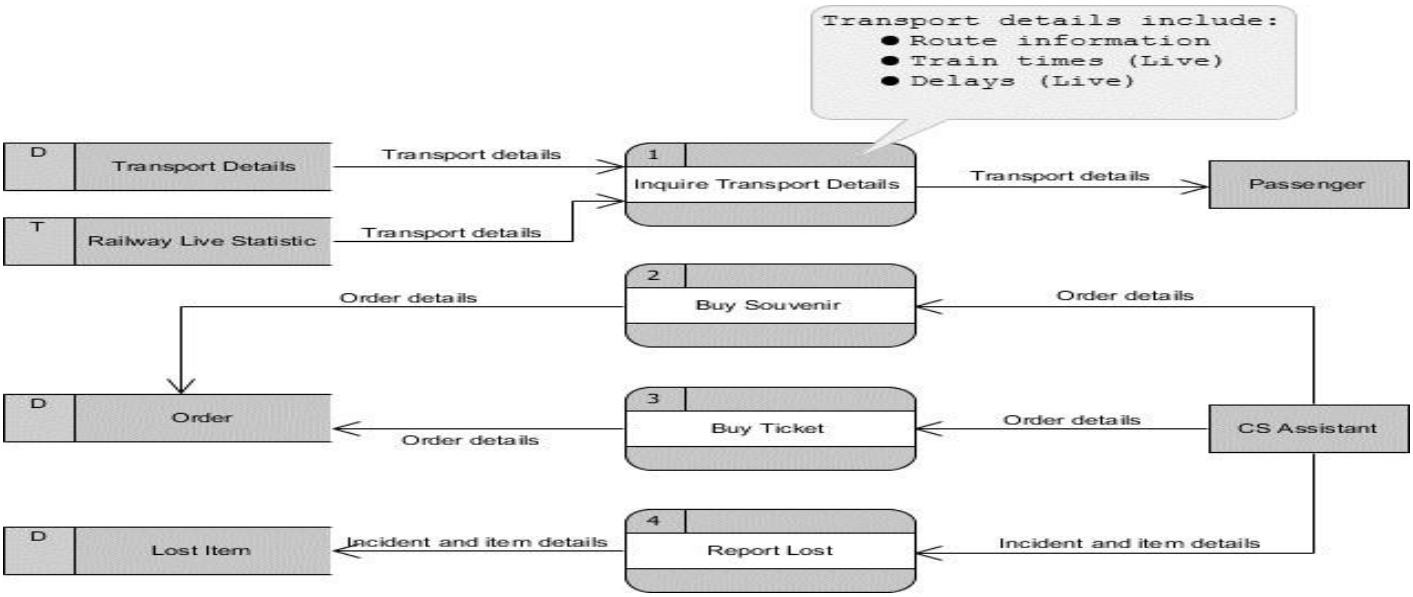
4.b Non Functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User Friendly, Easily Accessible.
NFR-2	**Security**	IBM Digital Security Certificate (SSL) for Database.
NFR-3	**Reliability**	Providing Quality Content.
NFR-4	**Performance**	Quick Access, Flexible, and Responsive
NFR-5	**Availability**	24/7 Support
NFR-6	**Scalability**	Good performance for large Customers and workload

5.PROJECT DESIGN

5.a Data Flow Diagrams

The data flow diagram of customer care registry project normally consists of overall application dataflow and processes of the customer process. It contains all the userflow and their entities such as customer feedback, salesman, product, transaction, offering, schedule.



