TITLE	**CUSTOMER CARE REGISTRY**
TEAM ID	**PNT2002TMID16135**

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Customer care is more than just providing great customer service. It's proactive approach to providing information, tools, and services to customers at each point they interact with a brand. This Application has been developed to help the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to a customer they will be notified with an email alert. Customers can view the status of the ticket till the service is provided. The main role and responsibility of the admin are to take care of the whole process. Starting from Admin login followed by the agent creation and assigning the customer's complaints. Finally, He will be able to track the work assigned to the agent and a notification will be sent to the customer.User can register for an account. After the login, they can create the complaint with a description of the problem they are facing. Each user will be assigned with an agent. They can view the status of their complaint.

1.2 PURPOSE

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

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

2.1.1 TITLE- CUSTOMER CARE REGISTRY

AUTHOR- Yusuf Indra Wibowo1

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 ini review customersatisfaction 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 risearch library is that: 1) Product Quality affects Customer Satisfaction; 2) Service Quality affects Customer Satisfaction; 3) Product Quality affects complaint level; 4) Service Quality affects complaint level; and 5) Customer Satisfaction affects complaint level.

2.1.2 TITLE- CUSTOMER CARE REGISTRY

AUTHOR- Shruthi Sivaprakasam J. Jayashree R. Shanmuga Priyan

DESCRIPTION

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

2.1.3 TITLE- CUSTOMER CARE REGISTRY

AUTHOR- Mona N. Shah, Vineet Raitani, Aditya Oza and Kunal Gupta(2017)

DESCRIPTION

Customer Satisfaction Study Of The Mumbai Metro Service". In this study they investigated about the service quality of the metro service based on the performance leading to

customer satisfaction. The survey was conducted and analyzed with SPSS tool. This survey is based on Gap 5 SERVQUAL model and identified the level of satisfaction with their parameter

2.1.4 TITLE- CUSTOMER CARE REGISTRY

AUTHOR- Rathod Piyush, Dr.Rajiv Bhatt and Dr. Jayesh Pitroda(2016)

DESCRIPTION

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

2.2 REFERENCES

1)Lucas, Robert (2015). Customer Service Skills For Success. New York: McGraw-Hill. ISBN 978-0-07-354546-2.

2)Buchanan, Leigh (1 March 2011). "A Customer Service Makeover". Inc. magazine. Retrieved 29 Oct 2012.

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

4)Bordoloi, Sanjeev (2019). Service Management Operations, Strategy, Information Technology. New York: McGraw-Hill. ISBN 978-1-260-09242-4.

2.3 PROBLEM STATEMENT DEFINITION

A Customer had occur a problem when they apply a ticket they need to recovery a solution or result .So the customer will contact a customer care for arise this issue. After the customer complaint, the company could identify that problem and solved this issue. Now the company wants to avoid these kinds of problems and technical issues So the company needs the customer satisfaction. This customer care registry helps to solve the issues and its find customer satisfaction

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours 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 challenges.

Figure 3.1.1 Empathy Map Canvas

3.2 IDEATION AND BRAINSTORMING

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

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

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

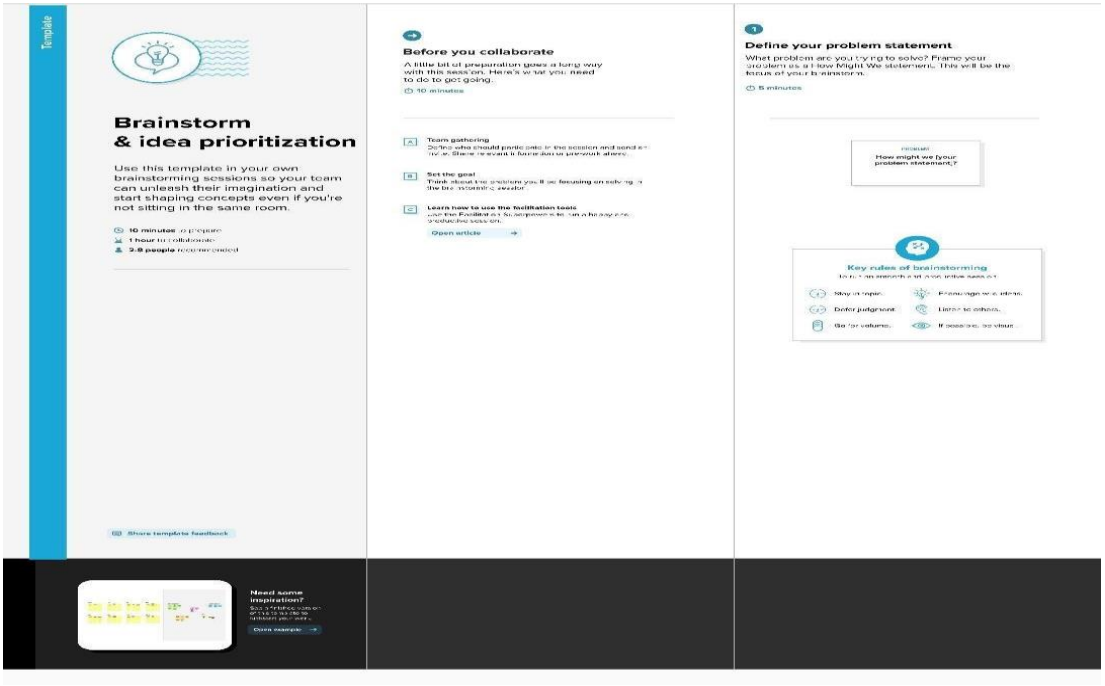


Figure 3.2.1 Team Gathering, Collaboration and Select the Problem Statement

Step-2: Brainstorm, Idea Listing and Grouping

The idea listing and grouping is used to organize and analyse large numbers of ideas by categorising them. By organising and reorganising ideas, students gain a better appreciation of, and dialogue about, their ideas. As students create idea clusters, new contexts and connections among themes emerge.

