CUSTOMER CARE REGISTRY

A PROJECT REPORT

Submitted by

HARINI R	**732219EC034**
DEEPIKA S	**732219EC019**
KEERTHANA N	**732219EC048**
KEERTHANA N	**732219EC049**

TEAM ID : PNT2022TMID19702

NANDHA COLLEGE OF ENGINEERING

(An Autonomous Institution)

Perundurai,Erode

TABLE OF CONTENTS:

1. INTRODUCTION

Project Overview

- A Web page is designed for the public where they can send their issues to customer care website.
- After registering the problem, the customer can directly interact to chatbot regarding the services offered by web portal.
- They can also get recommendations based on information provide by them in the web portal regarding their issues.
- Handling customer complaints and providing solution to them regarding their issues and also collecting and analyzing customer feedback
- The Customer Service Desk is a web based project. Customer Service also known as Client Service is the provision of service to customers' Its significance varies by product, industry and domain.
- In many cases customer services is more important if the information relates to a service as opposed to a Customer. Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition.

1.2 Purpose

The Purpose of our Project is

- An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through and e-mail.
- The system should have capability to integrate with any Service Provider from any domain or industry like Banking. Telecom Insurance. etc.
- Customer Service also known as Client Service is the provision of service to customers Its significance varies by product industry and domain.
- In many cases customer services is more important if the information relates to a service as opposed to as Customer Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition

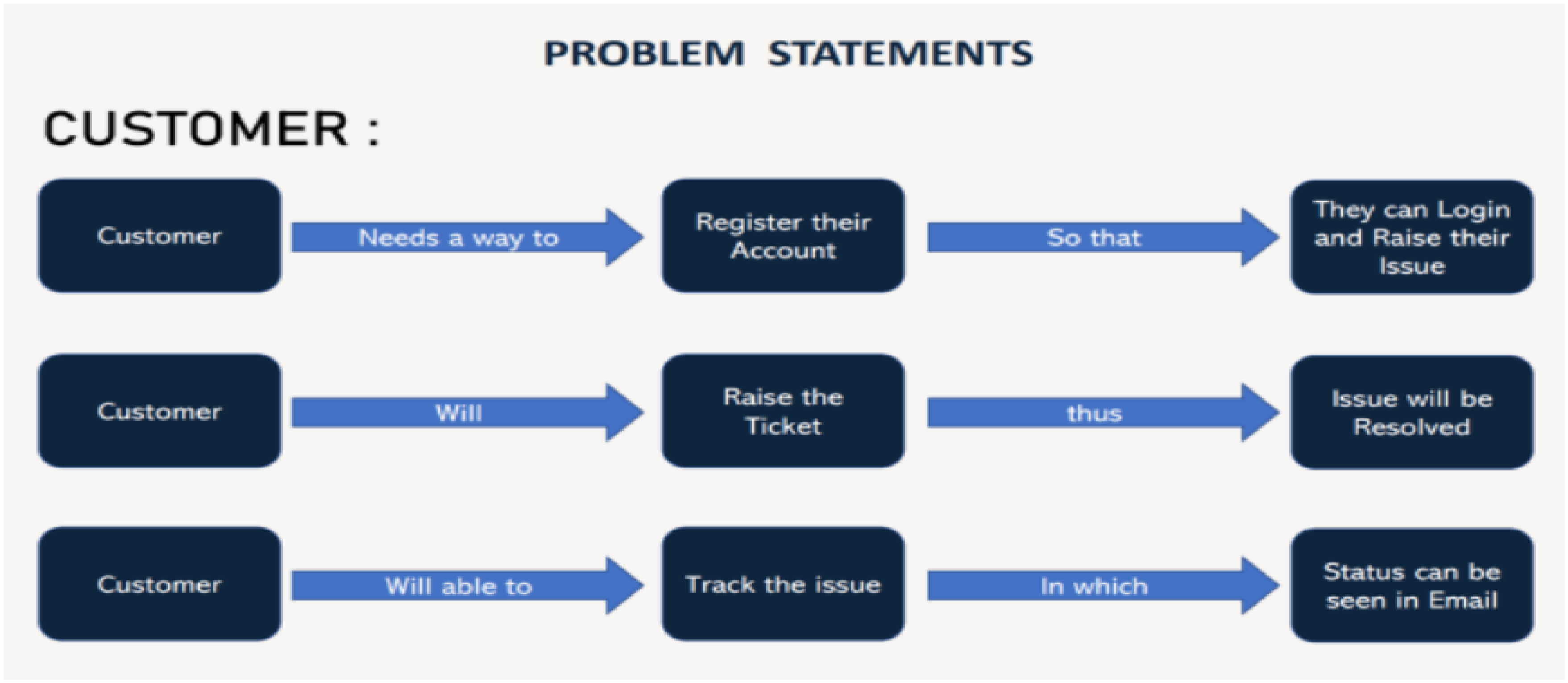
2.LITERATURE SURVEY

Problem Statement Definition

A problem statement is a concise description of the problem or issues a project seeks to address.

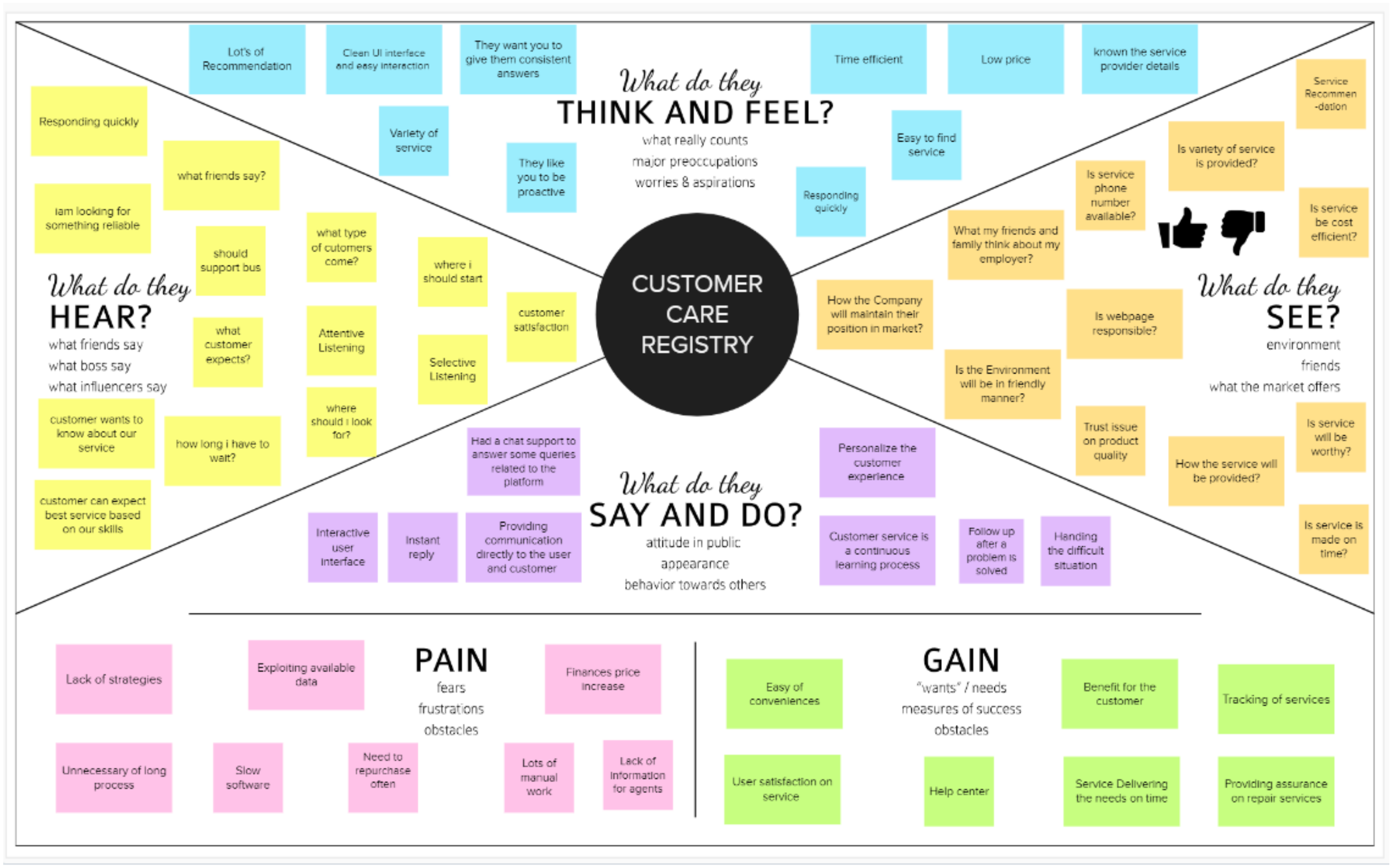
The problem statement identifies the current state, the desired future state and any gaps between the two.

A problem statement is an important communication tool that can help ensure everyone working on a project knows what the problem they need to address is and why the project is important



3.IDEATION & PROPOSED SOLUTION

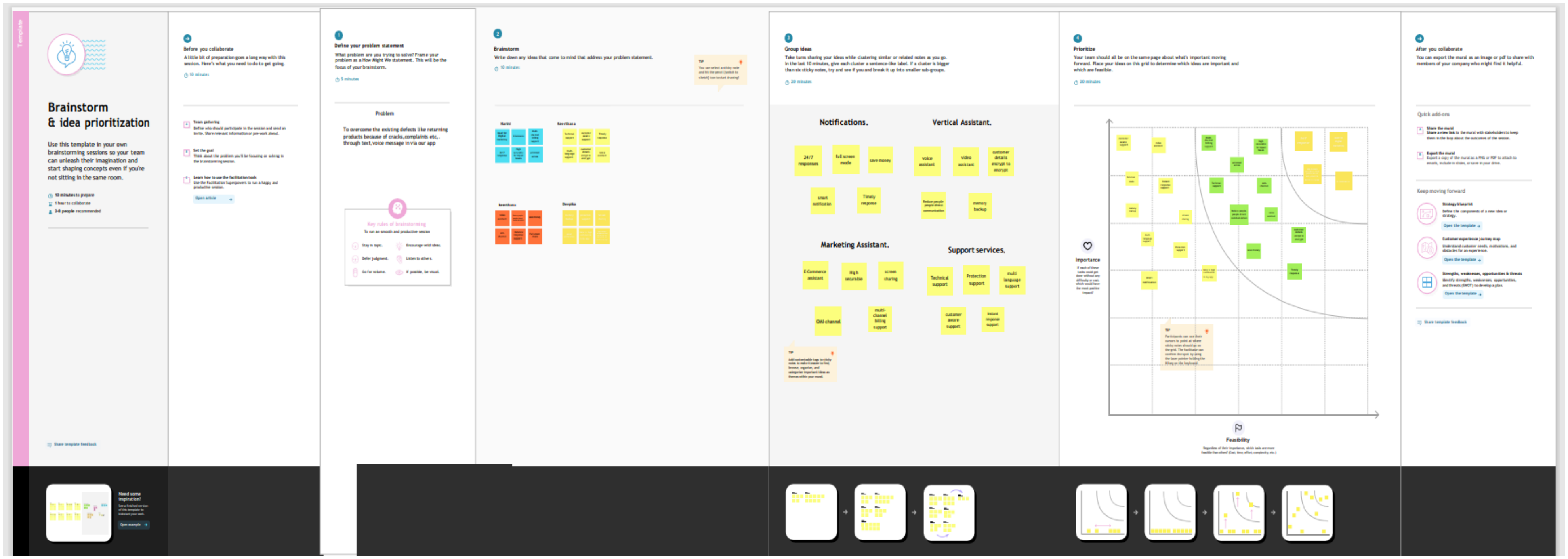
Empathy Map Canvas An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community



CUSTOMER CARE REGISTRY
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Lot's of Recommendation
Clean UI interface and easy interaction
They want you to give them consistent answers
Variety of service
They like you to be proactive
Time efficient
Low price
known the service provider details
Easy to find service
Responding quickly
What do they HEAR?
what friends say
what boss say
what influencers say
Responding quickly
what friends say?
Iam looking for something reliable
should support bus
what type of cutomers come?
where i should start
customer satisfaction
what customer expects?
Attentive Listening
Selective Listening
customer wants to know about our service
how long i have to wait?
where should i look for?
customer can expect best service based on our skills
What do they SEE?
environment
friends
what the market offers
Service Recommen -dation
Is variety of service is provided?
Is service phone number available?
Is service be cost efficient?
What my friends and family think about my employer?
How the Company will maintain their position in market?
Is webpage responsible?
Is the Environment will be in friendly manner?
Trust issue on product quality
Is service will be worthy?
How the service will be provided?
Is service is made on time?
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Had a chat support to answer some queries related to the platform
Personalize the customer experience
Interactive user interface
Instant reply
Providing communication directly to the user and customer
Customer service is a continuous learning process
Follow up after a problem is solved
Handing the difficult situation
PAIN
fears
frustrations
obstacles
Lack of strategies
Exploiting available data
Finances price increase
Unnecessary of long process
Slow software
Need to repurchase often
Lots of manual work
Lack of information for agents
GAIN
"wants" / needs
measures of success
obstacles
Easy of conveniences
Benefit for the customer
Tracking of services
User satisfaction on service
Help center
Service Delivering the needs on time
Providing assurance on repair services

Ideation & Brainstorming

In a brainstorming session, participants are encouraged to freely share any ideas that may come to mind.
According to the theory, by coming up with a lot of ideas, the brainstorming group is more likely to find a workable solution to the problem they are trying to solve.
With the creation of various brainstorming software tools, such Brightidea and Idea wake, the distinction between ideation and brainstorming has gotten a little bit more hazy.
These software applications are made to inspire staff members to come up with fresh suggestions for enhancing business operations and, eventually, bottom-line profitability.
The applications frequently mix the ideation and brainstorming processes in that they can be used by individual employees, but businesses can replicate brainstorming sessions by having multiple employees use the software to produce fresh ideas for a particular problem.



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
10 minutes

Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

1 Define your problem statement
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.
5 minutes

Problem
To overcome the existing defects like returning products because of cracks,complaints etc,. through text,voice message in via our app

Key rules of brainstorming
To run an smooth and productive session
Stay in topic. Encourage wild ideas. Defer judgment. Listen to others. Go for volume. If possible, be visual.

2 Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes

3 Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes

Notifications.
24/7 responses · full screen mode · save money · smart notification · Timely response

Vertical Assistant.
voice assistant · video assistant · customer details encrypt to encrypt · Reduce people people direct communication · memory backup

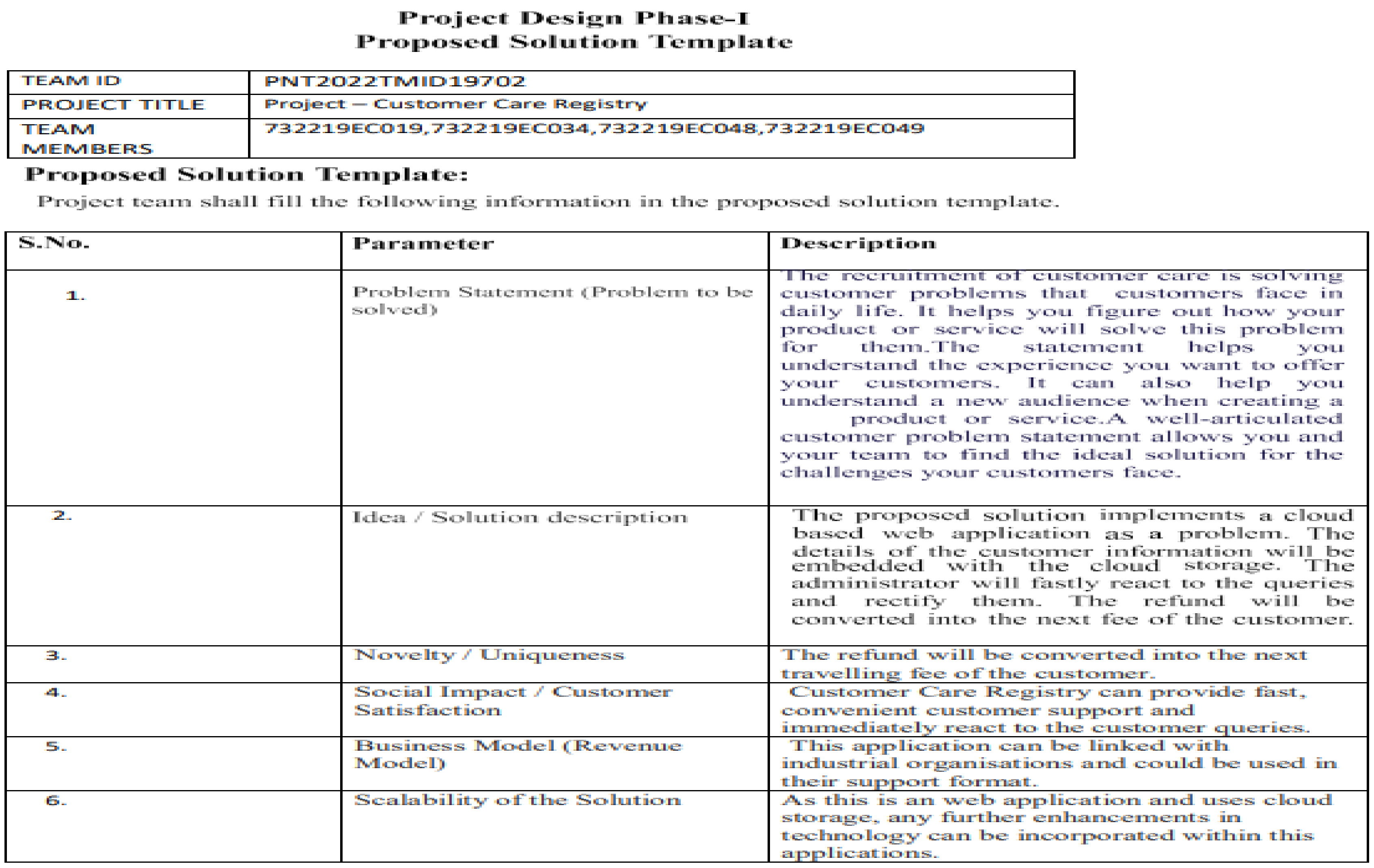
Marketing Assistant.
E-Commerce assistant · High securable · screen sharing · OMI-channel · multi-channel billing support