5.b Solution & Technical Architecture

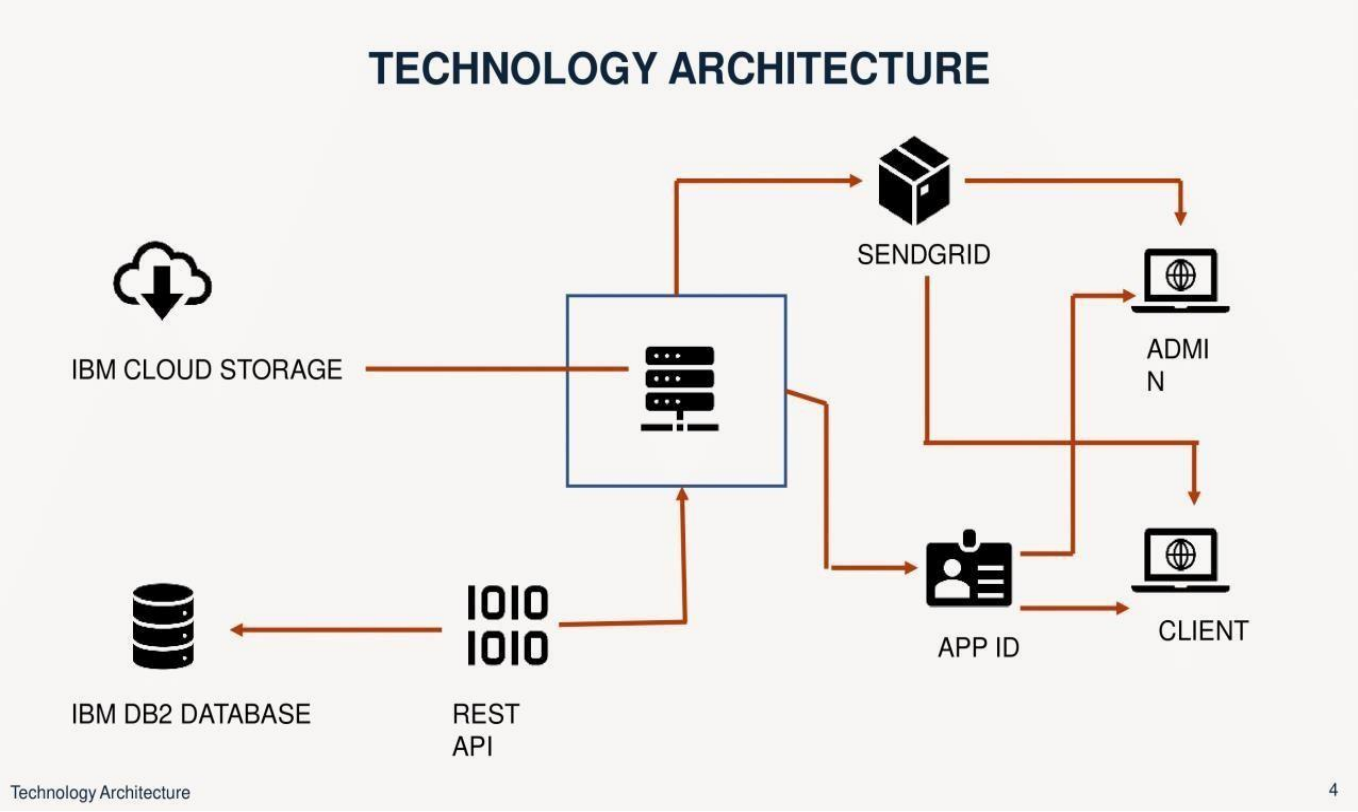


Table 1 - Components and technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application Eg) Web UI, Mobile App, Chatbot, etc	HTML, CSS, Javascript/Angular Js/React Js, Bootstrap etc.
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type , Configuration etc	MySQL,etc.
6.	Cloud Database	Database service on cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirement	IBM Block storage or other storage service or Local Filesystem.
8.	Infrastructure	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration:	Local, Cloud Foundry,etc

Table 2 – Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	python flask
2.	Security Implementation	List all the security/access controls implemented, use of firewalls etc.	e.g., encryption, intrusion detection software, antivirus, firewalls

3.	Scalable Architecture	Justify the scalability of architecture (3-tier, Micro-services)	supports higher workloads without any fundamental changes to it.
4.	Availability	Justify the availability of the application (eg, use of load balances, distributed servers etc.)	High availability enables your IT infrastructure to continue functioning even when some of its components fail.
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN) etc.	Performance technology, therefore, is a field of practice that uses various tools, processes, and ideas in a scientific, systematic manner to improve the desired outcomes of individuals and organizations

5.c User Stories

User Stories

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

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

Admin (Mobile user)	Login	USN-1	As a admin, I can login to the appliaction by entering Correct email and password	I can access my account/dashboard	High	Sprint-1
	Dashboard	USN-2	As an admin I can see all the orders raised in the entire system and lot more	I can assign agents by seeing those order.	High	Sprint-1
	Agent creation	USN-3	As an admin I can create an agent for clarifying the customers queries	I can create agents.	High	Sprint-2
	Assignment agent	USN-4	As an admin I can assign an agent for each order created by the customer.	Enable agent to clarify the queries.	High	Sprint-1
	Forgot password	USN-5	As an admin I can reset my password by this option in case I forgot my old password.	I get access to my account.	High	Sprint-1

6.PROJECT DEVELOPMENT AND PLANNING

6.a Sprint Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers,research publications etc.	3 OCTOBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	6 OCTOBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	10 OCTOBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	16 OCTOBER 2022

Problem Solution Fit	Prepare problem - solution fit document.	16 OCTOBER 2022
Solution Architecture	Prepare solution architecture document.	17 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	22 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	23 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram	27 OCTOBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	28 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	9 NOVEMBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	19 NOVEMBER (AS PLANNED)

