HX8001-PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

CUSTOMER CARE REGISTRY
URL'S USING IBM WATSON STUDIO

TEAM ID: PNT2022TMID50230

FACULTY MENTOR NAME: MRS. A. JENIFUS SELVARANI

TEAM LEADER: R. ISWARYA

TEAM MEMBERS: S. SUDALAIVADIVU @ GAYATHRI

V. GEETHAI

N. SELVA MARI

ABSTRACT

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 able to track the work assigned to the agent and a notification 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.

TABLE OF CONTENTS

CHAPTER 1

Introduction

1.1 project overview

This applicaton has baeen developed to help of customer in processing their complaints. The customer can raise the ticket with a description of the issues. An agent will be assigned to the customer to solve a problem. Whenever the agent is assinged to a customer they will be notified with an email alert. Customer can view the status of the ticket till the source is service is provided.

ADMIN: The main role and the responsibility of the admin are take to care of the whole process. Starting from admn login followed by the agent and asinging the customer's complaints. Finally he will be to track the work assigned to the agent and a notification will be sent to the customer.

USER: They can register for an account. After the login they can create the compliant with a descripted of the problem they are facin each user will assigned to the admin. They can view the status of their complaints.

PURPOSE:

The purpose of the project is toprovide a common platform to the customer to clarify their queriesHaving expert agents in the platform for bettter answering Customer's tickets are answered quickly by the agentsWhile doing so,the formar asks questionLater answer those question as quickly and as legimately as possibleCustomer can raise as many tickets as they wantCustomers and agents can also sumbit their feedback to the admin, for the betterment of the platform

CHAPTER 2

LITRETURE SURVEY

2.1 Existing System:

- Reviews and rating in the e-commerce websit are not reliable
- Even more,so they are often been giving by the manufactures themselves
- Reviews are not from the authentic individuals
- After buying common platform available to us,the customer,to have our doubts clearred
- If it is existing,we are not getting fast by the time the reply comes issues has been cleared or of not worth of being cleaned to the customer.

2.2 PROBLEM STATEMENT:

I am Iswarya and i am regular customer in famous e-commerce websites like Amazon,Filpkart,Messho. I order regularly.The problem have is in most times,Idon't have any relaiable sources to clear my doubts in some of the products i buy.

There are reviews and customer rating is thosse but somehow, I don't feel they are authentic and real.It would make if those websites were from a real expert,and i could clarify careful all my doubts in a single platform.Of course I would and instant replies from a real expert who knows about the products I am asking for.

2.3 problem Statement

i am iswarya and I am a regular customer in famous e-commerce web like amazon Flipkart i order regularty .the problem I have is that in most times Idon't have any reliable provider to cleadr my doubts in some of the products I buy platform of course ,I would need install replies from a real export who knows about the product i am there are reviews and customer rating in those websites but somehow, and real.

CHAPTER 3

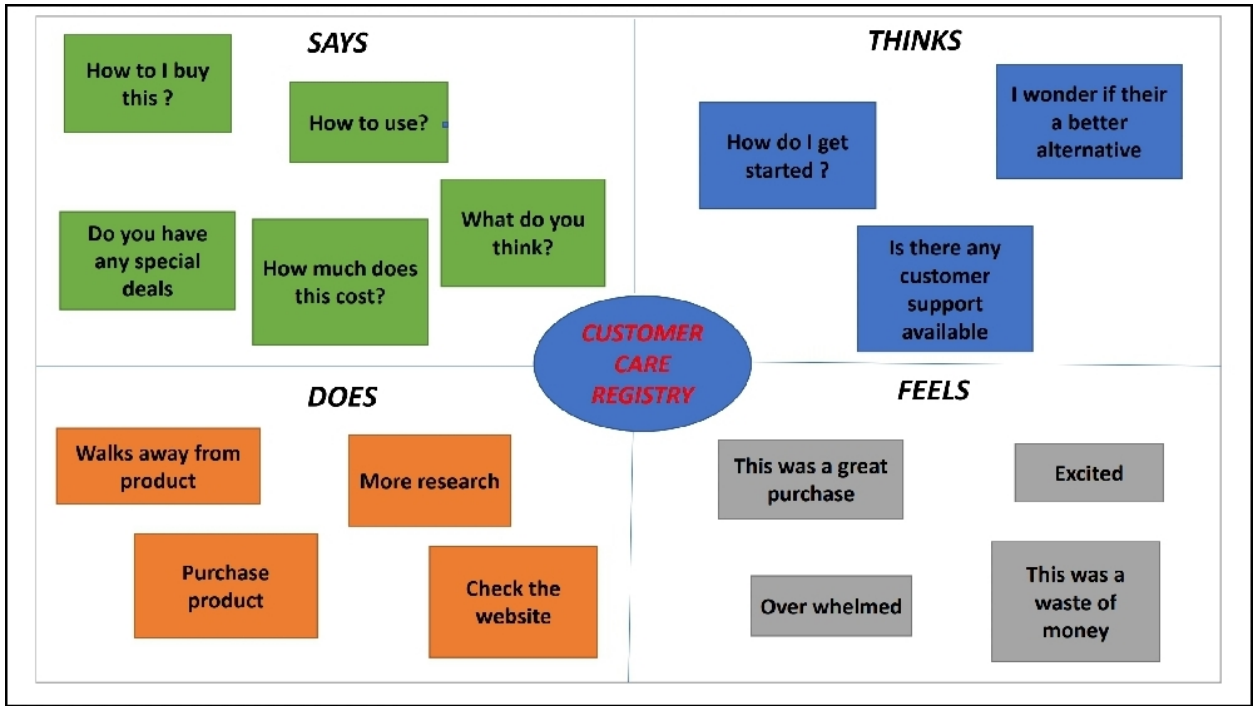
IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:

Empathy Map is a simple,easy-to-digest visual that captures knowledgable about a user's behaviour's and attitudes.

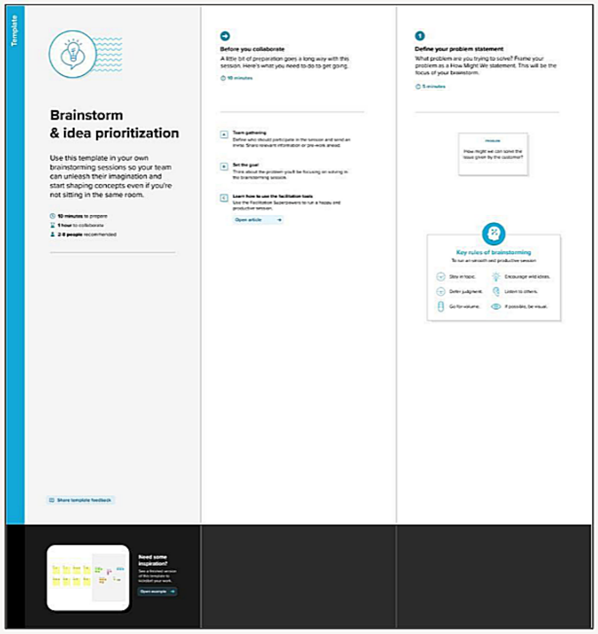
It is useful tool to help teams tobetter their usersCreating an effective solution requires understanding the user problem and the person who us experience it.

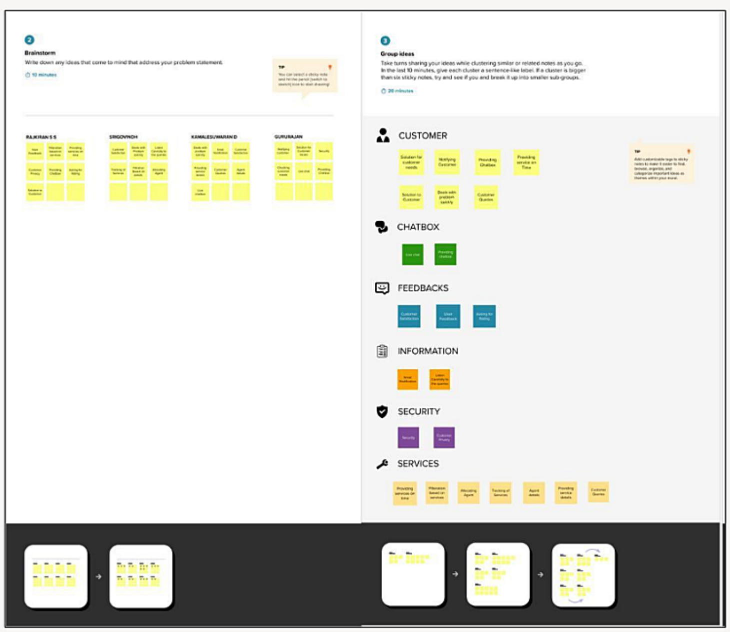
The excertise of creating the map helps to participants consider things from the user's perspective along with his or her goals challenges.