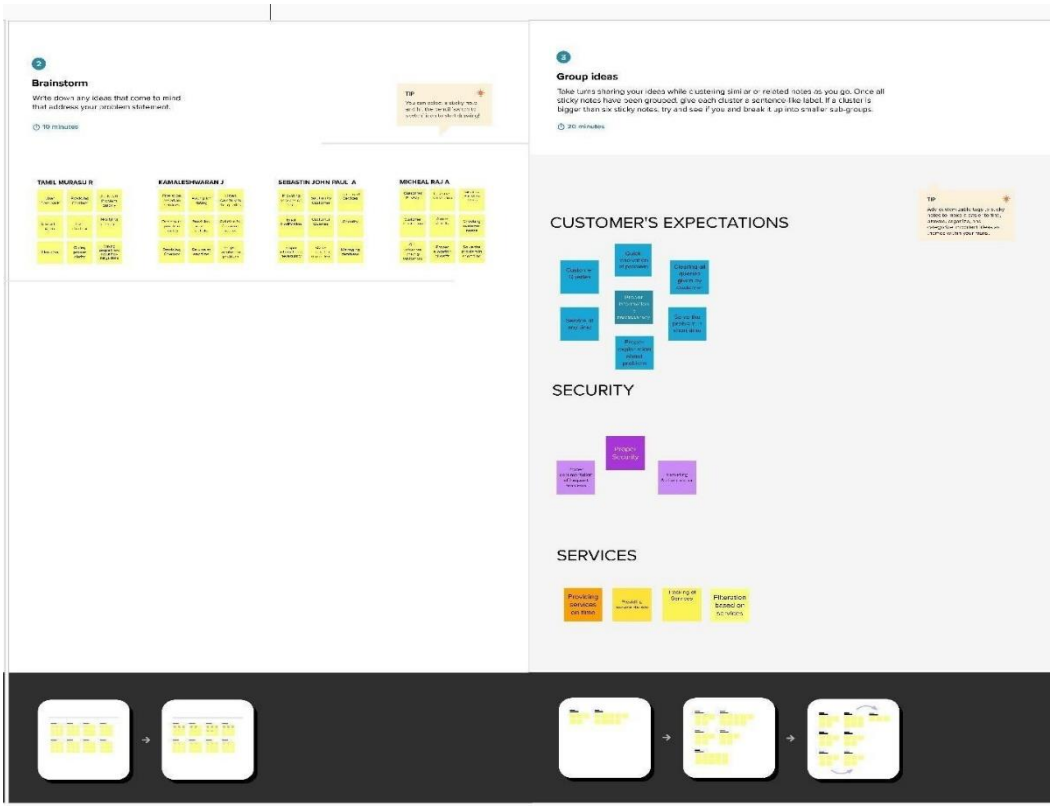


Figure 3.2.2 Brainstorm, Idea Listing and Grouping

Step-3: Idea Prioritization

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

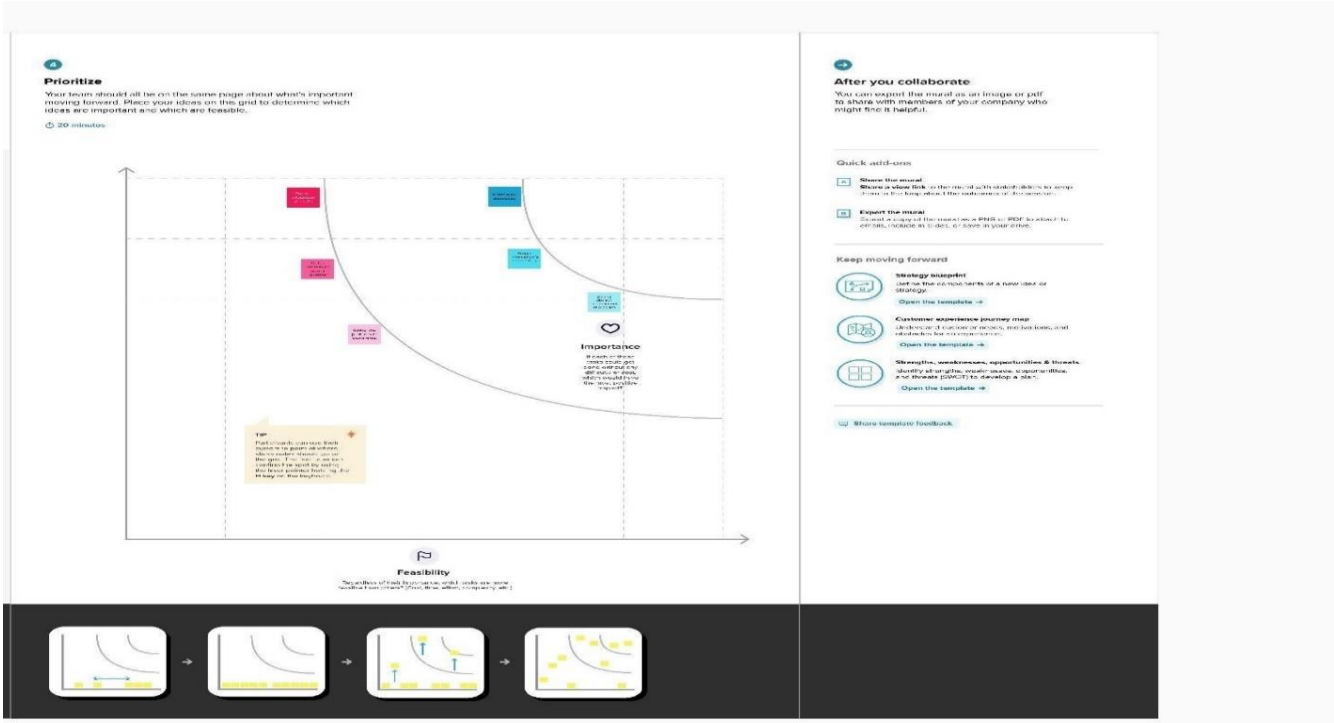


Figure 3.2.1 Idea Prioritization

3.3 PROPOSED SOLUTION

S.NO	PARAMETER	DESCRIPTION
01	Problem Statement (Problem to be solved)	To solve customer issues using Cloud Application Development.
02	Idea / Solution description	Assigned Agent routing can be solved by directly routing to the specific agent about theissue using the specific Email. Automated Ticket closure by using daily sync of the daily database. Status Shown to the Customer can display the status of the ticket to the customer.Regular data retrieval in the form of retrieving lost data.
03	Novelty / Uniqueness	Assigned Agent Routing, Automated Ticket Closure, Status Shown to the Customer, andBackup 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
06	Scalability of the Solution	The real goal of scaling customer service is providing an environment that will allow your customer service specialists to be as efficient as possible. An environment where they will be able to spend less time on gruntwork and more time on actually resolving critical customer issues