Product Backlog, Sprint Schedule, and Estimation

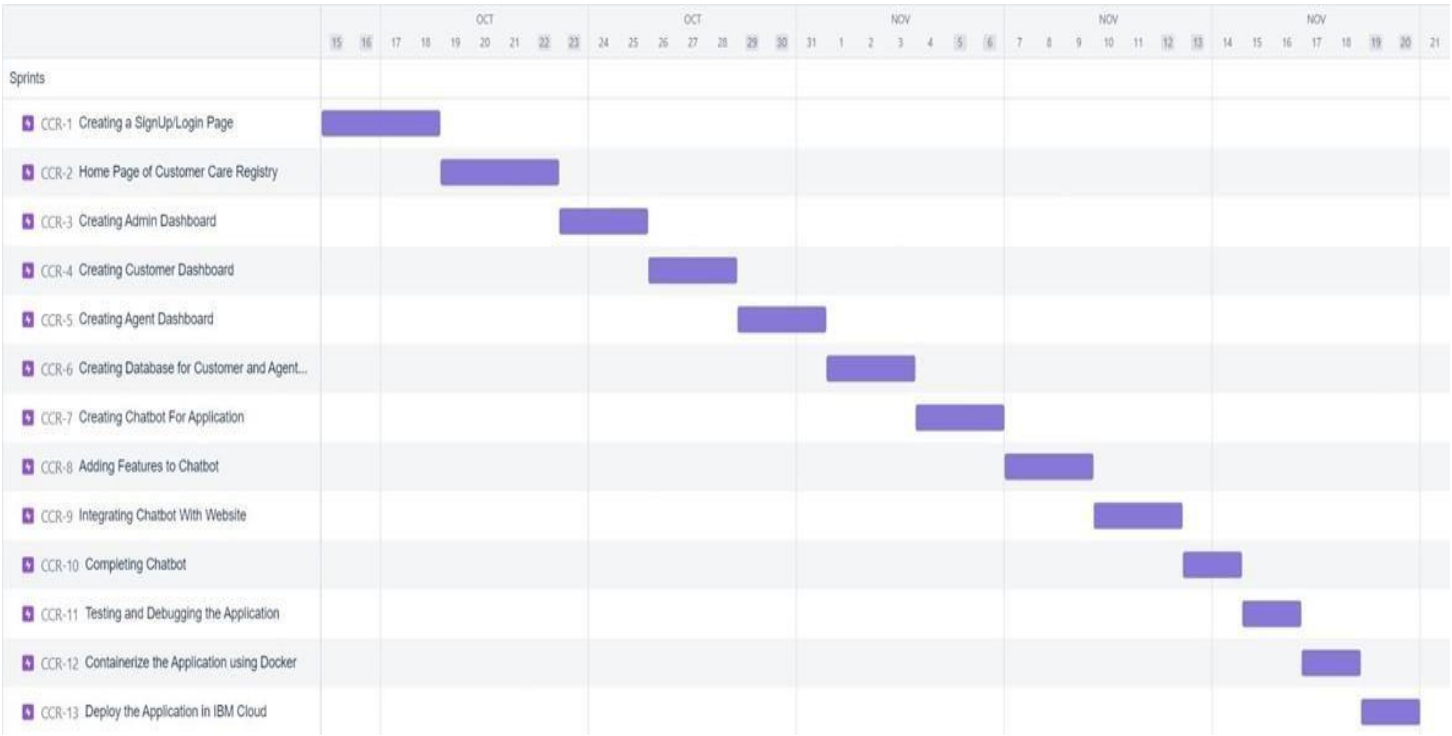
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Ilakkiya S Preetha T
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Niha amreen M Ilakkiya S
		USN-3	As a user, I can register for the application through Gmail	2	Low	Preetha T Niha amreen M

Sprint-2	Login	USN-4	As a user, I can log into the application by entering email & password	2	Medium	Praveena A Ilakkiya S Niha amreen M
	Dashboard	USN-5	As a user, I can see the issues/tickets raised by me and I can raise a new issues/ticket	2	High	Preetha T Ilakkiya S
		USN-6	As a user, I can raise a new issues/ticket with detailed descriptions of the issue	2	High	Ilakkiya S Praveena A
Sprint-3		USN-7	As a user, I can see the admin assigned an agent to the issue/ticket raised by me	1	Low	Niha Amreen M Preetha T
	Notification	USN-8	As a user, I can receive an email notification alert once an agent is assigned to the issue	2	Medium	Ilakkiya S Niha Amreen M Preetha T
		USN-9	As a user, I can see the live status of the issues/tickets raised by me	1	Medium	Praveena A Niha amreen M
Sprint-4	Chat	USN-10	As a user, I can chat with an agent more about the issue and I can get detailed explanation of the issue status	2	High	Niha amreen M Praveena A
	Graph	USN-11	As a user, I can get the detailed visual graphs based on the past issues	2	Medium	Preetha T Ilakkiya S
		USN-12	As a user, I can modify the active issues raised by me	2	Medium	Niha amreen M Preetha T Praveena A

