CUSTOMER CARE REGISTRY

**TEAM ID: PNT2022TMID36073**

**TEAM MEMBERS:**

|  |
| --- |
| **HARISH BABU.R.L (TEAM LEADER)** |
| **KATARI LAKSHMIPATHI** |
| **PAVAN KUMAR.D** |
| **MOHAN.A** |

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Table of content** | PAGE NO. |
| 1 | **INTRODUCTION** |  |
|  | 1.1 Project Overview | 3 |
|  | 1.2 Purpose | 3 |
| **2** | **LITERATURE SURVEY** |  |
|  | 2.1 Existing problem | 5 |
|  | 2.2 References | 5 |
|  | 2.3 Problem Statement Definition | 7 |
| **3** | **IDEATION & PROPOSED SOLUTION** |  |
|  | 3.1 Empathy Map Canvas | 8 |
|  | 3.2 Ideation & Brainstorming | 9 |
|  | 3.3 Proposed Solution | 11 |
|  | 3.4 Problem Solution fit | 13 |
| **4** | **REQUIREMENT ANALYSIS** |  |
|  | 4.1 Functional requirement | 14 |
|  | 4.2 Non-Functional requirements | 14 |
| **5** | **PROJECT DESIGN** |  |
|  | 5.1 Data Flow Diagrams | 15 |
|  | 5.2 Solution & Technical Architecture | 15 |
|  | 5.3 User Stories | 16 |
| **6** | **PROJECT PLANNING & SCHEDULING** |  |
|  | 6.1 Sprint Planning & Estimation | 17 |
|  | 6.2 Sprint Delivery Schedule | 17 |
|  | 6.3 Reports from JIRA | 17 |
| **7** | **CODING & SOLUTIONING** |  |
|  | 7.1 Feature 1 | 19 |
|  | 7.2 Feature 2 | 21 |
|  | 7.3 Database Schema (if Applicable) | 22 |
| **8** | **TESTING** |  |
|  | 8.1 Test Cases | 24 |
|  | 8.2 User Acceptance Testing | 24 |
| **9** | **RESULTS** |  |
|  | 9.1 Performance Metrics | 26 |
| **10** | **ADVANTAGES & DISADVANTAGES** | 28 |
| **11** | **CONCLUSION** | 29 |
| **12** | **FUTURE SCOPE** | 30 |
| **13** | **APPENDIX (**source code ,GitHub and Demo link) | 32 |

* 1. PROJECT OVERVIEW

# INTRODUCTION

Customer care and customer service together help create a positive customer experience, or the overall impression a person has when interacting with your company. Both are vital, but there are subtle differences in how they are implemented. High-quality customer care is proactive. The needs of customers throughout the buyer’s journey are anticipated, making customers feel supported. That, in turn, helps create an emotional connection between the customer and the company. Customer service is reactive. Here, the focus is on helping customers solve problems or answer questions before purchase, either in a self-serve fashion or via the customer support team. Customer care is more than just providing great customer service. It’s a proactive approach to providing information, tools, and services to customers at each point they interact with a brand. If a company neglects customer care, it can negatively impact the customer service experience. For example, when a website chatbot can’t provide key information about a product, customers are more likely to get frustrated and reach out to a customer service agent for help. Consumer expectations are extremely high, putting increased pressure on companies to improve their customer relationships. This can lead to lost information when the same person reaches out via multiple channels. When a customer service agent doesn’t know the whole story and the customer has to repeatedly share the problem, it leaves both people frustrated. They can register for an account. After the login, they can create a complaint with a description of the problem they are facing. Each user will be assigned an agent. They can view the status of their complaint.

* Customers get the insights they need to make an informed purchase.
* Customer satisfaction can increase and customer loyalty can improve.
* Customer service agents spend less time on routine tasks and answering commonly asked questions, enabling agents to do more meaningful task.
  1. PURPOSE

There are two sides to customer service objectives. First, there are the goals and KPIs customer service teams attempt to achieve. Then, there’s customer service resume objectives. It’s important to understand the connection between the two: Writing a strong customer service resume objective starts with understanding the objectives of the field and its depth and possibilities. To provide insight into both levels of customer service objectives. The prime objective of customer service is to answer customer questions quickly and effectively, resolve issues with empathy and care, document pain points to share with internal teams, nurture relationships, and improve brand credibility. Great customer service can make people loyal to your brand, products, and services for years to come.

A strong customer service resume objective underscores your skills and experiences in contributing to customer service's overall goals and objectives. Meeting key customer service KPIs doesn’t just involve answering phones and emails. It's a whole world of solutions development, intuition, empathy, brand management, time management-and the soft skills that help connect people and create trust. I guide my team toward giving the best service possible. Sometimes, we’re not delivering good news. But the objective is to do that with compassion and empathy and in a way that we give the customer constructive next steps to move forward. We also know that as a newer, younger brand, customers may be wary of our credibility. It usually takes a few consistently excellent customer experiences to feel connected and loyal to the brand. That awesome experience starts from the very first touchpoint, whether it be web, email, brick and mortar, or Instagram, and carries through to when they're wearing our product

* 1. EXISTING PROBLEM

# LITERATURE SURVEY

A strong customer problem statement should provide a detailed description of your customer’s current situation. Consider how they feel, the financial and emotional impact of their current situation, and any other important details about their thoughts or feelings.

Customer Satisfaction is an attitude that is decided based on the experience obtained. Satisfaction is an assessment of the characteristics or privileges of a product or service, or the product itself, that provides a level of consumer pleasure with regard to meeting consumer consumption needs.

Customer Satisfaction is the customer's response to the evaluation of perception of differences in initial expectations prior to purchase (or other performance standards) and the actual performance of the product as perceived after wearing or consuming the product in question.

The level of complaint is how high the complaint or delivery of dissatisfaction, discomfort, irritation, and anger over the service of the service or product. The dimension or indicator of complaint level is the high level of complaint.

Product Quality affects Customer Satisfaction, where the dimensions or indicators of Product Quality are quality products, in accordance with the price offered, and ease of use affects the dimensions or indicators of Customer Satisfaction in relation to subscription decisions.

* 1. **REFERENCES**

1] Uddi Executive Overview: Enabling Service Oriented Architecture, 2004 Oct.

1. “Web Services Architecture”, [http://www.w3.org/TR/ws-arch/.](http://www.w3.org/TR/ws-arch/) Date Accessed: 11/02/2004.
2. UDDI V3 Specification. [http://uddi.org/pubs/uddi-v3.00-published-20020719.html.](http://uddi.org/pubs/uddi-v3.00-published-20020719.html) 19/07/2002.
3. Christensen E, Curbera F, Meredith G, Weerawarana S, Web Services Description Language (WSDL) 1.1, W3C Note, 2001
4. Ali A S, Rana O F, Ali R A, Walker D W, UDDIe: an extended registry for Web services, SAINT-w’03 Proceedings of the 2003

Symposium on Applications and the Internet Workshops, 2003 Jan, pp .85-89.

1. Tsai W T, Paul R, Cao Z, Yu L, Saimi A, Xiao B, Verification of Web Services Using an Enhanced UDDI Server ,Proceedings of The

Eighth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems

,2003 Jan, pp 131-38.

1. Liu J, Gu N, Zong Y Ding Z, Zhang S, Zhang Q Service Registration and Discovery in a Domain-Oriented UDDI Registry ,

Proceedings of the 2005 The Fifth International Conference on Computer and Information Technology (CIT’05),2005, pp .276-83.

1. Jian W, Zhaohui W, Similarity-based Web Service Matchmaking Proceedings of the 2005 IEEE International Conference on Services

Computing (SCC’05), 2005.