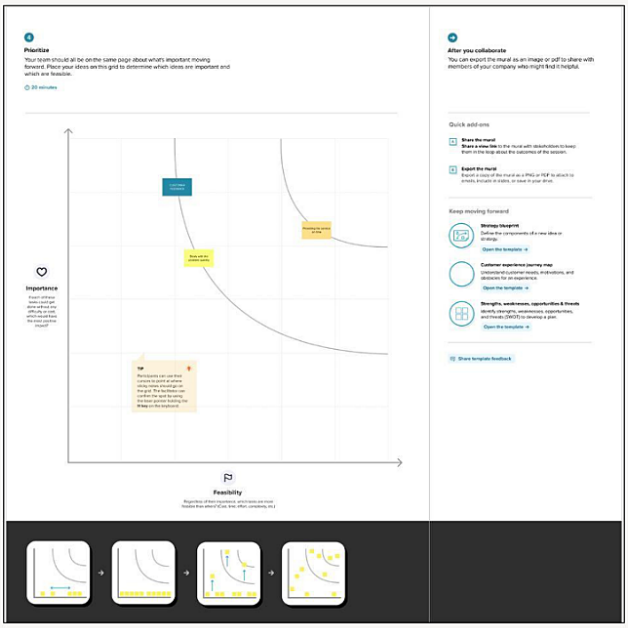


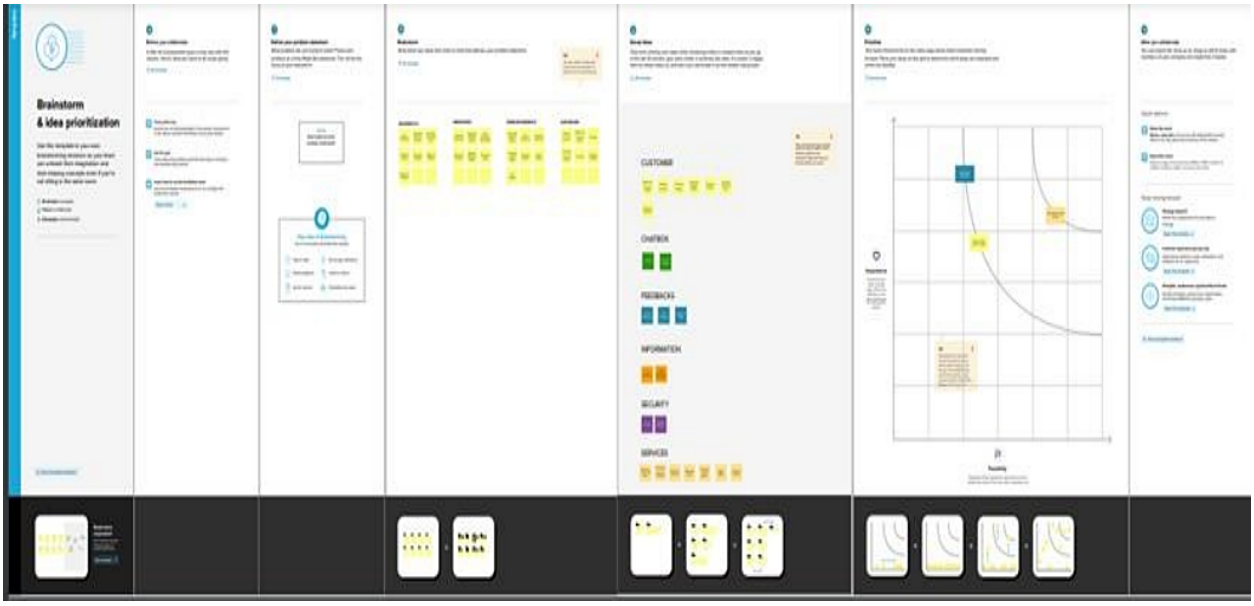
3.2 IDEATION AND BRAINSTROMING

Brainstroming provides a free and open environment that encourages everyone within a team toparticipate 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 paricipants and encouraged to collaborate helping and each other develop a rich number of creative solution.