Table 3.3.1 Proposed Solution

3.4 PROBLEM SOLUTION FIT

-

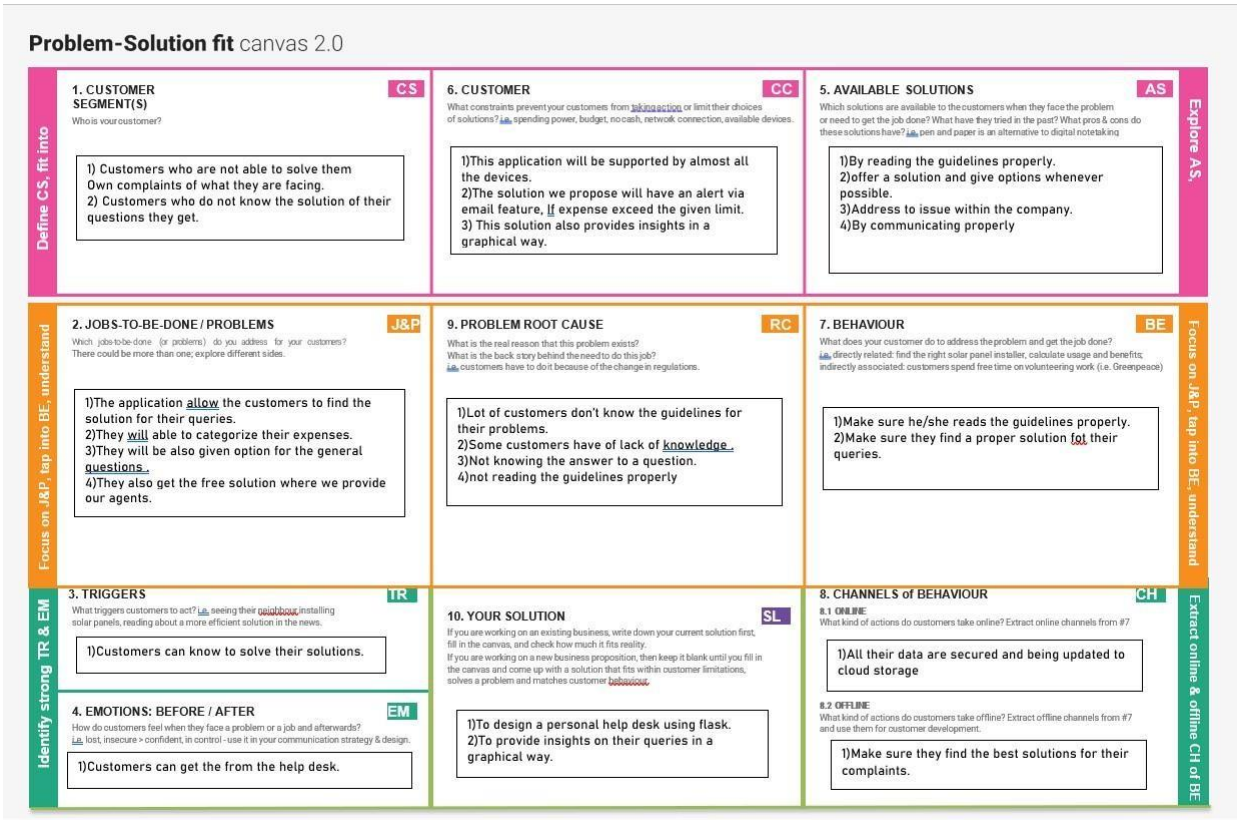


Figure 3.4.1 Problem Solution Fit

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

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

Table 4.1.1 Functional Requirement

4.2 NON-FUNCTIONAL REQUIREMENT

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	To provide optimal usability for our proposed solution we have mainly concentrated on easier navigation throughout our website. For user, they can easily login with their credentials and also they can register by themselves either with unique valid email id or with their mobile number if they don't have any prior account. After good navigation we have concentrated on visual clarity and developed web application which looks pleasant and simple thus making easier accessible to any aged person. For the first time users, Guide tour will also be available in order to provide better user satisfaction. Also, made our web application flexible to all type ofdevices such as android, mac and desktops.
NFR-2	Security	Before any user trying to login their account to any new device ,verification code will be sent either to their registered email id or to their registered mobile number. Only after entering their code, they will be allowed to login. That code will also made expire within particular time limit. Also notification will be sent for each and every customer
NFR-3	Reliability	Providing Quality Content
NFR-4	Availability	24/7 Support
NFR-5	Scalability	Good performance for large Customers and workload

