PROJECT REPORT

19/11/2022

Customer Care Registry

Project Name	:	Customer Care Registry
Project Domain	:	Cloud Application Development
College	:	Chennai Institute of Technology
Team ID	:	**PNT2022TMID24848**
Team Size	:	4
Team Members	:	A. Tenali Karthikeya B. Gadamsetty Harikishan C. Bysani Lakshmi Vivek D. Pakam Tharun
Faculty Mentor	:	Mr. T Sampath
Industry Mentor	:	Mr. Vasudeva Hanush
Github Link	:	Click Here
Project Demo Link	:	Click Here

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1	**INTRODUCTION**	
	1.1 Project Overview	4
	1.2 Project Purpose	4
2	**LITERATURE SURVEY**	
	2.1 Existing Problem	5
	2.2 References	5
	2.3 Problem Statement Definition	5
3	**IDEATION AND PROPOSED SOLUTION**	
	3.1 Empathy Map Canvas	6
	3.2 Ideation and Brainstorming	7
	3.3 Proposed Solution	9
	3.4 Problem Solution Fit	10
4	**REQUIREMENT ANALYSIS**	
	4.1 Functional Requirements	11
	4.2 Non-functional Requirements	12
5	**PROJECT DESIGN**	
	5.1 Data Flow Diagram	13
	5.2 Solution and Technical Architecture	14
	5.3 User Stories	15
6	**PROJECT PLANNING AND SCHEDULING**	
	6.1 Sprint Planning and Estimation	16
	6.2 Sprint Delivery Schedule	16
	6.3 Reports from JIRA	17
7	**CODING AND SOLUTIONING**	
	7.1 Main app.py source code	19
	7.2 Login page source code	24
	7.3 Registration source code	26
	7.4 IBM Cloud Connection code	28

8	**TESTING**	
	8.1 Test Cases	30
	8.2 User Acceptance Testing	31

9	**RESULTS**	
	9.1 Performance Metrics	32
10	**ADVANTAGES AND DISADVANTAGES**	33
11	**CONCLUSION**	34
12	**FUTURE SCOPE**	34
13	**APPENDIX**	35

1. INTRODUCTION

PROJECT OVERVIEW

1.1

Short Description:

This Application has been developed to help the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to a customer, they will be notified with an email alert. Customers can view the status of the ticket till the service is provided.

Admin: The main role and responsibility of the admin are to take care of the whole process. Starting from Admin login followed by the agent creation and assigning the customer's complaints. Finally, He will be able to track the work assigned to the agent and a notification will be sent to the customer.

User: They can register for an account. After the login, they can create the complaint with a description of the problem they are facing. Each user will be assigned with an agent. They can view the status of their complaint.

1.2 PURPOSE

The purpose of the whole project is to:

- Provide a common platform to the customers to clarify their queries
- Having expert agents in the platform for better answering
- Customer's tickets (queries) are answered quickly by the agents
- Customers and Agents can chat with one another for better understanding
- While doing so, the former asks questions
- Later, answers those questions as quickly and as legitimately as possible
- Customers can raise as many tickets as they want
- Customers and Agents can also submit their feedbacks to the Admin, for the betterment of the platform

2. LITERATURE SURVEY

2.1 Existing Problem

- Reviews and rating in the e-commerce websites are not reliable
- Even more so, they are often been given by the manufactures themselves
- Reviews are not from the authentic individuals
- After buying the products, I am left with no option to clear my doubts
- There is no common platform available to us, the customers, to have our doubts cleared
- If it is existing, we are not getting fast replies. By the time, the reply comes, the issue might have been cleared or of not worth of being cleared to the customers

2.2 References

https://www.helpdesk.com/

https://freshdesk.com/helpdesk-software

https://freshdesk.com/resources/case-study/hamleys

https://pulsedesk.com/

https://www.redpoints.com/blog/amazon-fake-reviews/

2.3 Problem Statement Definition

I am Ramu and I am a regular customer in famous e-commerce websites like Amazon, Flipkart. I order regularly. The problem I have is that in most times, I don't have any reliable sources to clear my doubts in some of the products I buy.

