PROJECT REPORT

Customer Care Registry

Project Name : Customer Care Registry

Project Domain : Cloud Application Development

College : Adhiparasakthi Engineering College

SPOC : Dr. C. Dhaya, ph.D.,

Team ID : PNT2022TMID38567

Team Size : 4

Team Leader : Dharani shree A

Team Member: Jyothisri P

Team Member: Kalaivani A

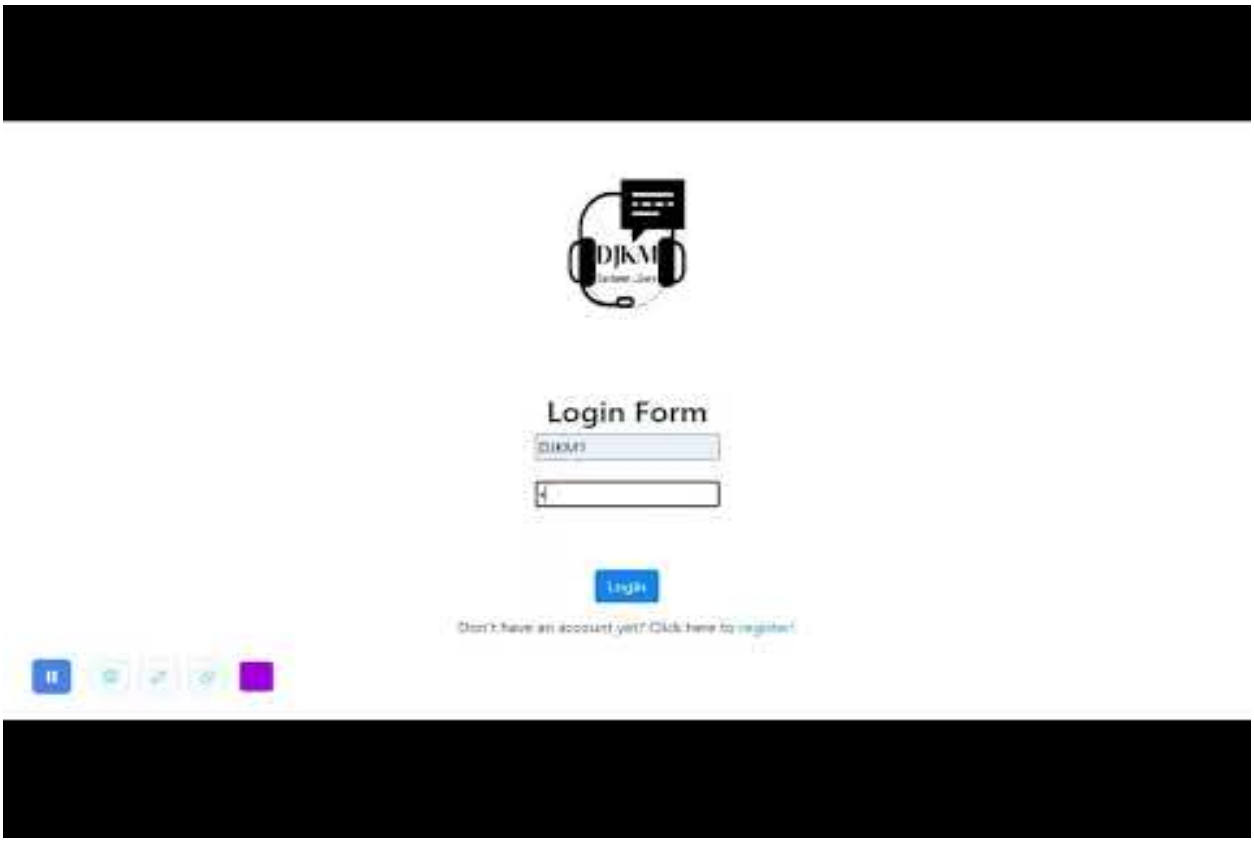
Team Member: Malini M

Team Mentor :Mrs. Srividya, M.E

Team Evaluator : Charmadhurai J

Github Link : https://github.com/IBM-EPBL/IBM-Project-14418-1659585439

Project Demo Link :



Project Report

1. **INTRODUCTION**
    1. Project Overview
    2. Purpose
2. **LITERATURE SURVEY**
    1. Existing problem
    2. References
    3. Problem Statement Definition

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

Source Code

GitHub & Project Demo Link

CHAPTER 1

INTRODUCTION

Customer is the king because he keeps every business afloat. Whether an organisation offers a product or service, it cannot remain in business if it cannot find a group of people willing to become its customers. Taking care of a customer's needs and solving their problems is called customer service. Customer service begins the moment you connect with the customer to fulfil his needs and continues even after the requirements are met. The services might be required before, during and after the customer purchases a product or service 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.

1.1 Project Overview

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.

ADMIN: 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 sent to the customer.

User: They 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

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.

CHAPTER 2

LITERATURE SURVEY

A literature review is a comprehensive summary of previous research on a topic. The literature review surveys scholarly articles, books, and other sources relevant to a particular area of research. The review should enumerate, describe, summarise, objectively evaluate and clarify this previous research.

2.1 Existing problem

- That the customers are categorised based on Purchase behaviours, historical ordering patterns and frequency of purchase customise customer care and promotions are given.
- Customer trust chatbots to provide the required support. Chatbots represent a potential means for automating customer service.
- The assists consumers in decision making. Based on the computer or Social Actors Paradigm.
- We employ the software as a service(SaaS) model which introduces drastic improvement to the situation , as the service provider can now have direct access to the user data and access to the user data and analyse it if agreed appropriately with the customer.

2.2 References

- NA Harun, SH Huspi, NA Lahad - “Question classification framework for helpdesk ticketing support system using machine learning” - 2021.
- F Alqodri, PHP Agustyana, A Masytho - “Helpdesk ticket support system based on fuzzy tahani algorithm” – 2021.
- H Hardianto, IM Shofi, D Khairani - “Integration of the helpdesk system with messaging service: A case study” -2021.
- GMD silva, S Thakare - “Real world smart chatbot for customer care using a software as a service (SAAS) architecture” - 2017

2.3 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.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

• 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 help teams to better understand their users
• Creating an effective solution requires understanding the true problem and the person who us 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

3.2 Ideation & Brainstorming

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

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

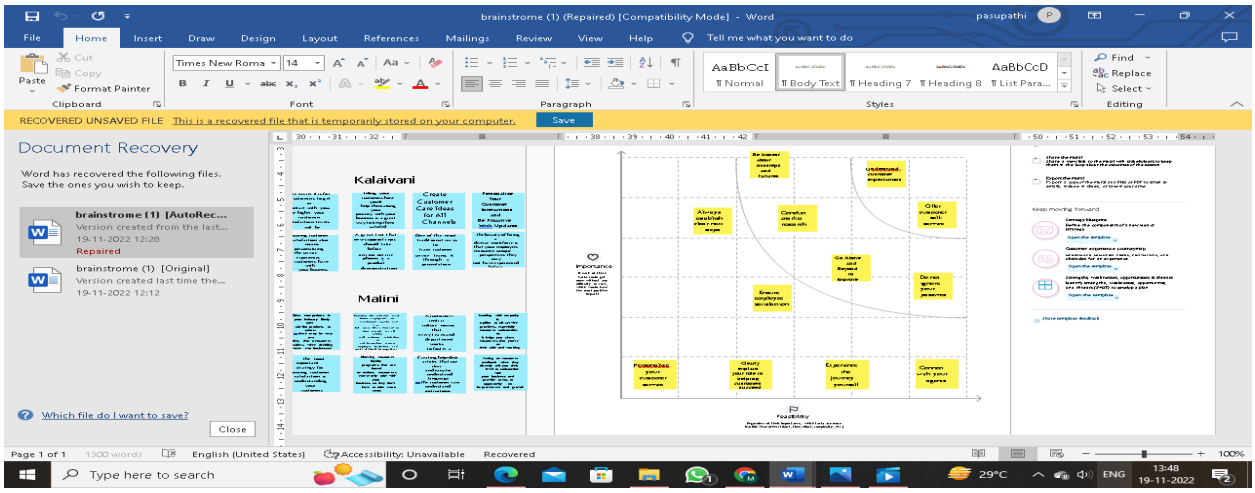
Team Leader: Dharani shree A
Team Members: Jyothisri P
Kalaivani A
Malini M