Table 4.2.1 Non-Functional Requirement

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

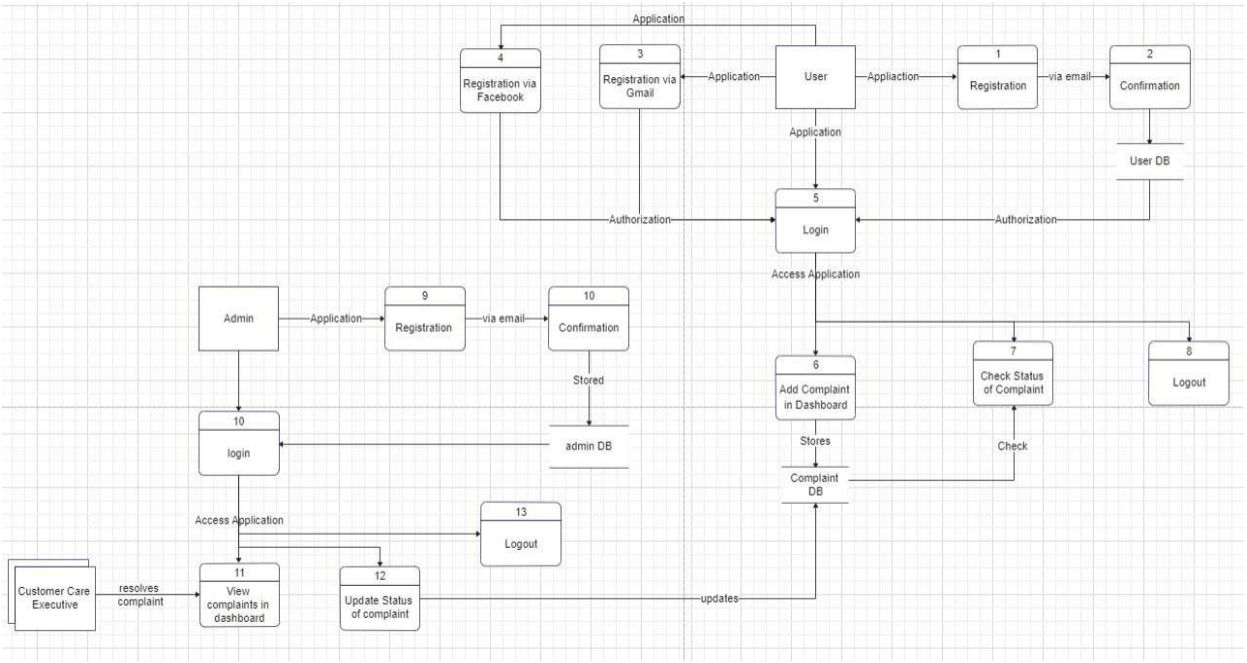


Figure 5.1.1 Data Flow of Customer care registry

5.2 SOLUTION AND TECHNICAL ARCHITECTURE

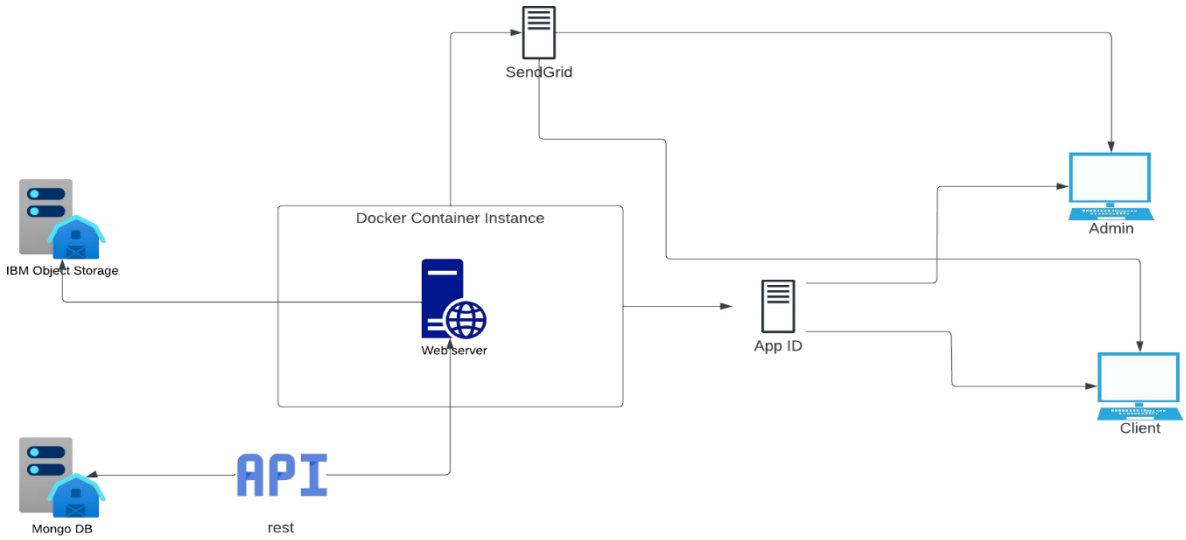


Figure 5.2.1 Solution Architecture

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story No	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email,password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-2
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user , I can register the complaint in the register complaint page	Ican register complaint	High	Sprint-1
		USN-7	As a user , I can view the status of the complaint.	I can view status of complaint	Medium	Sprint-1
		USN-8	As a user, I can logout of the application	I can logout from the application	Low	Sprint-2

Customer Care	Executive Dashboard	USN-8	As a customer care Executive, I can resolve a complaint registered by user.	Ican provide solution to a problem.	High	Sprint-1
Administrator	Registration	USN-9	As an admin, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-10	As an admin, I will receive confirmation email once I have registered for the application	I can receive confirmatio n email & click confirm	High	Sprint-1
	Login	USN-11	As an admin I can log intoapplication(admin panel) by entering email & password		High	Sprint-1
	Dashboard	USN-12	As an admin, I can update the status of the complaint to the user with the help of customer care executive.	I can satisfy the customer on his/her query.	Medium	Sprint-2
		USN-13	As an admin , I can logout from the application	I can logout from the application	Low	Sprint-2

Table 5.3.1 User Story

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	User Type	Functional Requirements	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Customer (Web User)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	2	High	Bavya, Hemalatha
Sprint-1		Login	USN-2	As a customer, I can login to the application by entering correct email and password	1	High	Hemalatha
Sprint-1		Dashboard	USN-3	As a customer, I can see all the tickets raised by meand lot more	3	High	Monisha
Sprint-2		Ticket creation	USN-4	As a customer I can create a new ticket with the detailed description of my query	2	High	Kavindra
Sprint-3		Address column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	3	High	Hemalatha, Kavindra