There are reviews and customer ratings in those websites, but somehow, I don't feel they are authentic and real. It would make my world if those replies were from a real expert, and I could clarify all my doubts in a single platform. Of course, I would need instant replies.

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas

- Empathy Map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes
- It is a useful tool to help teams to better understand their users
- Creating an effective solution requires understanding the true problem and the person who us experiencing it
- The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges

3.2 Ideation and Brainstorming

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

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

Team Gathering:

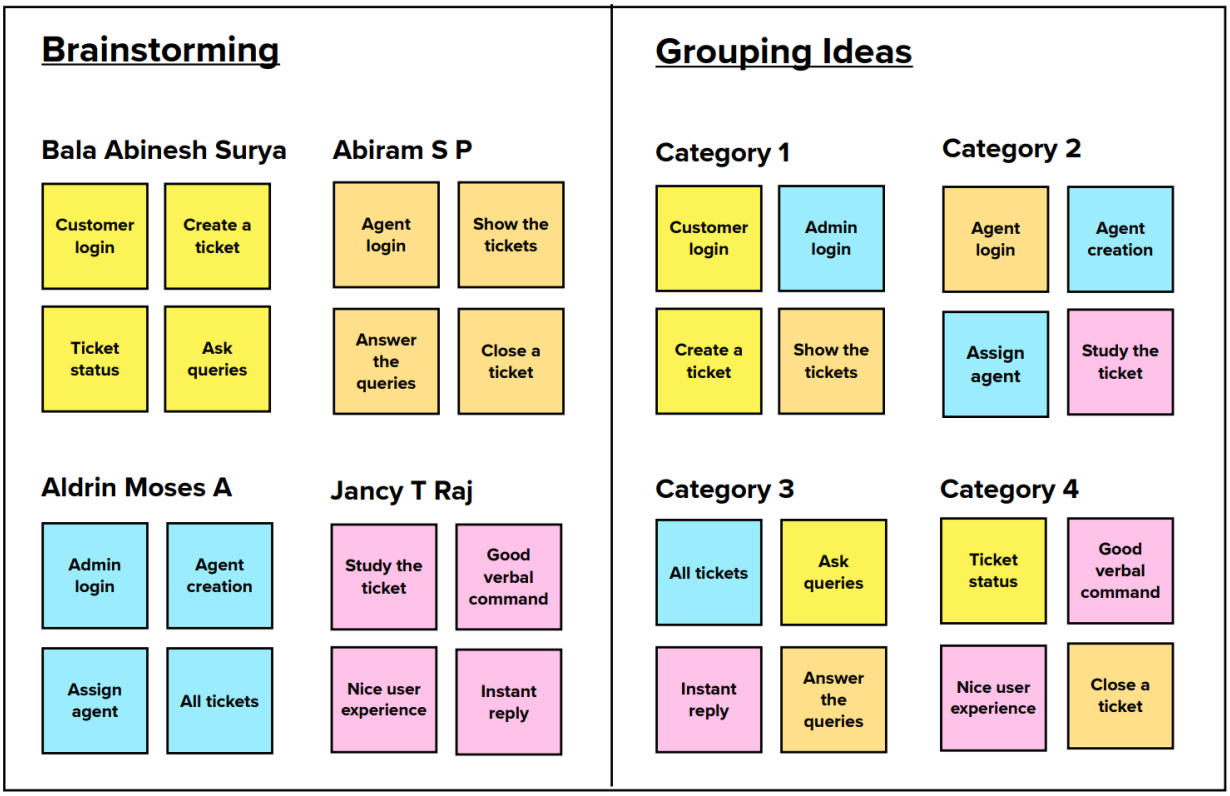
Team Members	
Team Leader	Tenali Karthikeya
Team Members	Pakam Tharun
	Gadamsetty Harikishan
	Bysani Lakshmi Vivek

Problem Statement:

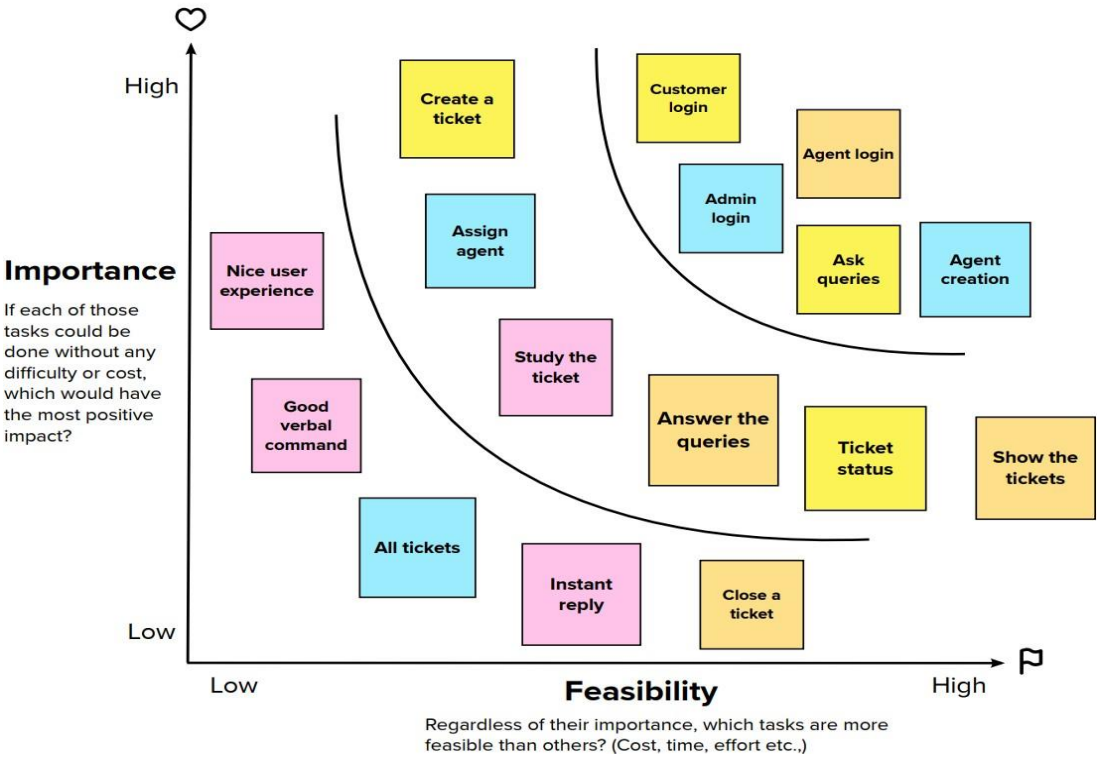
I am Surya and I am a regular customer in famous e-commerce websites like Amazon, Flipkart. I order regularly. The problem I have is that in most times, I don't have any reliable sources to clear my doubts in some of the products I buy.