1. Tretola G, Zimeo E , Structure Matching for Enhancing UDDI Queries Results , IEEE International Conference on Service-Oriented

Computing and Applications(SOCA'07),2007.

1. Ayorak E, Bener a, Super Peer Web Service Discovery Architecture, IEEE Proceedings

- International Conference on Data Engineering, 2007 pp .287-94.

1. Libing He W, Wu Y, Jianqun D A novel interoperable model of distributed UDDI - proceedings of the 2008 IEEE International

Conference on Networking, Architecture, and Storage - IEEE NAS 2008 Jun, pp .153-54.

1. Nawaz F, Qadir K, Ahmad H F, and SEMREG-Pro: A Semantic based Registry for Proactive Web Service Discovery using Publish

Subscribe Model, Fourth International Conference on Semantics, Knowledge and Grid, IEEE Xplore, 2008 Dec, pp .301-08.

1. LIANG1 Q, CHUNG2 J A Federated UDDI System for Concurrent Access to Service Data, IEEE International Conference on e-

Business Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, and DKEEE’08, 2008 Oct, pp .71-

78.

1. Vandan T, Nirmal D, InderjeetGarg S, Garg N, Soni P. An Improved Discovery Engine for Efficient and Intelligent discovery of Web

Service with publication facility, SERVICES 2009 - 5th 2009 World Congress on Services, 2009, pp .63-70.

1. Ma C, Song M, Xu K, Zhang X.Web Service Discovery Research and Implementation Based on Semantic Search Engine, IEEE 2nd

Symposium on Web Society, Beijing. 2010 Aug 16-17, pp .672-77.

1. Rajendran T.Balasubramanie P, Flexible and Intelligent Architecture for Quality Based Web Service Discovery with an Agent- Based

Approach, International Conference on Communication and Computational Intelligence (INCOCCI), Erode. 2010 Dec 27-29, pp .617-

22.

1. Johnsen F T, Hafsøe T, Eggen A, Griwodz C, Halvorsen P, Web Services Discovery across Heterogeneous Military Networks, IEEE

Communications Magazine, 2010 Oct, 48(10), pp. 84-90.

1. Ourania H, Georgios B, Mara N, Dimosthenis A, “A Specialized Search Engine for Web Service Discovery” IEEE 19th International

Conference on Web Services,USA. 2012 June 24-29, pp. 448-55.

1. Raj R J R, Sasipraba T., Web Service Recommendation Framework Using Qos Based Discovery and Ranking Process 3rd
   1. **PROBLEM STATEMENT DEFINITION**

A customer problem statement outlines problems that your customers face. It helps you figure out how your product or service will solve this problem for them.

The statement helps you understand the experience you want to offer your customers. It can also help you understand a new audience when creating a new product or service.

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.

A Customer Problem Statement is a detailed description of an issue that needs to be addressed. This document thoroughly elaborates on the problem that your product or your service solves for your particular customers. It takes into consideration your customer’s unique pain points and how your product goals about solving their situation. A customer problem statement helps you and your team understand the detailed experience you are attempting to transform by analyzing and empathizing with your customers.

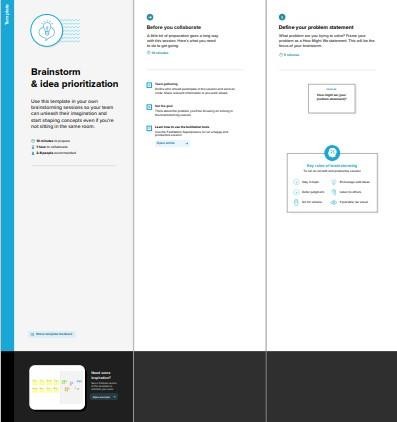
The customer problem statement is a critical component of a project. It benefits everyone involved with the project because it helps people understand why they’re working on the project, providing clarity on the reasons behind the product or service. Team members will consider how your customers will be impacted by your project, what their thoughts and needs are, and thus come up with truly effective and valuable ways to improve their experience.