Sprint-4		Forget password	USN-6	As a customer, I can reset my password by this optionin case I forgot my old password	2	Medium	Monisha, Kavindra
Sprint-4		Ticket details	USN-7	As a customer, I can see the current status of mytickets	2	Medium	Kavindra, Bavya
Sprint-3	Agent (Web User)	Login	USN-1	As an agent, I can login to the application by entering correct email and password	2	High	Monisha
Sprint-3		Dashboard	USN-2	As an agent, I can see all the ticketsassigned to me by the admin	3	High	Kavindra
Sprint-3		Address Column	USN-3	As an agent, I get to have conversations with the customer and clear his/her queries	3	High	Hemalatha, Kavindra
Sprint-4		Forget Password	USN-4	As an agent, I can reset my password by this option in case I forgot my old password	2	Medium	Monisha, Kavindra
Sprint-1	Admin (Web User)	Login	USN-1	As an admin, I can login to the application by entering correct email and password	1	High	Bavya, Monisha
Sprint-1		Dashboard	USN-2	As an admin, I can see all the tickets raised in the entire system and lot more	3	High	Monisha, Kavindra
Sprint-2		Agent Creation	USN-3	As an admin, I can create an agent for clarifying the customer’s queries	2	High	Monisha, Hemalatha

Sprint-2		Assigning agent	USN-4	As an admin, I can assign an agent for each ticketcreated by the customer	3	High	Hemalatha, Bavya
Sprint-4		Forget password	USN-4	As an admin, I can reset my password by this option in case I forgot my old password	2	Medium	Kavindra, Bavya

Table 6.1.1 Sprint Planning & Estimation

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total story points	Duration	Sprint start date	Sprint end date	Story points completed	Sprint release date
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022
Sprint-2	7	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022
Sprint-3	11	6 Days	07 Nov 2022	12 Nov 2022	11	12 Nov 2022
Sprint-4	8	6 Days	14 Nov 2022	19 Nov 2022	8	19 Nov 2022

Table 6.2.2 Sprint Delivery Schedule

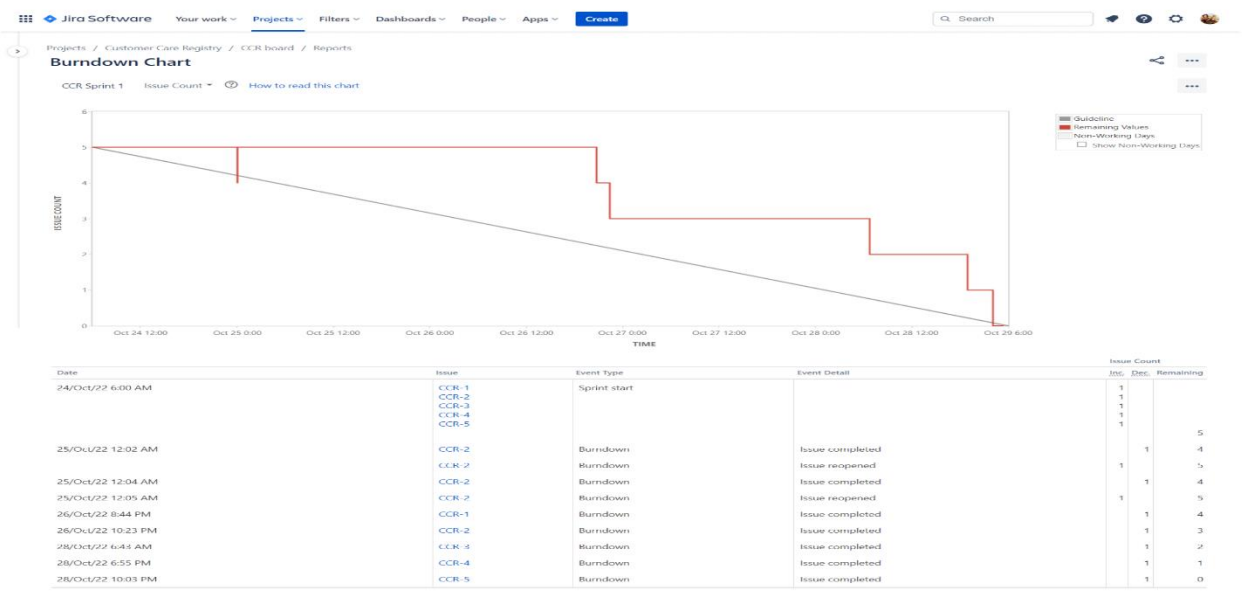
Velocity:

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) periteration unit (story points per day)

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

6.3 REPORTS FROM JIRA



Date	Issue	Event Type	Event Detail	Issue Count Inc.	Dec.	Remaining
24/Oct/22 6:00 AM	CCR-1 CCR-2 CCR-3 CCR-4 CCR-5	Sprint start		1 1 1 1 1		5
25/Oct/22 12:02 AM	CCR-2	Burndown	Issue completed		1	4
	CCR-2	Burndown	Issue reopened	1		5
25/Oct/22 12:04 AM	CCR-2	Burndown	Issue completed		1	4
25/Oct/22 12:05 AM	CCR-2	Burndown	Issue reopened	1		5
26/Oct/22 8:44 PM	CCR-1	Burndown	Issue completed		1	4
26/Oct/22 10:23 PM	CCR-2	Burndown	Issue completed		1	3
28/Oct/22 6:43 AM	CCR-3	Burndown	Issue completed		1	2
28/Oct/22 6:55 PM	CCR-4	Burndown	Issue completed		1	1
28/Oct/22 10:03 PM	CCR-5	Burndown	Issue completed		1	0