Customer Problem Statement :

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love. A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

Step-2: Brainstorm, Idea Listing and Grouping

Step-3: Idea Prioritisation



3.3 Proposed Solution

Project team shall fill the following information in the proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	A Customer has a problem when they apply a ticket they need to recover a solution or result. So the customer will contact a customer care for arise the issue. After the customer complaint, the company could identify that problem and solve this issue. Now the company wants to avoid these kinds of problems and technical issues So the company needs customer satisfaction. This customer care registry helps to solve the issues and find customer satisfaction.
2.	Idea / Solution description	Customer service solutions are products or services that businesses use to gain a deeper understanding of their customers' needs and expectations. They work to streamline and improve customer communications, therefore increasing customer satisfaction.
3.	Novelty / Uniqueness	Respond promptly. Know your product or service. Listen to your customers. Say thank you. Get to know your customers. Ask for feedback. Use the feedback you receive.

4.	Social Impact / Customer Satisfaction	Deliver personalised service on the most popular social media channels. Increase customer satisfaction and brand reputation by joining or starting conversations on the social channels preferred by each customer. Intelligently route social posts to the right agents or team.
5.	Business Model (Revenue Model)	A revenue model dictates how a business will charge customers for a product or service to generate revenue. Revenue models prioritise the most effective ways to make money based on what is offered and who pays for it
6.	Scalability of the Solution	Improve customer service practices and processes. Invest in better tools. Scale your self-service and knowledge base. Build a larger customer service team.

3.4 Problem Solution Fit

Project Title: Customer Care Registry

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID38567

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. Kids

1)The customers who are not able to solve their queries.
2)The customers can solve their problems by raising the tickets.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

1)This application is supported by all the devices.
2)The solution we propose will have an alert via email feature

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem
or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

1)By communicating properly with an agent.
2)By reading the guidelines properly.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

1)Customer can find the solution for the query that he/she are raised.
2)They can also solve the raised query by using chatbot.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do
this job?
i.e. customers have to do it because of the change in regulations.

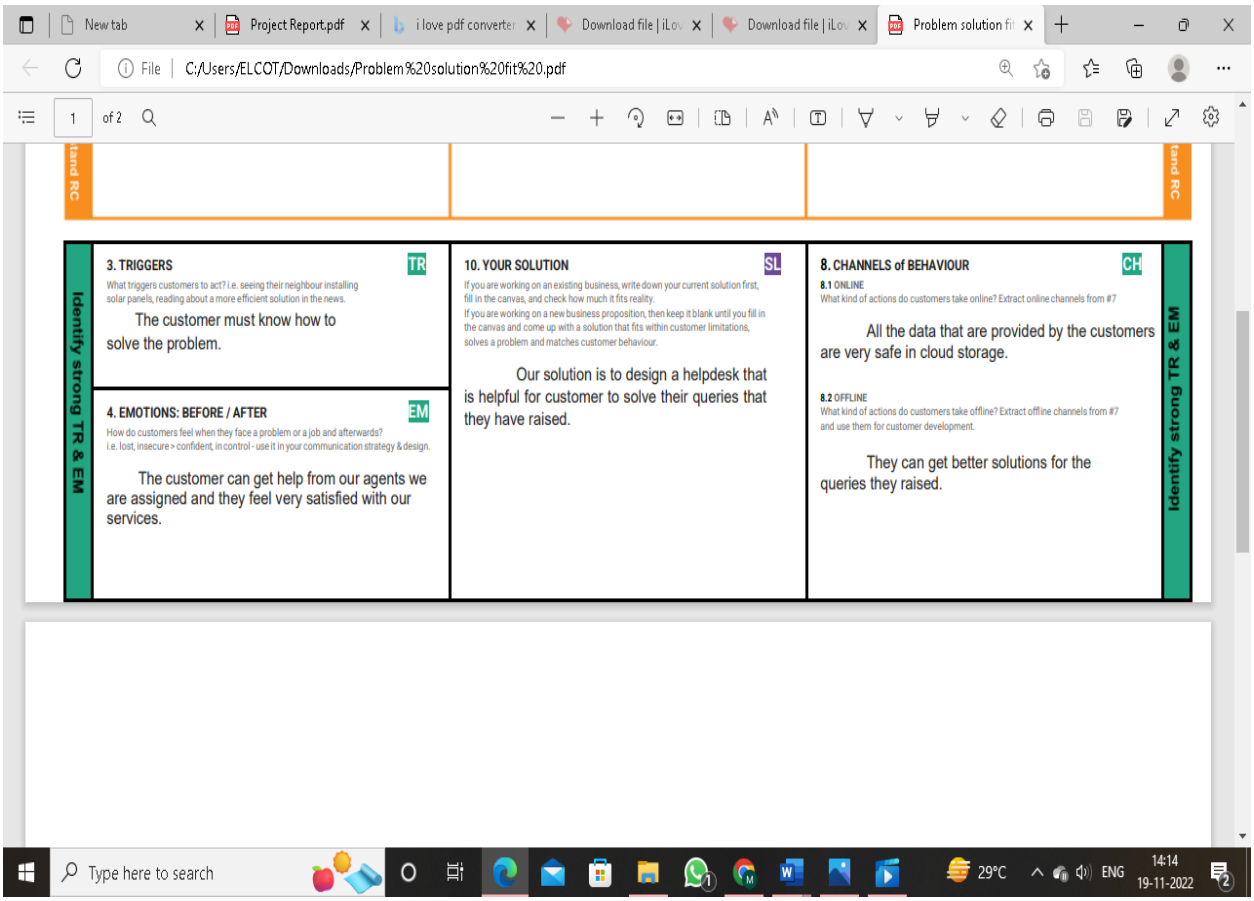
1)Not reading the guidelines properly.
2)some of the customers have lack of knowledge.
3)Lots of customers have not reads the guidelines properly.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

1)All the customers must read the guidelines properly to avoid the problem.
2)All the customer should find a proper solution for their queries.

Focus on J&P, tap into BE, understand RC



3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

The customer must know how to solve the problem.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

The customer can get help from our agents we are assigned and they feel very satisfied with our services.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

Our solution is to design a helpdesk that is helpful for customer to solve their queries that they have raised.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

All the data that are provided by the customers are very safe in cloud storage.

8.2 OFFLINE

What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

They can get better solutions for the queries they raised.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirements

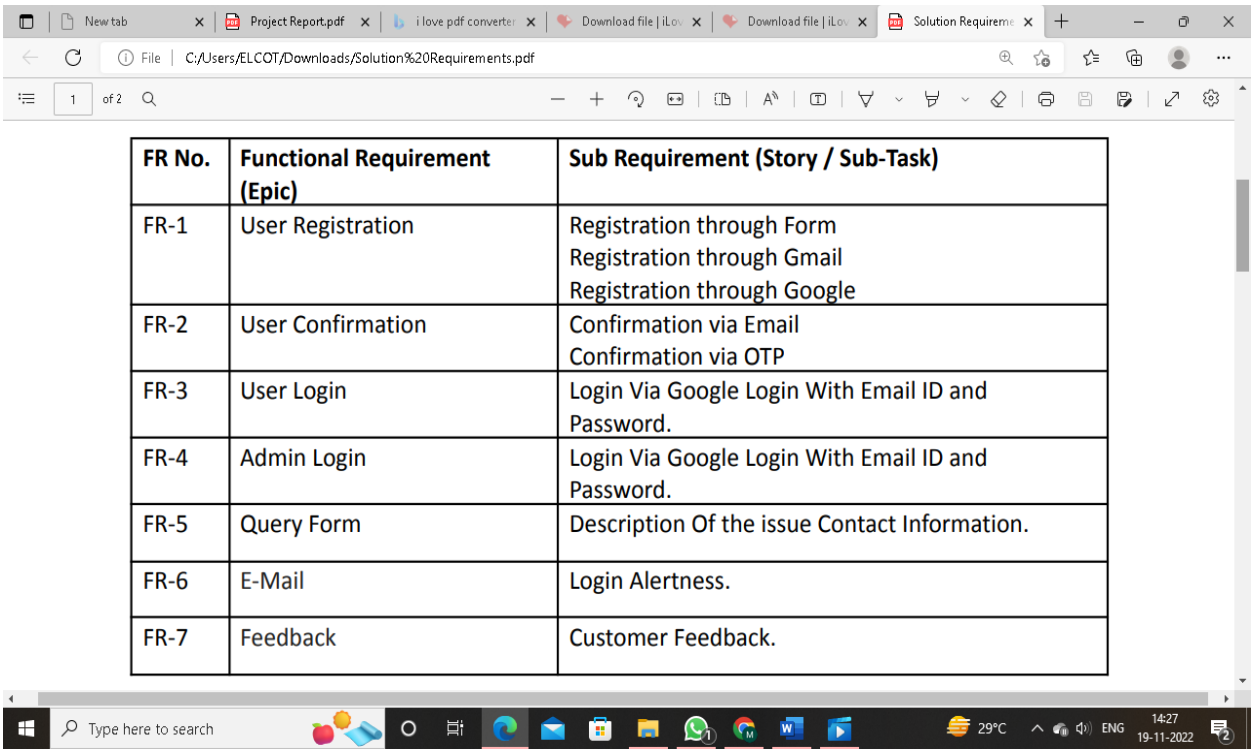
➢ A functional requirement defines a function of a system or its component, where a function is described as a specification of behaviour between inputs and outputs.

➢ It specifies “what should the software system do?”

➢ Defined at a component level

➢ Usually easy to define

➢ Helps you verify the functionality of the software



FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through Google
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login Via Google Login With Email ID and Password.
FR-4	Admin Login	Login Via Google Login With Email ID and Password.
FR-5	Query Form	Description Of the issue Contact Information.
FR-6	E-Mail	Login Alertness.
FR-7	Feedback	Customer Feedback.

4.2 Non-functional Requirements

- A non-functional requirement defines the quality attribute of a software system
- It places constraint on “How should the software system fulfil the functional requirements?”
- It is not mandatory
- Applied to system as a whole
- Usually more difficult to define
- Helps you verify the performance of the software

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	To Provide a solution to Customer's Problem.
NFR-2	**Security**	Authentication using login ID and Password.
NFR-3	**Reliability**	Tracking Of Problem Ticket Status through E-Mail.
NFR-4	**Performance**	It Can be accessed on all devices with Browser Comatibility.

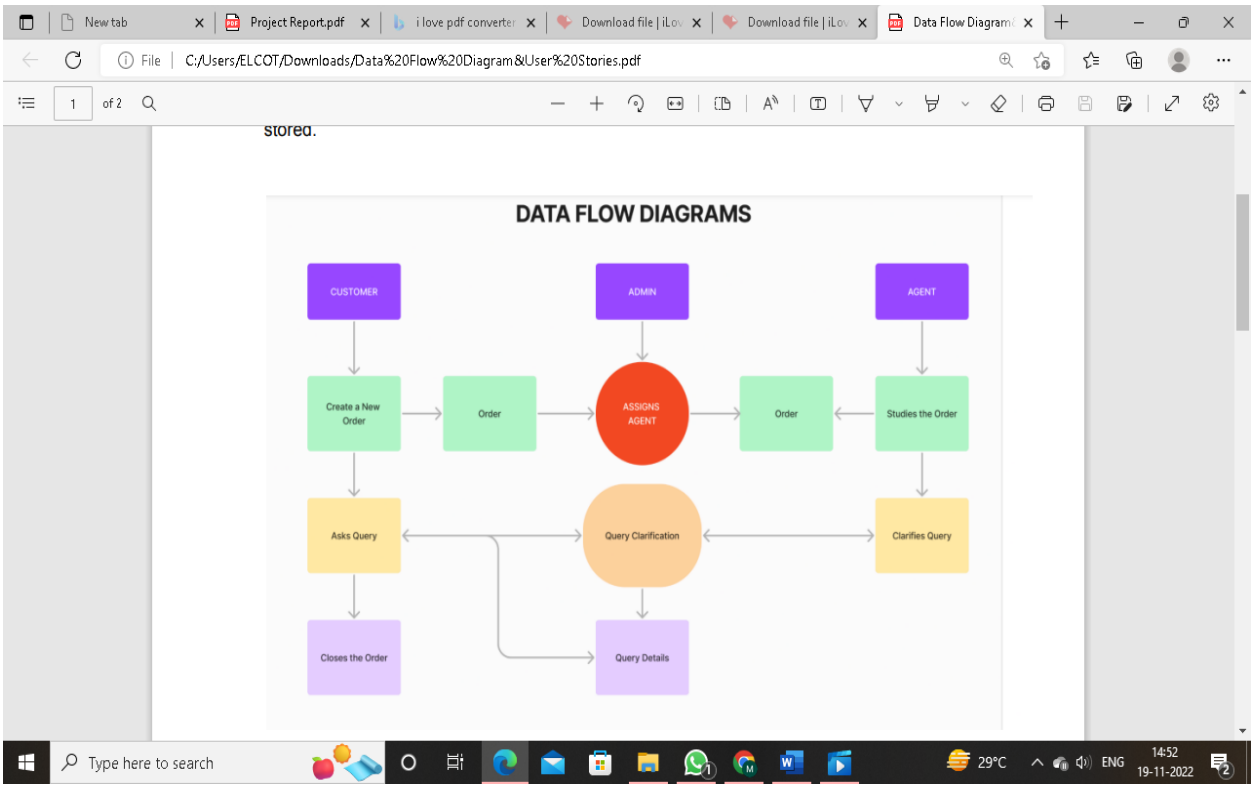
NFR-5	**Availability**	24/7 Services.
NFR-6	**Scalability**	Agents Scalability as per the number of customers.