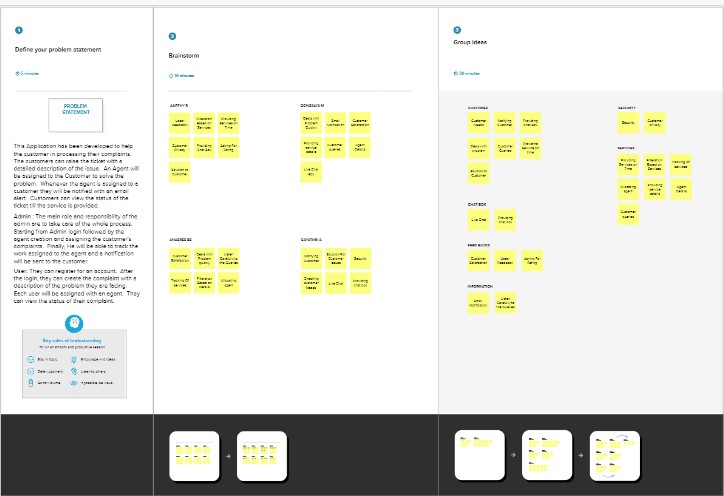
# IDEATION & PROPOSED SOLUTION

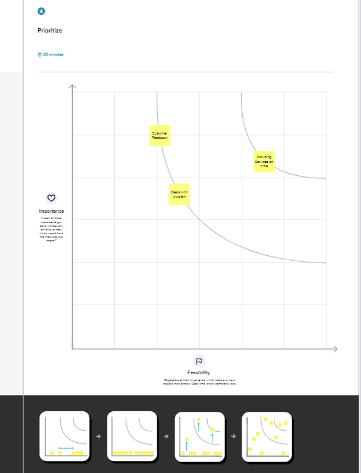
* 1. **EMPATHY MAP CANVAS**



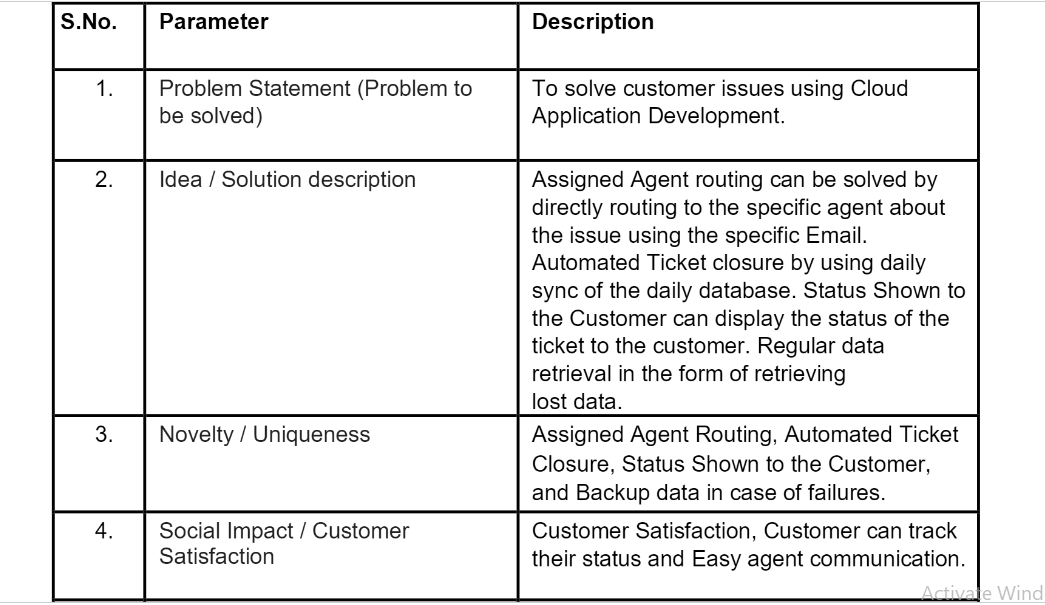
* 1. **IDEATION & BRAINSTORMING**



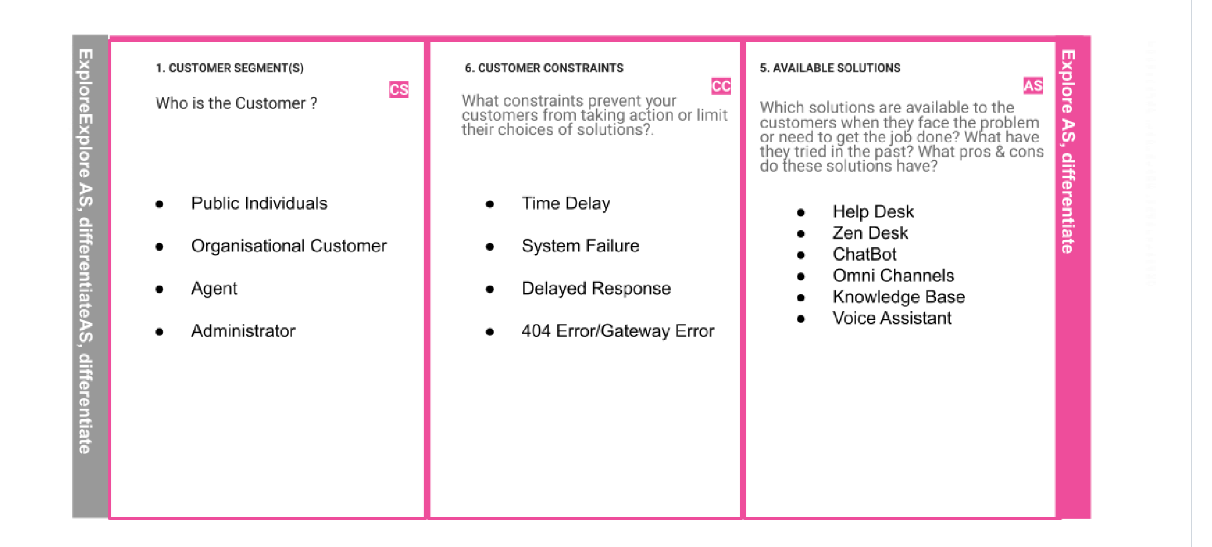




* 1. **PROPOSED SOLUTION**

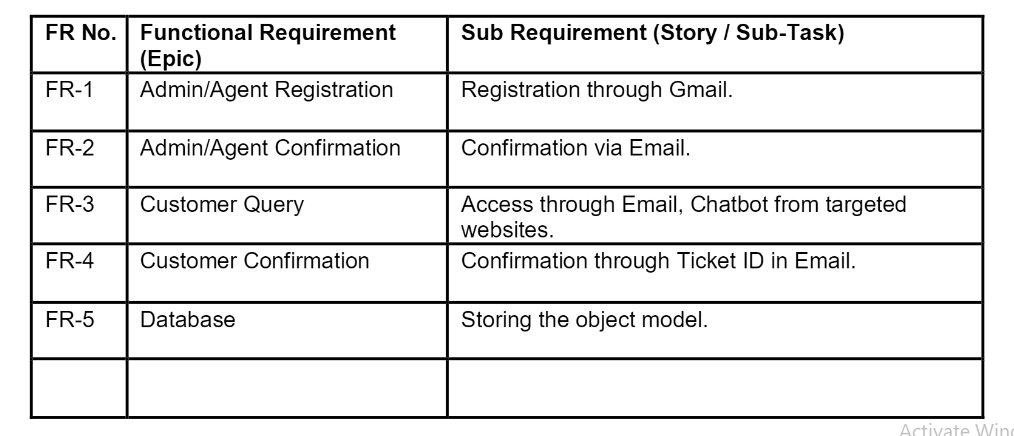


**PROBLEM SOLUTION FIT**

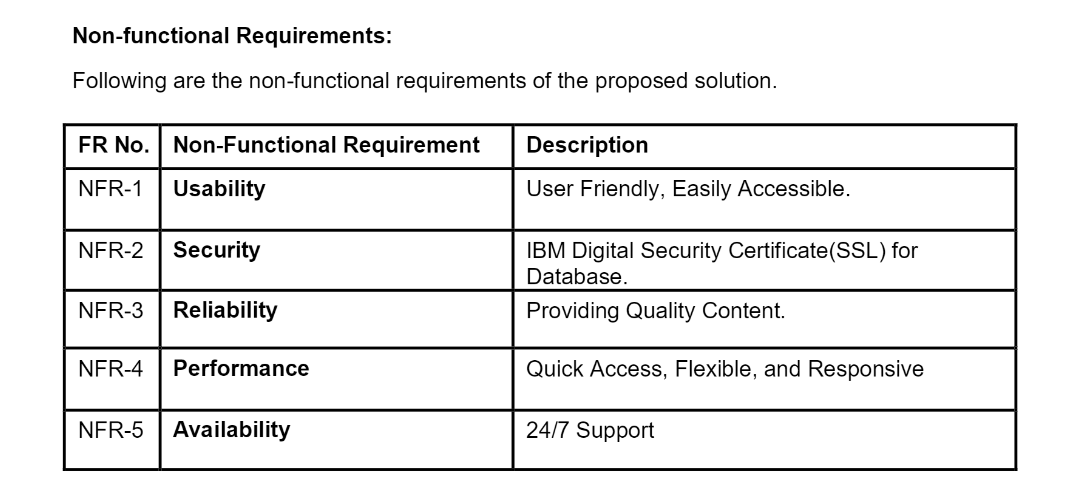
****

# REQUIREMENT ANALYSIS

* 1. **FUNCTIONAL REQUIREMENT**

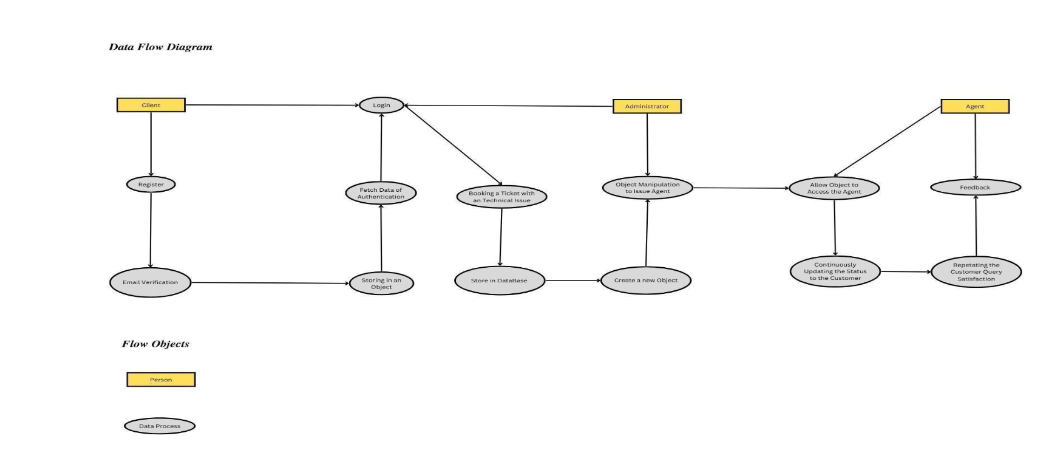
****

* 1. NON-FUNCTIONAL REQUIREMENT

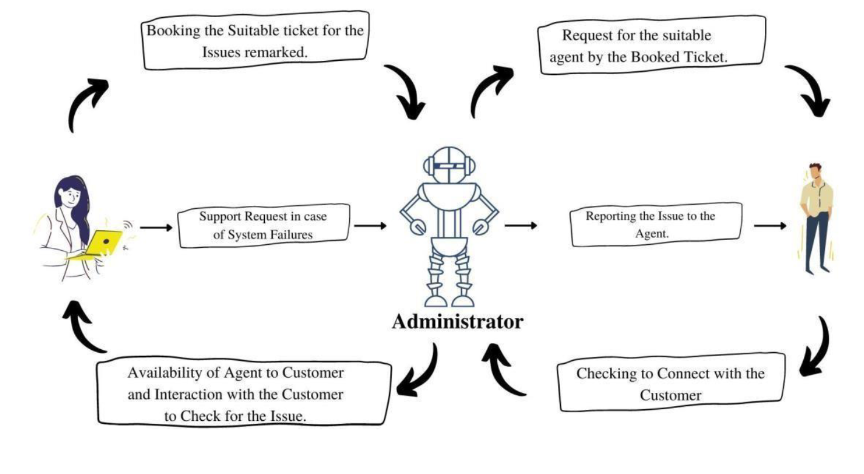


* 1. **DATA FLOW DIAGRAMS**