Figure 6.3.1 Reports from JIRA

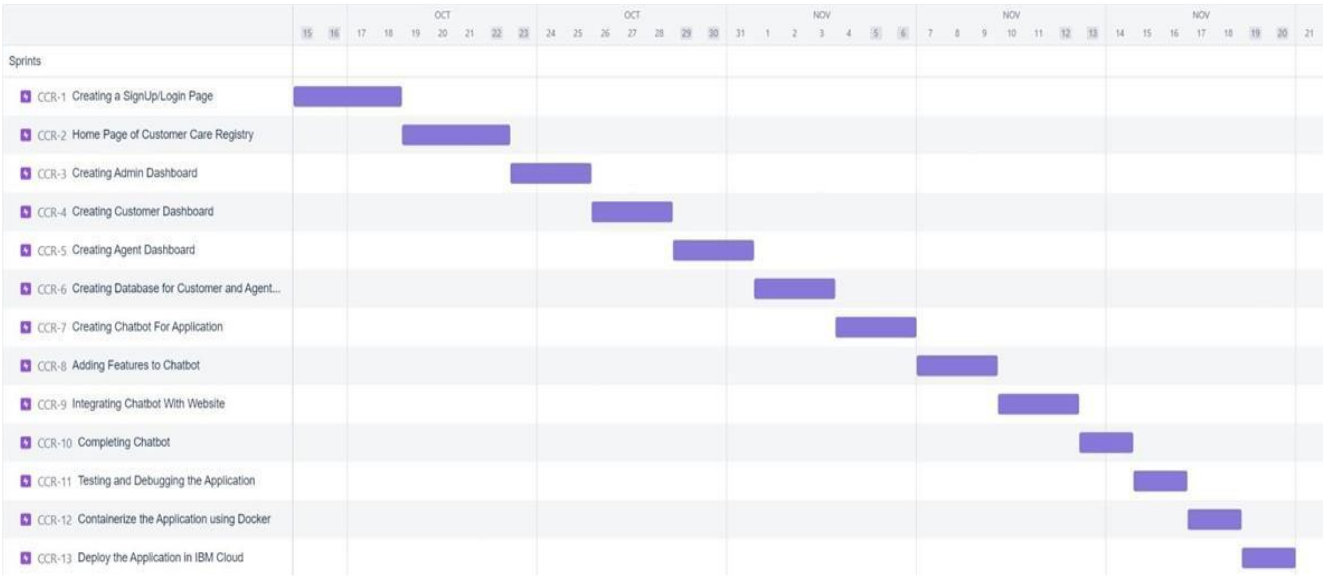


Figure 6.3.2 Reports from JIRA

CHAPTER 7

CODING AND SOLUTIONING

College graduates with prior programming expertise or technical degrees are recruited and transitioned into professional positions with Alabama firms and organizations through the highly competitive Coding Solutions job accelerator andtalent refinement programme at no cost to the graduates. We provide a pool of varied, well-trained, techs-savvy individuals that wants to launch and advance their career in Alabama. The mission of veteran- and woman-owned Coding Solutions is to mobilize the next generation of IT talent and provide them the tools and resources they require to make your business successful. Innovative talent is necessary for innovative technologies. We wish to provide Coding Solutions prospects to assist you expand your Alabama team. Our applicants are swiftly hired at the top of the list by growing businesses forlucrative, long-term positions.

7.1 Features 1

7 Main types of customer needs

- User-friendly
- Empathy
- Fairness
- Control
- Alternatives
- Information

7.2 Features 2

- Complaint Tracking
- Email Alert
- 24/7 Monitoring

CHAPTER 8

TESTING

8.1 TEST CASES

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

Characteristics of a good test case:

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

S.no	Scenario	Input	Excepted output	Actual output
1	Admin Login Form	User name and password	Login	Login success.
2	Employee Login Form	User name and password	Login	Login success.
3	User Registration Form	User basic details	Registered successfully	User basic details are stored in the database.
4	User Login Form	User name and password	Login	Login success.

Table 8.1.1 Test Cases

8.2 USER ACCEPTANCE TESTING

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

8.2.1 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	Subtotal
By Design	10	3	1	2	17
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	40
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	13	12	25	78