CHAPTER 5

PROJECT DESIGN

5.1 Data Flow Diagram

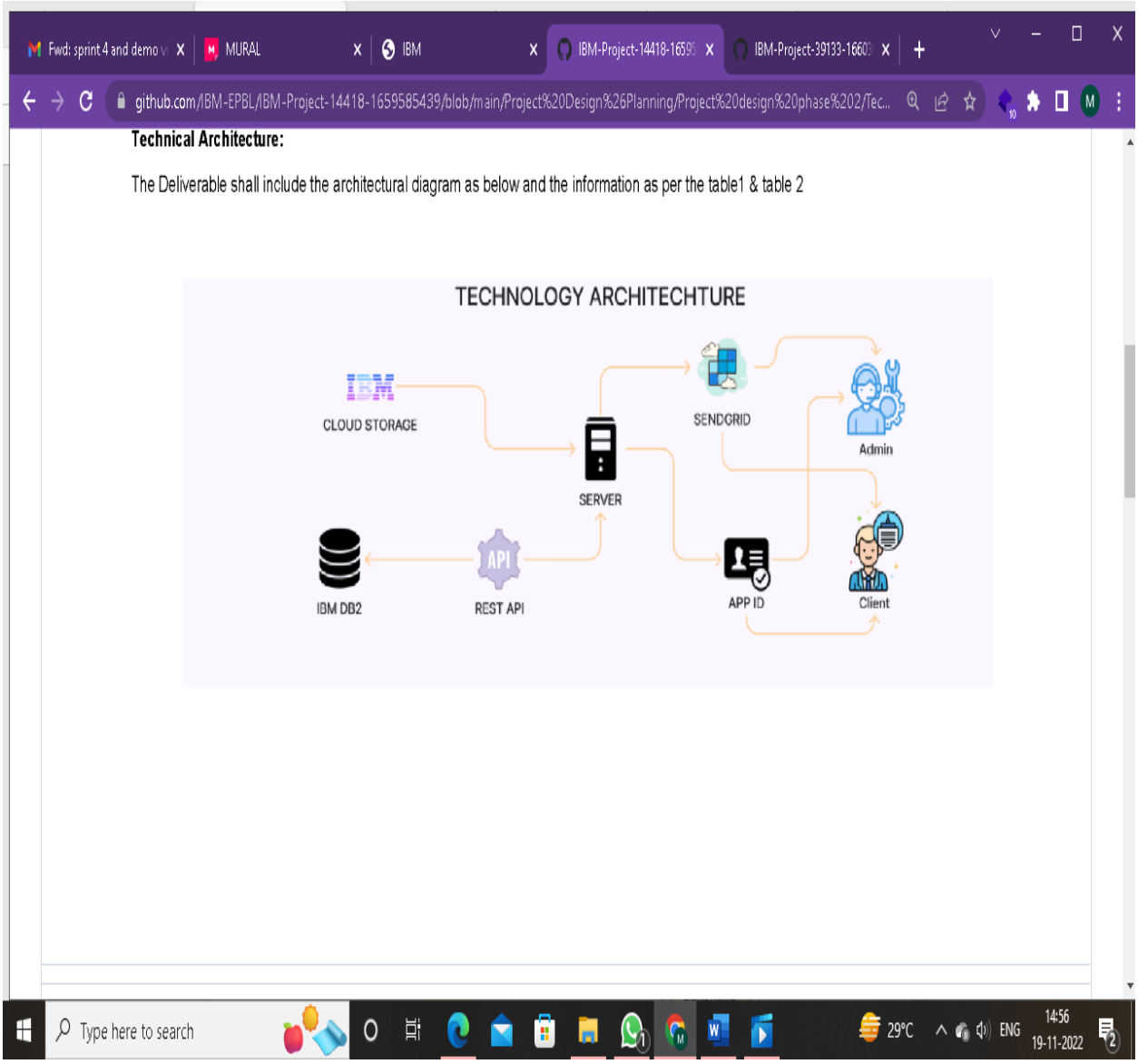
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2

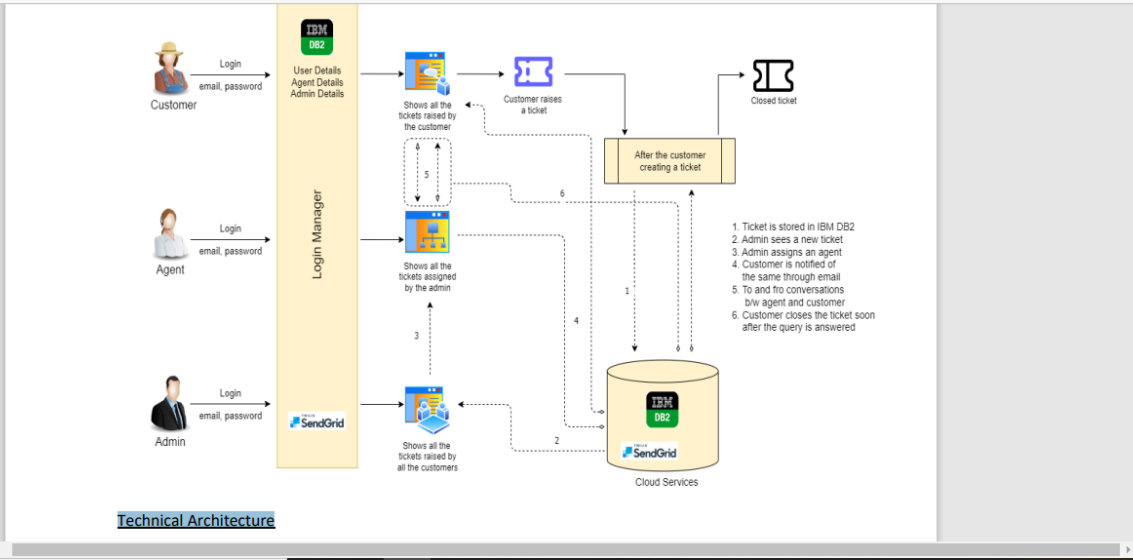
Solution and Technical Architecture

Technical Architecture



Solution Architecture

	Agent Creation	USN-3	As an admin, I can create an agent for clarifying the customer queries.	I can create agents	High	Sprint-2
	Assignment Agent	USN-4	As an admin, I can Assign an agent for each Thread created by the customer.	Enable agents 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 again.	High	Sprint-1



Technical Architecture

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a Customer, I can register for the application by entering my email, password, and confirming my password	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a Customer, I will receive a confirmation email.	I can receive confirmation email & click confirm	High	Sprint-1
	Dashboard	USN-3	As a Customer, I can see all the orders raised by me.	I can get all the info needed in my dashboard.	Low	Sprint-2
	Order Creation	USN-4	As a Customer, I can place my order with a 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 agent and get my queries clarified.	My queries are clarified.	High	Sprint-3
	Forgot Password	USN-6	As a Customer, I can see the current status of the order.	I can get access to my account again.	Medium	Sprint-4
	Order Details	USN-7	As a Customer, I can see the current status of the order.	I can get a better understanding.	Medium	Sprint-4
Agent	Login	USN-1	As an agent, I can Login to the	I can access my	High	Sprint-3

CHAPTER 6

PROJECT DESIGN AND PLANNING

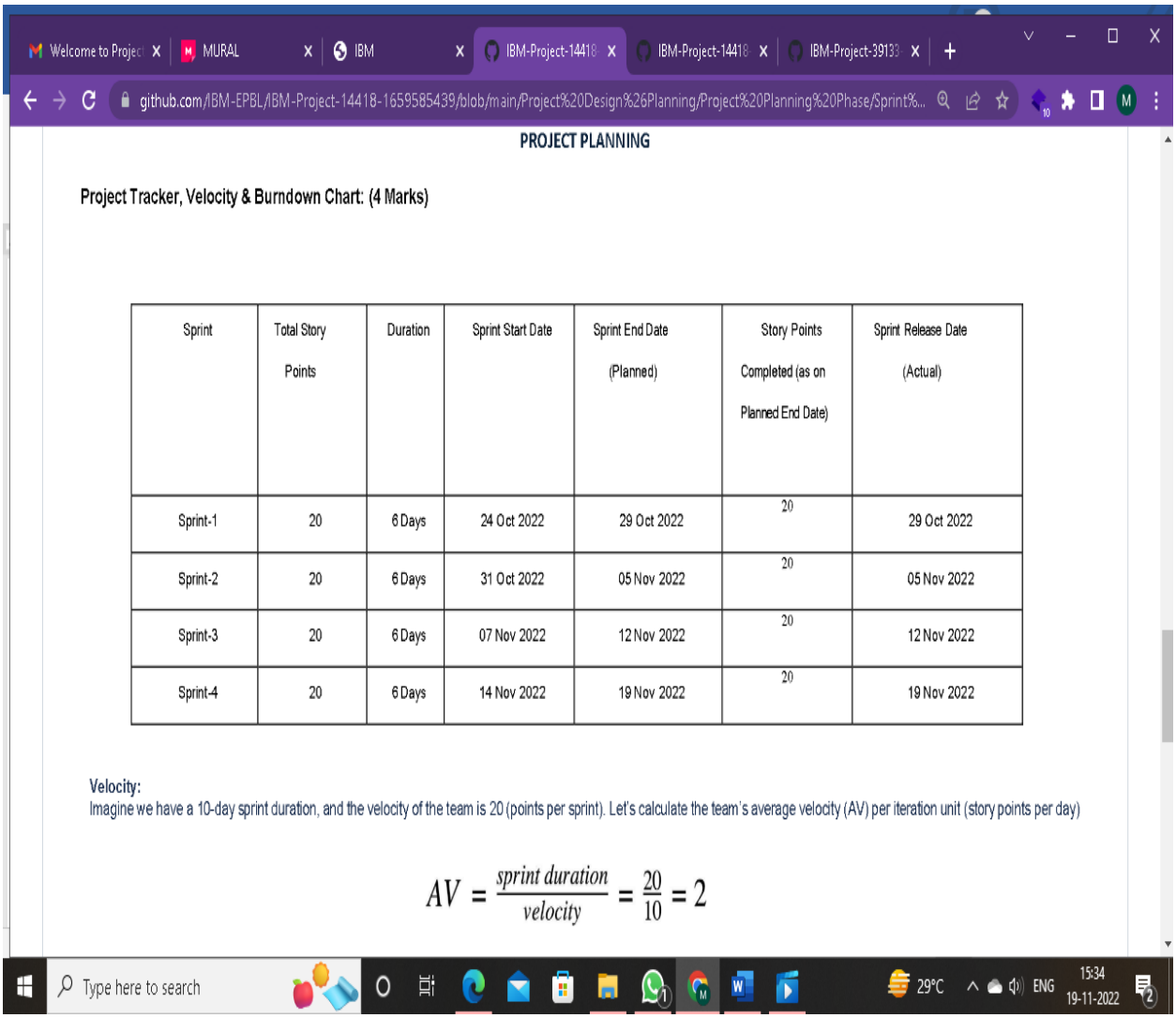
1. Sprint Planning and Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and gothrough the services available on the webpage	20	High Low	Dharani Shree.A Kalaivani.A
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the availability and have a trackof all the things that the users are going to service	20	High Medium	Jyothisri.P Malini.M
Sprint-3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding the services. Get the recommendations based on information provided by the user.	20	High Medium Low	Dharani Shree.A Kalaivani.A Jyothisri.P Malini.M
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes and deployment the application.Create the documentation and final submit the application	20	High	Dharani Shree.A Kalaivani.A Jyothisri.P Malini.M