Support services.
Technical support · Protection support · multi language support · customer aware support · instant response support

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4 Prioritize
20 minutes

Proposed Solution



Project Design Phase-I
Proposed Solution Template

TEAM ID	PNT2022TMID19702
PROJECT TITLE	Project – Customer Care Registry
TEAM MEMBERS	732219EC019,732219EC034,732219EC048,732219EC049

Proposed Solution Template:

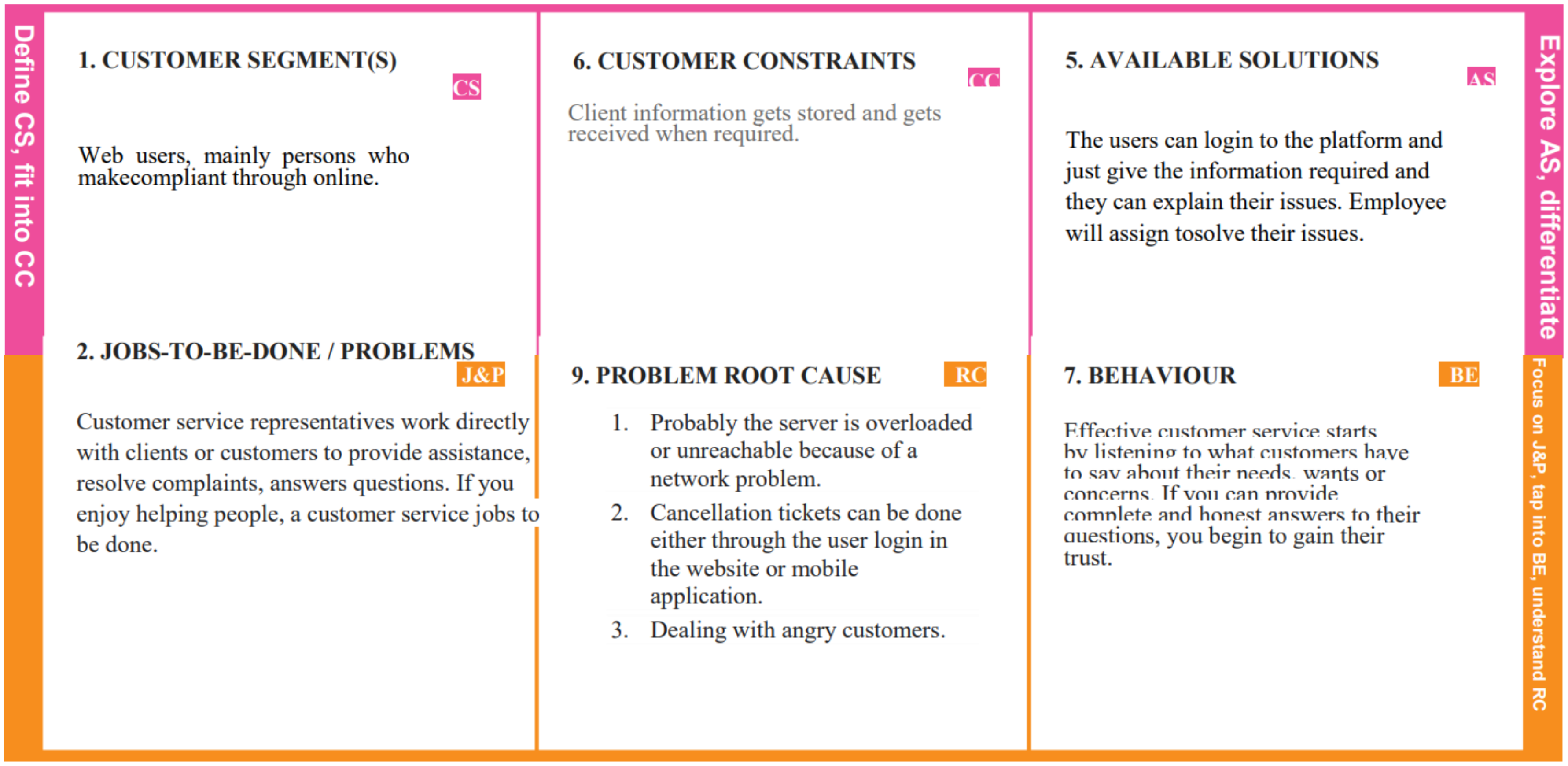
Project team shall fill the following information in the proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The recruitment of customer care is solving customer problems that customers face in daily life. 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 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.
2.	Idea / Solution description	The proposed solution implements a cloud based web application as a problem. The details of the customer information will be embedded with the cloud storage. The administrator will fastly react to the queries and rectify them. The refund will be converted into the next fee of the customer.
3.	Novelty / Uniqueness	The refund will be converted into the next travelling fee of the customer.
4.	Social Impact / Customer Satisfaction	Customer Care Registry can provide fast, convenient customer support and immediately react to the customer queries.
5.	Business Model (Revenue Model)	This application can be linked with industrial organisations and could be used in their support format.
6.	Scalability of the Solution	As this is an web application and uses cloud storage, any further enhancements in technology can be incorporated within this applications.

.

Problem Solution fit

Project Title: CUSTOMER CARE REGISTRY **Project Design Phase-I - Solution Fit Template** **Team ID:** PNT2022TMID19702



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Web users, mainly persons who makecompliant through online.

6. CUSTOMER CONSTRAINTS CC

Client information gets stored and gets received when required.

5. AVAILABLE SOLUTIONS AS

The users can login to the platform and just give the information required and they can explain their issues. Employee will assign tosolve their issues.

Explore AS, differentiate

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

Customer service representatives work directly with clients or customers to provide assistance, resolve complaints, answers questions. If you enjoy helping people, a customer service jobs to be done.

9. PROBLEM ROOT CAUSE RC

1. Probably the server is overloaded or unreachable because of a network problem.
2. Cancellation tickets can be done either through the user login in the website or mobile application.
3. Dealing with angry customers.

7. BEHAVIOUR BE

Effective customer service starts by listening to what customers have to say about their needs, wants or concerns. If you can provide complete and honest answers to their questions, you begin to gain their trust.

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

Identify strong TR & EM

3. TRIGGERS

TR

Not knowing the criteria for solving the queries. User can know about the platform through browsing or via friends

4. EMOTIONS: BEFORE / AFTER

EM

Before: Getting fault product from the online Website.

After: Queries clear for the fault product.

10. YOUR SOLUTION

SL

Our solution to solving the queries. To solve the queries agent is assign to the user. User explains theirqueries so the agent will solve the problem.

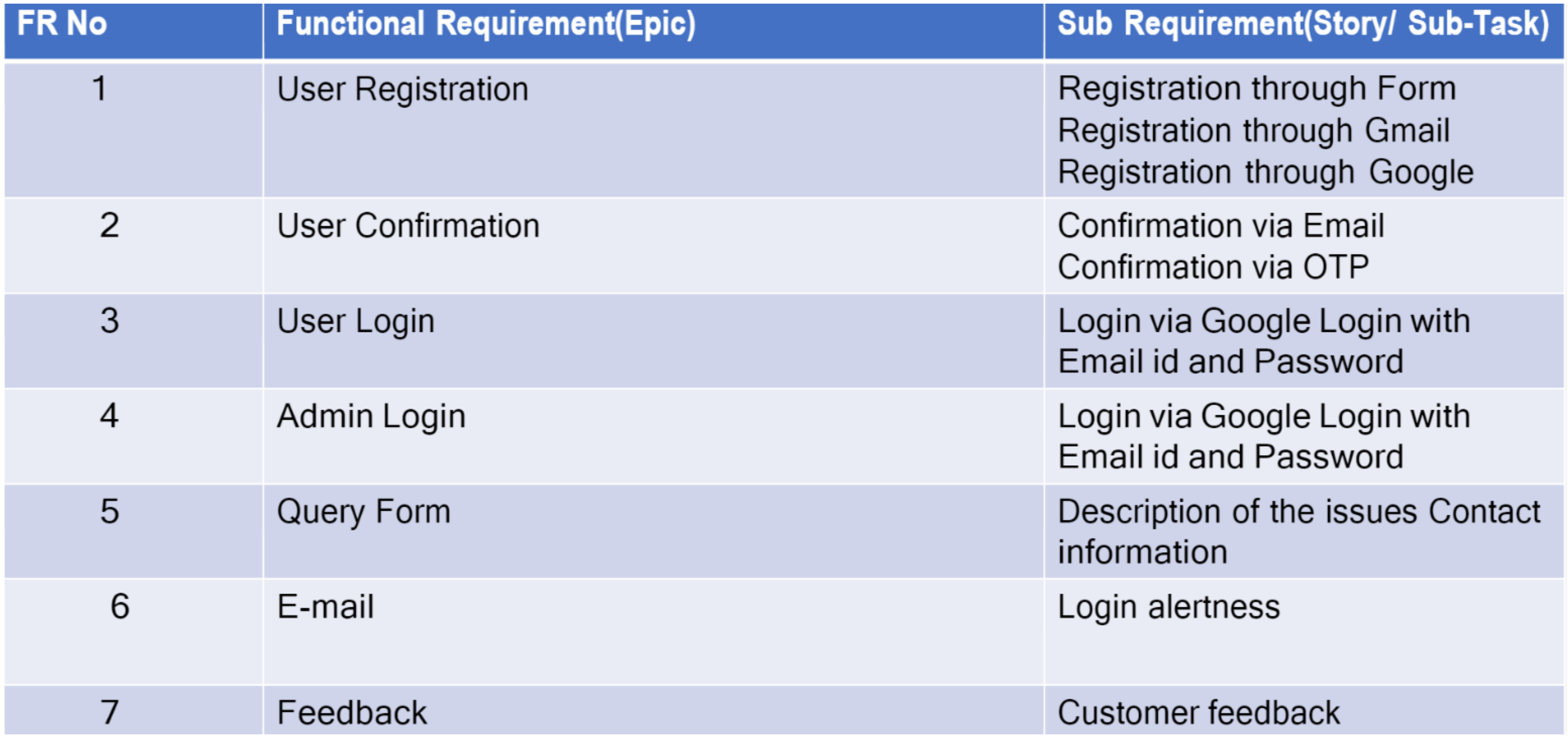
8.CHANNELS of BEHAVIOUR

CH

Online: login to the website and explaintheir issues of the product.

4.REQUIREMENTS ANALAYSIS

Functional requirements



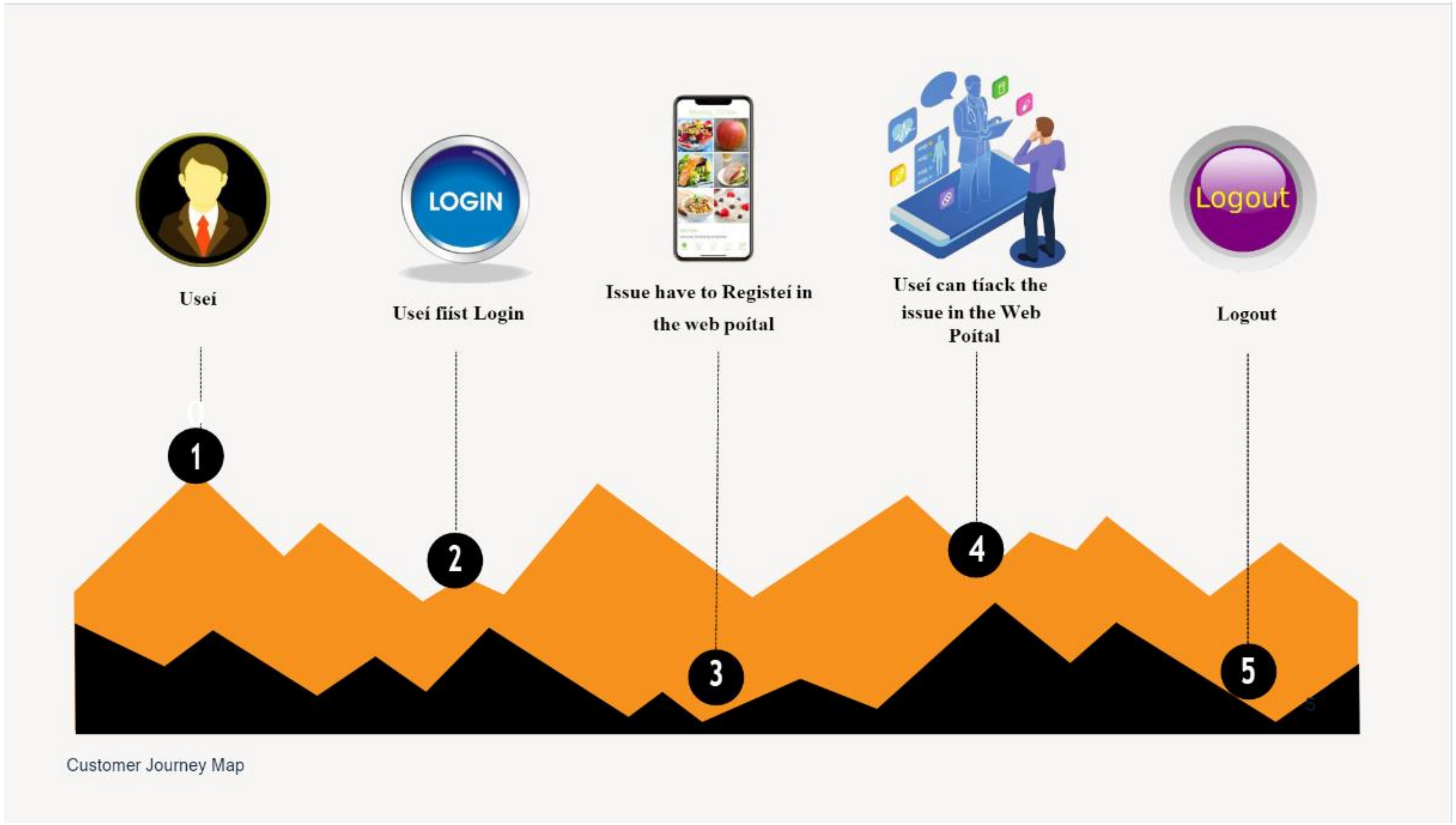
FR No	Functional Requirement(Epic)	Sub Requirement(Story/ Sub-Task)
1	User Registration	Registration through Form Registration through Gmail Registration through Google
2	User Confirmation	Confirmation via Email Confirmation via OTP
3	User Login	Login via Google Login with Email id and Password
4	Admin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

Non-Functional requirement

FR No	Non-Functional Requirement	Description
1	Usability	To provide the solution to the problem
2	Security	Track of login authentication
3	Reliability	Tracking of decade status through email
4	Performance	Effective development of web application
5	Availability	24/7 service
6	Scalability	Agents scalability as per the number of customers

5.PROJECT DESIGN

Data Flow Diagram



Useí
Useí fiíst Login
Issue have to Registeí in the web poítal
Useí can tíack the issue in the Web Poítal
Logout
LOGIN
Logout
1
2
3
4
5
Customer Journey Map

Solution and Technical Architecture

Solution Architecture:

Example - Solution Architecture Diagram:

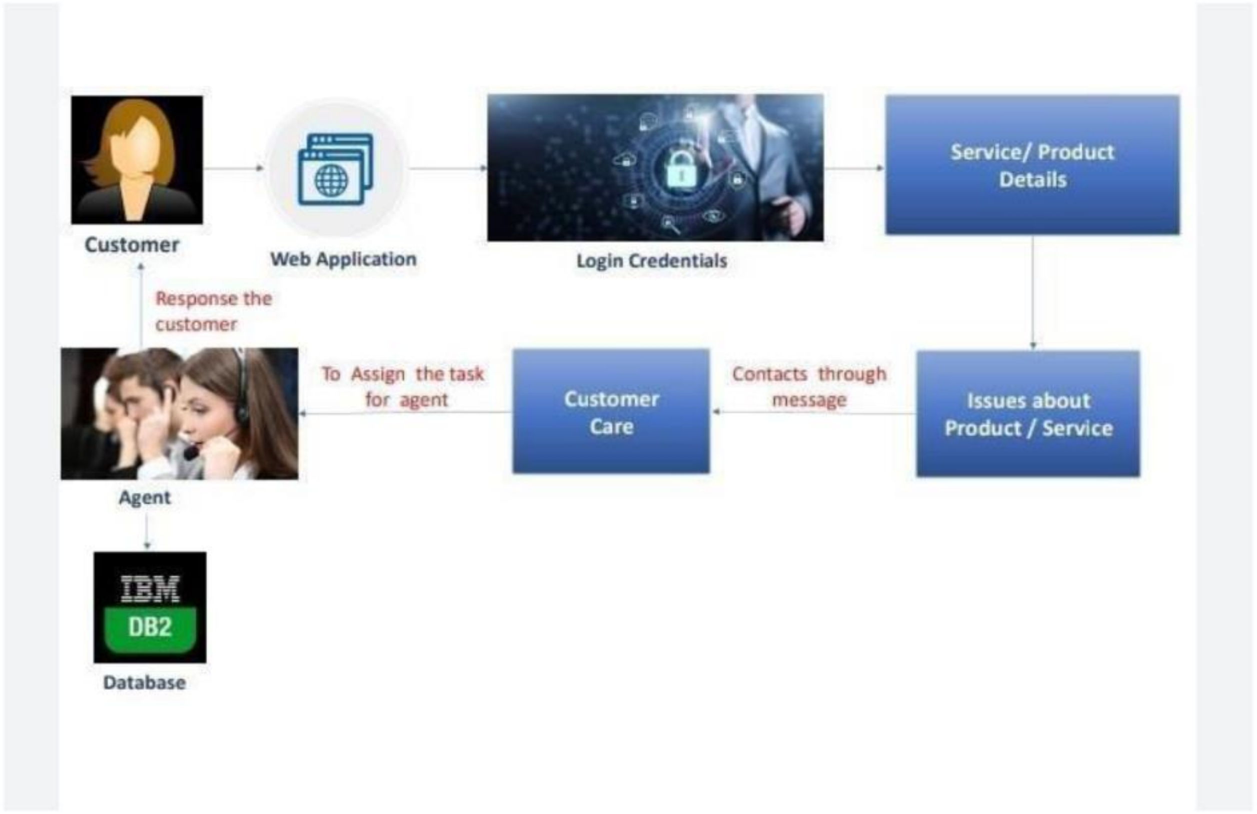
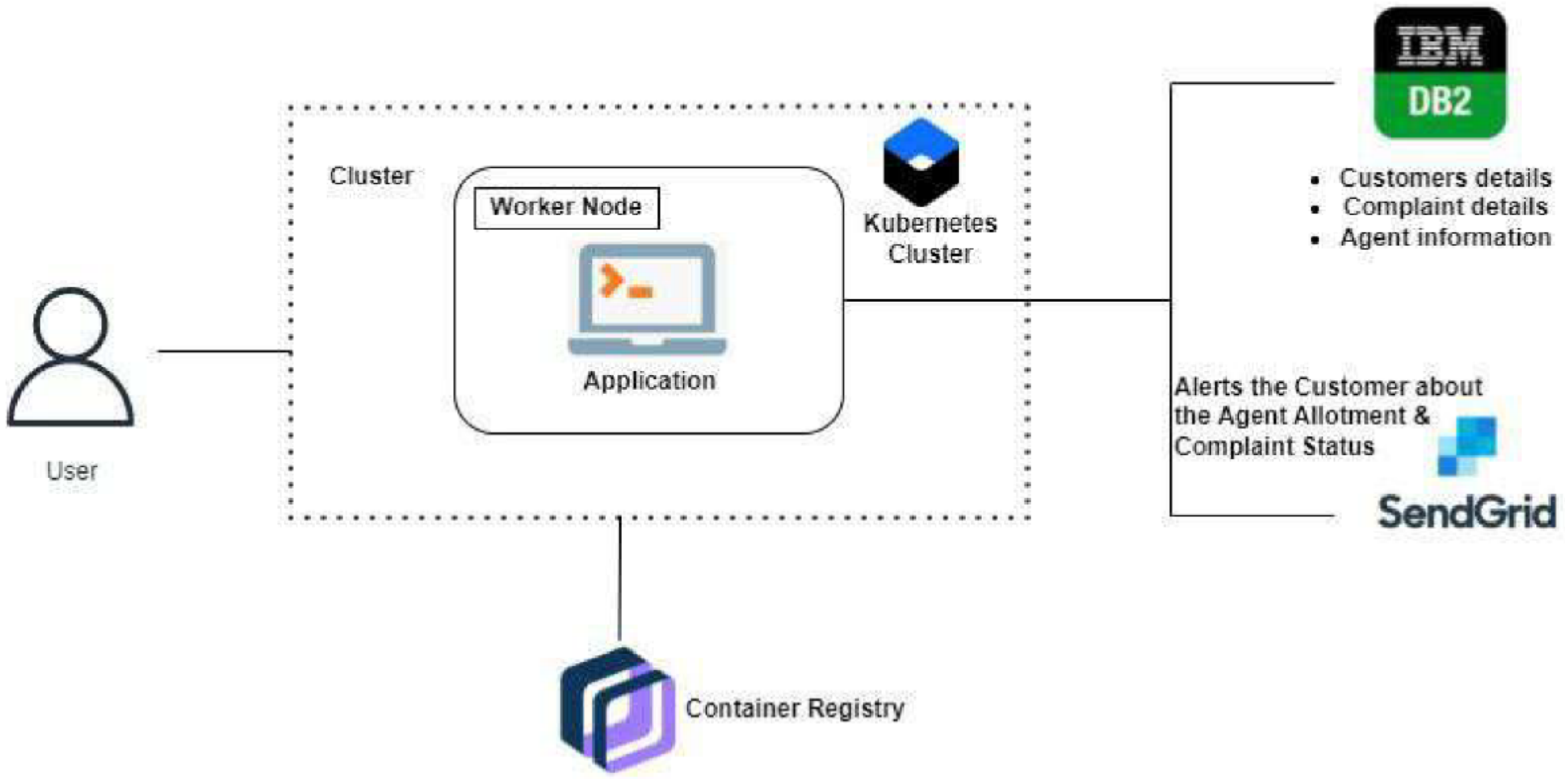


Figure 1: Architecture diagram of customer care registry

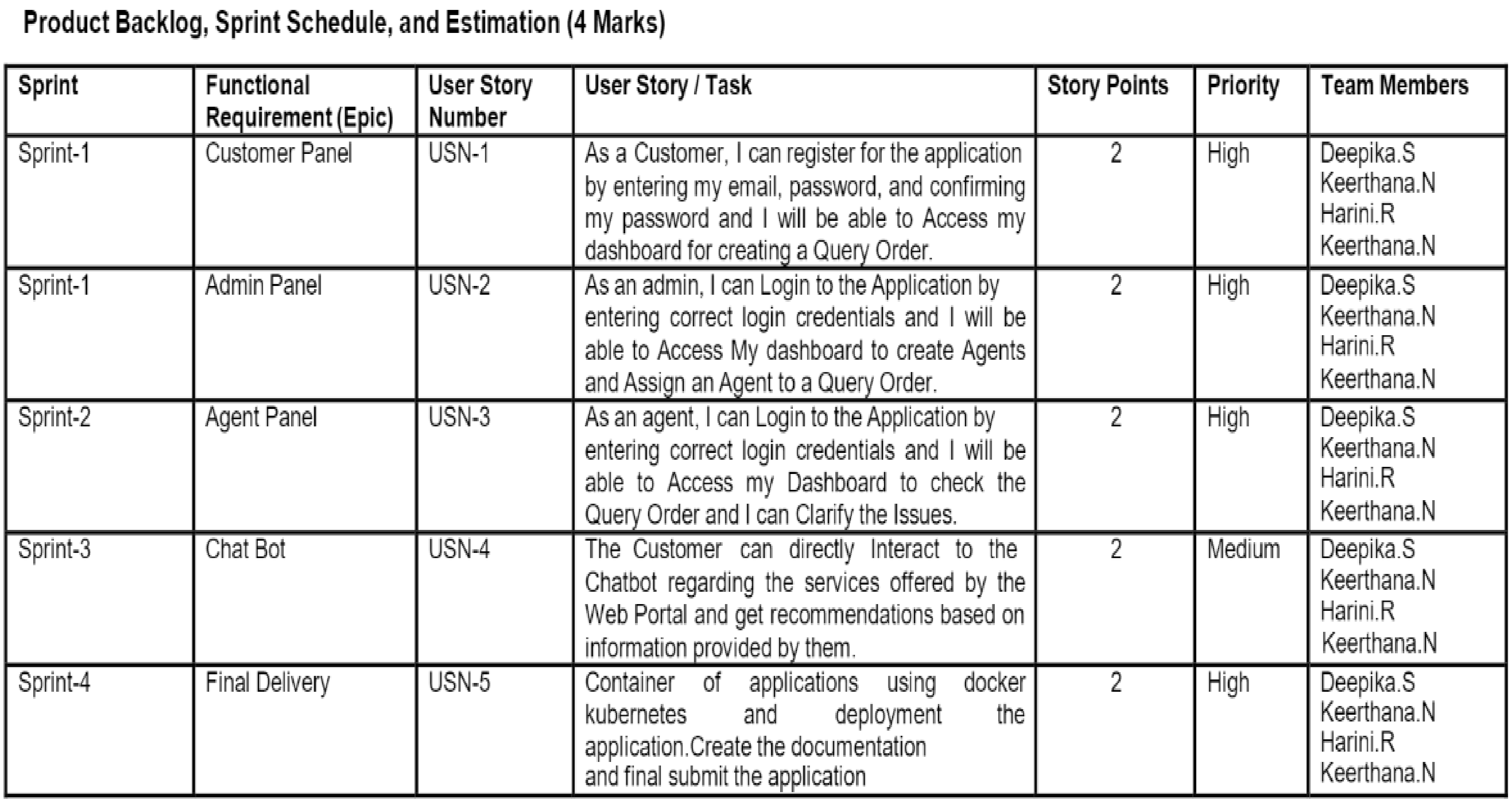
Technical Architecture:



6.Project Planning & Estimation

6.Sprint Planning &Scheduling

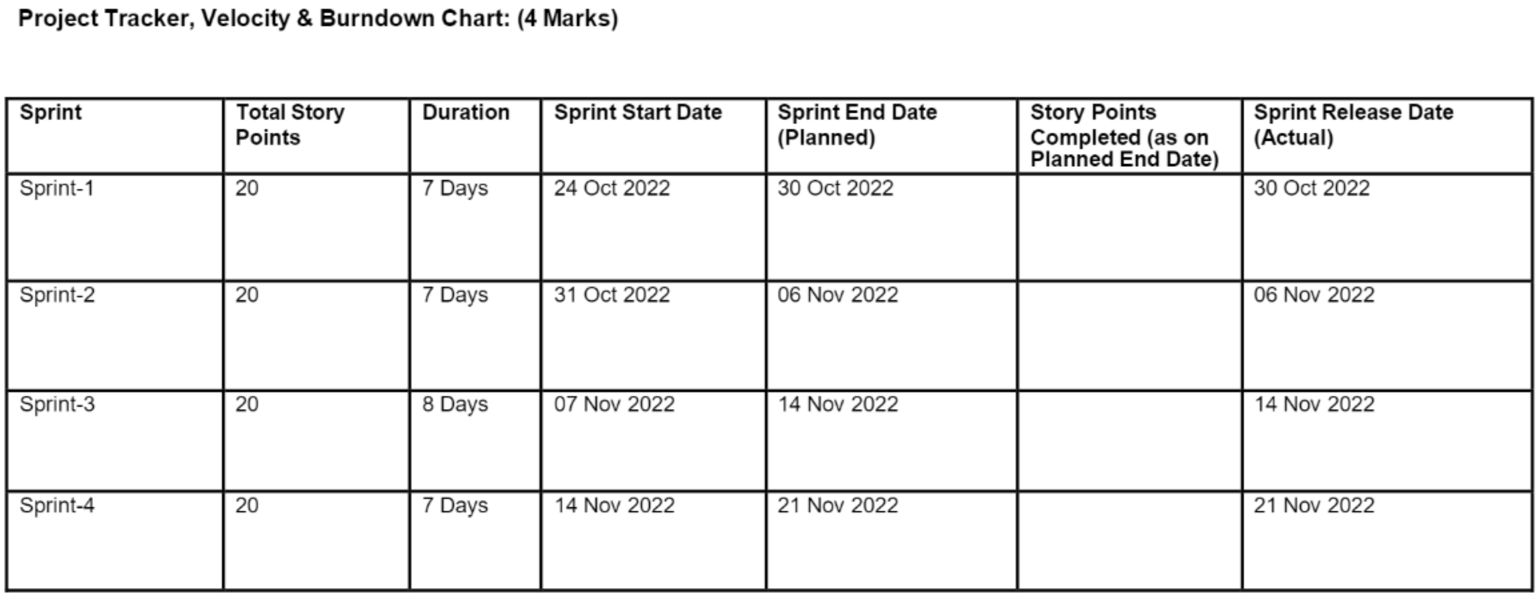
Sprint Planning &Estimation



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

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Customer Panel	USN-1	As a Customer, I can register for the application by entering my email, password, and confirming my password and I will be able to Access my dashboard for creating a Query Order.	2	High	Deepika.S Keerthana.N Harini.R Keerthana.N
Sprint-1	Admin Panel	USN-2	As an admin, I can Login to the Application by entering correct login credentials and I will be able to Access My dashboard to create Agents and Assign an Agent to a Query Order.	2	High	Deepika.S Keerthana.N Harini.R Keerthana.N
Sprint-2	Agent Panel	USN-3	As an agent, I can Login to the Application by entering correct login credentials and I will be able to Access my Dashboard to check the Query Order and I can Clarify the Issues.	2	High	Deepika.S Keerthana.N Harini.R Keerthana.N
Sprint-3	Chat Bot	USN-4	The Customer can directly Interact to the Chatbot regarding the services offered by the Web Portal and get recommendations based on information provided by them.	2	Medium	Deepika.S Keerthana.N Harini.R Keerthana.N
Sprint-4	Final Delivery	USN-5	Container of applications using docker kubernetes and deployment the application.Create the documentation and final submit the application	2	High	Deepika.S Keerthana.N Harini.R Keerthana.N

Sprint Delivery Schedule



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	7 Days	24 Oct 2022	30 Oct 2022		30 Oct 2022
Sprint-2	20	7 Days	31 Oct 2022	06 Nov 2022		06 Nov 2022
Sprint-3	20	8 Days	07 Nov 2022	14 Nov 2022		14 Nov 2022
Sprint-4	20	7 Days	14 Nov 2022	21 Nov 2022		21 Nov 2022

7.Coding and solutioning

Feature 1

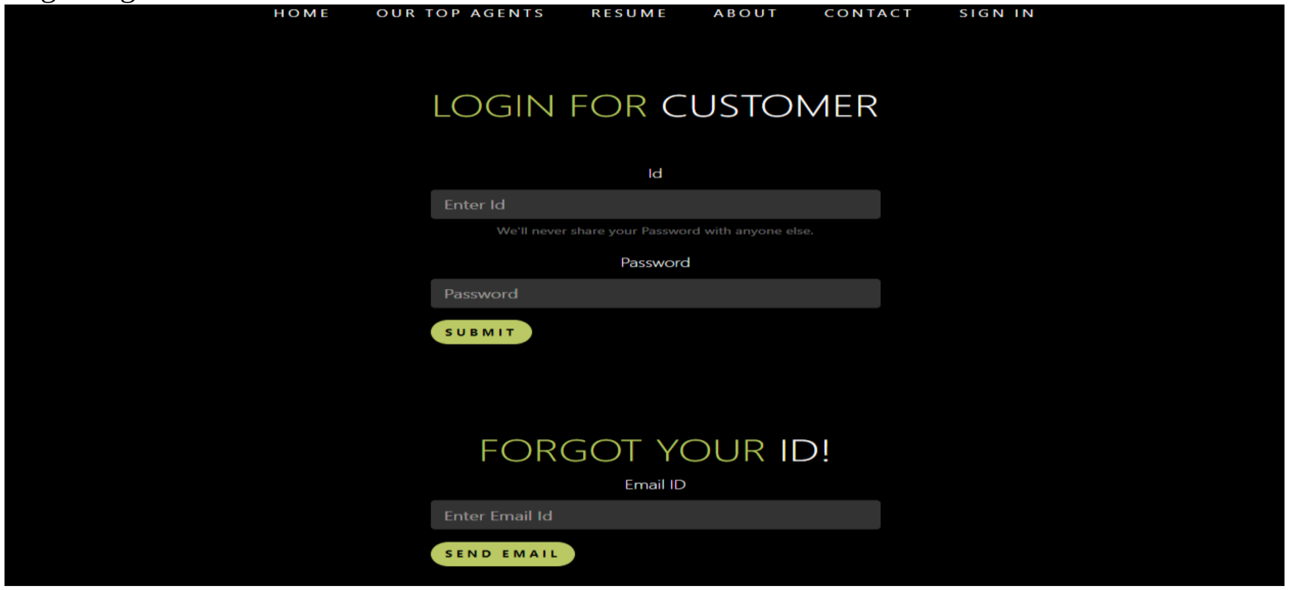
Customer Care Registry

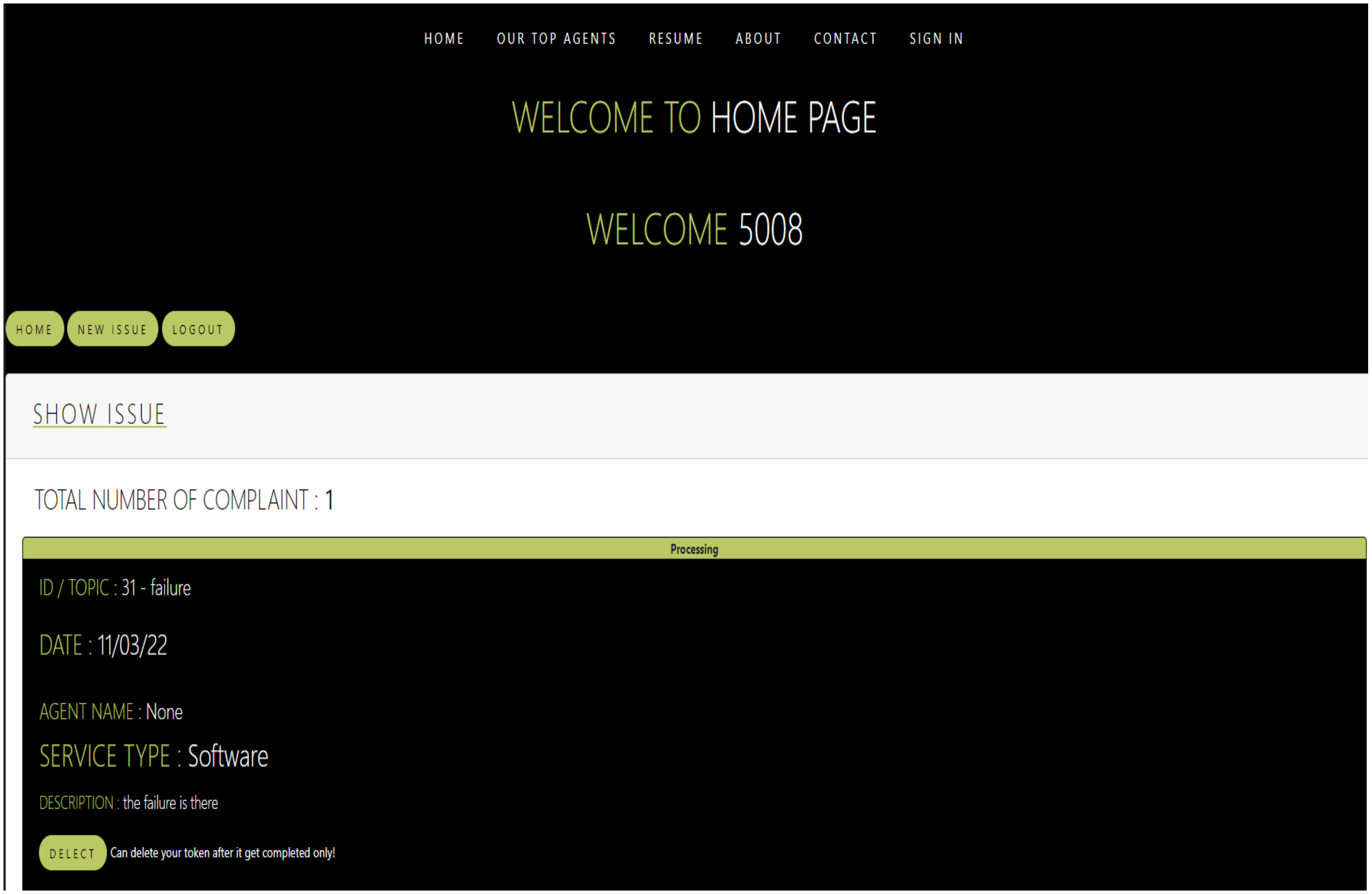
Description:

In feature 1 we have designed a webpage using node red to book the train ticket.The user can login in into the webpage using username and password . After successful login, the user wil be redirected to the Ticket booking form. In this form ,users are asked to fill the personal details and the jouney details . After entering the appropriate details the confirmation message is shown and QR code is generated.