Brainstorm
& idea prioritization


CUSTOMER
CHATBOX
FEEDBACKS
INFORMATION
SECURITY
SERVICES

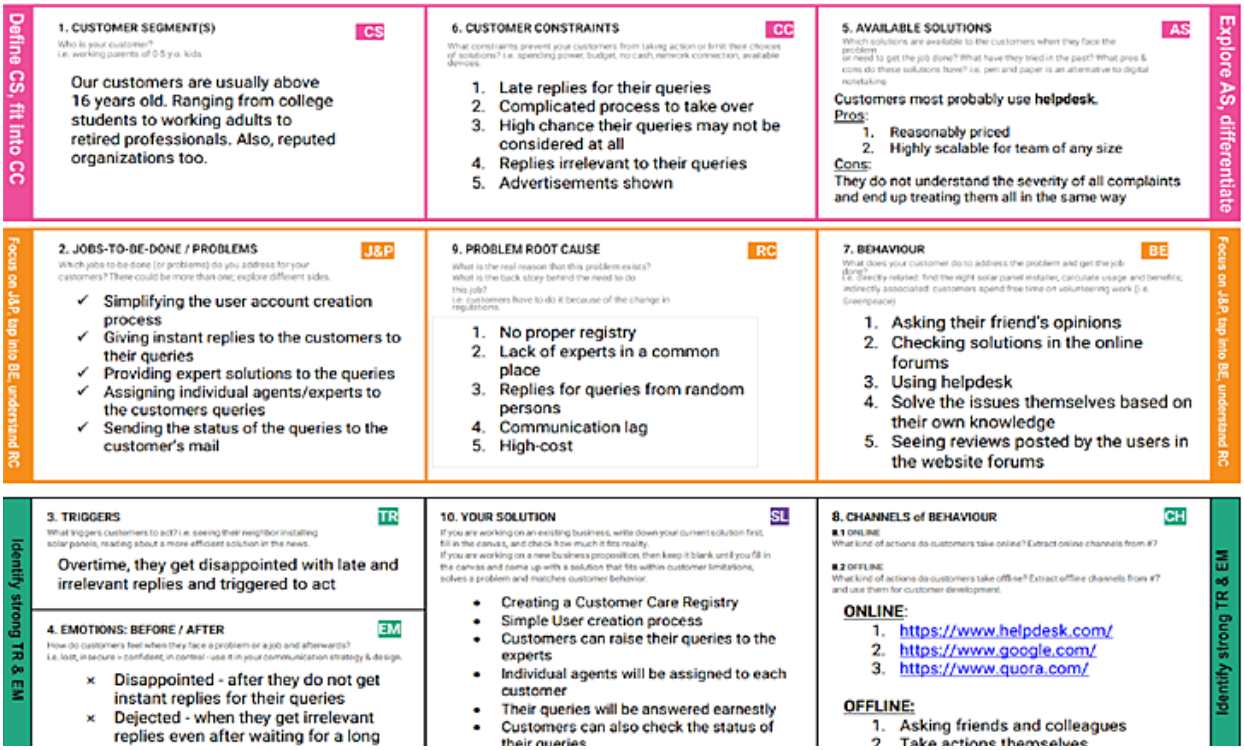

Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
Importance
Feasibility
After you collaborate
You can export the mural as an image or pdf to share with members of your company who might find it helpful.
Quick add-ons
Keep moving forward
Open the template
Share template feedback


Brainstorm
& idea prioritization
CUSTOMER
CHITECH
FEEDBACKS
INFORMATEN
SECURITY
SERVICES

3.3 Proposed Solution

S.NO	Parameter	decriptions
1.	problem statement	I am selvamari I am regular customer in famous e-comerce websutes like amazon,flipkard. i order regularly .the problem I have is th t is that mosttimes. I don't have any resourcesto clear my doubts in some of the products.
2.	Idea/solutiondescription	creating customer care registry ,where the customer canraise their queries in form of tickets.An agent will be assigned to them for reply their issue.
3	Noviety	The agents experts in the producrt domain and they will communicate well with the customers
4	social impact	Customer will be satisfied with the instant and valid replies also it creates a doublites socitety that boosts sales.
5	business model	customer can be changed minimal amonut based on the number of queries they can raise perod of time
6	salabilty of the solution	may be in the future,may be across platform mobile application be developer making customer careregistry more aceesible the users

3.4 Problem Fit:



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional requirement

A functional requiremnts defines functon of a system,where a functions is described a specification behaviour between input and output.

- It defines "What behaviour a software systen"
- define at a component level'usually easy to define
- helps you verify functionality of the software

S.NO	Functional requirements	Non-Functional requirements
1	user registration	registration throught signupform
2	forgot password	resetting the passwordby sending otp
3	user login	login throught login form
4	agent creation	create agent login form
5	dashboard(cuatomer)	show all the tickets by the customer