Table 8.2.1.1 Defect Analysis

8.2.2 TESTCASE ANALYSIS

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

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

Table 8.2.2.1 Testcase analysis

CHAPTER 9

RESULTS

9.1 PERFORMANCE METRICS

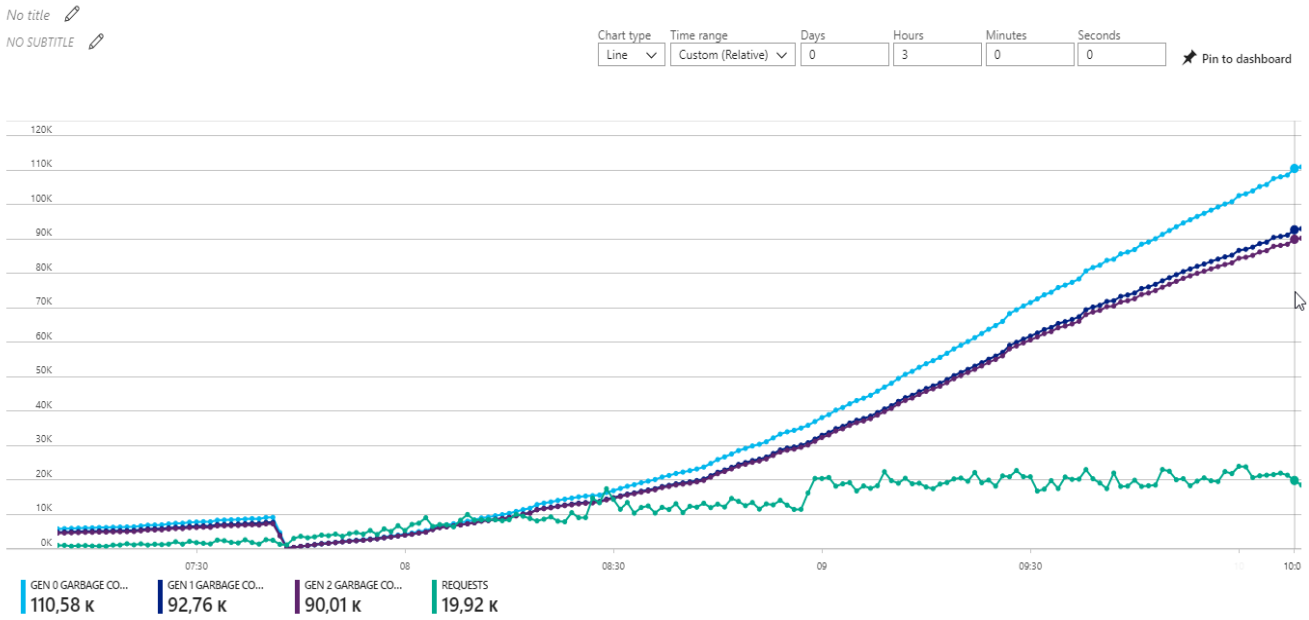


Figure 9.1.1 Performance Metrics

Figure 9.1.2 Performance Metrics

CHAPTER 10

ADVANTAGES AND DIS ADVANTAGES

ADVANTAGES

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

DISADVANTAGES

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

CHAPTER 11

CONCLUSION

Application software has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly, and has required option, which can be utilized by the user to perform the desired operations. Application meets the information requirements specified to a great extent. The system has been designed keeping in view the present and future requirements in mind and made very flexible. The goals that are achieved by the software are Instant access, improved productivity, Optimum utilization of resources, Efficient management of records, Simplifications of the operations, Less processing time and getting required information, User friendly, Portable and flexible for further enhancement. The system has the benefits of easy access because it is be developed as a platform independent web application, so the admin can maintain a proper contact with their users, which may be access anywhere. All communications between the police and administrator has done through the online, so this communication cost also is reduced.

CHAPTER 12

FUTURE 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.

CHAPTER 13

APPENDIX

SOURCE CODE

```
from flask import Flask, render_template, flash, request, session,send_file
from flask import render_template, redirect, url_for, request

import ibm_db
import pandas
import ibm_db_dbi
from sqlalchemy import create_engine

engine = create_engine(&#39;sqlite://&#39;,
echo = False)

dsn_hostname = &quot;b70af05b-76e4-4bca-a1f5-
23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud&quot;
dsn_uid = &quot;crc83247&quot;
dsn_pwd = &quot;eHGBftxhodLDnNpM&quot;

dsn_driver = &quot;{IBM DB2 ODBC DRIVER}&quot;
dsn_database = &quot;BLUDB&quot;
dsn_port = &quot;32716&quot;
dsn_protocol = &quot;TCPIP&quot;
dsn_security = &quot;SSL&quot;

dsn = (
&quot;DRIVER={0};&quot;
&quot;DATABASE={1};&quot;
&quot;HOSTNAME={2};&quot;
&quot;PORT={3};&quot;
&quot;PROTOCOL={4};&quot;
&quot;UID={5};&quot;
&quot;PWD={6};&quot;
&quot;SECURITY={7};&quot;).format(dsn_driver, dsn_database, dsn_hostname, dsn_port,
dsn_protocol, dsn_uid, dsn_pwd,dsn_security)
```

```
try:
conn = ibm_db.connect(dsn, &quot;&quot;, &quot;&quot;)
print (&quot;Connected to database: &quot;, dsn_database, &quot;as user: &quot;, dsn_uid, &quot;on
host: &quot;,
dsn_hostname)

except:
print (&quot;Unable to connect: &quot;, ibm_db.conn_errormsg() )

app = Flask(__name__)
app.config[&#39;DEBUG&#39;]
app.config[&#39;SECRET_KEY&#39;] = &#39;7d441f27d441f27567d441f2b6176a&#39;

@app.route(&quot;/&quot;)
def homepage():

return render_template(&#39;index.html&#39;)

@app.route(&quot;/AdminLogin&quot;)
def AdminLogin():

return render_template(&#39;AdminLogin.html&#39;)

@app.route(&quot;/UserLogin&quot;)
def UserLogin():
return render_template(&#39;UserLogin.html&#39;)

@app.route(&quot;/NewUser&quot;)
def NewUser():
return render_template(&#39;NewUser.html&#39;)

@app.route(&quot;/NewComplaint&quot;)
def NewComplaint():
user = session[&#39;uname&#39;]
return render_template(&#39;NewComplaint.html&#39;,uname=user)
```

```
@app.route(&quot;/NewAgent&quot;)
def NewAgent():

conn = ibm_db.connect(dsn, &quot;&quot;, &quot;&quot;)
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = &quot;SELECT * FROM agenttb where &quot;
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql(&#39;booktb1&#39;, con=engine, if_exists=&#39;append&#39;)
data = engine.execute(&quot;SELECT * FROM booktb1&quot;).fetchall()

return render_template(&#39;NewAgent.html&#39;,data=data)

@app.route(&quot;/AdminHome&quot;)
def AdminHome():
conn = ibm_db.connect(dsn, &quot;&quot;, &quot;&quot;)
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = &quot;SELECT * from regtb &quot;
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql(&#39;Employee_Data&#39;,
con=engine,
if_exists=&#39;append&#39;)

# run a sql query
data = engine.execute(&quot;SELECT * FROM Employee_Data&quot;).fetchall()

return render_template(&#39;AdminHome.html&#39;,data=data)

@app.route(&quot;/UserHome&quot;)
def UserHome():
user = session[&#39;uname&#39;]

conn = ibm_db.connect(dsn, &quot;&quot;, &quot;&quot;)
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = &quot;SELECT * FROM regtb where UserName= &#39;&quot; + user + &quot;&#39;
&quot;
```

```
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql(&#39;booktb1&#39;, con=engine, if_exists=&#39;append&#39;)
data = engine.execute(&quot;SELECT * FROM booktb1&quot;).fetchall()
return render_template(&#39;UserHome.html&#39;,data=data)

@app.route(&quot;/UserComplaint&quot;)
def UserComplaint():
user = session[&#39;uname&#39;]

conn = ibm_db.connect(dsn, &quot;&quot;, &quot;&quot;)
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = &quot;SELECT * FROM booktb where UserName= &#39;&quot; + user + &quot;&#39;
&quot;
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql(&#39;booktb1&#39;, con=engine, if_exists=&#39;append&#39;)
data = engine.execute(&quot;SELECT * FROM booktb1&quot;).fetchall()

return render_template(&#39;UserComplaint.html&#39;,data=data)

@app.route(&quot;/AdminComplaintInfo&quot;)
def AdminComplaintInfo():

conn = ibm_db.connect(dsn, &quot;&quot;, &quot;&quot;)
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = &quot;SELECT * FROM booktb &quot;
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql(&#39;booktb1&#39;, con=engine, if_exists=&#39;append&#39;)
data = engine.execute(&quot;SELECT * FROM booktb1&quot;).fetchall()

return render_template(&#39;AdminComplaintInfo.html&#39;,data=data)

@app.route(&quot;/adminlogin&quot;, methods=[&#39;GET&#39;, &#39;POST&#39;])
def adminlogin():
error = None
```

if request.method == 'POST':
if request.form['uname'] == 'admin' or request.form['password'] == 'admin':