6.b Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	10	07 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	10	07 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	10	14 Nov 2022

6.c Reports from JIRA



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

College graduates with prior programming expertise or technical degrees are recruited and transitioned into professional positions with Alabama firms and organisations 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 mobilise 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.

7.a Feature 1

7 Main types of customer needs:

- Friendliness
- Empathy
- Fairness
- Control
- Alternatives
- Information
- Guidance

7.b Feature 2

- Complaint Tracking
- Email Alert
- 24/7 monitoring and servicing

8.TESTING

8.a Test Cases

TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	1.Enter URL and click go 2.Scroll down 3.Verify login/Singup popup displayed or not	http://169.51.204.215:30106/	Login/Signup popup should display	Working as expected	PASS	Successfull			ILAKKIYA S
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	1. Enter URL and click go 2.Click on Sigup button for User 3. Verify login/Singup popup with below UI elements: a.id text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	http://169.51.204.215:30106/	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	PASS	Successful			**NIHA AMREEN M**
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter valid password in password text box 5.Click on login button	ID: 5342 password: Testing123	User should navigate to user account homepage	Working as expected	PASS	Successful			PREETHA T
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter valid password in password text box 5.Click on login button	ID: 5342 password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			PRAVEENA A
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter Invalid password in password text box 5.Click on login button	ID: 5342 password: Testing123678686876876	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			ILAKKIYA S
LoginPage_TC_OO6	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	ID: 5342 password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			PREETHA T

LoginPage_TC_OO7	Functional	Login page	Verify User is able to log into application with Valid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	ID: 5434 password: Testing123	Application should show 'correct email or password ' validation message.	Working as expected	PASS	Successful			NIHA AMREEN M
LoginPage_TC_OO8	Functional	Login page for ADMIN	Verify User is able to log into application with Valid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter Valid ID in ID text box 4.Enter valid password in password text box 5.Click on login button	ID: 1111 password: 5678	Application should show 'correct email or password ' validation message.	Working as expected	PASS	Successful			PRAVEENA A
LoginPage_TC_OO9	UI	ADMIN PAGE	Verify all the Customer database is visible	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	http://169.51.204.215:30106/	Customer database is visible	Working as expected	PASS	Successful			PREETHA T

LoginPage_TC_O10	Functional	USER REGISTER	Verify Id sent to customer email address	1.Enter URL(http://169.51.204.215:30106/) and click go 1.Register the account by giving credentials 2. Click on button Submit	http://169.51.204.215:30106/	Email sent successfully	Working as expected	PASS	Successful			ILAKKIYA S
LoginPage_TC_O11	Functional	AGENT REGISTER	Verify AGENT is able to log into application with Valid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	ID: 5342 password: Testing123	ID sent successfully	Application should show 'correct email or password ' validation message.	PASS	Successful			NIHA AMREEN M
LoginPage_TC_O12	Functional	Login page for ADMIN	Verify User is able to log into application with InValid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3.Enter InValid ID in ID text box 4.Enter Invalid password in password text box 5.Click on login button	ID: 1111 password: 5678	Application should show 'Incorrect ID or password ' validation message.	Working as expected	PASS	Successful			PREETHA T
LoginPage_TC_O13	UI	Home page for Agent	Verify user is able to see the agent home page when user finish on submitting Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2. To the Agent Login page and submit Your Credentials	ID: 1111 password: 5678	AGENT Home Page popup should display	Working as expected	PASS	Successful			ILAKKIYA S

LoginPage_TC_O14	UI	Home page for USER	Verify user is able to see the User home page when user finish on submitting Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and submit Your Credentials	http://169.51.204.215:30106/	USER Home Page popup should display	Working as expected	PASS	Successful			PREETHA T
LoginPage_TC_O15	UI	Home page for ADMIN	Verify user is able to see the ADMIN home page when user finish on submitting Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and submit Your Credentials	http://169.51.204.215:30106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			NIHA AMREEN M
LoginPage_TC_O16	Functional	AGENT PAGE	On delete Button the user Credentials will be delected.	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the Admin Page and delect on User Credentials	http://169.51.204.215:30106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			ILAKKIYA S