2. Sprint Delivery Plan



PROJECT PLANNING

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

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

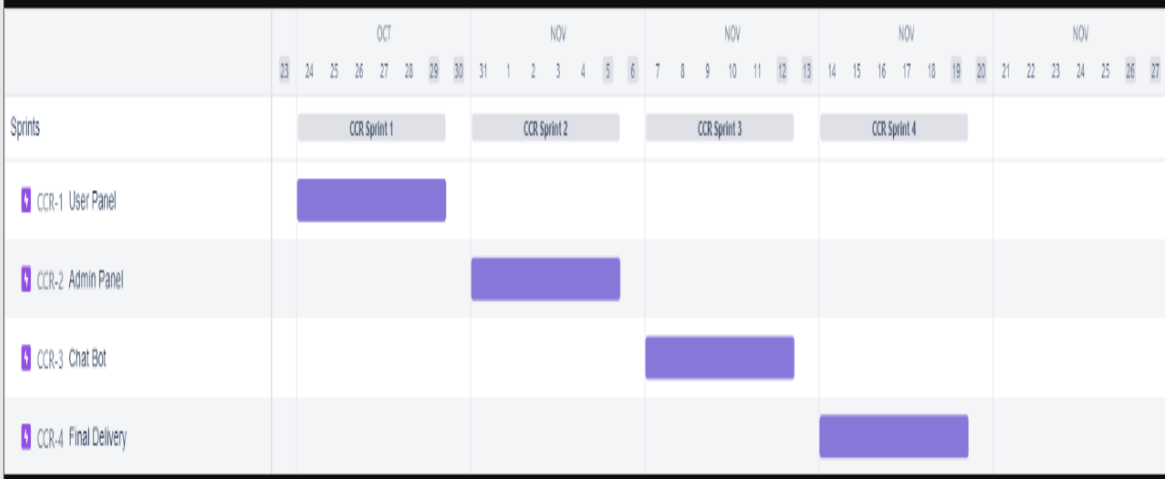
$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

6.3Reports from JIRA

Sprint 1,2,3,4 – Burndown Chart

Welcome to Project | MURAL | IBM | customer_care_reg | IBM-Project-14418 | IBM-Project-39133

raw.githubusercontent.com/IBM-EPBL/IBM-Project-14418-1659585439/main/Project%20Design%26Planning/Project%20Planning%20Phase/Jir...



Type here to search

Poll... ENG 15:44 19-11-2022

Roadmap


Epics
PS-1 CREATING REGISTER/LOGIN PAGE
PS-2 HOME PAGE OF CUSTOMER CARE REGISTRY
PS-3 CREATING AGENT LOGIN PAGE
PS-4 CREATING DATABASE FOR AGENT AND CUSTOM...
PS-5 COMPLETING THE USER PANEL
PS-6 CREATING UI FOR ADMIN PANEL
PS-7 CREATING DATABASE CONNECTION FOR USER P...
PS-8 CREATING CHATBOT FOR APPLICATION
PS-10 ADDING FEATURES OF CHATBOT
PS-11 INTEGRATE CHATBOT WITH WEBSITE
PS-12 COMPLETING CHATBOT
PS-13 TESTING & DEBUGGING THE APPLICATION
PS-14 CONTAINERIZE THE APPLICATIONS
PS-15 DEPLOY THE APPLICATION

CHAPTER 7

CODING AND SOLUTIONING

7.1 Admin assigning an agent to a ticket Code:

```
@admin.route('/admin/update/<agent_id>/<ticket_id>')
@login_required
def assign(agent_id, ticket_id):
    '''
        Assigning an agent to the ticket
    '''
    from .views import admin

    if(hasattr(admin, 'email')):
        # query to update the ASSIGNED_TO of a ticket
        assign_agent_query = '''
            UPDATE tickets SET assigned_to = ? WHERE ticket_id = ?
        '''

        stmt = ibm_db.prepare(conn, assign_agent_query)
        ibm_db.bind_param(stmt, 1, agent_id)
        ibm_db.bind_param(stmt, 2, ticket_id)

        ibm_db.execute(stmt)

        return "None"

    else:
        # logging out
        return redirect(url_for('blue_print.logout'))
```

Explanation:

• User creates a ticket by describing the query.

• Admin views the newly created ticket in the dashboard.

• In the dropdown given, admin selects an agent.

• Once selected, using fetch() the request is sent to the server.

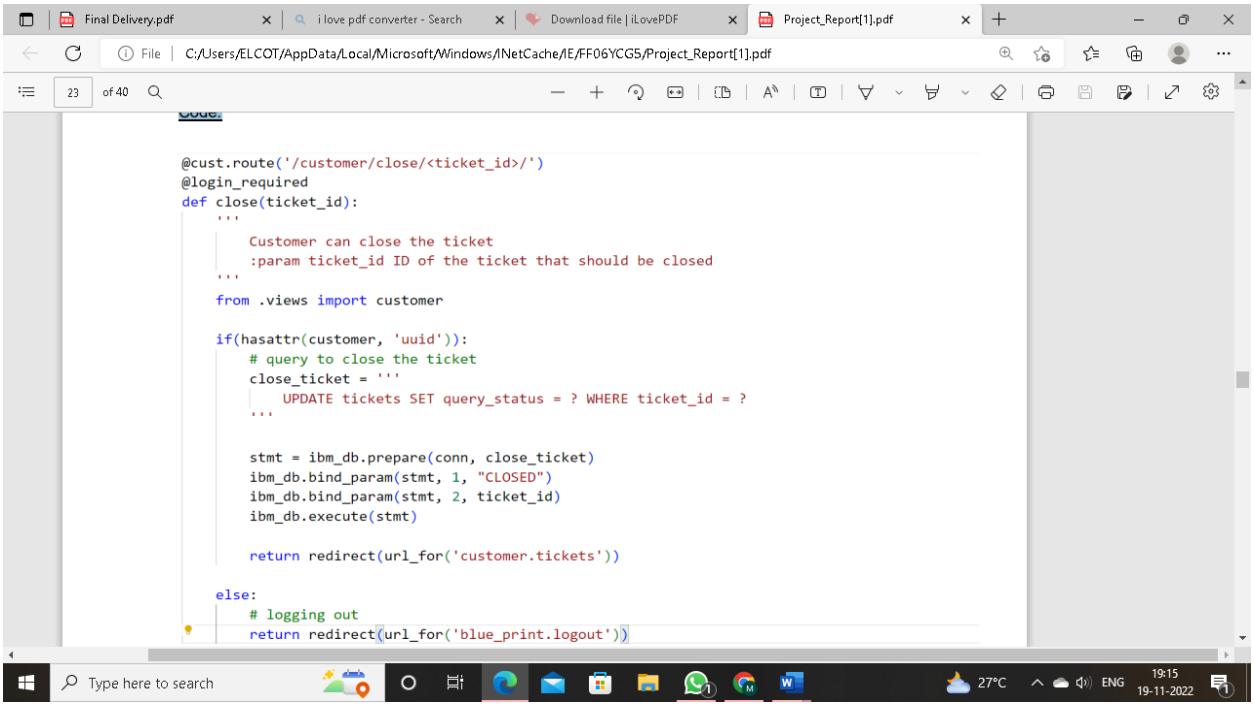
• The request URL contains both the Ticket ID and the selected Agent ID.