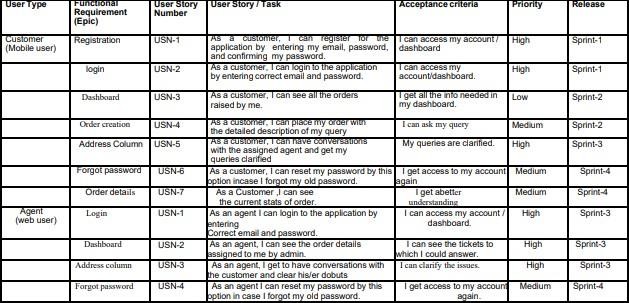
# PROJECT DESIGN

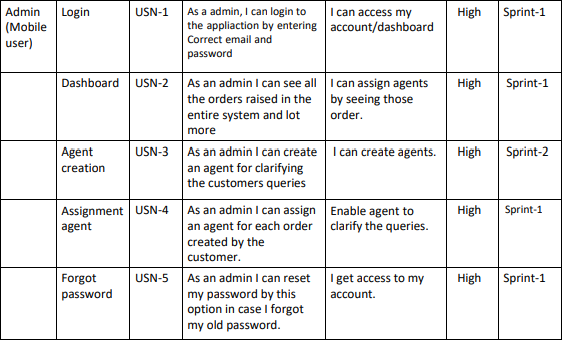


**SOLUTION AND TECHNICAL ARCHITECTURE**



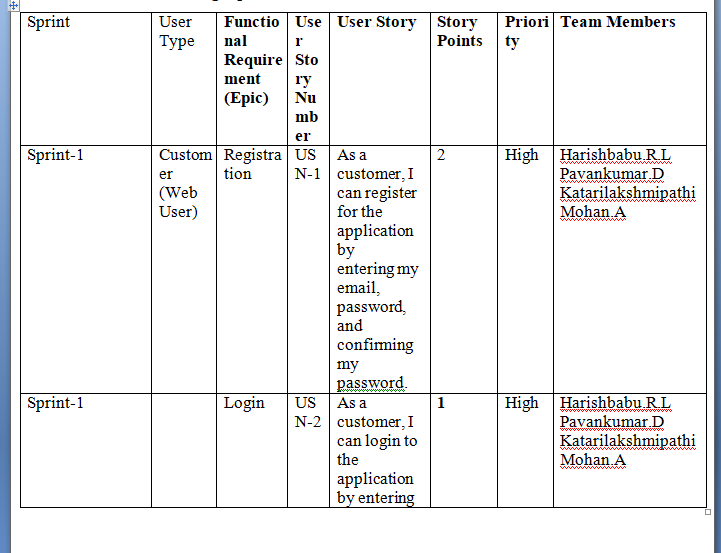
* 1. **USER STORIES**



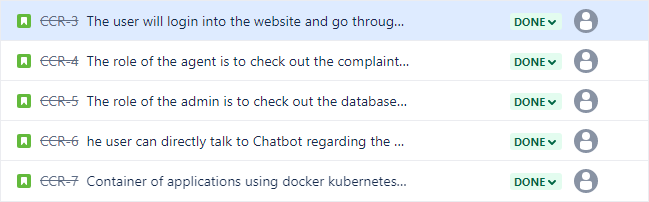


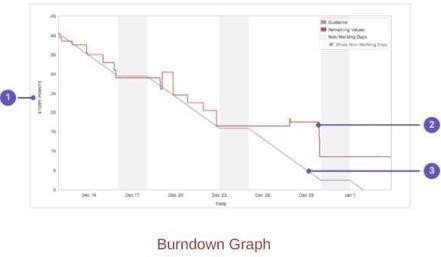
# PROJECT PLANNING & SCHEDULE

* 1. **SPRINT DELIVERY SCHEDULE**

****

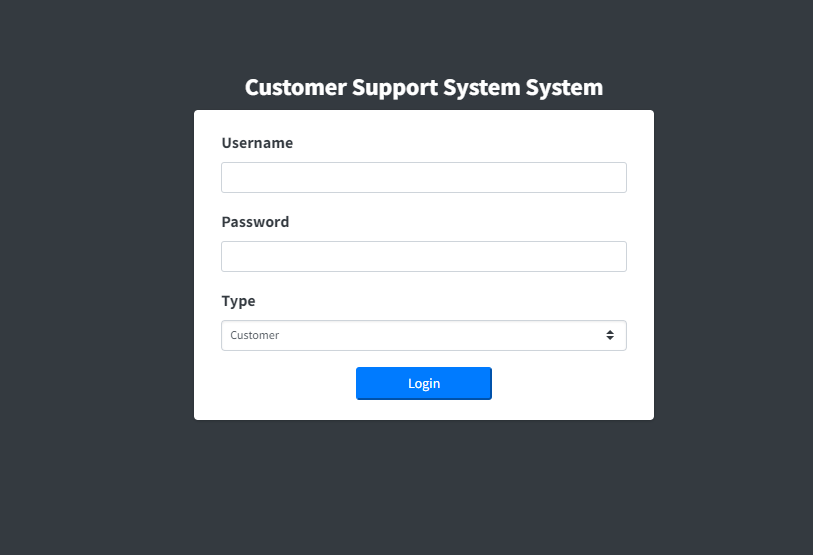
* 1. **REPORTS FROM JIRA**

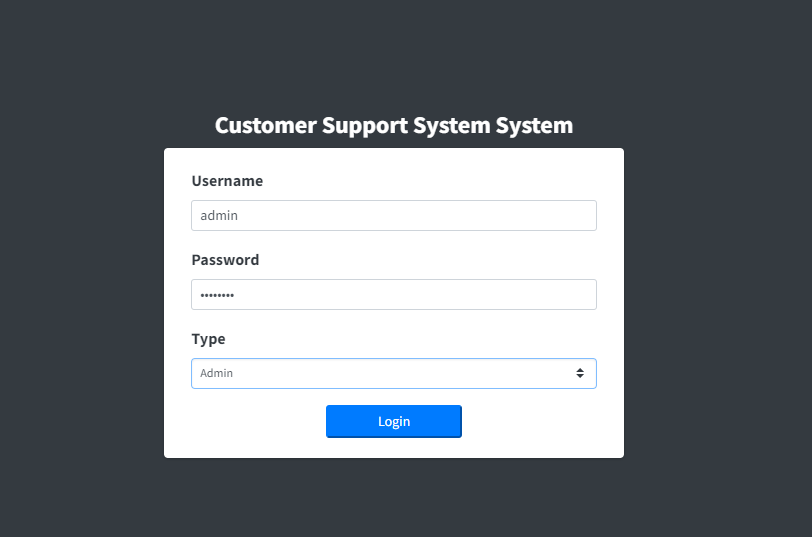




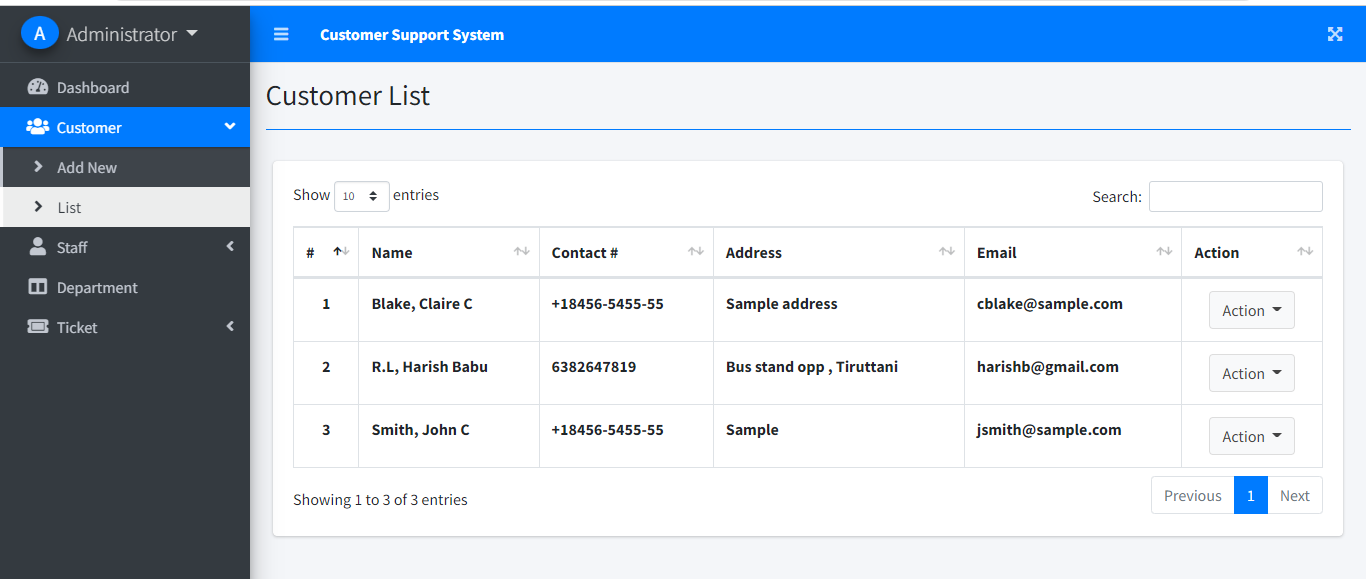
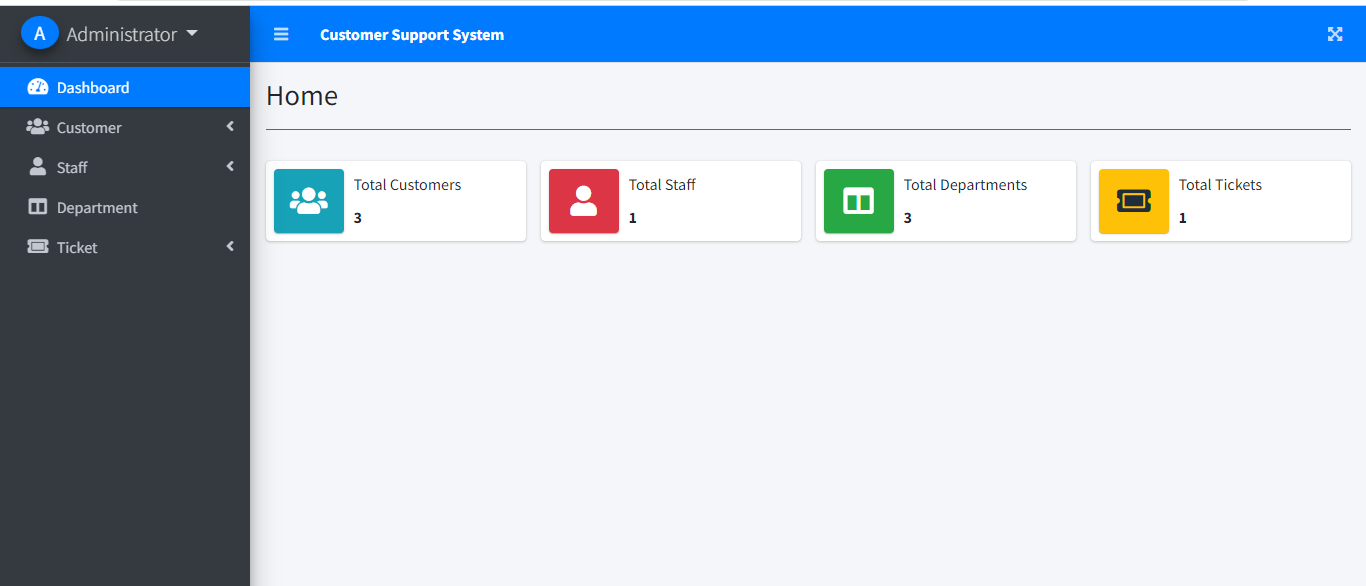
* 1. **FEATURE 1**

# CODING & SOLUTIONING

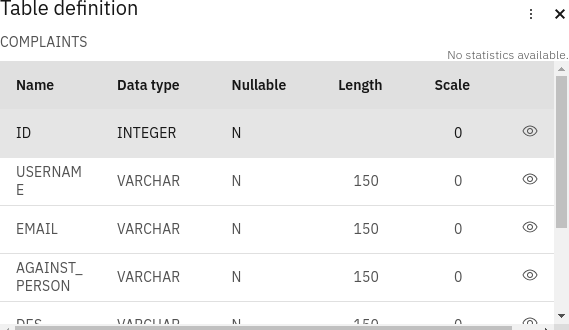