8.b User Acceptance Testing

Purpose of Document

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

Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtot al
By Design	5	0	0	2	7
External	0	2	0	0	2
Fixed	12	11	35	45	103
Not Reproduced	0	5	0	0	5
Skipped	0	0	0	0	0
Totals	17	18	35	47	117

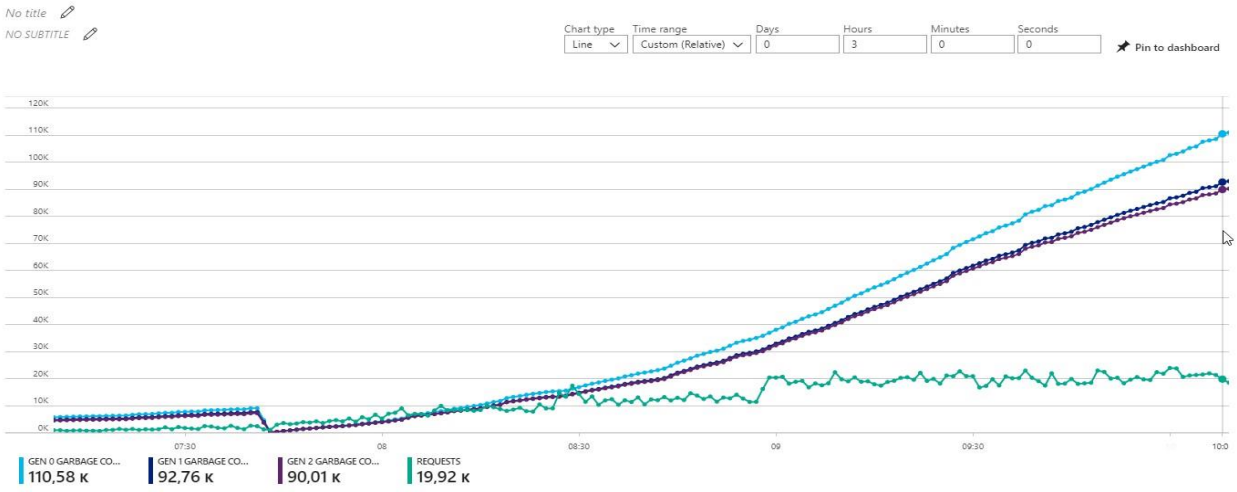
Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Client Application	72	0	0	72
Security	7	0	0	7
Exception Reporting	5	0	0	5
Final Report Output	4	0	0	4

9.RESULTS

9.1 Performance Metrics



10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

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

DISADVANTAGES:

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

11.CONCLUSION

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

12. FUTURE SCOPE

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

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

13. APPENDIX

Source Code

```
# Project : Customer Care Registry
# Team ID : PNT2022TMId29528
 index.py
 from flask import Flask, render_template, request, redirect, url_for, session, flash, jsonify from
 flask_mysqldb import MySQL import MySQLdb.cursors import ibm_db import re, random,
 smtplib, os, time, datetime from flask_mail import Mail, Message


 app = Flask(__name__)

 app.secret_key = '12345'

 conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=21fecfd8-47b7-4937-
 840dd791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31864;SECURITY=SSL;
 SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=rtp84701;PWD=DJ4gX1wChdTCGZPz","")
```

```
mail= Mail(app)

app.config['MAIL_SERVER']='smtp.gmail.com' app.config['MAIL_PORT'] =
465 app.config['MAIL_USERNAME']='customercareregistry22@gmail.com'
app.config['MAIL_PASSWORD'] = 'vxzttcjvdvrqeeve'
app.config['MAIL_USE_TLS'] = False app.config['MAIL_USE_SSL'] = True
mail = Mail(app)



@app.route('/', methods =['GET', 'POST']) def index():    if
request.method == 'POST' and 'email' in request.form:
    email = request.form['email']      cursor =
mysql.connection.cursor(MySQLdb.cursors.DictCursor)      cursor.execute('SELECT *
FROM subscriptions WHERE email = % s', (email, ))      subscriptions = cursor.fetchone()
if subscriptions:
       flash('This Email Is Already Subscribed')      else:
       ts = time.time()
       timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')          cursor =
mysql.connection.cursor(MySQLdb.cursors.DictCursor)          cursor.execute('INSERT INTO subscriptions
VALUES (%s, % s, % s)', (None, email, timestamp, ))          mysql.connection.commit()          flash('You
have successfully Subscribed')    return render_template('index.html')



@app.route('/customerlogin', methods =['GET', 'POST']) def
customerlogin():
  msgdecline = ''    if request.method == 'POST' and 'cemail' in request.form and 'cpassword' in
request.form:
    cemail = request.form['cemail']              cpassword = request.form['cpassword']              cursor =
mysql.connection.cursor(MySQLdb.cursors.DictCursor)                cursor.execute('SELECT * FROM
customers_details WHERE customer_email = % s AND customer_password = % s', (cemail, cpassword, ))
customers_details = cursor.fetchone()      if customers_details:
       session['loggedin'] = True          session['cemail'] =
customers_details['customer_email']          msgsuccess = 'Logged in
successfully !'          return redirect(url_for('welcome'))      else:
       msgdecline = 'Incorrect Email / Password !'    return
render_template('customerlogin.html', msgdecline = msgdecline)



@app.route('/agentlogin', methods =['GET', 'POST']) def agentlogin():    msgdecline = ''    if
request.method == 'POST' and 'aemail' in request.form and 'apassword' in request.form:
    aemail = request.form['aemail']      apassword = request.form['apassword']      cursor =
mysql.connection.cursor(MySQLdb.cursors.DictCursor)      cursor.execute('SELECT * FROM
```

```
agent_information WHERE agent_email = % s AND agent_password = %s', (aemail, apassword,))
agent_information = cursor.fetchone()       if agent_information:
        session['loggedin'] = True
        session['aemail']          =          agent_information['agent_email']
msgsuccess       =       'Logged       in       successfully       !'       return
redirect(url_for('agentdashboard'))
    else:
        msgdecline = 'Incorrect Email / Password !'    return
render_template('agentlogin.html', msgdecline = msgdecline)



@app.route('/adminlogin', methods =['GET', 'POST']) def adminlogin():    msgdecline = ''    if request.method
== 'POST' and 'adminusername' in request.form and 'adminpassword' in request.form:
    adminusername = request.form['adminusername']          adminpassword = request.form['adminpassword']
cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)          cursor.execute('SELECT * FROM
admin_details WHERE admin_username = % s AND admin_password = % s', (adminusername,
adminpassword, ))       admin = cursor.fetchone()       if admin:
        session['loggedin'] = True          session['adminusername'] =
admin['admin_username']          msgsuccess = 'Logged in successfully
!'          return redirect(url_for('admindashboard'))       else:
        msgdecline = 'Incorrect Username / Password !'    return
render_template('adminlogin.html', msgdecline = msgdecline)



@app.route('/customerregister', methods =['GET', 'POST']) def customerregister():       msgdecline = ''       if
request.method == 'POST' and 'cname' in request.form and 'cemail' in request.form and 'cpassword' in
request.form and 'cconfirmpassword' in request.form :
    cname = request.form['cname']       cemail = request.form['cemail']       cpassword =
request.form['cpassword']       cconfirmpassword = request.form['cconfirmpassword']       cursor =
mysql.connection.cursor(MySQLdb.cursors.DictCursor)       cursor.execute('SELECT * FROM
customers_details WHERE customer_email = % s', (cemail, ))       user_registration = cursor.fetchone()       if
user_registration:
        msgdecline = 'Account already exists ! Try Login'       elif
cpassword != cconfirmpassword:
        msgdecline = 'Password did not match !'       else:
        ts = time.time()
        timestamp          =          datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d          %H:%M:%S')
cursor.execute('INSERT INTO customers_details VALUES (%s, % s, % s, % s, % s)', (None, cname, cemail,