• Using the shown SQL query, the assigned_to column of the tickets table is set to agent_id where the ticket_id column = ticket_id.

• Then, the dashboard of the admin gets refreshed.

7.2 Customer closing a ticket

Code:



```
@cust.route('/customer/close/<ticket_id>/')
@login_required
def close(ticket_id):
    '''
        Customer can close the ticket
        :param ticket_id ID of the ticket that should be closed
    '''
    from .views import customer

    if(hasattr(customer, 'uuid')):
        # query to close the ticket
        close_ticket = '''
            UPDATE tickets SET query_status = ? WHERE ticket_id = ?
        '''

        stmt = ibm_db.prepare(conn, close_ticket)
        ibm_db.bind_param(stmt, 1, "CLOSED")
        ibm_db.bind_param(stmt, 2, ticket_id)
        ibm_db.execute(stmt)

        return redirect(url_for('customer.tickets'))

    else:
        # logging out
        return redirect(url_for('blue_print.logout'))
```

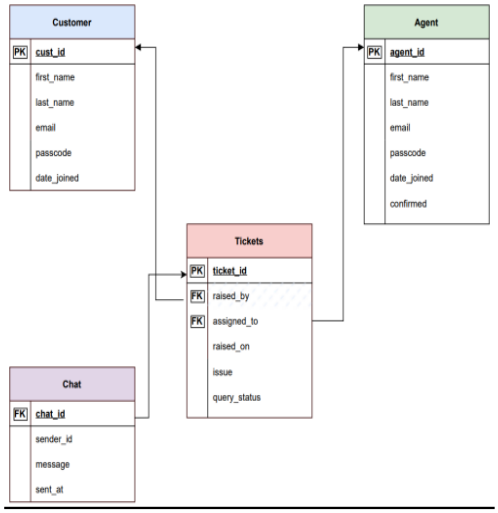
Explanation:

- User creates a ticket by describing the query
- Admin assigns an agent to this ticket
- The customer and the agent, chat with each other, in the view of clearing the customer's doubts
- Once the customer is satisfied, the customer decides to close the ticket
- Using fetch() the request is sent to the server. The requested URL contains the Ticket ID
- Using the shown SQL query, the status of the ticket is set to "CLOSED"
- Thus the ticket is closed
- Then the customer gets redirected to the all-tickets page

7.3 Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

it useful.



CHAPTER 8

TESTING

8.1 Test Cases

The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not. Test case designing includes preconditions, case name, input conditions, and expected result. A test case is a first level action and derived from test scenarios.

Test case gives detailed information about testing strategy, testing process, preconditions, and expected output. These are executed during the testing process to check whether the software application is performing the task for that it was developed or not.

Test case helps the tester in defect reporting by linking defect with test case ID. Detailed test case documentation works as a full proof guard for the testing team because if developer missed something, then it can be caught during execution of these full-proof test cases.

To write the test case, we must have the requirements to derive the inputs, and the test scenarios must be written so that we do not miss out on any features for testing. Then we should have the test case template to maintain the uniformity, or every test engineer follows the same approach to prepare the test document.

8.2 User Acceptance Testing

1. 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).

2. 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	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

3. 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

CHAPTER 9

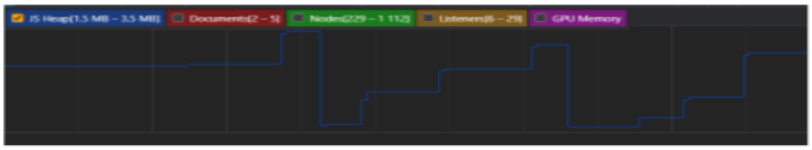
RESULTS

9.1 Performance Metrics:

CPU usage:

✓ Since all the operations run using Flask is in server-side, the client (browser) need not worry about the CPU usage. Just rendering the page, static contents take place in the client-side.

✓ Memory for client-side functions (Javascript) is allocated using heap. It can be either increased based upon the requirement or removed from the heap.

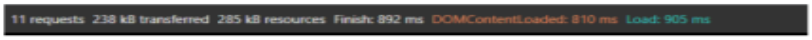


Errors:

- ✓ Since all the backend functions are done using flask, any exceptions / errors rising are well-handled. Though they appear, user's interaction with the site is not affected in any way

Latency and Response time:

It takes less than a second to load a page in the client. From this it is evident that there is low latency



CHAPTER 10

ADVANTAGES AND DISADVANTAGES

Advantages:

✓ Customers can clarify their doubts just by creating a new ticket.

✓ Customer gets replies as soon as possible.

✓ Not only the replies are faster, the replies are more authentic and practical.

✓ Customers are provided with a unique account, to which the latter can login at any time.

✓ Very minimal account creation process.

✓ Customers can raise as many tickets as they want.

✓ Application is very simple to use, with well-known UI elements.

✓ Customers are given clear notifications through email, of all the processes related lo login, ticket creation etc.,

✓ Customers' feedbacks are always listened

✓ Free of cost

Disadvantages:

× Only web application is available right now (as of writing).
× UI is not so attractive, it's just simple looking. × No automated replies.
× No SMS alerts.
× Supports only text messages while chatting with the Agent.
× No tap to reply feature.
× No login alerts.
× Cannot update the mobile number.
× Account cannot be deleted, once created.
× Customers cannot give feedback to the agent for clarifying the queries

CHAPTER 11

CONCLUSION

Thus, there are many customer service applications available on the internet. Noting down the structural components of those applications and we built a customer care registry application. It will be a web application build with Flask (Python micro-web framework), HTML, JavaScript. It will be a ticket-based customer service registry.

Customers can register into the application using their email, password, first name and last name. Then, they can login to the system, and raise as tickets as they want in the form of their tickets.

These tickets will be sent to the admin, for which an agent is assigned. Then, the assigned agent will have a one-to-one chat with the customer and the latter's queries will be clarified. It is also the responsibility of the admin, to create an agent.

CHAPTER 12

FUTURE SCOPE

Our application is not finished yet. There are many rooms for improvement. Some of them will be improved in the future versions

- ✓ Attracting and much more responsive UI throughout the application.
- ✓ Releasing cross-platform mobile applications.
- ✓ Incorporating automatic replies in the chat columns.
- ✓ Deleting the account whenever customer wishes to.
- ✓ Supporting multi-media in the chat columns.
- ✓ Creating a community for our customers to interact with one another.
- ✓ Call support.
- ✓ Instant SMS alerts

CHAPTER 13

APPENDIX

Flask:

✓ Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.
✓ It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

JavaScript:

✓ JavaScript, often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.
✓ As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries.

IBM Cloud:

✓ IBM cloud computing is a set of cloud computing services for business offered by the information technology company.

IBM Kubernetes:

✓ Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management.

Docker:

✓ Docker is a set of platforms as a service product that use OS-level virtualization to deliver software in packages called containers.