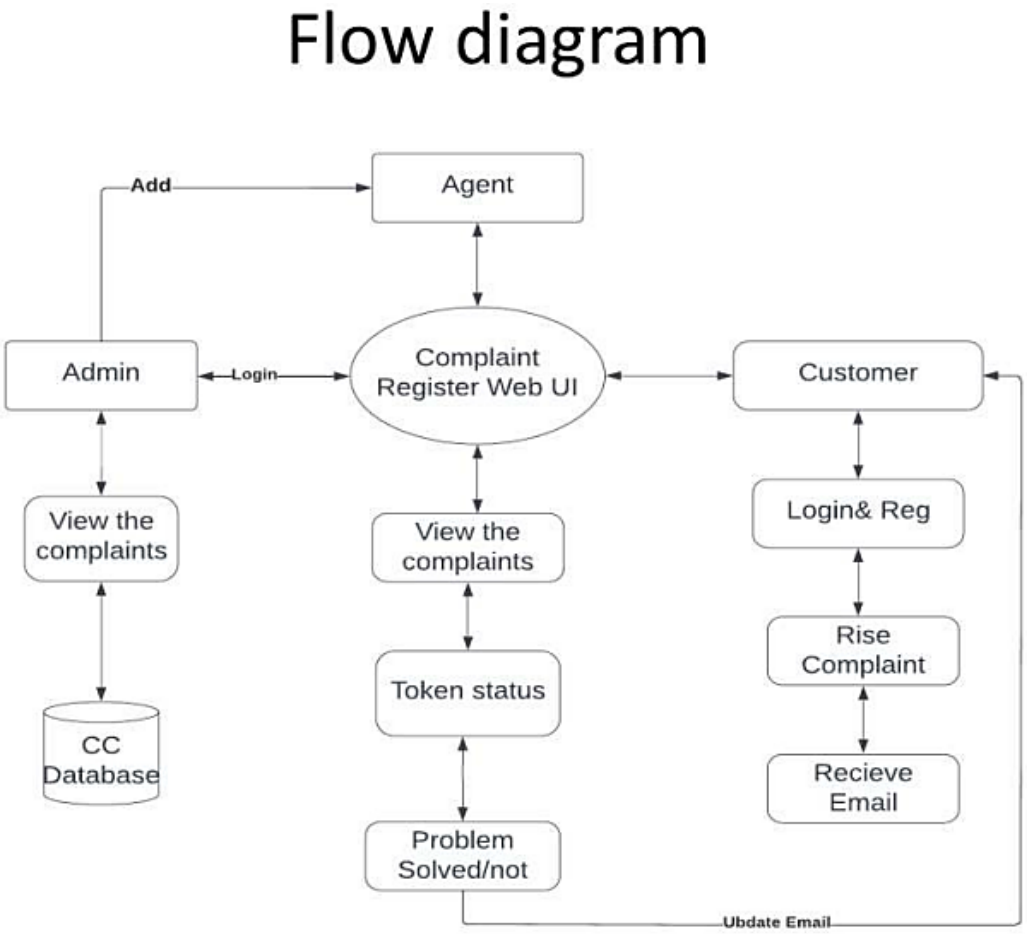
4.1 Non functional requirements:

- A non functional requirement defines the quality attribute software system
- it is not mandatory
- applied system as whole
- usually more difficult to define
- helps you verify the performance of the software

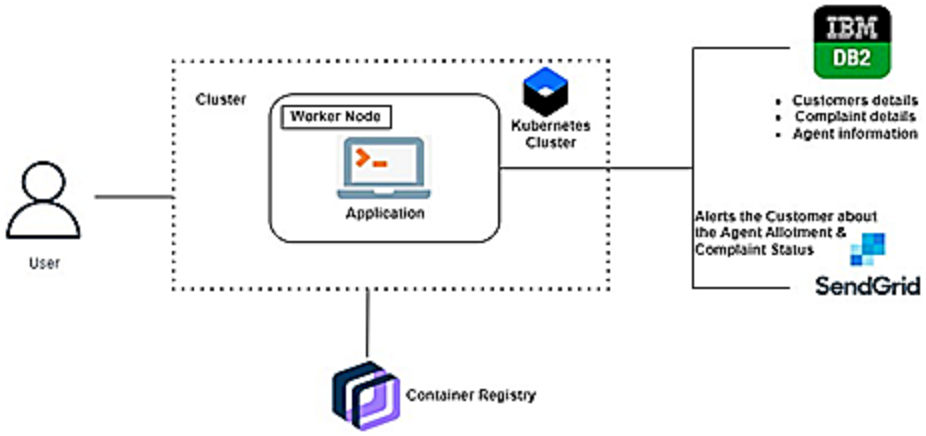
CHAPTER 5

PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer. It's tempting to think that user stories are, simply put, software system requirements. But they're not. A key component of agile software development is putting people first, and a user story puts end users at the center of the conversation. These stories use non-technical language to provide context for the development team and their efforts. After reading a user story, the team knows why they are building, what they're building, and what value it creates. User stories are one of the core components of an agile program. They help provide a user-focused framework for daily work — which drives collaboration, creativity, and a better product overall.

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey& Information Gathering	Literature survey on theselected project & gatheringinformation byreferring the,technical papers,research publications etc.	25 OCTOBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvasto capture the user Pains & Gains, Prepare listof problemstatements	26 OCTOBER 2022
Ideation	List the by organizing the brainstorming session andprioritize thetop 3 ideas based on thefeasibility & importance.	27 OCTOBER 2022
Proposed Solution	Prepare the proposed solutiondocument, which includes thenovelty, feasibility of idea, business model, social impact, scalability of solution, etc.	28 OCTOBER 2022
Problem Solution Fit	Prepare problem - solutionfitdocument.	29 OCTOBER 2022
Solution Architecture	Prepare solution architecturedocument.	30 OCTOBER 2022
Customer Journey	Prepare the customer journeymaps to understand the user interactions & experiences with the application (entry to exit).	30 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	31 OCTOBER 2022