cpassword, timestamp, ))          mysql.connection.commit()          flash('You have successfully registered ! Try
Login')          try:
            mailmsg = Message('Customer Care Registry', sender = 'Registration Successful', recipients
= ['{}', cemail])
```

```
        mailmsg.body = "Hello {},\nYou have successfully registered on Customer Care
Registry".format(cname)          mail.send(mailmsg)          except:          pass          return
redirect(url_for('customerlogin'))    elif request.method == 'POST':
    msgdecline = 'Please fill out the form !'    return
render_template('customerregister.html', msgdecline = msgdecline)



@app.route('/agentregister', methods =['GET', 'POST']) def
agentregister():    if not session.get("adminusername"):
    return redirect("/adminlogin")    else:        msgdecline = "        if request.method == 'POST' and 'aname' in
request.form and 'aemail' in request.form and 'ausername' in request.form and 'apassword' in request.form and
'aconfirmpassword' in request.form :
        aname = request.form['aname']          aemail = request.form['aemail']          ausername =
request.form['ausername']          apassword = request.form['apassword']          aconfirmpassword =
request.form['aconfirmpassword']          cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
cursor.execute('SELECT * FROM agent_information WHERE agent_email = % s', (aemail, ))
agent_information = cursor.fetchone()          if agent_information:
        msgdecline = 'Account already exists ! Try Login'
else:          ts = time.time()
        timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
cursor.execute('INSERT INTO agent_information VALUES (%s, % s, % s, % s, % s, % s)', (None, aname,
aemail, ausername, apassword, timestamp,))          mysql.connection.commit()          flash('Agent Has been
successfully registered !')
        try:
          mailmsg = Message('Customer Care Registry', sender = 'Registration Successful', recipients = ['{}',
aemail])                mailmsg.body = "Hello, You have been Successfully Registered as Agent"
mail.send(mailmsg)          except:          pass          return redirect(url_for('agentlogin'))        elif
request.method == 'POST':        msg = 'Please fill out the form !'    return render_template('agentregister.html',
msgdecline = msgdecline)



@app.route('/welcome', methods =['GET', 'POST']) def
welcome():
  if not session.get("cemail"):
    return redirect("/customerlogin")    else:
    msgsuccess = "          msgdecline = "          cmail = session['cemail']          mycursor =
mysql.connection.cursor(MySQLdb.cursors.DictCursor)          mycursor.execute('SELECT * FROM
complaint_details WHERE customer_email = %s ORDER BY timestamp DESC', (cmail,))          data =
mycursor.fetchall()          mycursor.execute('SELECT customer_name FROM customers_details WHERE
customer_email
= %s', (cmail,))        cname = mycursor.fetchone()        if request.method == 'POST' and 'name' in request.form
and 'email' in request.form and 'category' in request.form and 'subject' in request.form and 'description' in
request.form :
```

```
        name = request.form['name']         email =
request.form['email']         category =
request.form['category']         subject =
request.form['subject']         description =
request.form['description']         ticketno =
random.randint(100000, 999999)         ts = time.time()
        timestamp = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')         cursor =
mysql.connection.cursor(MySQLdb.cursors.DictCursor)         cursor.execute('INSERT INTO
complaint_details VALUES (%s, % s, % s, % s, % s, % s, % s, % s, % s )', (ticketno, name, email, category,
subject, description, timestamp , "pending" , "pending" , ))         mysql.connection.commit()         try:
            mailmsg = Message('Customer Care Registry', sender = 'Request Received', recipients = ['{}', email])
            mailmsg.body = "Hello {},\n\nThanks for contacting Customer Care Registry\nWe have received your
complain\nYour Ticket Number: {}\nCategory: {}\nSubject: {}\nDescription: {}\n\nWe strive to provide
excellent service, and will respond to your request as soon as possible.".format(name, ticketno, category,
subject, description)             mail.send(mailmsg)         except:             pass
        flash ('Your complaint is successfully submitted !')         return
redirect(url_for('welcome'))       elif request.method == 'POST':
        msgdecline = 'Please fill out the form !'    return render_template('welcome.html', msgsuccess =
msgsuccess, data=data, cname=cname)



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

    for x in agent:
        agent_name = agent[x]

    mycursor2 = mysql.connection.cursor(MySQLdb.cursors.DictCursor)         mycursor2.execute('SELECT *
FROM complaint_details WHERE agent_name = %s ORDER BY timestamp DESC', (agent_name, ))       data
= mycursor2.fetchall()       if request.method == 'POST' and 'status' in request.form :
        status = request.form['status']                ticketno = request.form['ticketno']                cursor =
mysql.connection.cursor(MySQLdb.cursors.DictCursor)         cursor.execute('UPDATE complaint_details SET
status = %s WHERE ticket_no = %s', (status, ticketno,) )         mysql.connection.commit()         msg = 'Your
complaint is successfully solved !'

        mailcursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)         mailcursor.execute('SELECT
customer_email FROM complaint_details WHERE ticket_no = %s', (ticketno,) )
        customer_mail = mailcursor.fetchone()

        for x in customer_mail:
```

```
        cemail = customer_mail[x]
            try:
        mailmsg = Message('Customer Care Registry', sender = 'Your Ticket Status', recipients = ['{}', cemail])