****

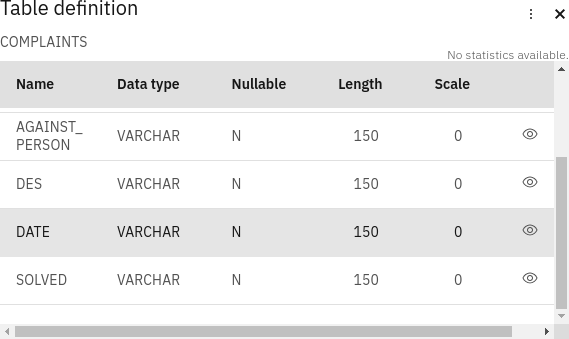


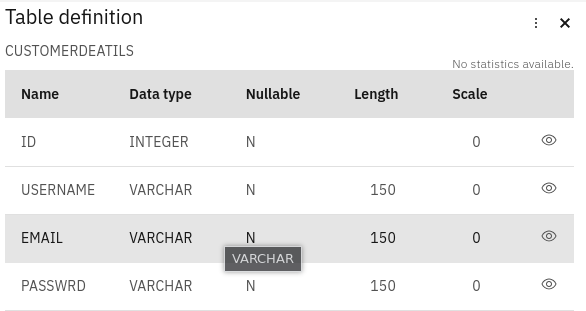
**FEATURE 2**



* 1. **DATABASE SCHEMA**







* 1. **TEST CASES**

# TESTING

* + 1. **FUNCTIONAL TESTING**

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

* + 1. **WHITE BOX TESTING:**

Testing based on an analysis of internal workings and structure of a piece of software. This testing can be done sing the percentage value of load and energy. The tester should know what exactly is done in the internal program. Includes techniques such as Branch Testing and Path Testing. Also known as Structural Testing and Glass Box Testing.