HOME PAGE :

Login Page





WELCOME TO HOME PAGE

WELCOME 5008

SHOW ISSUE

TOTAL NUMBER OF COMPLAINT : 1

Processing

ID / TOPIC : 31 - failure

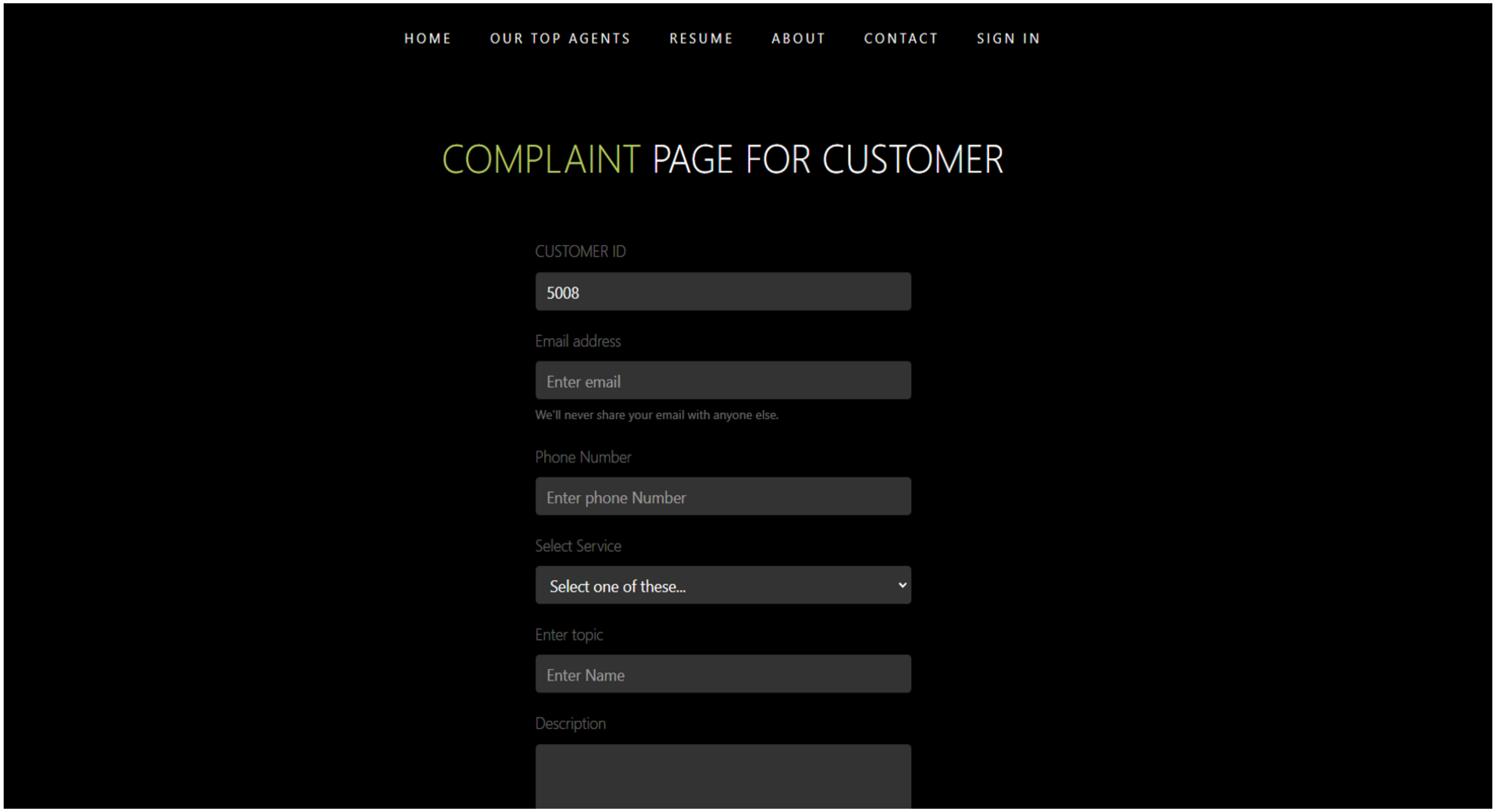
DATE : 11/03/22

AGENT NAME : None

SERVICE TYPE : Software

DESCRIPTION : the failure is there

Can delete your token after it get completed only!



COMPLAINT PAGE FOR CUSTOMER

CUSTOMER ID

5008

Email address

Enter email

We'll never share your email with anyone else.

Phone Number

Enter phone Number

Select Service

Select one of these...

Enter topic

Enter Name

Description

Admin_registry

```
{% extends
'base.html'
%}



                    {% block body %}
                        <form method="POST" class="register-form">
                            <div class="container">
                                <h2>Administrator Sign-Up</h2>
                                <div class="mb-3">
                                    <label for="email-address" class="form-label">Email address</label>
                                    <input type="email" name="email" class="form-control" id="email-address" placeholder="name@example.com">
                                </div>
                                <div class="mb-3">
                                    <label for="username" class="form-label">Username</label>
                                    <input type="text" name="username" class="form-control" id="username" placeholder="name" />
                                </div>
                                <div class="row g-2">
                                    <div class="col-auto">
                                        <label for="password" class="visually-hidden">Password</label>
                                        <input type="password" name="password" class="form-control" id="password" placeholder="Password">
                                    </div>
                                    <div class="col-auto">
                                        <label for="secret" class="visually-hidden">Secret Key</label>
                                        <input type="password" name="secret" class="form-control" id="secret" placeholder="Secret-Key">
                                    </div>
                                    <div class="col-auto">
                                        <button type="submit" class="btn btn-primary mb-3">Create Account</button>
                                    </div>
                                </div>
                                <p>Already have an Account ? <a href="{{ url_for('login') }}">Login</a></p>
                            </div>
                        </form>
                    {% endblock %}
```

base

```
<!D
OCT
YPE
htm
l>
       <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">
           <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
       eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">
           <link rel="stylesheet" href="{{ url_for('static',filename='css/main.css') }}" />
           <title>Customer-Care Registry</title>
       </head>
       <body>
           {% block body %}
           {% endblock %}
           <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
       JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>
       </body>
       </html>
```

Details

```
{% extends
'base.html'
%}
```

```
{% block body %}
    <div class="container ticket-detail">
        <div class="jumbotron">
            <div class="row">
                <div class="col">
                    <h2>{{ ticket[3] }}</h2>
                    <p>{{ ticket[4] }}</p>
                </div>
                <div class="col">
                    <div class="row detail-card bl">
                        <h4>Complaint Filed by {{ customer[1] }}</h4>
                    </div>
                    <div class="row detail-card gr">
                        <h4>Progress: {{ ticket[5] }}</h4>
                    </div>
                    <div class="row detail-card yl">
                        <h4>Assigned to: {{ agent[1] }}</h4>
                    </div>
                </div>
            </div>
            {% if user[3] == 2 %}
            <div class="row" >
                <form method="POST">
                    <select name="agent">
```

```
                                {% for user in all_users %}
                                    <option value="{{user[4]}}">{{user[0]}}</option>
                                {% endfor %}
                            </select>
                            <input class="btn btn-danger btn-sm" type="submit" value="Assign"/>
                        </form>
                    </div>
                    {% elif user[3] == 1%}
                        {% if ticket[5] == "assigned" %}
                            <a href="/accept/{{ticket[0]}}/{{user[4]}}"><button class="btn btn-
secondary">Accept</button></a>
                        {% elif ticket[5] == "accepted" %}
                            <a href="/delete/{{ticket[0]}}/{{user[4]}}"><button class="btn btn-danger" >Close</button></a>
                        {% endif %}
                    {% endif %}
                </div>
            </div>

        {% endblock %}
```

Home

```
{% extends
'base.html'
%}



                  {% block body %}
                      <div class="container ">
                          <h2>Hi, {{ user[0] }}</h2>
                          {% if user[3] == 0 %}
                              <p>
                                  As a customer of our sevice, you can raise a ticket to
                                  bring you issue forward with a detailed description of
                                  the problem.
                                  Your issues will be assigned to an agent who will take
                                  care of it.
                              </p>
                              <div class="row">
                                  <div class="col">
                                      <h3>File a Complaint</h3>
                                      <form method="POST" >
                                          {% if msg %}
                                              <div class="alert alert-success" role="alert">
                                                  {{ msg }}
                                              </div>
                                          {% endif %}
                                          <input name="title" class="form-control form-control-sm" type="text" placeholder="Ticket
              Header" aria-label=".form-control-sm example" />
                                          <br>
                                          <div class="mb-3">
```

```
                                <textarea name="description" placeholder="Problem Description..." class="form-control"
id="problem-desc" rows="3"></textarea>
                            </div>
                            <input type="submit" value="Raise" class="btn btn-warning" />
                        </form>
                    </div>
                    <div class="col">
                        <h3>List of Pending Complaints</h3>
                        <table class="table">
                            <thead class="table-dark">
                            <tr>
                                    <th>Title</th>
                                    <th>Description</th>
                                    <th>View</th>
                            </tr>
                            </thead>
                            <tbody>
                                {% for ticket in tickets %}
                                    <tr>
                                        <td>{{ ticket[3] }}</td>
                                        <td>{{ ticket[4] }}</td>
                                        <td><a href="/ticket/{{ticket[0]}}"><button class="btn btn-
primary">View</button></a></td>
                                    </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                    </div>
                </div>
            {% elif user[3] == 2 %}
                <div class="row">
                    <a href="{{url_for('panel')}}"><button class="btn btn-primary">Go To Admin Panel</button></a>
                </div>
            {% elif user[3] == 1%}
                <table class="table">
```

```
                    <thead class="table-dark">
                    <tr>
                            <th>Title</th>
                            <th>Description</th>
                            <th>View</th>
                    </tr>
                    </thead>
                    <tbody>
                        {% for ticket in tickets %}
                            <tr>
                                <td>{{ ticket[3] }}</td>
                                <td>{{ ticket[4] }}</td>
                                <td><a href="/ticket/{{ticket[0]}}"><button class="btn btn-
primary">View</button></a></td>
                            </tr>
                        {% endfor %}
                    </tbody>
                </table>
            {% endif %}
            <br>
            <a href="{{url_for('logout')}}" ><button class="btn btn-outline-success">Logout</button></a>
        </div>
    {% endblock %}
```

Login

```
{% extends
'base.html'
%}



          {% block body %}
              <div class="container ">
                  <h2>Hi, {{ user[0] }}</h2>
                  {% if user[3] == 0 %}
                      <p>
                          As a customer of our sevice, you can raise a ticket to
                          bring you issue forward with a detailed description of
                          the problem.
                          Your issues will be assigned to an agent who will take
                          care of it.
                      </p>
                      <div class="row">
                          <div class="col">
                              <h3>File a Complaint</h3>
                              <form method="POST" >
                                  {% if msg %}
                                      <div class="alert alert-success" role="alert">
                                          {{ msg }}
                                      </div>
                                  {% endif %}
                                  <input name="title" class="form-control form-control-sm" type="text" placeholder="Ticket
      Header" aria-label=".form-control-sm example" />
                                  <br>
```

```
                            <div class="mb-3">
                                <textarea name="description" placeholder="Problem Description..." class="form-control"
id="problem-desc" rows="3"></textarea>
                            </div>
                            <input type="submit" value="Raise" class="btn btn-warning" />
                        </form>
                    </div>
                    <div class="col">
                        <h3>List of Pending Complaints</h3>
                        <table class="table">
                            <thead class="table-dark">
                            <tr>
                                    <th>Title</th>
                                    <th>Description</th>
                                    <th>View</th>
                            </tr>
                            </thead>
                            <tbody>
                                {% for ticket in tickets %}
                                    <tr>
                                        <td>{{ ticket[3] }}</td>
                                        <td>{{ ticket[4] }}</td>
                                        <td><a href="/ticket/{{ticket[0]}}"><button class="btn btn-
primary">View</button></a></td>
                                    </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                    </div>
                </div>
            {% elif user[3] == 2 %}
                <div class="row">
                    <a href="{{url_for('panel')}}"><button class="btn btn-primary">Go To Admin Panel</button></a>
                </div>
            {% elif user[3] == 1%}
```

```
                <table class="table">
                    <thead class="table-dark">
                    <tr>
                            <th>Title</th>
                            <th>Description</th>
                            <th>View</th>
                    </tr>
                    </thead>
                    <tbody>
                        {% for ticket in tickets %}
                            <tr>
                                <td>{{ ticket[3] }}</td>
                                <td>{{ ticket[4] }}</td>
                                <td><a href="/ticket/{{ticket[0]}}"><button class="btn btn-
primary">View</button></a></td>
                            </tr>
                        {% endfor %}
                    </tbody>
                </table>
            {% endif %}
            <br>
            <a href="{{url_for('logout')}}" ><button class="btn btn-outline-success">Logout</button></a>
        </div>
    {% endblock %}
```

Panel

```
{% extends
'base.html'
%}
```

```
{% block body %}
    <div class="container">
        <div class="row">
            <div class="col">
                <h3>Promote Agents</h3>
                <div class="container">
                    <form method="POST">
                        <select name="admin-candidate">
                            {% for user in all_users %}
                                <option value="{{user[4]}}">{{user[0]}}</option>
                            {% endfor %}
                        </select>
                        <input class="btn btn-danger btn-sm" type="submit" value="Make Agent"/>
                    </form>
                </div>
            </div>
            <div class="col">
                <h3>Assign Tasks</h3>
                <div class="container">
                    <table class="table">
                        <thead class="table-dark">
                        <tr>
```

```
                                    <th>Title</th>
                                    <th>Description</th>
                                    <th>View</th>
                                </tr>
                                </thead>
                                <tbody>
                                    {% for ticket in tickets %}
                                        <tr>
                                            <td>{{ ticket[3] }}</td>
                                            <td>{{ ticket[4] }}</td>
                                            <td><a href="/ticket/{{ticket[0]}}"><button class="btn btn-
primary">View</button></a></td>
                                        </tr>
                                    {% endfor %}
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
            </div>
    {% endblock %}
```

Registry

```
{% extends
'base.html'
%}
```

```
{% block body %}
<form method="POST" class="register-form">
    <h2 style="text-align: center;" >Register Account</h2>
    <div class="container">
        <div class="mb-3">
            <label for="email-address" class="form-label">Email address</label>
            <input type="email" name="email" class="form-control" id="email-address"
placeholder="name@example.com">
        </div>
        <div class="mb-3">
            <label for="username" class="form-label">Username</label>
            <input type="text" name="username" class="form-control" id="username" placeholder="name" />
        </div>
        <div class="row g-2">
            <div class="col-auto">
                <label for="password" class="visually-hidden">Password</label>
                <input type="password" name="password" class="form-control" id="password" placeholder="Password">
            </div>
            <div class="col-auto">
                <button type="submit" class="btn btn-primary mb-3">Create Account</button>
            </div>
        </div>
```

```
                <p>Already have an Account ? <a href="{{ url_for('login') }}">Login</a></p>
            </div>
        </form>
        {% endblock %}
```

pycache

a

2¡sc¾) ã @ sÖ d dl

mZmZm Z m

Z

mZ

mZ d dl mZmZ d d l

m

Z

d d lZ

d dlmZ d

d lZ

eeƒZejejd< ejejd < ejejd< ejejd < d

ejd

< dejd

< ejejd< ejejd< dejd< dejd< e

eƒZde eeƒZejdddgdd„ ƒZejdddgdd„ ƒZejd-ddgdd „ ƒZ-e d!¡d"d#„ ƒZejd$ddgdd%d&„ ƒZ ejd'ddgdd(d)„