        mailmsg.body = "Hello, \nYour complaint has been successfully solved\nYour Ticket Number: {}".format(ticketno) mail.send(mailmsg) except: pass return redirect(url_for('agentdashboard')) elif request.method == 'POST': msg = 'Please fill out the form !' return render_template('agentdashboard.html', msg = msg, data=data, agent_name=agent_name)



@app.route('/admindashboard', methods =['GET', 'POST']) def admindashboard(): if not session.get("adminusername"):
    return redirect("/adminlogin") else:
    msg = ''
    mycursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor) mycursor.execute('SELECT * FROM complaint_details ORDER BY timestamp DESC') data = mycursor.fetchall() mycursor.execute('SELECT * FROM agent_information') agent = mycursor.fetchall() mycursor.execute('SELECT COUNT(status) AS pending FROM complaint_details WHERE status = %s', ("pending",)) pending = mycursor.fetchall() mycursor.execute('SELECT COUNT(status) AS assigned FROM complaint_details WHERE status = %s', ("Agent Assigned",)) assigned = mycursor.fetchall() mycursor.execute('SELECT COUNT(status) AS completed FROM complaint_details WHERE status = %s', ("Closed",)) completed = mycursor.fetchall() if request.method == 'POST' and 'agentassign' in request.form :
        agentassign = request.form['agentassign'] adminusername = request.form['adminusername'] ticketno = request.form['ticketno'] cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor) cursor.execute('UPDATE complaint_details SET agent_name = %s WHERE ticket_no = %s', (agentassign, ticketno,) ) cursor.execute('UPDATE complaint_details SET status = %s WHERE ticket_no = %s', ("Agent Assigned", ticketno,) ) mysql.connection.commit() msg = 'Your complaint is Assigned to Agent !'

        mailcursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor) mailcursor.execute('SELECT customer_email FROM complaint_details WHERE ticket_no = %s', (ticketno,) ) customer_mail = mailcursor.fetchone()

        for x in customer_mail:
          cemail = customer_mail[x]
              try:
          mailmsg = Message('Customer Care Registry', sender = 'Agent Assigned', recipients = ['{}', cemail])
          mailmsg.body = "Hello,\nWe have received your complaint and agent {} has been Successfully Assigned\nYour Ticket Number: {}\n\nYou will be notified when your complain will be solved.".format(agentassign, ticketno) mail.send(mailmsg) except: pass
```

```
        return redirect(url_for('admindashboard'))       elif
request.method == 'POST':          msg = 'Please fill out the
form !'
  return          render_template('admindashboard.html',          msg          =          msg,          data=data,          agent=agent,
pending=pending, assigned=assigned, completed=completed)



@app.route('/admininanalytics') def
admininanalytics():
  mycursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)              mycursor.execute('SELECT
COUNT(agent_name) AS JenTile FROM complaint_details WHERE agent_name = %s', ("Jen Tile",))   JenTile
= mycursor.fetchall()          mycursor.execute('SELECT COUNT(agent_name) AS AllieGrater FROM
complaint_details WHERE agent_name = %s', ("Allie Grater",))        AllieGrater = mycursor.fetchall()
mycursor.execute('SELECT COUNT(agent_name) AS RaySin FROM complaint_details WHERE agent_name
= %s', ("Ray Sin",))      RaySin = mycursor.fetchall()      mycursor.execute('SELECT COUNT(category) AS
Category1 FROM complaint_details WHERE category = %s', ("Product Exchange or Return",))
  category1 = mycursor.fetchall()      mycursor.execute('SELECT COUNT(category) AS Category2 FROM
complaint_details WHERE category = %s', ("Product Out of Stock",))      category2 = mycursor.fetchall()
mycursor.execute('SELECT COUNT(category) AS Category3 FROM complaint_details WHERE category =
%s', ("Payments & Transactions",))        category3 = mycursor.fetchall()        mycursor.execute('SELECT
COUNT(category) AS Category4 FROM complaint_details WHERE category = %s', ("Product Delivery",))
category4 = mycursor.fetchall()        mycursor.execute('SELECT COUNT(category) AS Category5 FROM
complaint_details WHERE category = %s', ("Other",))    category5 = mycursor.fetchall()    print(category1)
  return          render_template('admininanalytics.html',          JenTile=JenTile,          AllieGrater=AllieGrater,
RaySin=RaySin,       category1=category1, category2=category2, category3=category3, category4=category4,
category5=category5)



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

```
            msgdecline = 'Oops! Something went wrong! Email not sent'          else:
            msgdecline = 'This email is not registered!'
    return render_template('customerforgotpassword.html', msgdecline = msgdecline)



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



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



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

```
    msgdecline = ''       if request.method == 'POST' and 'otp' in
request.form :
        otp = int(request.form['otp'])          if int(session['otp'])
== otp:             msgsuccess = 'success'             return
redirect(url_for('changepassword'))          else:
           msgdecline = 'You have entered wrong OTP'         elif request.method ==
'POST':          msg = 'Please fill out the form !'       return
render_template('enterotp.html', msgdecline = msgdecline)

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



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

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

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

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



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

Github And Project Demo Link

https://github.com/IBM-EPBL/IBM-Project-52498-1661007198.git

https://youtu.be/voGpjQ0257k