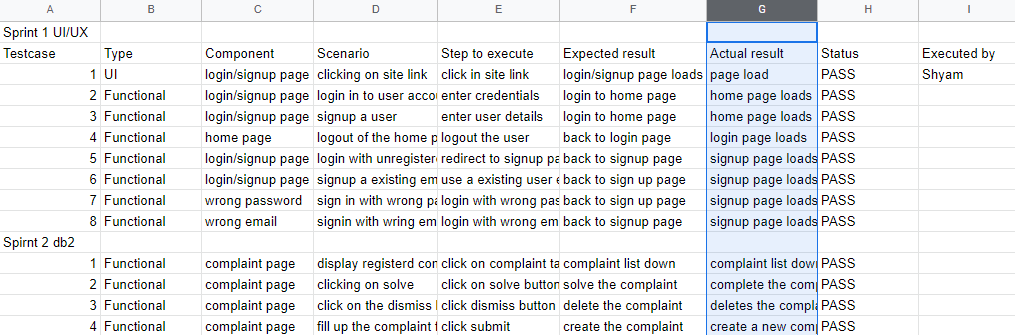
* + 1. **BLACK BOX TESTING:**

Testing without knowledge of the internal workings of the item being tested. Tests are usually functional. This testing can be done by the user who has no knowledge of how the shortest path is found.

* 1. **USER ACCEPTANCE TESTING**

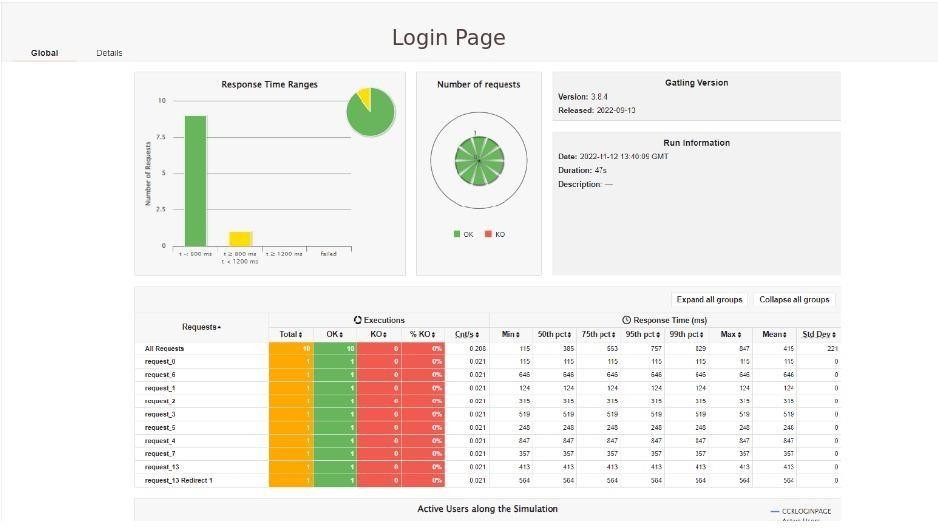
Acceptance testing can be defined in many ways, but a simple definition is the succeeds when the software functions in a manner that can be reasonable expected by the customer. After the acceptance test has been conducted, one of the two possible conditions exists. This is to fine whether the inputs are accepted by the database or other validations. For example accept only numbers in the numeric field, date format data in the date field. Also the null check for the not null fields. If any error occurs then show the error messages. The function of performance characteristics to specification and is accepted. A deviation from specification is uncovered and a deficiency list is created. User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

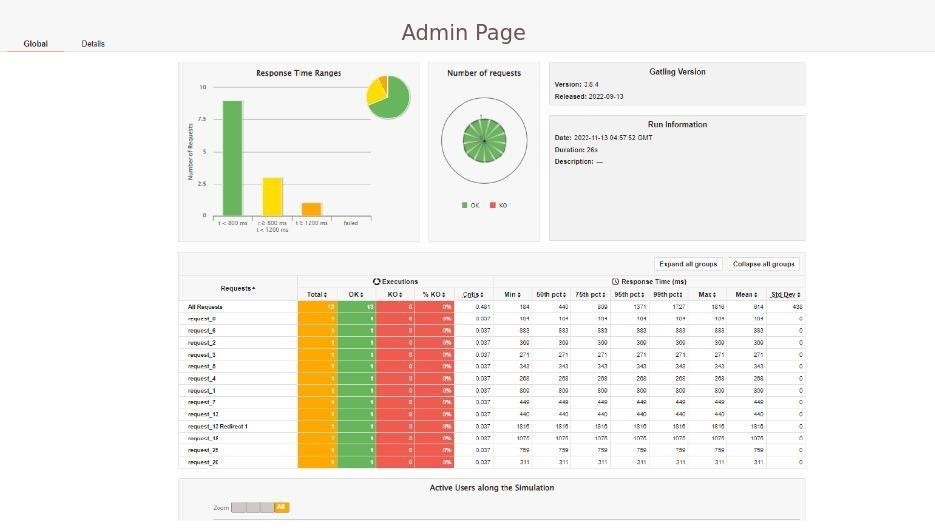
* 1. **TEST RESULTS**

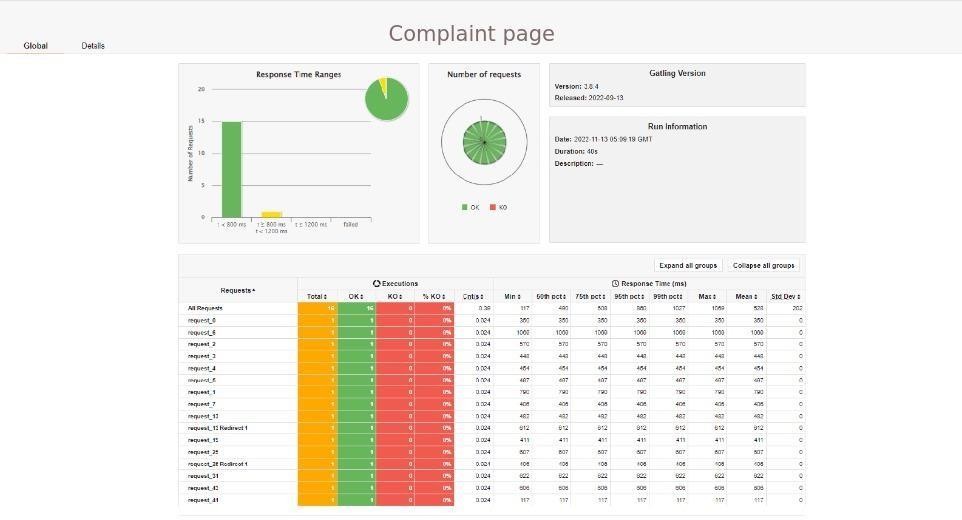


**9.1 PERFORMANCE METRICES**

# RESULTS









**Advantage**

# ADVANTAGES &DISADVANTAGES

* Flow sheet is a powerful tool to monitor clinical data and track trends
* Provides a dashboard of who needs what
* Provides total population data reporting with no chart abstraction
* Generates revenue (it shows when services are needed)
* Provides outreach information at fingertips
* Improves team-based care
* Smaller software package than EHRs
* Creating loyal customers through good customer service can provide businesses with lucrative long-term relationships.
* Customer loyalty. Loyal customers have many benefits for businesses