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

Step-2: Brainstorm, Idea Listing and Grouping



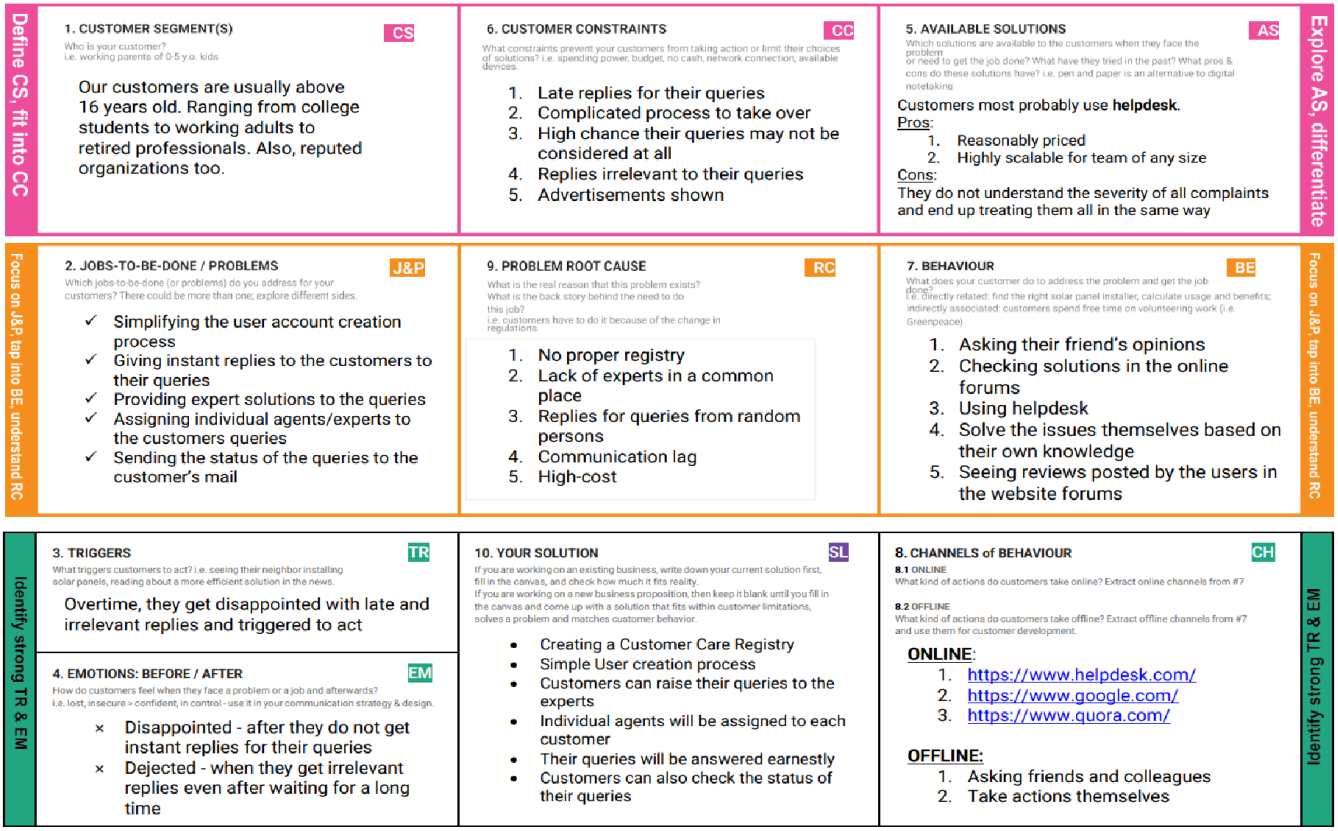
Step-3: Idea Prioritization



3.3 Proposed Solution

S. No.	Parameter	Description
•	Problem Statement (Problem to be solved)	I am Surya and I am a regular customer in famous e-commerce websites like Amazon, Flipkart. I order regularly. The problem I have is that in most times, I don't have any reliable sources to clear my doubts in some of the products I buy. There are reviews and customer ratings in those websites, but somehow, I don't feel they are authentic and real. It would make my world if those replies are from a real expert and I could clarify all my doubts in a single platform. Of course, I would need instant replies from a real expert who knows about the products I am asking for.
•	Idea / Solution description	Creating a Customer Care Registry, where the customers can raise their queries in form of tickets. An agent will be assigned to them for replying/clarifying their issue.
•	Novelty / Uniqueness	The agents are experts in the product domain and they will communicate well with the customers
•	Social Impact / Customer Satisfaction	Customers will be satisfied with the instant and valid replies. Also, it creates a doubtless society, that boosts sales.
•	Business Model (Revenue Model)	Customers can be charged a minimal amount based on the number of queries (tickets) they can rise in a said period of time.
•	Scalability of the Solution	May be in the future, may be a cross-platform mobile application may be developed, making this customer care registry much more accessible to the users.

3.4 Problem Solution Fit



4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

- A functional requirement defines a function of a system or its component, where a function is described as a specification of behaviour between inputs and outputs.
- It specifies “what should the software system do?”
- Defined at a component level
- Usually easy to define
- Helps you verify the functionality of the software