Data Flow Diagrams	Draw the data flow diagrams and submit for review.	31 OCTOBER 2022
Technology Architecture	Prepare thetechnology architecture diagram.	01 NOVEMBER 2022
Prepare Milestone & ActivityList	Prepare the milestones & activity list of the project.	01NOVEMBER 2022

6.2 Sprint Delivery Schedule

=

Use the below template to create product backlog and sprint 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) per iteration unit (story points per day)

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

CHAPTER 7

CODING & SOLUTIONING

7.1 Feature 1

```
from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysqldb import MySQL

import MySQLdb.cursors
import re
from sendemail import usermail,agentmail
app = Flask(__name__)

app.secret_key='a'

app.config['MYSQL_HOST'] = 'remotemysql.com'
app.config['MYSQL_USER'] = 'cmD4MqEFox'
app.config['MYSQL_PASSWORD'] = 'szmvOT91Pl'
app.config['MYSQL_DB'] = 'cmD4MqEFox'
mysql = MySQL(app)
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/registertemp',methods=["POST","GET"])
def registertemp():
    return render_template("register.html")

@app.route('/uploaddata',methods =['GET','POST'])
def register():
    msg = ''
    if request.method == 'POST':
        firstname = request.form['firstname']
        lastname = request.form['lastname']
        username = request.form['username']
```

```
        email = request.form['email']
        password = request.form['password']
        address = request.form['address']
        cursor = mysql.connection.cursor()

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM users WHERE username = % s', (username, ))
    account = cursor.fetchone()
    print(account)
    if account:
        msg = 'Account already exists !'
    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:
        cursor.execute('INSERT INTO users VALUES (NULL,% s,% s,% s,% s,% s,%
s)',(firstname,lastname,username,email,password,address,))
        mysql.connection.commit()
        msg = 'Dear % s You have successfully registered!'%(username)
    return render_template('register.html',a = msg,indicator="success")

@app.route('/login',methods=["POST","GET"])
def login():
    return render_template("login.html")

@app.route('/logindata',methods=["POST","GET"])
def logindata():
    global userid
    msg = ''
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM users WHERE username = % s AND password = % s',
(username, password, ))
        account = cursor.fetchone()
        print (account)
        if account:
            session['id'] = account[0]
```

```
            userid =  account[0]
            session['username'] = account[1]
            return redirect(url_for('dashboard'))
        else:
            msg = 'Incorrect username / password !'
            return render_template('login.html', b = msg,indicator="failure")
    else:
        return render_template("login.html",msg="ERROR")
@app.route('/home')
def dashboard():
    if 'id' in session:
        username = session['username']
        return render_template('userdashboard.html',name=username)

@app.route('/profile',methods=["POST","GET"])
def profile():
    if 'id' in session:
        uid = session['id']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM users WHERE id = % s', (uid,))
        cursor.connection.commit()
        acc = cursor.fetchone()
        return
render_template('userprofile.html',fullname=acc[1]+acc[2],username=acc[3],email=acc[4],addres
s=acc[6])

@app.route('/addcomplaint',methods=["POST","GET"])
def comp():
    if 'id' in session:
        return render_template('userlodgecomp.html')

@app.route('/complaint',methods=["POST","GET"])
def complaint():
    if request.method == "POST":
        if 'id' in session:
            msg = ''
            uid=session['id']
            selectcategory = request.form['selectcategory']
            selectsubcategory = request.form['selectsubcategory']
            complainttype = request.form['type']
```

```
        state = request.form['state']
        complaint = request.form.get('complaint')
        date = request.form['date']
        cursor = mysql.connection.cursor()
        cursor.execute("SELECT * FROM users WHERE id = % s",(uid,))
        acc = cursor.fetchone()
        email = acc[4]
        cursor.execute('INSERT INTO complaintdetails VALUES (NULL,% s,% s,% s,% s,% s,% s,%
s,% s)',(uid,selectcategory,selectsubcategory,complainttype,state,complaint,date,'pending'))
        cursor.connection.commit()
        msg = 'You have successfully registered your complaint'
        TEXT1 = """\<!DOCTYPE html>
            <html>
            <body>
              <div class="containter" style="display: block;">
                <h3 style="font-size: 24px; font-family:serif"> Dear """+acc[1]+" "+acc[2]+""", </h3>
                <div class="side" style="width: 400px; height: 150px; padding:30px; border-
radius:10px; position:relative; left:100px;" >
                  <div class="details"style="position:relative;left:60px; font-size:20px;text-
align:left;">
                    <p style=" font-weight:bold;">Your complaint has been succesfully
registered...!!</p>
                    <p style=" font-weight:bold;">One of our agent is assigned , will surely
resolve your complaint as soon as possible</p>
                    <p style=" font-weight:bold;">Please check the complaint history tab for
status of complaint.</p>
                  </div>
                </div>
              </div>
            </body>
            </html>"""
        TEXT = """\<!DOCTYPE html>
            <html>
            <body>
              <div class="containter" style="display: block;">
                <h3 style="font-size: 24px; font-family:serif"> New Complaint from """+acc[1]+"
"+acc[2]+""" </h3>
                <div class="side" style="width: 400px; height: 150px; padding:30px; border-
radius:10px; position:relative; left:100px;" >
                  <div class="details"style="position:relative; top:20px; left:60px; font-
```

```
size:20px;text-align:left;">
                        <p style=" font-weight:bold;"> Complaint Description :</p>
                        <p>"""+complaint+"""</p>
                    </div>
                </div>
            </div>
        </body>
        </html>"""
    agentemail = 'agentcustomercareregistry@gmail.com'
    usermail(TEXT1,email)
    agentmail(TEXT,agentemail)
    return render_template('userlodgecomp.html',a = msg)


@app.route('/view',methods=["POST","GET"])
def view():
  if 'id' in session:
    return render_template('comphistory.html')

@app.route('/comphistory',methods=['POST','GET'])
def compview():
  if 'id' in session:
    uid=session['id']
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM complaintdetails WHERE userid = % s',(uid,))
    cursor.connection.commit()
    comp = cursor.fetchall()
    return render_template('usercomphist.html',complaints = comp)

@app.route('/admin')
def admin():
  return render_template('admin.html')

@app.route('/adminpage')
def adminpage():
  return render_template('admin dashboard.html')

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