SOURCE CODE

App.py

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

session import ibm_db

import re


app = Flask(__name__,template_folder="static")


app.secret_key = 'a'


conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.
bs2io90l08kqb1od8lcg.databases.appain.cloud;PORT=32536;SECURITY=SSL;SSLServerCe
rtif icate=DigiCertGlobalRootCA.crt;UID=mhd17189;PWD=VzsvTyQHZz3IMA5B",",")
```

```
@app.route('/')

def homer():
    return render_template('home.html')

@app.route('/login',methods =['GET',
'POST']) def login():
    global
    userid
    msg = ''

if request.method == 'POST' : username
= form['username'] password =
request.form['password']
        sql = "SELECT * FROM users WHERE username =? AND
        password=?" stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,userna
        me)
        ibm_db.bind_param(stmt,2,passwo
        rd) ibm_db.execute(stmt)
        account =
        ibm_db.fetch_assoc(stmt)  print
        (account)
```

```
    if                          account:
        session['loggedin']         =
        True
            session['id'] = account['USERNAME']
            userid= account['USERNAME']
            session['username'] =
            account['USERNAME'] msg = 'Logged in
            successfully !'


            msg = 'Logged insuccessfully !'
            return render_templ ('dashboard.html', msg =
        msg) else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg =
    msg)
 @app.route('/register', methods =['GET',
 'POST']) def registet():


msg = ''
 if request.method == 'POST' :
    username = request.form['username']
    email = request.form['email']
```

```
password = request.form['password']
    sql = "SELECT * FROM users WHERE username
    =?" stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.execute(stmt)
    account =
    ibm_db.fetch_assoc(stmt)
    print(account)
    if account:
        msg = 'Account alreadexists !'
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
    msg = 'Invalid email address !'
    elif not re.match(r'[A-Za-z0-9]+', username):
        msg = 'name must contain only characters and
    numbers !' else:
        insert_sql = "INSERT INTO users VALUES (?, ?,
        ?)" prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prep_stmt, 1, username)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3,
        password) ibm_db.execute(prep_stmt)
        msg = 'You have succesfully
```

```
    registered !' elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('register.html', msg =
 msg) @app.route('/dashboard')
def dash():
      return
 render_template('dashboard.html')
 @app.route('/apply',methods ['GET',
 'POST']) def apply():
    msg = ''
  if request.method == 'POST' :
    username = request.form['username']
    email = request.form['email']
        Complaint=
        request.form['Complaint'] #skills =
        request.form['skills']
        #jobs = request.form['s']
        sql = "SELECT * FROM users WHERE username
        =?" stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account =
```

```
            ibm_db.fetch_assoc(stmt)
            print(account)
            if account:

                msg = 'there is only 1 job position! for you'
                return render_template('apply.html', msg =
                msg)
            insert_sql = "INSERT INTO Complaint VALUES (?, ?,
            ?)" prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, Complaint)
            #ibm_db.bind_param(prep_tmt, 5,
            jobs) ibm_db.execute(prep_stmt)
            msg = 'You have successfully applied for
            job !' session['loggedin'] = True
            TEXT = "Hello,a new application for job position" +Complaint+"is requested"
        elif request.method == 'POST':
            msg = 'Please fill out the form !'
        return render_template('apply.html', msg =
    msg) @app.route('/display')
def display():
    #print(session["username"],session['id'])
```

```
  #cursor = ibm_db.connect(conn,)

    # cursor.execute('SELECT * FROM Complaint WHERE userid = % s',

    (session['id'],)) # account = cursor.fetchone()

        return render_template('display.html',account =

  ibm_db) @app.route('/logout')

def logout():

  session.pop('loggedin', None)

  session.pop('id', None)

  session.pop('username',

  None)



    return

  render_template('home.html') if __

  name  == '_main_':

     app.run(debug=True,host='0.0.0.0')
```

Reg.html

```
<!DOCTYPE html>

  <html lang="en">

  <head>

      <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>DJKM_Customer_Care | LOGIN</title>

        <!-- favicon -->

        <!-- <link rel="shortcut icon" href="/assets/img/favicon.ico" type="image/x-icon"> -->

        <!-- <link rel="icon" href="/assets/img/favicon.ico" type="image/x-icon"> -->

        <link rel="icon" type="image/png" sizes="16x16" href="/assets/img/favicon-32x32.png">

        <!-- bootstrap css cdn -->

        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrp/4.5.2/css/bootstrap.min.css" integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuOnIxN0hoP+VmmDGMN 5t9UJ0Z"
crossorigin="anonymous">

        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css">

        <!-- css stylesheet -->

        <link rel="stylesheet" href="css/style.css">

        <!-- font styles cdn -->

        <link rel="preconnect" href="https://fonts.gstatic.com">
        <link href="https://fonts.googleapis.com/css2?family=egreya&display=swap"
rel="stylesheet">

        <link href="https://fonts.googleapis.com/css2?family=Alegreya:wght@600&display=swap"
rel="stylesheet">

</head>

<body>

    <!-- bootstrap navbar -->

    <div class="logo mt-3 text-center">

        <a class="main-logo-img mt-5" href="#"><img src="static/img/smartinternz.png"
alt="sheep-logo" height="250px" width="200px">
```

```
            <!-- <a class="navbar-brand" href="index.html">JobPortal</a> -->

           </a>

    </div>

      <!-- navbar ends -->

    <!-- Login form -->

    <div class="login text-center mt-5">

        <h2> Register Form </h2>

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

        <div class="msg">{{ msg }}</div>

            <!-- <input type="text" placeholder="fullname" id="fullname"> </br></br> -->

                    <input type="text" name="username" placeholder="Enter Your Username"
id="username" required></br></br>

                                <input type="email" name="email" placeholder="Enter Your Email ID"
id="email" required></br></br>

                                <input type="password" name="ssword" placeholder="Enter Your
Password" id="password" required></br></br>



            </br>

            </br>

        <button type="submit" id="button" class="btn btn-primary"> Register </button>
        </form>

    </div>

    <div class="note mt-3 text-center"> <!--Register form -->

    <p> already have an account ? please login <a href="/login">login! </a> </p>

       </div>

</body>
```

```
</html>
```

Login.html

```
    <!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DJKM_Customer_Care | LOGIN</title>
        <!-- favicon -->
        <!-- <link rel="shortcut icon" href="/assets/img/favicon.ico" type="image/x-icon"> -->
        <!-- <link rel="icon" href="/assets/img/favicon.ico" type="image/x-icon"> -->
        <link rel="icon" type="image/png" sizes="16x16" f="/assets/img/favicon-32x32.png">
        <!-- bootstrap css cdn -->
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGM N5t9UJ0Z"
crossorigin="anonymous">
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css">
        <!-- css stylesheet -->
        <link rel="stylesheet" href="css/style.css">
        <!-- font styles cdn -->
        <link rel="preconnect" href="https://fonts.gstatic.com">
```

```
        <link href="https://fonts.googleapis.com/css2?familAlegreya&display=swap"
rel="stylesheet">

        <link href="https://fonts.googleapis.com/css2?family=Alegreya:wght@600&display=swap"
rel="stylesheet">

</head>
<body>

    <!-- bootstrap navbar -->

    <div class="logo mt-3 text-center">

        <a class="main-logo-img mt-5" href="#"><img src="static/img/smartinternz.png"
alt="sheep-logo" height="250px" width="200px">

            <!-- <a class="navbar-brand" href="index.html">JobPortal</a> -->

          </a>

    </div>

      <!-- navbar ends -->

    <!-- Login form -->

    <div class="login text-center mt-5">

        <h2> Login Form </h2>

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

        <div class="msg">{{ msg }}</div>

            <input type="text" name="username" placeholr="Enter Your Username" id="username"
required></br></br>
                                <input type="password" name="password" placeholder="Enter Your
Password" id="password" required></br></br>

            </br>

            </br>

        <button type="submit" id="button" class="btn btn-primary"> Login </button>

        </form>
```

```
    </div>


    <div class="note mt-3 text-center"> <!--Register form -->

    <p> Don't have an account yet? Click here to <a href="register">register! </a> </p>

    </div>
  </body></html>
```

Dash.html

```
<!DOCTYPE html>

  <html lang="en">

  <head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=ege">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>DJKM_Customer_Care | HOME</title>