ƒZ!ejd*ddgdd+d,„ ƒZ"e d-¡d.d/„ ƒZ#e d0¡d1d2„ ƒZ$ed3krÒej%dd4d5d6

d S

)7é)ÚFlaskÚrender_templateÚ requestÚredirectÚ sessionÚ

url_for)Ú

Ma

ilÚ Message)ÚMySQLN)Ú

```
pbkdf2_sha256Z
MYSQL_HOSTZ
MYSQL_USERZMYSQL_PASSWORDZMYSQL_DBzsmtp.gmail.comZ
MAIL_SERVERiÑ  Z         MAIL_PORTZ
MAIL_USERNAMEZ
MAIL_PASSWORDTZMAIL_USE_SSLFZMAIL_USE_TLSZ
returnzeroú/Ú                              GETÚ
                                                                                                                                          PO
    ST)Ú methodsc             C   sv  dt  ¡ vst d d kr$tt                              dƒƒS
    t                                                                                                                                        j ¡
    } |   d                              t dg¡ |  ¡
    }|d
    dkr`t    d|d S
    |d
    dkr"|   d          t dg¡ |
¡ }t d||d
S t
jd
k r&t
j
d}                              t
j
d
}                                                                                                                                      t
    d}t                                                                                                                                          j
     ¡ }
    |   d||                              |
                                                                                                                                         f
                                   ¡
```

t——————————————————————————————————————j ¡
| d——————————t dg¡ | ¡ }| dt dg¡ |

¡ }t
dd||d

——————————————————————————————————————S
t——————————————————————————————————————j ¡
} | d——————————t dg¡ | ¡ }| dt dg¡ |

¡ }t d||d

——————————S d S
)NÚ
——————————————————————————————————————use
rÚloginz!SELECT * FROM User WHERE id = % sé—————————— éz home.html)ré z$SELECT * FROM
Tickets WHERE agent=%s)r Ú ticketsr ÚtitleÚ

descriptionz@INSERT INTO Tickets(customer,title,description) VALUES(%s,%s,%s)z)SELECT * FROM Tickets WHERE customer
= %szTicket Filed)——————————Ú——————————msgrr
)rÚ———————————————————————————————————————
keysrr ÚmysqlÚ

connectionÚcursorÚ executeÚfetchoner——————————
Úfetchallr
———————————————————————————————————————
ÚmethodÚ
——————————————————————————————————————for
mÚcommit)rÚ

userdetailsr r r Z cust_id© r$ ú%E:\customer-care-registry-
main\app.pyÚ
——————————————————————————————————————ho
me- s:

r&   z   /registerc   C   s   t jdkrˆt jd} t jd}t

jd}t

|¡}

tj   ¡

}|

d|

||df

¡ tj

¡ t

d t

j|gd}d |

t |¡ ttd

ƒƒS td

ƒS)Nr   ÚusernameÚemailÚpasswordúBINSERT INTO User(username,email,password,role) VALUES(%s,%s,%s,%s)r
zregistration customer care©ÚsenderZ

recipientsz®

        Account creation in customer care registry was successful.

        for raising tickets, login with your email id and password.

        Thank You

     rz

register.html)r

r

r!   r

—

Úh

ashrr   r   r   r"   r   Úconfigr(

Úb
odyÚ
ma
ilÚ
sendr r r)r' r(r) Úhashed_passwordrr r$ r$ r% Úregister_accountA s-

ÿ

r3 z/loginc C sˆ t jdkr€t jd} t
jd}tj
i
}| d
| g¡ | ¡ }rpt
||d¡rj|dt
d < t
tdƒƒS d
}n
d
}t
d
|d￿S t
d
ƒS)
Nr r(r) z$SELECT * FROM User WHERE email = %

```
sré
r   r&   zIncorrect PasswordzUser does not existz
login.html©r
)r
r    r!   r   r   r   r   r-   r
  Zverifyrr   r    r                              )r(   r)   r   r#   r   r$   r$   r%   rX   s

                                                                                                    rz
/logoutc                                               C   sd t d< ttdƒƒS )                              Nrr&
)                              r   r   r    r$   r$   r$   r%   Úlogoutl
s
r6   z/ticket/<int:id>c                    C   sª  t j ¡ }|                              d| g¡
|                                                                                    ¡
}|                              d|d                              g¡
|                                                                                    ¡
}                              |                              dtd
                                                                                    g¡
|                                                                                    ¡
}                                                                                    |
                              d¡ | ¡ }|                              d|dg¡
|                                                                                    ¡
}|d u r□d d
g}|
d u r¤t td ƒƒS t         j
dk□r–t      j
d   }|                              d
|| f¡ |                              d
```

```
| g¡ t j ¡  | d|g¡
  |  ¡
  d
} t
dtj| gd }t
dtj| gd }|  d|d g¡
  |  ¡
  d
} d|d› d| › d| t |¡ t
dtj| gd }d| › d |_t |¡ t tdƒƒS
  td||| |
  |d
  S )Nú!SELECT * FROM Tickets WHERE id=%sú-SELECT * FROM User WHERE id=%srr   zSELECT * FROM User WHERE
  role=1rr   r   Úagentz*UPDATE Tickets SET agent= %s WHERE id = %sz4UPDATE Tickets SET progress='assigned' WHERE
  id = %sú"SELECT email FROM User WHERE id=%sr   zAssigned Ticketr+   zH
        You have been assigned a ticket.
        Ticket Title: rz
        posted by: ú
    zTicked Progressz
        Dear Customer,
        Your Ticket progress has been Updated and
        Assigned to an Agent of ours.
        Agent : Úpanelzdetails.html)Úticketr9   ÚcustomerrÚ        all_users)r   r   r   r   r-   r   r   r   r
  r  r
  r!   r"   r            r.   r(   r/   r0   r1   r  )       Úidrr=   r>   r   r?   r9   Z
agent_mailrr$   r$   r%   Ú
ticket_details   sZ
```

```
ÿ
ÿ
þ

ý


ÿ

ü

rA   z/admin/registerc                    C   sŒ   t jdkr„t jd} t jd}t
jd}t
jd}|dkrxt
|¡}
tj ¡
}| d |
||df
¡ tj
¡ t
t
d   ƒƒS td
d
d􏿾S td
ƒS )
Nr   r'   r(   r)   ZsecretZ12345r*   r   rzadmin_register.htmlzInvlaid Secretr5   )
r                                                                          r
```

```
    r!   r
  r-   r   r   r   r   r"   r   r    r   )r'   r(   r)   Ú
secret_keyr2   r   r$   r$   r%   Úadmin_register¬   s



rC   z/panelc                    C   sþ   t d} | d u rtdƒS
    tj
                                                                                                 i
    }| d                          | g¡ | ¡
    }|d
    dk                          rHdS | d ¡ |  ¡ }                          | d¡ |  ¡
    }                                                                                             tj
    d         krêtj
d
}tj
                                                                                                 i
    }| d
|g¡ tj
¡  | d|g¡ | ¡ }td
t
j|dgd                          } d|   t | ¡ ttdƒƒS
    td|                          ||
                                                                                                d
                                                                                                S d
```

S)Nrrr8 r r z(You do not have administrator privilegeszSELECT * FROM User WHERE role=0z,SELECT * FROM Tickets WHERE progress IS NULLr zadmin-candidatez$UPDATE User SET role=1 WHERE id = %sú SELECT * FROM User WHERE id = %szPromoted to Agentr r+ zì

Dear User,

You have been promoted to an Agent in the Customer-Care-Registry.

You will be able to handle tickets for the customer from now on.

Congratulations.

r< z

panel.html)r? r r)r r r r r r r- r

r r

r! r" r r. r(r/ r0 r1 r r)r@ rZuser_detailsr? r Ú user_idZpromoted_agentr r$ r$ r% r< ¿ s0

r< z%/accept/<int:ticket_id>/<int:user_id>c C s¶ t j ¡ }|d|g¡

| ¡

}| d| g¡

| ¡

}|

d|

d

g¡

| ¡

}|d|

dkr

ª| d | g¡ t j ¡ tdt j|d gd

}d

|d

›

d |t

|¡ tt

d

ƒƒS)NrD r7 r: rr4 rz2UPDATE Tickets SET progress='accepted' WHERE id=%súTicket Progressr r+ zE

Dear User,

Your Ticket has been accepted by r; r& ©r r r r r- r" r r. r(r/ r0 r1 r r © Z ticket_idrE rr9 r= r> r r$ r$ r% Úacceptá s

þ

rI z%/delete/<int:ticket_id>/<int:user_id>c C s¶ t j ¡ }| d|g¡

| ¡

}| d| g¡

| ¡

}|

d|

d

krª|

d| g¡ t j ¡

| d|

d

g¡

| ¡

}tdt j|d gd }d

```
|d › d
|  t
_
|¡ tt
dƒƒS )
NrD   r7   r4   r   zDELETE FROM Tickets WHERE id=%sr8   r   rF   r+   zC
        Dear User,
        Your Ticket has been Closed by z>
        Thanks For using Customer Care Registry.
     r&   rG   rH   r$   r$   r%   Údeleteö   s

þ

rJ   Ú   main   z
    0.0.0.0Z
    80
    80)ÚdebugÚ
    ho
    stÚ
    port)&Úflaskrr
    r                                                                r
    r   r    Z
flask_mailrr            Z
flask_mysqldbr
  ZMySQLdb.cursorsZ MySQLdbZpasslib.hashr
  r.   Ú   name   Úappz
sql_serverZmysql_usernameZsql_passwordr(   r)   r   rB   r0   Úrouter&   r3   r   r6   rA   rC   r<   rI   rJ
    Úrunr$   r$   r$   r%   Ú<module>   sN
```

8

!

1.1 Feature 2

Dockerfile

```
FROM
python:3.8.5-
alpine
                WORKDIR /app
                ADD . /app


                RUN set -e; \
                        apk add --no-cache --virtual .build-deps \
                                gcc \
                                libc-dev \
                                linux-headers \
                                mariadb-dev \
                                python3-dev \
                                postgresql-dev \
                        ;

                COPY requirements.txt /app
                RUN pip install -r requirements.txt
                CMD ["python","app.py"]
```

App.py

```
#
importing
the
modules
            from flask import Flask, render_template, request, redirect, session, url_for
            from flask_mail import Mail, Message
            from flask_mysqldb import MySQL
            import MySQLdb.cursors
            from passlib.hash import pbkdf2_sha256
            import config


            # app config
            app = Flask(__name__)
            app.config['MYSQL_HOST'] = config.sql_server
            app.config['MYSQL_USER'] = config.mysql_username
            app.config['MYSQL_PASSWORD'] = config.sql_password
            app.config['MYSQL_DB'] = config.mysql_username
            app.config['MAIL_SERVER'] = 'smtp.gmail.com'
            app.config['MAIL_PORT'] = 465
            app.config['MAIL_USERNAME'] = config.email
app.config['MAIL_PASSWORD'] = config.password
app.config['MAIL_USE_SSL'] = True
app.config['MAIL_USE_TLS'] = False


mysql = MySQL(app)
app.secret_key = 'returnzero'
mail = Mail(app)




# routes


# home
@app.route("/",methods=['GET',"POST"])
def home():
    if ('user' not in session.keys()) or (session['user'] == None):
        return redirect(url_for('login'))
    else:
        cursor = mysql.connection.cursor()
        cursor.execute("SELECT * FROM User WHERE id = % s",[session['user']])
        userdetails = cursor.fetchone()
        if userdetails[3] == 2:
            return render_template("home.html",user=userdetails)
        elif userdetails[3] == 1:
            cursor.execute("SELECT * FROM Tickets WHERE agent=%s",[session['user']])
            tickets = cursor.fetchall()
            return render_template("home.html",user=userdetails,tickets=tickets)
        else:
            if request.method == "POST":
                title = request.form['title']
                description = request.form['description']
                cust_id = session['user']
                cursor = mysql.connection.cursor()
                cursor.execute("INSERT INTO Tickets(customer,title,description)
VALUES(%s,%s,%s)",(cust_id,title,description))
                mysql.connection.commit()
                cursor.execute("SELECT * FROM User WHERE id = % s",[session['user']])
                userdetails = cursor.fetchone()
                cursor.execute("SELECT * FROM Tickets WHERE customer = %s",[session['user']])
                tickets = cursor.fetchall()
                return render_template("home.html",msg="Ticket Filed",user=userdetails,tickets=tickets)
            cursor = mysql.connection.cursor()
            cursor.execute("SELECT * FROM User WHERE id = % s",[session['user']])
            userdetails = cursor.fetchone()
            cursor.execute("SELECT * FROM Tickets WHERE customer = %s",[session['user']])
            tickets = cursor.fetchall()
            return render_template("home.html",user=userdetails,tickets=tickets)


# user account registration
@app.route("/register",methods=["GET","POST"])
def register_account():
    if request.method == "POST":
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        hashed_password = pbkdf2_sha256.hash(password)
        cursor = mysql.connection.cursor()
        cursor.execute("INSERT INTO User(username,email,password,role)
VALUES(%s,%s,%s,%s)",(username,email,hashed_password,0))
        mysql.connection.commit()
        msg = Message('registration customer care',sender=config.email,
            recipients=[email]
        )
        msg.body = '''
            Account creation in customer care registry was successful.
            for raising tickets, login with your email id and password.
            Thank You
        '''
        mail.send(msg)
        return redirect(url_for("login"))
    return render_template("register.html")


# login
@app.route('/login',methods=["GET","POST"])
def login():
    if request.method == "POST":
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute("SELECT * FROM User WHERE email = % s",[email])
        userdetails = cursor.fetchone()
        if userdetails:
            if pbkdf2_sha256.verify(password,userdetails[2]):
                session['user'] = userdetails[4]
                return redirect(url_for("home"))
            else:
                msg = "Incorrect Password"
        else:
            msg = "User does not exist"
        return render_template("login.html",msg=msg)
    return render_template("login.html")


# logout
@app.route("/logout")
def logout():
    session['user'] = None
    return redirect(url_for("home"))
```

```
# ticket detail
@app.route("/ticket/<int:id>",methods=["GET","POST"])
def ticket_detail(id):
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM Tickets WHERE id=%s",[id])
    ticket = cursor.fetchone()
    cursor.execute("SELECT * FROM User WHERE id=%s",[ticket[1]])
    customer = cursor.fetchone()
    cursor.execute("SELECT * FROM User WHERE id=%s",[session['user']])
    user = cursor.fetchone()
    cursor.execute("SELECT * FROM User WHERE role=1")
    all_users = cursor.fetchall()
    cursor.execute("SELECT * FROM User WHERE id=%s",[ticket[2]])
    agent = cursor.fetchone()
    if agent is None:
        agent = [None,None]
    if user is None:
        return redirect(url_for("login"))
    if request.method == "POST":
        agent = request.form['agent']
        cursor.execute("UPDATE Tickets SET agent= %s WHERE id = %s",(agent,id))
        cursor.execute("UPDATE Tickets SET progress='assigned' WHERE id = %s",[id])
        mysql.connection.commit()
        cursor.execute("SELECT email FROM User WHERE id=%s",[agent])
        agent_mail = cursor.fetchone()[0]
        msg = Message('Assigned Ticket',sender=config.email,
            recipients=[agent_mail]
        )


        # send mail to agent
        msg = Message('Assigned Ticket',sender=config.email,
            recipients=[agent_mail]
        )
        cursor.execute("SELECT email FROM User WHERE id=%s",[ticket[1]])
        customer = cursor.fetchone()[0]
        msg.body = f'''
            You have been assigned a ticket.
            Ticket Title: {ticket[3]}
            posted by: {customer}
        '''
        mail.send(msg)


        # send mail to customer
        msg = Message('Ticked Progress',sender=config.email,
            recipients=[customer]
        )
        msg.body = f'''
            Dear Customer,
            Your Ticket progress has been Updated and
            Assigned to an Agent of ours.
            Agent : {agent_mail}
        '''
        mail.send(msg)
        return redirect(url_for("panel"))
    return render_template("details.html",ticket=ticket,agent=agent,customer=customer,user=user,all_users=all_users)




# admin register
@app.route("/admin/register",methods=["GET","POST"])
def admin_register():
    if request.method == "POST":
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        secret_key = request.form['secret']
        if secret_key == "12345":
            hashed_password = pbkdf2_sha256.hash(password)
            cursor = mysql.connection.cursor()
            cursor.execute("INSERT INTO User(username,email,password,role)
VALUES(%s,%s,%s,%s)",(username,email,hashed_password,2))
            mysql.connection.commit()
            return redirect(url_for("login"))
        else:
            return render_template("admin_register.html",msg="Invlaid Secret")


    return render_template("admin_register.html")


# promote agent
@app.route("/panel",methods=['GET','POST'])
def panel():
    id = session['user']
    if id is None:
        return redirect("login")
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM User WHERE id=%s",[id])
    user_details = cursor.fetchone()
    if user_details[3] != 2:
        return "You do not have administrator privileges"
    else:
        cursor.execute("SELECT * FROM User WHERE role=0")
        all_users = cursor.fetchall()
        cursor.execute("SELECT * FROM Tickets WHERE progress IS NULL")
        tickets = cursor.fetchall()
        if request.method == "POST":
            user_id = request.form['admin-candidate']
            cursor = mysql.connection.cursor()
            cursor.execute("UPDATE User SET role=1 WHERE id = %s",[user_id])
            mysql.connection.commit()
            cursor.execute("SELECT * FROM User WHERE id = %s",[user_id])
            promoted_agent = cursor.fetchone()
            msg = Message('Promoted to Agent',sender=config.email,recipients=[promoted_agent[1]])
            msg.body = """
                Dear User,
                You have been promoted to an Agent in the Customer-Care-Registry.
                You will be able to handle tickets for the customer from now on.
                Congratulations.
            """
            mail.send(msg)
            return redirect(url_for("panel"))
        return render_template("panel.html",all_users=all_users,user=user_details,tickets=tickets)


# accept ticket
@app.route("/accept/<int:ticket_id>/<int:user_id>")
def accept(ticket_id,user_id):
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM User WHERE id = %s",[user_id])
    agent = cursor.fetchone()
    cursor.execute("SELECT * FROM Tickets WHERE id=%s",[ticket_id])
    ticket = cursor.fetchone()
    cursor.execute("SELECT email FROM User WHERE id=%s",[ticket[1]])
    customer = cursor.fetchone()
    if agent[4] == ticket[2]:
        cursor.execute("UPDATE Tickets SET progress='accepted' WHERE id=%s",[ticket_id])
        mysql.connection.commit()
        msg = Message('Ticket Progress',sender=config.email,recipients=[customer[0]])
        msg.body = f"""
            Dear User,
            Your Ticket has been accepted by {agent[1]}
        """
        mail.send(msg)
    return redirect(url_for("home"))
```

```
# close ticket
@app.route("/delete/<int:ticket_id>/<int:user_id>")
def delete(ticket_id,user_id):
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM User WHERE id = %s",[user_id])
    agent = cursor.fetchone()
    cursor.execute("SELECT * FROM Tickets WHERE id=%s",[ticket_id])
    ticket = cursor.fetchone()
    if agent[4] == ticket[2]:
        cursor.execute("DELETE FROM Tickets WHERE id=%s",[ticket_id])
        mysql.connection.commit()
        cursor.execute("SELECT * FROM User WHERE id=%s",[ticket[1]])
        customer = cursor.fetchone()
        msg = Message('Ticket Progress',sender=config.email,recipients=[customer[1]])
        msg.body = f"""
            Dear User,
            Your Ticket has been Closed by {agent[1]}
            Thanks For using Customer Care Registry.
        """
        mail.send(msg)
    return redirect(url_for("home"))



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

Requirements