```
    email = request.form['email']
    password = request.form['password']
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM admininfo  WHERE email = % s and password = %
s',(email,password))
    cursor.connection.commit()
    logged = cursor.fetchone()
    if(logged):
        msg = 'successfully loggedin'
        return render_template("admin dashboard.html",a=msg)
    else:
        return render_template("admin.html",a="Incorrect email/password")

@app.route('/adcomplainthist',methods=['POST','GET'])
def adcomplainthist():
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM complaintdetails')
        cursor.connection.commit()
        comp = cursor.fetchall()
        return render_template('admincomphist.html',complaints = comp)

@app.route('/loggout')
def loggout():
    if 'id ' in session:
        session.pop('id',None)
        session.pop('email',None)
        session.pop('password',None)
    return redirect(url_for('home'))

@app.route('/logout')
def logout():
    if 'id ' in session:
        session.pop('id',None)
        session.pop('name',None)
        session.pop('username',None)
    return redirect(url_for('home'))

@app.route('/agent',methods=["POST","GET"])
def agent():
    return render_template('agent.html')
```

```
@app.route('/agentdata',methods=["POST","GET"])
def agentdata():
    msg = ''
    username = request.form['username']
    password = request.form['password']
    cursor = mysql.connection.cursor()
    cursor.execute('INSERT INTO agentinfo VALUES (NULL,% s,% s)',(username,password))
    cursor.connection.commit()
    msg = 'Agent has been created successfully'
    return render_template('agent.html',a = msg)


@app.route('/solved/<no>')
def solved(no):
    i = no
    cursor = mysql.connection.cursor()
    cursor.execute("UPDATE complaintdetails SET status = % s WHERE id = % s",("Solved",i))
    cursor.connection.commit()
    return render_template('admincomphist.html',a="Complaint Resolved")

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

7.2 Feature 2

```
import ibm_db
import ibm_db_dbi
import pandas

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_hostname = "54a2f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_port = "32733"
dsn_protocol = "TCPIP"
dsn_uid = "fmn33477"
dsn_pwd = "JatelsTYIFVJ0iZG"
dsn = (
```

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


def run(query):
    conn = ibm_db.connect(dsn, "", "")
    print(query)
    create_table = ibm_db.exec_immediate(conn, query)
    return 1


def check(query):
    conn = ibm_db.connect(dsn, "", "")
    print(query)
    try:
        select = ibm_db.exec_immediate(conn, query)
        return ibm_db.num_rows(select)