**Disadvantage**

* Disease-specific, not longitudinal
* Does not include information necessary for billing
* Requires hardware, software and maintenance
* Requires data entry and data maintenance
* Parallel documentation system (i.e., some information has to be entered in two systems)
* Can’t stand alone, must have an additional documentation system.
* Experience burnout and stress. Working as a customer service representative requires you to maintain a friendly demeanour at all times, regardless of how customers act or how you personally feel

# CONCLUSION

Companies today are modernizing customer care, using advanced AI to ensure a positive customer experience starting from the first interaction and throughout the buyer's journey. To properly manage customer care, companies must understand how they are succeeding and what needs improvement. This requires establishing key performance indicators (KPIs) for customer service and creating a system of gathering metrics across channels. In conclusion, customer care, involves the use of basic ethics and any company who wants to have success and grow, needs to remember, that in order to do so, it must begin with establishing a code of ethics in regards to how each employee is to handle the dealing with customers. Customers are at the heart of the company and its growth or decline. Customer care involves, the treatment, care, loyalty, trust the employee should extend to the consumer, as well in life. This concept can be applied to so much more than just customer care. People need to treat others with respect and kindness; people should try to take others into consideration when making any decision. If more people were to practice this policy, chances are the world would be a better, more understanding place for all to exist. Thereby, the customer care registry would be far helpful and approachable. It offers easy tracking, recording and notification than any other means.

# FUTURE SCOPE

The current state of customer care registery, in so many companies, looks something like this:

* Customer acquisition is prioritised over retention
* Customer service investment projects are sidelined.
* Departmental efficiency is of highest priority.
* Businesses see employees in the customer service department as short-term and disposable. They are there to fulfil a specific, repetitive, purpose.
* Employees are considered unskilled and leaders hire accordingly.
* New agents view customer service as a ‘last resort’ or ‘short term’ job. People often see careers in customer support as unambitious.
* Agent training rarely goes beyond product and people skills.

In the next 3-5 years, we expect to see these **future customer care registry trends**:

* The shift from a primarily 'cost centre' to primarily 'growth centre' worldview.
* The job desk for a customer care registry director will focus more on leadership, innovation, and ability to drive company-wide improvement.
* Customer service will shift to become a strategic partner of marketing, sales, and product development. CS will help with direction, project prioritisation, and impact.
* A need for customer service leaders to take a highly strategic seat at the table. They’ll need to argue for investment in talent, technology, and innovation.
* A shift in performance metrics. Forget of resolved tickets. In the future, we'll measure performance based on of customers saved from the precipice of churn.
* A career in customer care registry will not be a last resort. Top graduates will prioritise getting an education in strategic customer interaction.
* Focus on ticket deflection will reduce because brands will view each customer interaction as an opportunity to learn, build a relationship, and grow profits. They deserve a well-trained, human touch.

Modern and developing technology enables this future to exist. With new technology, administrative tasks will tend toward zero.

* The sole purpose of the customer service is to meet the expectations of the customers so that they are satisfied with the outcome. These services are also available to understand the queries of the customers and ensure that they enjoy a cost-effective experience after purchasing any product from the respective company.

**SOURCE CODE**

# APPENDIX

from flask import Flask,render\_template,request,url\_for,session,redirect from flask\_mysqldb import MySQL

from sendmail import sendemail,forget\_password\_mail,updated\_password\_mail,solve\_mail

import json import ibm\_db import re

from random import randint from datetime import date

app = Flask( name )

# <http://remotemysql.com/>

# dsn\_hostname = "b0aebb68-94fa-46ec-a1fc- 1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud"

# dsn\_uid = "dmt13873"

# dsn\_pwd = "740yZ1Yq8Uj2E4qm" # dsn\_database = 'bludb'

# dsn\_port = 31249

conn = ibm\_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa- 46ec-a1fc- 1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=3124 9;SECURITY=SSL;SSLServerCertificate=src/DigiCertGlobalRootCA.crt;UID=d mt13873;PWD=740yZ1Yq8Uj2E4qm",'','') # type: ignore

print(conn)

print("connection successful...")

# database configuration

# app.config['MYSQL\_HOST'] = 'sql12.freesqldatabase.com' # app.config['MYSQL\_USER'] = 'sql12552843'

# app.config['MYSQL\_PASSWORD'] = 'zWIzHmXNi8' # app.config['MYSQL\_DB'] = 'sql12552843' app.secret\_key = "super secret key"

# mysql = MySQL(app)

@app.route('/') def home():

today = date.today()

current\_date = today.strftime('%d/%m/%Y') if "google\_token" in session:

session["current\_date"] = current\_date return render\_template('home.html')

if "username" in session: session["current\_date"] = current\_date return render\_template('home.html')

return render\_template('index.html')

# manually registration @app.route('/register',methods=["POST"]) def register():

if request.method == 'POST': name = request.form['uname']

mail = request.form['mail'] pwd = request.form['pwd']

cpwd = request.form['confirmpwd']

if not re.match(r'[^@]+@[^@]+\.[^@]+', mail): msg = 'Invalid email address !'

return render\_template('index.html',signupmsg=msg) if pwd != cpwd:

msg = 'Please enter correct confirm password'

return render\_template('index.html',signupmsg=msg) # check account is exists or not

# cursor = mysql.connection.cursor() rCheckQuery = ''

result = ibm\_db.exec\_immediate(conn,f"SELECT \* FROM customerdeatils WHERE email LIKE '{mail}'")

# cursor.execute('SELECT \* FROM customerdeatils WHERE email LIKE

% s',[mail])

# existing\_user = cursor.fetchone() # cursor.close()

existing\_user = ibm\_db.fetch\_row(result) #exits

if existing\_user:

msg = 'Account already exists please login.'

return render\_template('index.html',signupmsg = msg) # not exists

# cursor = mysql.connection.cursor()

# cursor.execute('INSERT INTO customerdeatils VALUES(null,% s,% s,% s)',(name,mail,pwd))

# mysql.connection.commit() # cursor.close()

regInsertQuery = f"INSERT INTO customerdeatils (username,email,passwrd) VALUES('{name}','{mail}','{pwd}')"

insertflag = ibm\_db.exec\_immediate(conn,regInsertQuery) msg = 'Your registration successfully completed.'

# send mail sendemail(mail,'Account\_creation')

return render\_template('index.html',signupmsg = msg) # admin page

@app.route('/admin/<which>') def admin(which):

if which == 'customers':

# cursor = mysql.connection.cursor()

result = ibm\_db.exec\_immediate(conn,'SELECT \* FROM customerdeatils') data = []

while ibm\_db.fetch\_row(result):

temp = [ibm\_db.result(result,0),ibm\_db.result(result,1),ibm\_db.result(result,2),ibm\_db.re sult(result,3)]

data.append(temp)

return render\_template('admin.html',customers=data,complaints=None) if which == 'complaints':

# cursor = mysql.connection.cursor()

result = ibm\_db.exec\_immediate(conn,'SELECT \* FROM complaints') data = []

while ibm\_db.fetch\_row(result):

temp = [ibm\_db.result(result,0),ibm\_db.result(result,1),ibm\_db.result(result,2),ibm\_db.re sult(result,3),ibm\_db.result(result,4),ibm\_db.result(result,5)]

data.append(temp)

return render\_template('admin.html',customers=None,complaints=data) # admin delete

@app.route('/Delete/<type>/<id>') def Delete(type,id):

if type == 'customers':

# cursor = mysql.connection.cursor()

result = ibm\_db.exec(conn,f'DELETE FROM customerdeatils WHERE id = "{id}"')

# mysql.connection.commit() # cursor.close()

return redirect(url\_for('admin',which='customers')) if type == 'complaints':