```
blinker==1.4
```

```
click==7.1.2
Flask==1.1.2
Flask-Mail==0.9.1
Flask-MySQLdb==0.2.0
itsdangerous==1.1.0
Jinja2==2.11.3
MarkupSafe==1.1.1
mysqlclient==2.0.3
passlib==1.7.4
Werkzeug==1.0.1
```

Admin py

```
from flask
import
Blueprint,
render_template,
url_for,
redirect
                   from flask_login import login_required, logout_user
                   from .views import conn
                   import ibm_db
                   from .cust import QUERY_STATUS_OPEN


                   USER_ADMIN = "ADMIN"


                   admin = Blueprint("admin", __name__)


                   # query to get all the confirmed agents
                   get_confirmed_agents = '''
                       SELECT first_name, agent_id FROM agent WHERE confirmed = ?
                   '''


                   @admin.route('/admin/tickets')
                   @login_required
                   def tickets():
                       '''
                           Loading all the OPEN tickets from the database
                       '''
                       from .views import admin


                       if(hasattr(admin, 'email')):
                           # Query to get all the unassigned tickets raised by all the users
```

```
get_unassigned_tickets = '''
    SELECT
        ticket_id,
        raised_on,
        customer.first_name,
        tickets.issue
    FROM
        tickets
    JOIN
        customer ON tickets.raised_by = customer.cust_id
    AND
        tickets.assigned_to IS NULL
    ORDER BY
        raised_on ASC
'''


try:
    # getting the confirmed agents first
    stm = ibm_db.prepare(conn, get_confirmed_agents)
    ibm_db.bind_param(stm, 1, True)
    ibm_db.execute(stm)


    agents = ibm_db.fetch_assoc(stm)
    agents_list = []


    while(agents != False):
        temp = []


        temp.append(agents['FIRST_NAME'])
        temp.append(agents['AGENT_ID'])
        agents_list.append(temp)
        print(temp)


        agents = ibm_db.fetch_assoc(stm)


    # Getting the unassigned tickets
    stmt = ibm_db.prepare(conn, get_unassigned_tickets)
    ibm_db.execute(stmt)


    tickets = ibm_db.fetch_assoc(stmt)
    tickets_list = []


    if tickets:
        # means there are still some unassigned tickets
        while tickets != False:
            temp = []


            temp.append(tickets['TICKET_ID'])
            temp.append(str(tickets['RAISED_ON'])[0:10])
            temp.append(tickets['FIRST_NAME'])
            temp.append(tickets['ISSUE'])


            tickets_list.append(temp)


            tickets = ibm_db.fetch_assoc(stmt)
            return render_template(
                'admin tickets.html',
                id = 0,
                tickets_to_show = True,
                tickets = tickets_list,
                msg = "These are the unassigned tickets",
                agents = agents_list,
                user = USER_ADMIN
            )


        else:
            # all the tickets may be assigned
            # may be, there are no tickets raised in the system at all
            return render_template(
                'admin tickets.html',
                id = 0,
                tickets_to_show = False,
                msg = "There is nothing to assign!",
                user = USER_ADMIN
            )


    except:
        # something fishy happened while getting the tickets
        # so alerting the admin
        return render_template(
            'admin tickets.html',
            id = 0,
            to_show = True,
            message = "Something wrong! Please TrY Again",
            user = USER_ADMIN
        )
    else:
        # logging out
        return redirect(url_for('blue_print.logout'))


@admin.route('/admin/agents')
@login_required
def agents():
    '''
        Returning all the confirmed agents from the database
    '''


    from .views import admin


    if(hasattr(admin, 'email')):
        # query to get all the confirmed agents
        get_confirmed = '''
            SELECT * FROM agent WHERE confirmed = ?
        '''


        try:
            stmt = ibm_db.prepare(conn, get_confirmed)
            ibm_db.bind_param(stmt, 1, True)
            ibm_db.execute(stmt)


            agents = ibm_db.fetch_assoc(stmt)
            agents_list = []


            if agents:
                # there are some confirmed agents
        while agents != False:
            temp = []


            temp.append(agents['AGENT_ID'])
            temp.append(str(agents['DATE_JOINED'])[0:10])
            temp.append(agents['FIRST_NAME'])
            temp.append(agents['LAST_NAME'])
            temp.append(agents['EMAIL'])


            agents_list.append(temp)


            agents = ibm_db.fetch_assoc(stmt)

        return render_template(
            'admin agents.html',
            id = 1,
            msg = "List of confirmed agents",
            agents_to_show = True,
            agents = agents_list,
            user = USER_ADMIN
        )


    else:
        # no confirmed agents present
        return render_template(
            'admin agents.html',
            id = 1,
            msg = "No agents present",
            agents_to_show = False,
            user = USER_ADMIN
        )
        except:
            # something happened while fetching the agents
            return render_template(
                'admin agents.html',
                id = 1,
                mmessage = "Something happened! Please try again",
                to_show = True,
                user = USER_ADMIN
            )


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


@admin.route('/admin/accept')
@login_required
def accept():
    '''
        Loading the agents info from the database who are not yet confirmed
    '''

    from .views import admin


    if(hasattr(admin, 'email')):
        # query to get all the agents from the database who are all not confirmed yet
        get_agents_query = '''
            SELECT * FROM agent WHERE confirmed = ?
        '''
```

```
agents_to_show = False
msg = ""


try:
    stmt = ibm_db.prepare(conn, get_agents_query)
    ibm_db.bind_param(stmt, 1, False)
    ibm_db.execute(stmt)


    agents = ibm_db.fetch_assoc(stmt)


    agents_list = []


    while agents != False:
        temp = []


        temp.append(agents['AGENT_ID'])
        temp.append(agents['EMAIL'])
        temp.append(agents['FIRST_NAME'])
        temp.append(agents['DATE_JOINED'])


        agents_list.append(temp)


        agents = ibm_db.fetch_assoc(stmt)


    if len(agents_list) >= 1:
        # there are some agents who are not yet confirmed
        msg = "These are the pending requests"
```

```
                agents_to_show = True


            else:
                agents_to_show = False
                msg = "There are no pending requests"


            return render_template(
                'admin acc agent.html',
                id = 2,
                agents = agents_list,
                agents_to_show = agents_to_show,
                msg = msg,
                user = USER_ADMIN
            )


        except:
            # something happened while admin either accepts/denies the agent
            return render_template(
                'admin acc agent.html',
                to_show = True,
                message = "Something went wrong!",
                id = 2,
                user = USER_ADMIN
            )


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


@admin.route('/admin/about')
```

```
@login_required
def about():
    '''
        Showing the about of the application to the admin
    '''


    from .views import admin


    if(hasattr(admin, 'email')):
        return render_template(
            'admin about.html',
            id = 3,
            user = USER_ADMIN
        )


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


@admin.route('/admin/support')
@login_required
def support():
    '''
        Showing all the feedbacks given by the agents and customers
    '''
    from .views import admin


    if(hasattr(admin, 'email')):
        # query to retrieve all the feedbacks submitted
        get_feedbacks_query = '''
            SELECT * FROM feedback ORDER BY RAISED_ON DESC
        '''


        try:
            stmt = ibm_db.prepare(conn, get_feedbacks_query)
            ibm_db.execute(stmt)
            feedbacks = ibm_db.fetch_assoc(stmt)
            feedbacks_list = []


            feedbacks_to_show = False


            while feedbacks != False:
                temp = []


                temp.append(str(feedbacks['RAISED_ON'])[0:10])
                temp.append(feedbacks['RAISED_BY'])
                temp.append(feedbacks['RAISED_NAME'])
                temp.append(feedbacks['FEED'])


                feedbacks_list.append(temp)

                feedbacks = ibm_db.fetch_assoc(stmt)


            if len(feedbacks_list) > 0:
                # some feedbacks are submitted
                msg = 'All the feedbacks from the customers and agents'
                feedbacks_to_show = True
```

```
            else:
                # no feedbacks submitted yet
                msg = 'No feedbacks yet!'


            return render_template('admin support.html',
                id = 4,
                user = USER_ADMIN,
                feedbacks = feedbacks_list,
                msg = msg,
                true = feedbacks_to_show
            )


        except:
            # something happened while fetching the feedbacks
            return render_template('admin support.html',
                id = 4,
                user = USER_ADMIN,
                to_show = True,
                message = 'Something went wrong! Please try again'
            )


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


@admin.route('/admin/<email>/<action>')
@login_required
def alter(email, action):
    '''
        Either accepting or denying the agent, as per the admin's decision
    '''
```

```
from .views import admin


if(hasattr(admin, 'email')):
    if action == "True":
        # admin chose to the accept the agent
        accept_query = '''
            UPDATE agent SET confirmed = ? WHERE email = ?
        '''


        stmt = ibm_db.prepare(conn, accept_query)
        ibm_db.bind_param(stmt, 1, True)
        ibm_db.bind_param(stmt, 2, email)


        ibm_db.execute(stmt)


    else:
        # admin must have chosen to delete the agent
        delete_query = '''
            DELETE FROM agent WHERE email = ?
        '''


        stmt = ibm_db.prepare(conn, delete_query)
        ibm_db.bind_param(stmt, 1, email)


        ibm_db.execute(stmt)


    return "None"
    else:
        # logging out
        return redirect(url_for('blue_print.logout'))



@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'))
```

Dockerfile

```
FROM
python:3.7
        WORKDIR /app
        ADD . /app
        COPY requirements.txt /app
        RUN python -m pip install -r requirements.txt
        EXPOSE 5000
        ENTRYPOINT [ "python" ]
        CMD [ "app.py" ]
```

```
.detail-
card{
                text-align: center;
                margin-top:0.5em !important;
                border-bottom: 1px black solid;
                border-radius: 5px;
                border: none;
                font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
            }


            .bl{
                background-color: rgb(151, 151, 245) !important;
            }


            .yl{
                background-color: rgb(243, 240, 88);
            }


            .gr{
                background-color: rgb(95, 204, 91);
            }


            /* login form */
            .login-form{
                margin-top:10em;
            }


            /* ticket detail */
            .ticket-detail{
                margin-top:15em;
                border: 1px black solid;
                padding:1em;
                border-radius:0.2em;
            }
```

App py

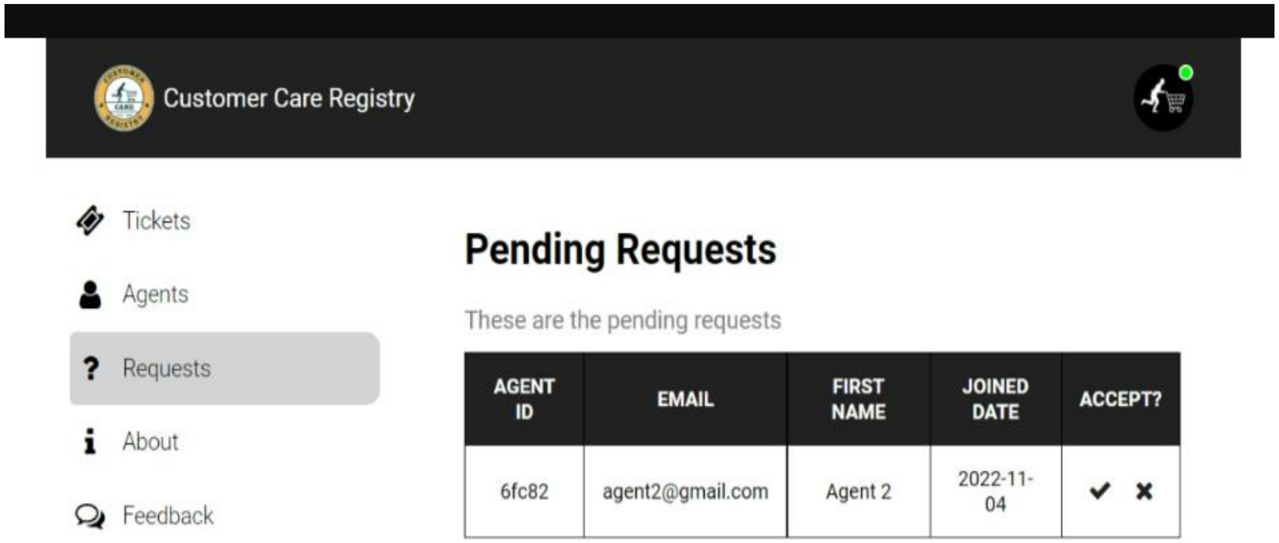
```
                        .register-form{
                            margin-top:10em;
                        }


                        body{
                            background-image: url('https://external-content.duckduckgo.com/iu/?u=http%3A%2F%2Fwww.pixelstalk.net%2Fwp-
                        content%2Fuploads%2F2016%2F04%2FPhotos-download-abstract-minimalist-wallpaper-HD.jpg&f=1&nofb=1');
                            background-repeat: no-repeat;
                            background-size: cover;
                        }
 FROM
 python:3.7
             WORKDIR /app
             ADD . /app
             COPY requirements.txt /app
             RUN python -m pip install -r requirements.txt
             EXPOSE 5000
             ENTRYPOINT [ "python" ]
             CMD [ "app.py" ]
```

Requirement