    except:
        return 0


def view(query):
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    print(query)
    try:
        select = ibm_db.exec_immediate(conn, query)
        result = []
        dictionary = ibm_db.fetch_assoc(select)
        while dictionary != False:
            result.append(dictionary)
            dictionary = ibm_db.fetch_assoc(select)
```

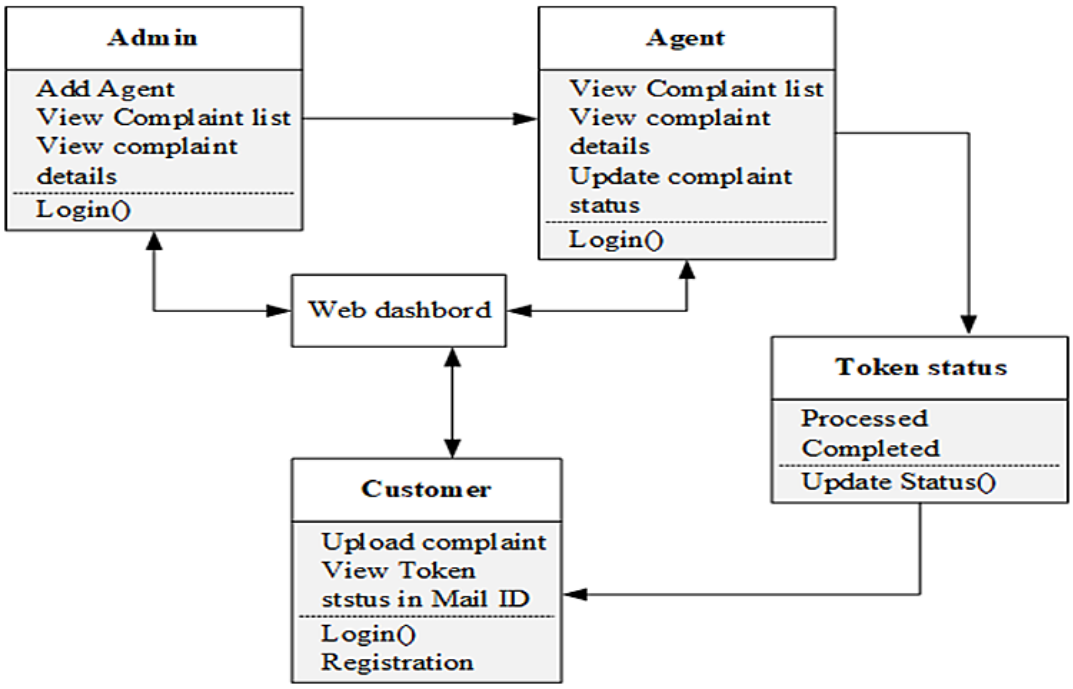
```
    return result

except:
    return "
```

7.3 Database Schema (if Applicable)

A database schema is the keleton structure that represents the logical view of the entire database.It defines how the data is organized and how the relationsamong them are associated. It formulates all the constraints that are to be applied on the data ..

Adatabase schema defines its entitites and relationship among them.It constraints a descriptive detail of the database ,which canbe designed to help programmers understand database and make it useful



CHAPTER 8

TESTING

8.1 Test Cases

The testcase defined as a group of conditions under which a tester determines whether a software application is working as per the customer requirements or not. testcase designing inckudes preconditions,casename,inputcondition,and expected result.

Testced edase gives detailed information about testing stratergy ,testing process, precondition,and excepted output.These are executed during the testing process to check whether the software application performing the task for that was developed or not.

8.2 User Acceptance testing

1.Performance of Document

the purpose of the document i sbrefly explain the test coverage and open issuses of the customer care registry project at the release to user acceptance testing

2.Defect Analysis

the report shows of resolved or closed bugs at each severity and how they were resolved.

CHAPTER 9

RESULTS

9.1 Performance Metrics

since all the operations using flask is in server side,the client need not worry.
about the cpu usage.just rrendering the page,static contents takes place the client side.
memory for clent side functions is allocated heap.It can be either increased based upon requirements or removed from from the reap.

CHAPTER 10

ADVANTAGES

customer can clarify doubts just by creating new ticket
customer gets as soon as possible
not only the replies more authric and practical
very minimal accout creation process
customer feedback are always listened
free cost
customer can raise tiket as they want

CHAPTER 11

CONCLUSION

Conclusionustomer application on the internet avaliab,e on the internet nothing down the structural components of those application bulit customer registry application

Customer can register into the application using their email, password, firstname, lastname.

Then can login to the system and arises they want from th ticket.

These tickets will be sent to the application assigned.

Then the assigned agent have one to one chat the customer and latters will be clarified .

It isalso the responsilblity of the domain to create an d\agent.

CHAPTER 12

FUTURE SCOPE

- Our application is not finished.there are many rooms for improvement.
- Attracking and much more respective UI thought the application
- Releasing cross platform mobile applications
- Call support
- Instants SMSalerts
- Incorporating automatic in the chat columns
- Deleting the account whenever customer wishes to supporting multi media in the chat columns
-

CHAPTER 13

APPENDIX

Source Code GitHub & Project Demo Link :

Source Code GitHub Link :

https://github.com/IBM-EPBL/IBM-Project-43444-1660716937

Project Demo Link :

https://drive.google.com/file/d/1w38luMXAGEJ12PMcVmfkEBNzlaGsaYWS/view?usp=share_link