    <meta charset="UTF-8">

        <!-- favicon -->

        <!-- <link rel="shortcut icon" href="/assets/img/favicon.ico" type="image/x-icon"> -->


        <!-- <link rel="icon" href="/assets/img/favicon.ico" type="image/x-icon"> -->

        <link rel="icon" type="image/png" sizes="16x16" href="/assets/img/favicon-32x32.png">

        <!-- bootstrap css cdn -->

        <link rel="stylesheet"
  href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-
  JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGM N5t9UJ0Z"
```

```
crossorigin="anonymous">

        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css">

        <!-- css stylesheet -->

        <link rel="stylesheet" href="css/style.css">

        <!-- font styles cdn -->     <link rel="preconnect" href="https://fonts.gstatic.com">

        <link href="https://fonts.googleapis.com/css2?family=Alegreya&display=swap"
rel="stylesheet">

        <link href="https://fonts.googleapis.com/css2?family=Alegreya:wght@600&display=swap"
rel="stylesheet">

</head>

<body>

        <!-- bootstrap navbar -->

        <nav class="navbar sticky-top navbar-expand-lg navbar-light">

            <div class="container-fluid">

              <a class="main-logo-img mt-3" href="#"><img src="static/img/smartinternz.png"
alt="sheep-logo" height="250px" width="200px">

                <!-- <a class="navbar-brand" href="index.html">JobPortal</a> -->

              </a>

              <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">

                <span class="navbar-toggler-icon"></span>

              </button>

                          <div class="row donate-sponsor">


                <a type="button" class="btn btn-success mr-1" id="donate"
href="/logout">LOGOUT</a>
```

```
                <a type="button" class="btn btn-warning mr-1" id="sponsor"
href="register">REGISTER</a>

                <a type="button" class="btn btn-primary mr-1" id="sponsor" href="display">MY
Complaint</a>

              </div>

            </div>

          </nav>

          <!-- navbar ends -->

        <!-- what we focus on -->

        <section class="our-focus">

            <div class="container">

              <h2 class="text-center mt-3">Welcome To DM_Customer_Care</h2>

              <div class="row ml-3 mt-3">

                <div class="col-lg-3 mr-5" id="focus-first">

                  <div class="card" style="width: 19rem;">

                    <!-- <img src="assets/img/home kids.jpg" class="card-img-top" alt="..."> -->

                  <!-- <div class="card-body">

                      <h5 class="card-title">Python</h5>

                     <p class="card-text">Skills for python

                     </p>

                     <a href="apply" class="btn btn-primary">Apply Now</a>

                    </div>

                  </div>

                </div>        <div class="col-lg-3 mr-5" id="focus-second">

                  <div class="card" style="width: 20rem;">-->
```

```
            <!-- <img src="assets/img/friendship day.JPG" class="card-img-top" alt="..."> -->
            <div class="card-body">
              <h5 class="card-title">Add your Complaint</h5>
              <p class="card-text">New complaint</p>
                <a href="apply" class="btn btn-primary">Apply Now</a>
            </div>
          </div>
        </div>
        <div class="col-lg-3 ml-5" id="focus-third">
          <div class="card" style="width: 20rem;">
            <!-- <img src="assets/img/health camp.jpg" class="card-img-top" alt="..."> -->
           <!-- <div class="card-body">
              <h5 class="card-title">HR Manager</h5>
              <p class="card-text">skills for hr manager</p>
                <a href="apply" class="btn btn-primar>Apply Now</a>
            </div>-->
          </div>
        </div>
      </div>
    </div>
  </section>
  <!-- focus section ends -->
<!-- footer starts -->
  <!-- Site footer -->
  </body>
```

```
</html>
```

Apply.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DJKM_Customer_Care | APPLY</title>
        <!-- favicon -->
        <!-- <link rel="shortcut icon" href="/assets/img/vicon.ico" type="image/x-icon"> -->
        <!-- <link rel="icon" href="/assets/img/favicon.ico" type="image/x-icon"> -->
        <link rel="icon" type="image/png" sizes="16x16" href="/assets/img/favicon-32x32.png">
        <!-- bootstrap css cdn -->
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrabootstrap.min.css"
integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGM
N5t9UJ0Z" crossorigin="anonymous">
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css">
        <!-- css stylesheet -->
        <link rel="stylesheet" href="css/style.css">
        <!-- font styles cdn -->
        <link rel="preconnect" href="https://fonts.gstatic.com">
        <link href="https://fonts.googleapis.com/css2?family=Alegreya&display=swap"
```

```
rel="stylesheet">
        <link href="https://fonts.googleapis.com/css2?family=Alegreya:wght@600&display=swap"
rel="stylesheet">
</head>
<body>
    <!-- bootstrap navbar -->
    <div class="logo mt-3 text-center">
        <a class="main-logo-img mt-5" href="#"><img src="/static/img/smartinternz.png"
alt="sheep-logo" height="250px" width="200px">
            <!-- <a class="navbar-brand" href="index.html">JobPortal</a> -->
          </a>
    </div>
      <!-- navbar ends -->
    <!-- Login form -->
    <div class="login text-center mt-5"><h2>Apply Now</h2>
        <div class="msg">{{ msg }}</div>
        <form action="/apply" method="post" class="mt-3">
            <!-- <input type="text" placeholder="fullname" id="fullname"> </br></br> -->
```

Display.html

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>DJKM_Customer_Care | HOME</title>

      <div class="msg">{{ msg }}</div>

    <meta charset="UTF-8">

        <!-- favicon -->

        <!-- <link rel="shortcut icon" href="/asseg/favicon.ico" type="image/x-icon"> -->
<!-- <link rel="icon" href="/assets/img/favicon.ico" type="image/x-icon"> -->

        <link rel="icon" type="/image/png" sizes="16x16" href="/assets/img/favicon-32x32.png">

        <!-- bootstrap css cdn -->

        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGM N5t9UJ0Z"
crossorigin="anonymous">

        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css">

        <!-- css stylesheet -->

        <link rel="stylesheet" href="css/style.css

        <!-- font styles cdn -->

        <link rel="preconnect" href="https://fonts.gstatic.com">

        <link href="https://fonts.googleapis.com/camily=Alegreya&display=swap"
rel="stylesheet">

        <link href="https://fonts.googleapis.com/css2?family=Alegreya:wght@600&display=swap"
rel="stylesheet">

</head>

<body>

        <!-- bootstrap navbar -->
```

```
<nav class="navbar sticky-top navbar-expand-lg navbar-light">

    <div class="container-fluid">

        <a class="main-logo-img mt-3" href="#"><img src="static/img/smartinternz.png"
alt="sheep-logo" height="250px" width="200px">

          <!-- <a class="navbar-brand" href="index.html">JobPortal</a> -->

        </a>

        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">

          <span class="navbar-toggler-icon"></span>

        </button>
        <div class="row donate-sponsor">

          <a type="button" class="btn btn-success mr-1" id="donate"
href="/logout">LOGOUT</a>

          <a type="button" class="btn btn-warning r-1" id="sponsor"
href="register">REGISTER</a>

          <a type="button" class="btn btn-primary mr-1" id="sponsor" href="display">MY
Complaint</a>

        </div>

    </div>

  </nav>

  <!-- navbar ends -->

<!-- what we focus on -->

<section class="our-focus">

    <div class="container">

        <h2 class="text-center mt-3">Your details</h2>

        <div class="border">
```

```
<table class="worddark"></br></br></br></br>

<tr>

        <td>username:</td>
        <td>{{ account[1] }}</td>

</tr>

<tr>

        <td>Email ID:</td>

        <td>{{ account[2] }}</td>

</tr>

<tr>

        td>Complaint:</td>
       <td>{{ account[3] }}</td>

</tr>

</table>

</section>

</body>

</html>
```

GITHUB AND PROJECT DEMO LINK

Github Rep Link: https://github.com/IBM-EPBL/IBM-Project-14418-1659585439

Project Demo Link:https://youtu.be/I_f6JQsl82Y