```
 flask
        flask_login
        ibm_db
```

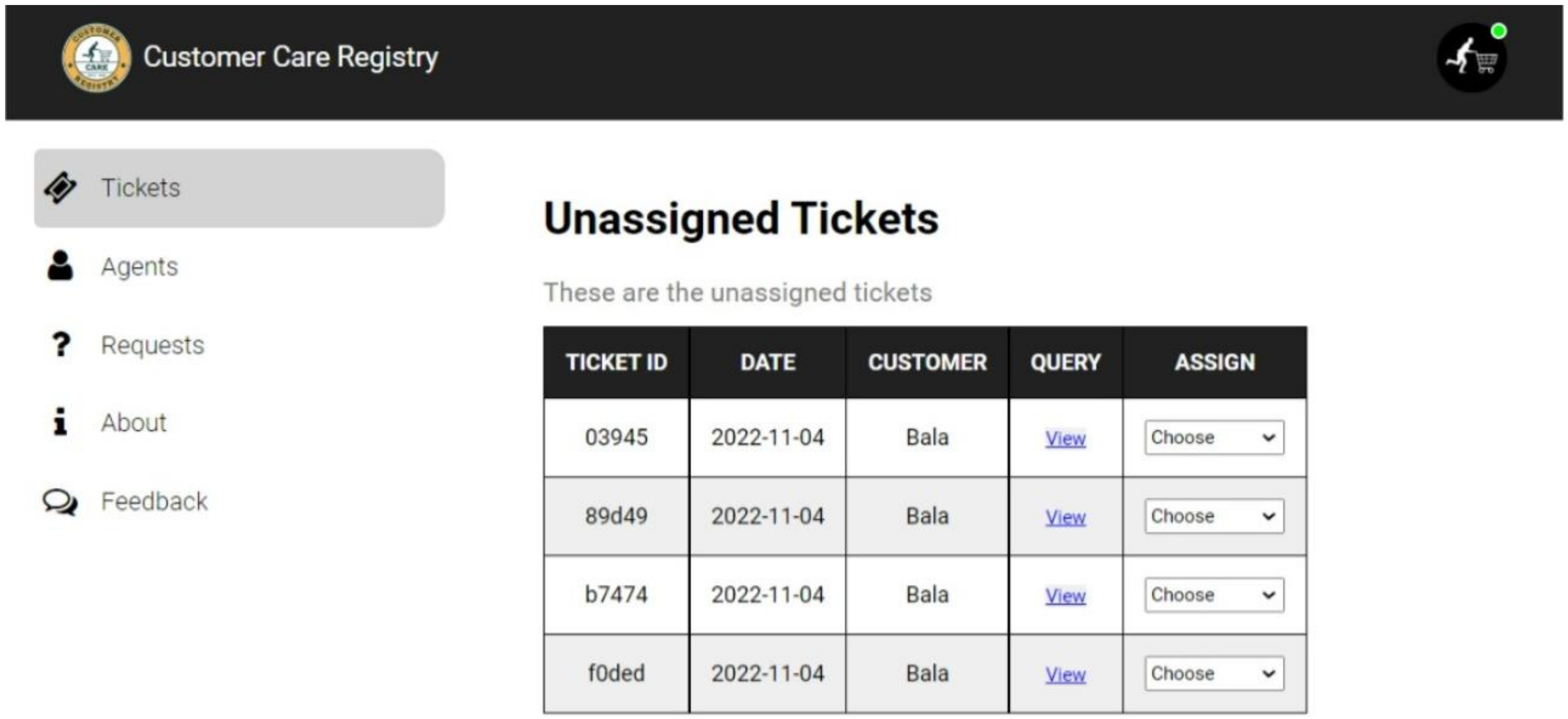
output



Pending Requests

These are the pending requests

AGENT ID	EMAIL	FIRST NAME	JOINED DATE	ACCEPT?
6fc82	agent2@gmail.com	Agent 2	2022-11-04	✔ ✖

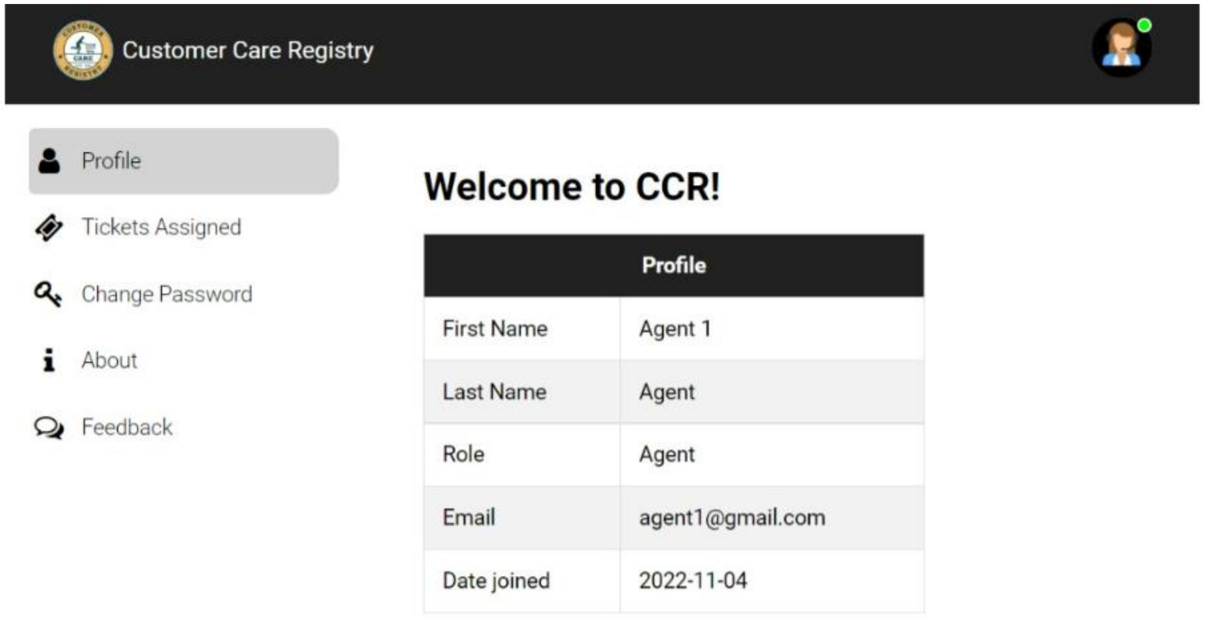


Unassigned Tickets

These are the unassigned tickets

TICKET ID	DATE	CUSTOMER	QUERY	ASSIGN
03945	2022-11-04	Bala	View	Choose
89d49	2022-11-04	Bala	View	Choose
b7474	2022-11-04	Bala	View	Choose
f0ded	2022-11-04	Bala	View	Choose

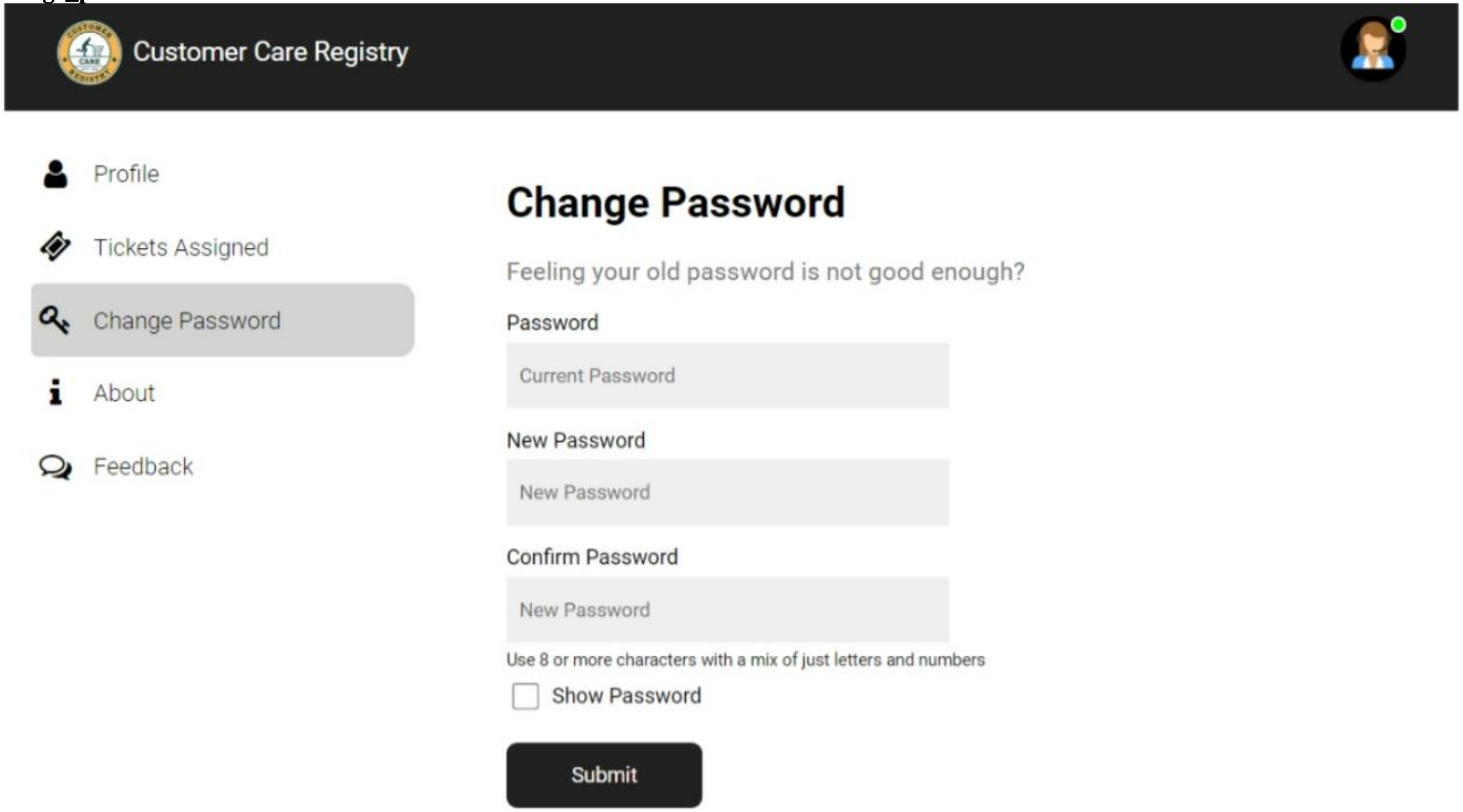
Agent dashboard



Welcome to CCR!

Profile	
First Name	Agent 1
Last Name	Agent
Role	Agent
Email	agent1@gmail.com
Date joined	2022-11-04

Change_password



Change Password

Feeling your old password is not good enough?

Password

Current Password

New Password

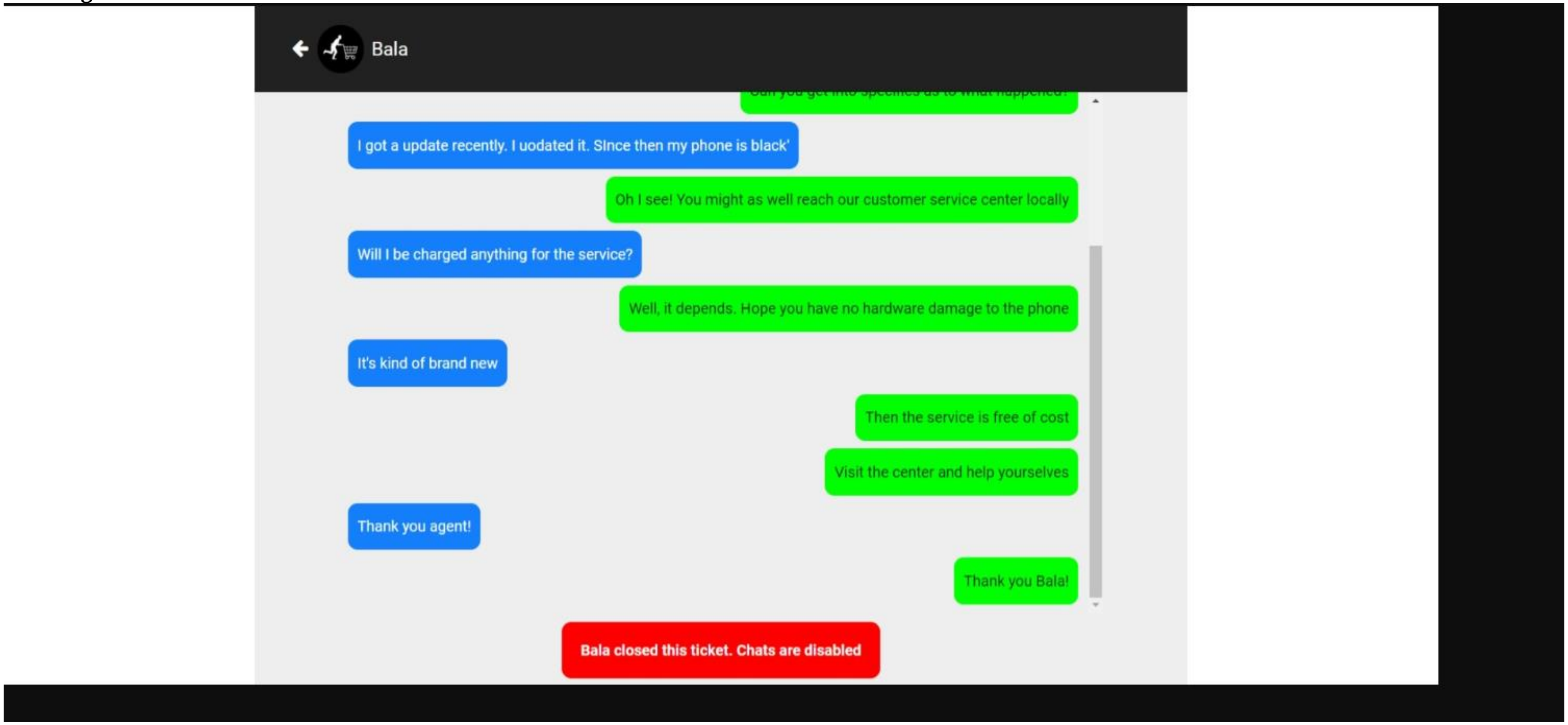
New Password

Confirm Password

New Password

Use 8 or more characters with a mix of just letters and numbers

Chatting with customer



I got a update recently. I uodated it. SInce then my phone is black'

Oh I see! You might as well reach our customer service center locally

Will I be charged anything for the service?

Well, it depends. Hope you have no hardware damage to the phone

It's kind of brand new

Then the service is free of cost

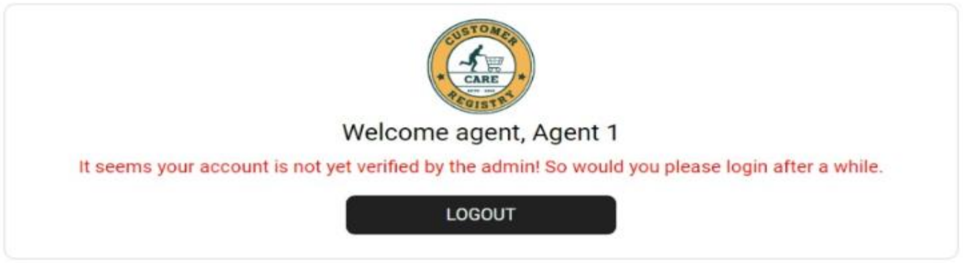
Visit the center and help yourselves

Thank you agent!

Thank you Bala!

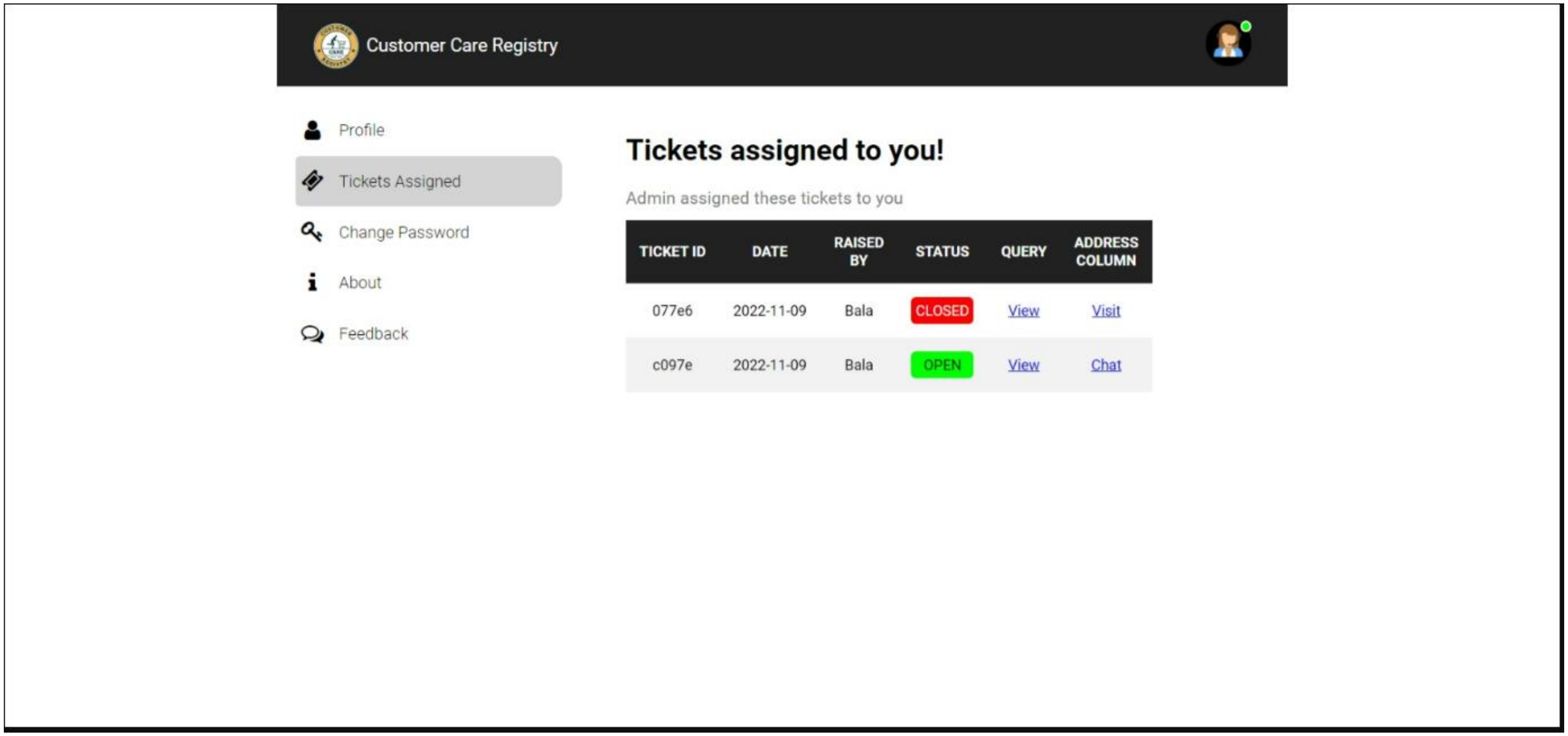
Bala closed this ticket. Chats are disabled

Not confirmed


Welcome agent, Agent 1

It seems your account is not yet verified by the admin! So would you please login after a while.

Ticket assigned

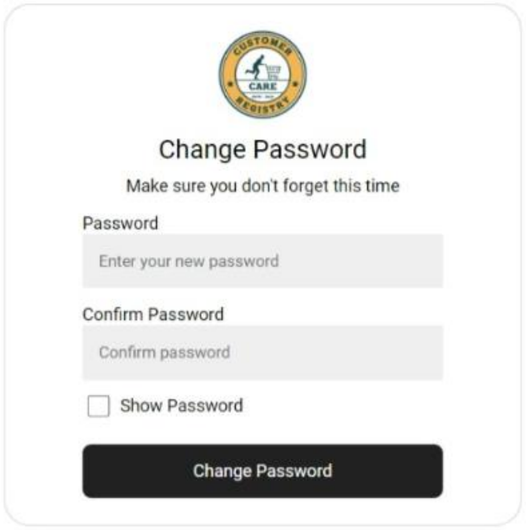


Tickets assigned to you!

Admin assigned these tickets to you

TICKET ID	DATE	RAISED BY	STATUS	QUERY	ADDRESS COLUMN
077e6	2022-11-09	Bala	CLOSED	View	Visit
c097e	2022-11-09	Bala	OPEN	View	Chat

Change password



Change Password

Make sure you don't forget this time

Password

Enter your new password

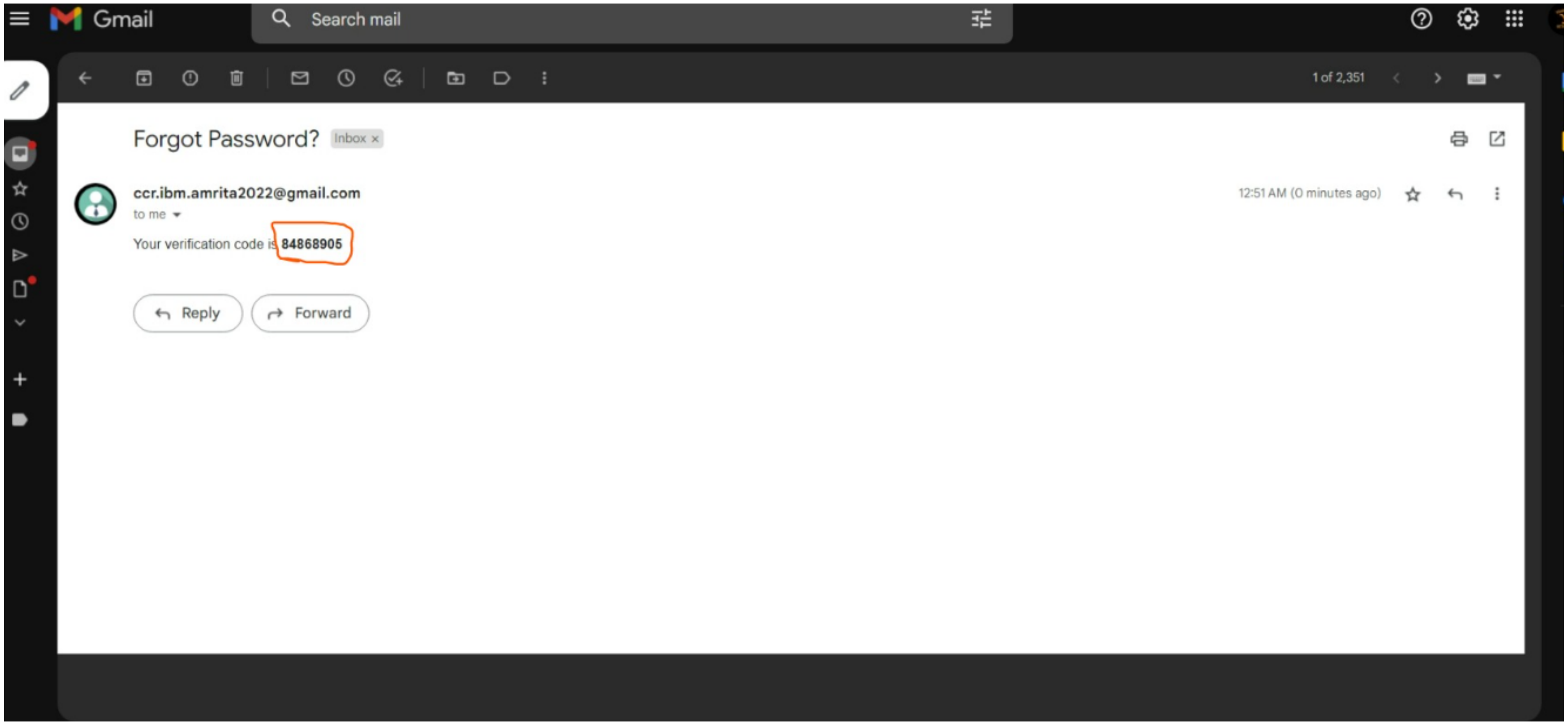
Confirm Password

Confirm password

Show Password

Change Password

Code sent to email



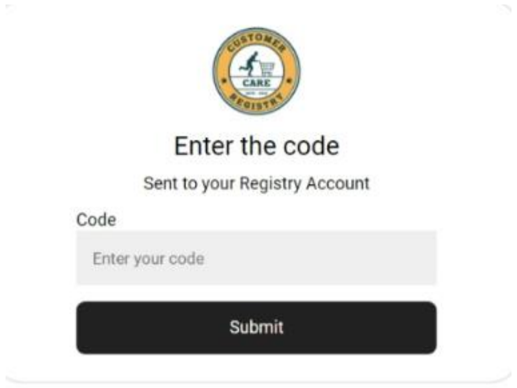
Forgot Password?

ccr.ibm.amrita2022@gmail.com
to me

12:51 AM (0 minutes ago)

Your verification code is 84868905

Enter the code


Enter the code

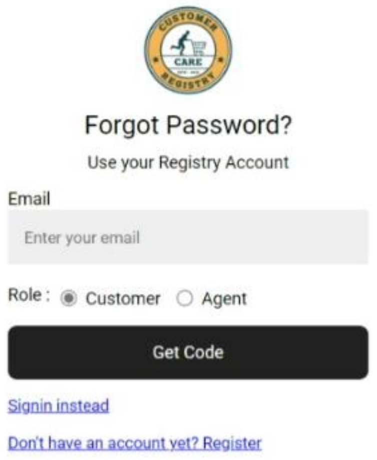
Sent to your Registry Account

Code

Enter your code

Submit

Forget password


Forgot Password?

Use your Registry Account

Email

Enter your email

Role : [✓ Customer] Agent

Get Code

Signin instead

Don't have an account yet? Register

8.TESTING

Test Cases Performed:

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass / Fail
61.	Customer forgot the password and trying to update the password with invalid email	1. Go to the site 2. Click "Forgot Password?" option in the Login form 3. Enter the email 4. Click "Get Code" button	Email = suryathayagmail.com Role = "Customer"	Customer should get an alert saying "Invalid email!"	As expected	Pass
62.	Customer forgot the password and trying to update the password with invalid email	1. Go to the site 2. Click "Forgot Password?" option in the Login form 3. Enter the email 4. Click "Get Code" button	Email = suryathaya@gmail.com Role = "Customer"	Customer should get an alert saying "Customer does not exist"	As expected	Pass

63.	Customer forgot the password and trying to update the password with valid email	1. Go to the site 2. Click “Forgot Password?” option in the Login form 3. Enter the email 4. Click “Get Code” button	Email = suryathaya10@gmail.com Role = “Customer”	Customer should receive an 8-digit code in the email and redirected to the code entering page	As expected	Pass
64.	Customer entering invalid code to change the password	1. Go to the site 2. Click “Forgot Password?” option in the Login form 3. Enter the email 4. Click “Get Code” button 5. Enter invalid code 6. Click “Submit” button	Email = suryathaya10@gmail.com Role = “Customer” Code = “bhuudbsgygdy2”	Customer should get an alert saying "Invalid code!"	As expected	Pass
65.	Customer entering valid code to change the password	1. Go to the site 2. Click “Forgot Password?” option in the Login form 3. Enter the email 4. Click “Get Code” button 5. Enter the valid code received in the email 6. Click “Submit” button	Email = suryathaya10@gmail.com Role = “Customer” Code = “87436601”	Customer should be redirected to the passwords entering page	As expected	Pass
66.	Customer entering the invalid passwords in the change password page	1. Go to the site 2. Click “Forgot Password?” option in the Login form 3. Enter the email 4. Click “Get Code” button 5. Enter the valid code received in the email 6. Click “Submit” button 7. Enter the passwords	Email = suryathaya10@gmail.com Role = “Customer” Code = “87436601” Password = 12345678 Confirm password = 87654321	Customer should get an alert saying “Passwords do not match!”	As expected	Pass

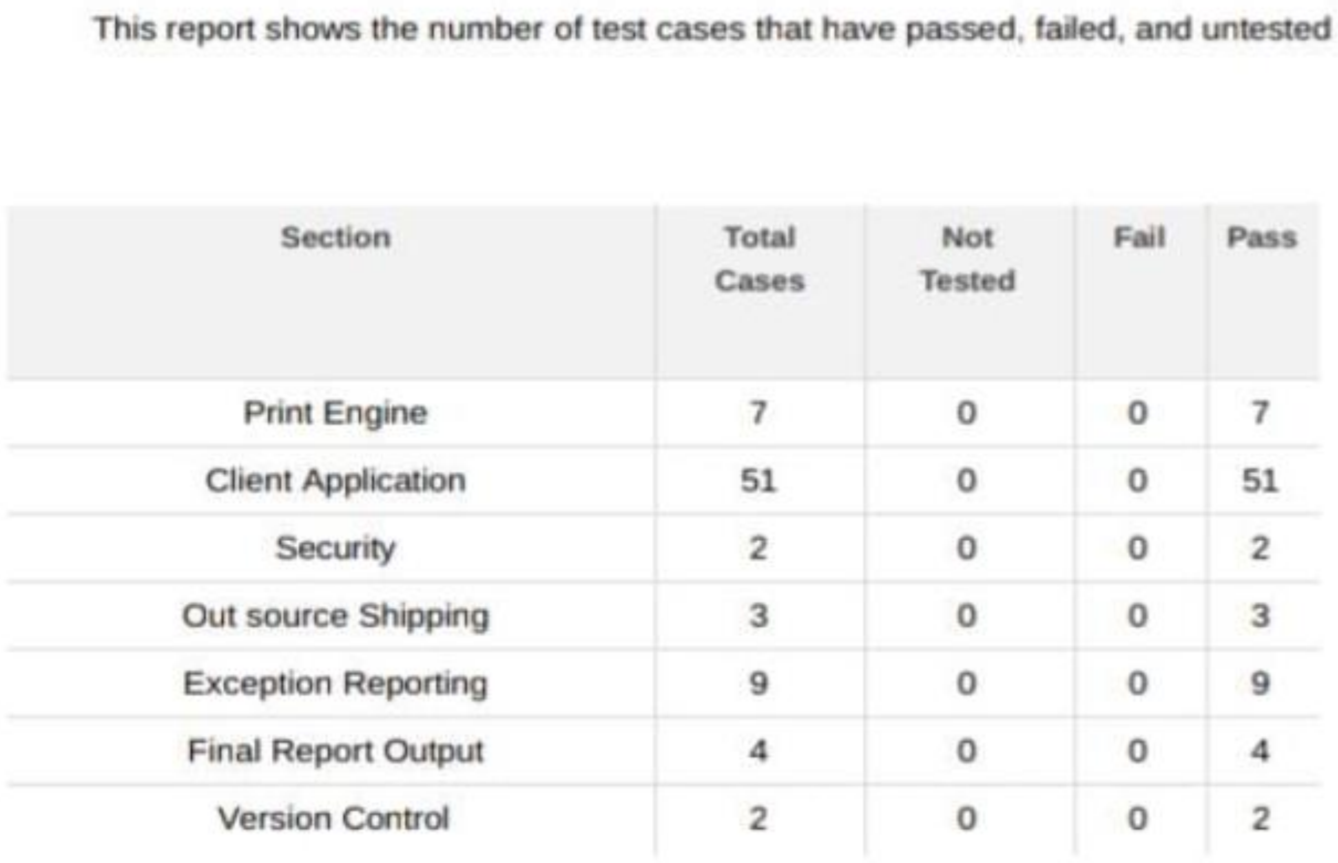
67.	Customer entering the new passwords in the change password page	1. Go to the site 2. Click “Forgot Password?” option in the Login form 3. Enter the email 4. Click “Get Code” button 5. Enter the valid code received in the email 6. Click “Submit” button 7. Enter the passwords	Email = suryathaya10@gmail.com Role = “Customer” Code = “87436601” Password = 12345678 Confirm password = 12345678	Customer's password gets updated. Then the customer is redirected to the login page to login	As expected	Pass
68.	Agent forgot the password and trying to update the password with invalid email	1. Go to the site 2. Click “Forgot Password?” option in the Login form 3. Enter the email 4. Click “Get Code” button	Email = agent1gmail.com Role = “Agent”	Agent should get an alert saying “Invalid email!”	As expected	Pass
69.	Agent forgot the password and trying to update the password with invalid email	1. Go to the site 2. Click “Forgot Password?” option in the Login form 3. Enter the email 4. Click “Get Code” button	Email = agent44@gmail.com Role = “Agent”	Agent should get an alert saying “Agent does not exist”	As expected	Pass
70.	Agent forgot the password and trying to update the password with valid email	1. Go to the site 2. Click “Forgot Password?” option in the Login form 3. Enter the email 4. Click “Get Code” button	Email = agent1@gmail.com Role = “Agent”	Agent should receive an 8-digit code in the email and redirected to the code entering page	As expected	Pass