# cursor = mysql.connection.cursor()

result = ibm\_db.exec\_immediate(conn,f'DELETE FROM complaints WHERE id = {id}')

# mysql.connection.commit() # cursor.close()

return redirect(url\_for('admin',which='complaints')) # manually login @app.route('/login',methods=['POST','GET'])

def login():

if request.method == 'POST': mail = request.form['mail1']

password = request.form['pwd1'] # login is admin or not

if mail == "admin" and password == 'admin@1810': return redirect(url\_for('admin',which='customers'))

# check account is exists or not

# cursor = mysql.connection.cursor()

query = "SELECT \* FROM customerdeatils WHERE email=? AND passwrd=?"

stmt = ibm\_db.prepare(conn, query) # type:ignore ibm\_db.bind\_param(stmt,1,mail) # type:ignore ibm\_db.bind\_param(stmt,2,password) # type:ignore ibm\_db.execute(stmt) # type:ignore

user = ibm\_db.fetch\_assoc(stmt) # type:ignore print(user,password)

#exists if user:

session["username"] = user['USERNAME'] session['mail'] = mail

return render\_template('home.html',username=session["username"],mail=session["mail"

])

else:

msg = 'mail or password is not valid.'

return render\_template('index.html',signinmsg=msg)

if request.method == "GET": return redirect(url\_for('home'))

# logout method @app.route('/logout') def logout():

if "username" in session: session.pop("username")

if "google\_token" in session: session.pop("google\_token") session.pop("mail")

if "mail" in session: session.pop("mail")

return redirect(url\_for('home'))

# complaint register @app.route('/complaint',methods=['POST']) def complaint():

if request.method == 'POST':

complaint\_name = request.form['complaint\_name'] name = request.form['name']

mail = request.form['email']

against\_person = request.form['against\_person'] date = request.form["date"]

des = request.form['complaint\_des'] # cursor = mysql.connection.cursor()

if not name == session["username"] or not mail == session["mail"]: msg = "please don't change username and mail."

return render\_template('home.html',msg=msg)

result = ibm\_db.exec\_immediate(conn,f"INSERT INTO complaints (username,email,against\_person,des,date,solved) VALUES('{name}','{mail}','{against\_person}','{des}','{date}','{'0'}')")

# mysql.connection.commit() # cursor.close()

sendemail(mail,'complaint\_creation')

msg = 'Complaint registerd you check out complaints section.' return render\_template('home.html',msg=msg)

# show complaints and progress @app.route('/showcomplaints') def showcomplaints():

# cursor = mysql.connection.cursor()

# cursor.execute("SELECT \* FROM complaints WHERE username= % s AND email=% s",(session["username"],session["mail"]))

# details = cursor.fetchall() # cursor.close()

query = "SELECT \* FROM complaints WHERE username=? AND email=?" stmt = ibm\_db.prepare(conn, query) # type:ignore ibm\_db.bind\_param(stmt,1,session["username"]) # type:ignore ibm\_db.bind\_param(stmt,2,session['mail']) # type:ignore ibm\_db.execute(stmt)

data = []

while ibm\_db.fetch\_row(stmt):

temp = [ibm\_db.result(stmt,0),ibm\_db.result(stmt,1),ibm\_db.result(stmt,2),ibm\_db.result (stmt,3),ibm\_db.result(stmt,4),ibm\_db.result(stmt,5),ibm\_db.result(stmt,6)]

print(temp) data.append(temp)

return render\_template('complaints.html',complaints=data)

# update complaint @app.route('/solve',methods=["POST"]) def solve\_complaint():

if request.method == "POST": c\_id = request.form['c\_id']

print(c\_id)

# cursor = mysql.connection.cursor()

# cursor.execute("UPDATE complaints SET solved = % s WHERE id = % s",('1',c\_id,))

query = "UPDATE complaints SET solved = '1' WHERE id = ?" # mysql.connection.commit()

stmt = ibm\_db.prepare(conn, query) # type:ignore ibm\_db.bind\_param(stmt,1,c\_id) # type:ignore ibm\_db.execute(stmt)

detail = ibm\_db.result(stmt,0) print(detail)

# cursor.execute("SELECT \* FROM complaints WHERE id = % s",[c\_id]) query2 = "SELECT \* FROM complaints WHERE id = ?"

stmt1 = ibm\_db.prepare(conn, query2) # type:ignore

ibm\_db.bind\_param(stmt1,1,c\_id) # type:ignore ibm\_db.execute(stmt1)

details = ibm\_db.result(stmt1,0) # cursor.close()

print(details)

# solve\_mail(session['mail'],'user')

return redirect(url\_for('showcomplaints')) return redirect(url\_for('showcomplaints'))

# # admin agent allot

# @app.route('/solve\_admin',methods=["POST"]) # def solve\_admin():

# if request.method == "POST":

# c\_id = request.form['c\_id']

# # cursor = mysql.connection.cursor()

# cursor.execute("SELECT \* FROM complaints WHERE id = % s",[c\_id]) # query = "SELECT \* FROM complaints WHERE id = ?"

# details = cursor.fetchone() # cursor.close()

# solve\_mail(details[3],'admin')

# return redirect(url\_for('admin',which='complaints')) # return redirect(url\_for('admin',which='complaints'))

# remove complaint @app.route('/dismiss',methods=["POST"]) def dismiss\_complaint():

if request.method == "POST":

c\_id = request.form["c\_id"]

# cursor = mysql.connection.cursor()

# cursor.execute("DELETE FROM complaints WHERE id = % s",[c\_id]) # mysql.connection.commit()

# cursor.close()

query = "DELETE FROM complaints WHERE id = ?" stmt = ibm\_db.prepare(conn, query) ibm\_db.bind\_param(stmt,1,c\_id) # type:ignore ibm\_db.execute(stmt)

return redirect(url\_for('showcomplaints')) return redirect(url\_for('showcomplaints'))

# send otp in user mail id @app.route('/send\_otp',methods=["POST","GET"]) def send\_otp():

if request.method == "POST": mail = request.form["mail"]

cursor = mysql.connection.cursor()

cursor.execute("SELECT \* FROM customerdeatils WHERE email = % s",[mail])

temp = cursor.fetchone() cursor.close()

if not temp:

return render\_template('forget.html',type='otp',msg1='Your account doesn\'t exist please register')

otp = randint(10 \*\* 5,10\*\*6) forget\_password\_mail(mail,otp) session["otp"] = otp

return render\_template('forget.html',type='update\_password',tempmail=mail) # forget password method @app.route('/forgetpassword/<type>',methods=["POST","GET"])

def forgetpassword(type): if type == 'otp':

return render\_template('forget.html',type=type) if request.method == "POST":

mail = request.form["mail"] otp = request.form["otp"]

pwd = request.form["password"] c\_pwd = request.form["con\_pwd"] print(otp,session['otp'])

if not pwd == c\_pwd:

msg = 'Please Enter Password properly'

return render\_template('forget.html',type='updatePassword',msg=msg) if not otp == str(session['otp']):

msg = "Your OTP is Incorrect."

return render\_template('forget.html',type='updatePassword',msg=msg) cursor = mysql.connection.cursor()

cursor.execute("UPDATE customerdeatils SET passwrd = % s WHERE email = % s",(pwd,mail))

mysql.connection.commit() cursor.close()

msg = 'password updated successfully' updated\_password\_mail(mail)

return render\_template('forget.html',type='updatePassword',msg=msg)

if name == ' main ':

app.run(host = '0.0.0.0',port = 8080,debug=True)

**GitHub & Project Demo Link**

**GitHub link:** https://github.com/IBM-EPBL/IBM-Project-28497-1660112909

**Demo link: https**://youtu.be/XVklWCRjMCo