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from regtb "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')

# run a sql query
data = engine.execute("SELECT * FROM Employee_Data").fetchall()
return render_template('AdminHome.html' , data=data)

else:
return render_template('index.html', error=error)

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

if request.method == 'POST':

username = request.form['uname']
password = request.form['password']
session['uname'] = request.form['uname']

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from regtb where UserName='" + username + "' and

password='" + password + "'"
dataframe = pandas.read_sql(selectQuery, pd_conn)

if dataframe.empty:
data1 = 'Username or Password is wrong'
return render_template('goback.html', data=data1)
else:
print("Login")
selectQuery = "SELECT * from regtb where UserName='" + username + "' and
password='" + password + "'"
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')

# run a sql query
data = engine.execute("SELECT * FROM Employee_Data").fetchall()

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

@app.route("/newuser", methods=['GET', 'POST'])
def newuser():
if request.method == 'POST':

name1 = request.form['name']

gender1 = request.form['gender']
Age = request.form['age']
email = request.form['email']
pnumber = request.form['phone']
address = request.form['address']

uname = request.form['uname']
password = request.form['psw']

conn = ibm_db.connect(dsn, "", "")

insertQuery = "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 + "','" +
Age + "','" + email + "','" + pnumber + "','" + address + "','" + uname + "','" + password + "')"
insert_table = ibm_db.exec_immediate(conn, insertQuery)
print(insert_table)

return render_template('UserLogin.html')

@app.route("/newage", methods=['GET', 'POST'])
def newage():
if request.method == 'POST':

name1 = request.form['name']
gender1 = request.form['gender']
Age = request.form['age']
email = request.form['email']
pnumber = request.form['phone']
address = request.form['address']

uname = request.form['uname']

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

insertQuery = "INSERT INTO agenttb VALUES ('" + name1 + "','" + gender1 + "','" +
Age + "','" + email + "','" + pnumber + "','" + address + "','" + uname + "')"
insert_table = ibm_db.exec_immediate(conn, insertQuery)
print(insert_table)

selectQuery = "SELECT * FROM agenttb "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()

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

@app.route("/newcom", methods=['GET', 'POST'])
def newcom():
if request.method == 'POST':

name = request.form['name']
com = request.form['com']
uname = session['uname']

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * FROM booktb"
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('booktb', con=engine, if_exists='append')
data2 = engine.execute("SELECT * FROM booktb").fetchall()

count = 0

for item in data2:
count += 1

Bookingid = "COMID00" + str(count)

insertQuery = "INSERT INTO booktb VALUES ('" + Bookingid + "','" + uname + "','" +
com + "','','')"
insert_table = ibm_db.exec_immediate(conn, insertQuery)
print(insert_table)

selectQuery = "SELECT * FROM booktb where UserName= '" + uname + "' "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()
return render_template('UserComplaint.html', data=data)

@app.route("/AgentAssign", methods=['GET'])
def AgentAssign():
cid = request.args.get('id')
session['cid'] = cid

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * FROM agenttb "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()

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

@app.route("/Action", methods=['GET'])
def Action():
cid = request.args.get('id')
session['cid'] = cid

return render_template('Action.html')

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

agid = request.form['agid']

cid = session['cid']

uname = session['uname']

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery1 = "SELECT * FROM regtb where UserName='" + uname + "'"
dataframe = pandas.read_sql(selectQuery1, pd_conn)

dataframe.to_sql('regtb', con=engine, if_exists='append')
data1 = engine.execute("SELECT * FROM regtb").fetchall()

for item1 in data1:
Mobile = item1[5]
Email = item1[4]
sendmsg(Email,"Assign Agent id"+agid)

insertQuery = "update booktb set AgentName='"+ agid +"' where ComplaintId='"+ cid +"'
"
insert_table = ibm_db.exec_immediate(conn, insertQuery)

alert = 'Agent Assign Send Notication'

return render_template('goback.html', data=alert)

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

com = request.form['com']

cid = session['cid']

uname = session['uname']

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery1 = "SELECT * FROM regtb where UserName='" + uname + "'"
dataframe = pandas.read_sql(selectQuery1, pd_conn)

dataframe.to_sql('regtb', con=engine, if_exists='append')
data1 = engine.execute("SELECT * FROM regtb").fetchall()

for item1 in data1:
Mobile = item1[5]
Email = item1[4]
sendmsg(Email,"Action Information "+com)

insertQuery = "update booktb set ACTIONINFO='"+ com +"' where ComplaintId='"+ cid
+"' "

insert_table = ibm_db.exec_immediate(conn, insertQuery)

alert = 'Action Info Saved Send Notication'

return render_template('goback.html', data=alert)

def sendmsg(Mailid,message):
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders

fromaddr = "sampletest685@gmail.com"
toaddr = Mailid

```
# instance of MIMEMultipart
msg = MIMEMultipart()

# storing the senders email address
msg[&#39;From&#39;] = fromaddr

# storing the receivers email address
msg[&#39;To&#39;] = toaddr

# storing the subject
msg[&#39;Subject&#39;] = &quot;Alert&quot;

# string to store the body of the mail
body = message

# attach the body with the msg instance
msg.attach(MIMEText(body, &#39;plain&#39;))

# creates SMTP session
s = smtplib.SMTP(&#39;smtp.gmail.com&#39;, 587)



# start TLS for security
s.starttls()

# Authentication
s.login(fromaddr, &quot;hneucvnontsuwgpj&quot;)

# Converts the Multipart msg into a string
text = msg.as_string()

# sending the mail
s.sendmail(fromaddr, toaddr, text)
```

```
# terminating the session
s.quit()
if __name__ == &#39;__main__&#39;:
app.run(debug=True, use_reloader=True)
```

GITHUB LINK

IBM-EPBL/IBM-Project-19179-1659694077