71.	Agent entering invalid code to change the password	1. Go to the site 2. Click "Forgot Password?" option in the Login form 3. Enter the email 4. Click "Get Code" button 5. Enter invalid code 6. Click "Submit" button	Email = agent1@gmail.com Role = "Agent" Code = "bhuudbsgygdy2"	Agent should get an alert saying "Invalid code!"	As expected	Pass
72.	Agent entering valid code to change the password	1. Go to the site 2. Click "Forgot Password?" option in the Login form 3. Enter the email 4. Click "Get Code" button 5. Enter the valid code received in the email 6. Click "Submit" button	Email = agent1@gmail.com Role = "Agent" Code = "87436601"	Agent should be redirected to the passwords entering page	As expected	Pass
73.	Agent entering the invalid passwords in the change password page	1. Go to the site 2. Click "Forgot Password?" option in the Login form 3. Enter the email 4. Click "Get Code" button 5. Enter the valid code received in the email 6. Click "Submit" button 7. Enter the passwords	Email = agent1@gmail.com Role = "Agent" Code = "87436601" Password = 12345678 Confirm password = 87654321	Agent should get an alert saying "Passwords do not match!"	As expected	Pass

74.	Agent entering the new passwords in the change password page	1. Go to the site 2. Click "Forgot Password?" option in the Login form 3. Enter the email 4. Click "Get Code" button 5. Enter the valid code received in the email 6. Click "Submit" button 7. Enter the passwords	Email = agent1@gmail.com Role = "Agent" Code = "87436601" Password = 12345678 Confirm password = 12345678	Agent's password gets updated. Then the customer is redirected to the login page to login	As expected	Pass

# Along with these test cases, test cases performed during the Sprint 1, 2, 3 are also performed.

User Acceptance Testing



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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Out source Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULTS

Performance Metrics

Software quality is a measurement of something intangible, “how good” a software product really is. Some of the aspects of software quality taken are

a. Scalability

b. Speed

c. Stability

d. Reliability

e. Security

f. Maintainability and code quality

LOAD TEST

Scenario Name	Load Test – Customer Care Registry
Scenario Type	Load Test – Duration 1 hour
Scenario Objective	To Simulate the peak load and to monitor the performance of the Website
Steps	The online load will be maintained at steady state
Entry Criteria	All the monitors are in ready state
Exit Criteria	Response met the criteria and test completion report is agreed

STRESS TEST

Scenario Name	Stress Test – Customer Care Registry
Scenario Type	Stress Test
Scenario Objective	Objective is to verify that the application can handle the projected growth and to discover the breaking point
Steps	Ramp up to 150% of peak volume and continuously increase load until breaking point
Entry Criteria	All the monitors are in place Test Data is set up Peak load test completed successfully
Exit Criteria	Test completion report is agreed upon as per expectation

ENDURANCE / SOAK TEST:

Scenario Name	Soak Test – Customer Care Registry
Scenario Type	Endurance – Duration 8 hours
Scenario Objective	To discover memory issues and bottlenecks that might occur under daily usage of the application
Steps	Steady state is maintained for 8 hours with half of the peak load
Entry Criteria	All the monitors are in place Test Data is set up Peak load test completed successfully
Exit Criteria	Test completion report is agreed upon as per expectation

ADVANTAGES:

- Customer loyalty Loyal customers have many benefits for businesses. 91% of customers say a positive customer service experience makes them more likely to make a further purchase (source: Salesforce Research).
- Also, investing in new customers is five times more expensive than retaining existing ones (source: Invesp). Creating loyal customers through good customer service can therefore provide businesses with lucrative long-term relationships.
- Increase profits These long-term customer relationships established through customer service can help businesses become more profitable.
- Businesses can grow revenues between 4% and 8% above their market when they prioritise better customer service experiences (source: Bain & Company).
- Creating a better customer service experience than those offered by competitors can help businesses to standout in their market place, and in turn make more sale
- Good customer service can help businesses turn leads into sales. 78% of customers say they have backed out of a purchase due to a poor customer experience (source: Glance). It is therefore safe to assume that providing good customer service will help to increase customer confidence and in turn increase conversion.
- Customer service can help businesses to improve the public perception of the brand, which can then provide protection if there is a slip up. 78% of customers will forgive a company for a mistake after receiving excellent service (source: Salesforce Research).
- Meanwhile, almost 90% of customers report trusting a company whose service they rate as “very good.” On the other hand, only 16% of those who give a “very poor” rating trust companies to the same degree(source: Qualtrics XM Institute).
- Creating positive customer experiences is vital in gaining customer trust and creating a strong public image.

DISADVANTAGES:

- The Consumer Protection Act in India has numerous restrictions and drawbacks, which are listed in this article.
  Only services for which a particular payment has been made are covered under the consumer protection act.
- However, it does not protect medical professionals, or hospitals, and covers cases when this act does not apply to free medical care. This act does not apply to mandatory services, such as water supply, that are provided by state agencies. Only two clauses related to the supply of hazardous materials are covered by this act. Consumer redress is not given any power by the consumer protection act. The consumer protection act focuses on the supply of ineffective products, but there are no strict regulations for those who produce it.

2. Conclusion

- It is a web-enabled project.
- With this project the details about the product will be given to the customers in detail with in a short span of time.
- Queries regarding the product or the services will also be clarified. It provides more knowledge about the various technologies.

11.APPENDIX

GITHUB LINK :

https://github.com/IBM-EPBL/IBM-Project-32080-1660207971