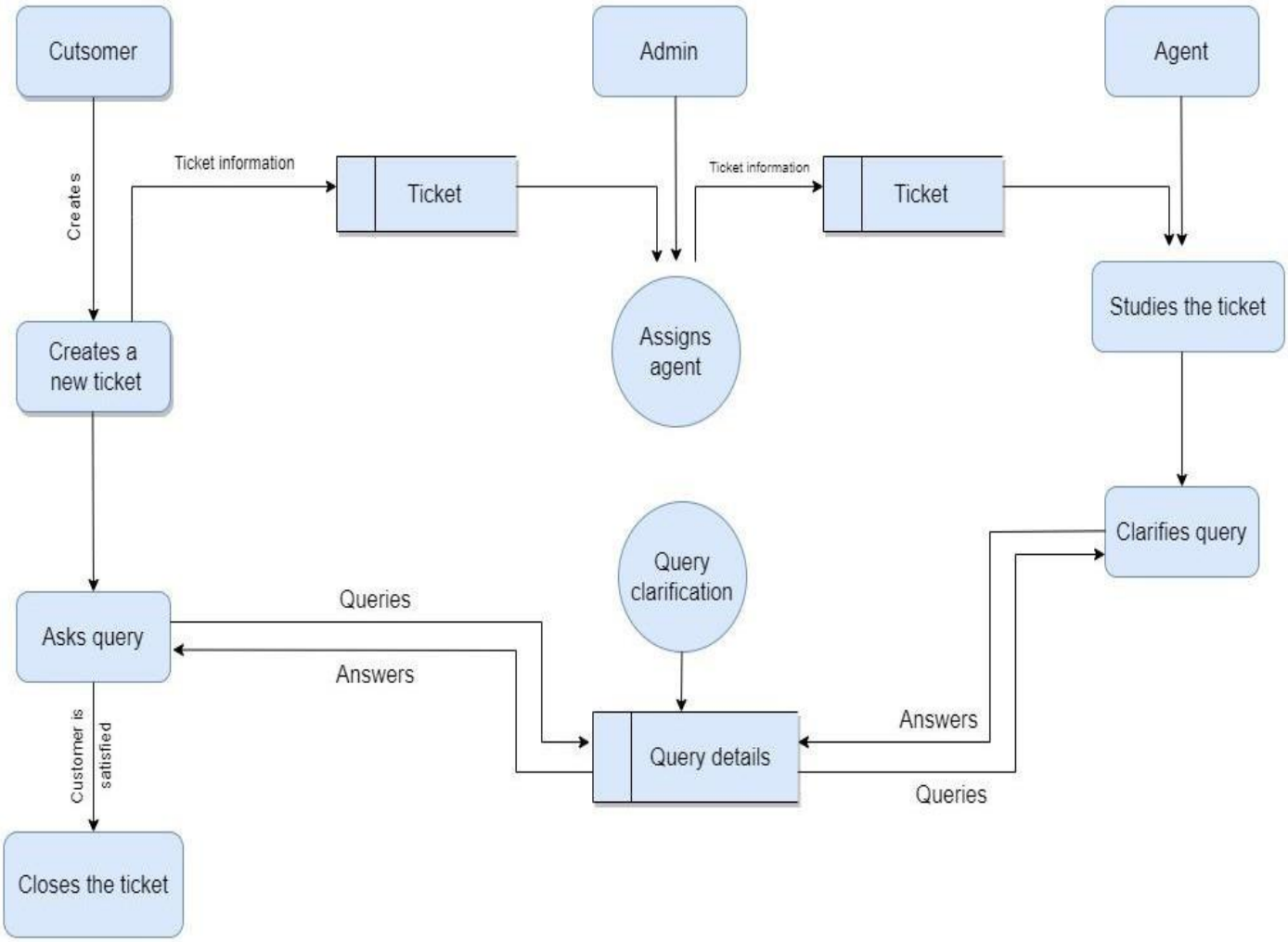
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Signup form (customer)
FR-2	Forgot Password	Resetting the password by sending an OTP to user’s mail (customer, agent, admin)
FR-3	User Login	Login through Login form (customer, agent, user)
FR-4	Agent creation (admin)	Create an agent profile with username, email and password
FR-5	Dashboard (customer)	Show all the tickets raised by the customer
FR-6	Dashboard (agent)	Show all the tickets assigned to the agent by admin
FR-7	Dashboard (Admin)	Show all the tickets raised in the entire system
FR-8	Ticket creation (customer)	Customer can raise a new ticket with the detailed description of his/her query
FR-9	Assign agent (admin)	Assigning an agent for the created ticket
FR-10	Ticket details (customer)	1. Showing the actual query, status, assigned agent details 2. Status of the ticket
FR-11	Address Column	Agent clarifies the doubts of the customer

4.1 Non-functional Requirements

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

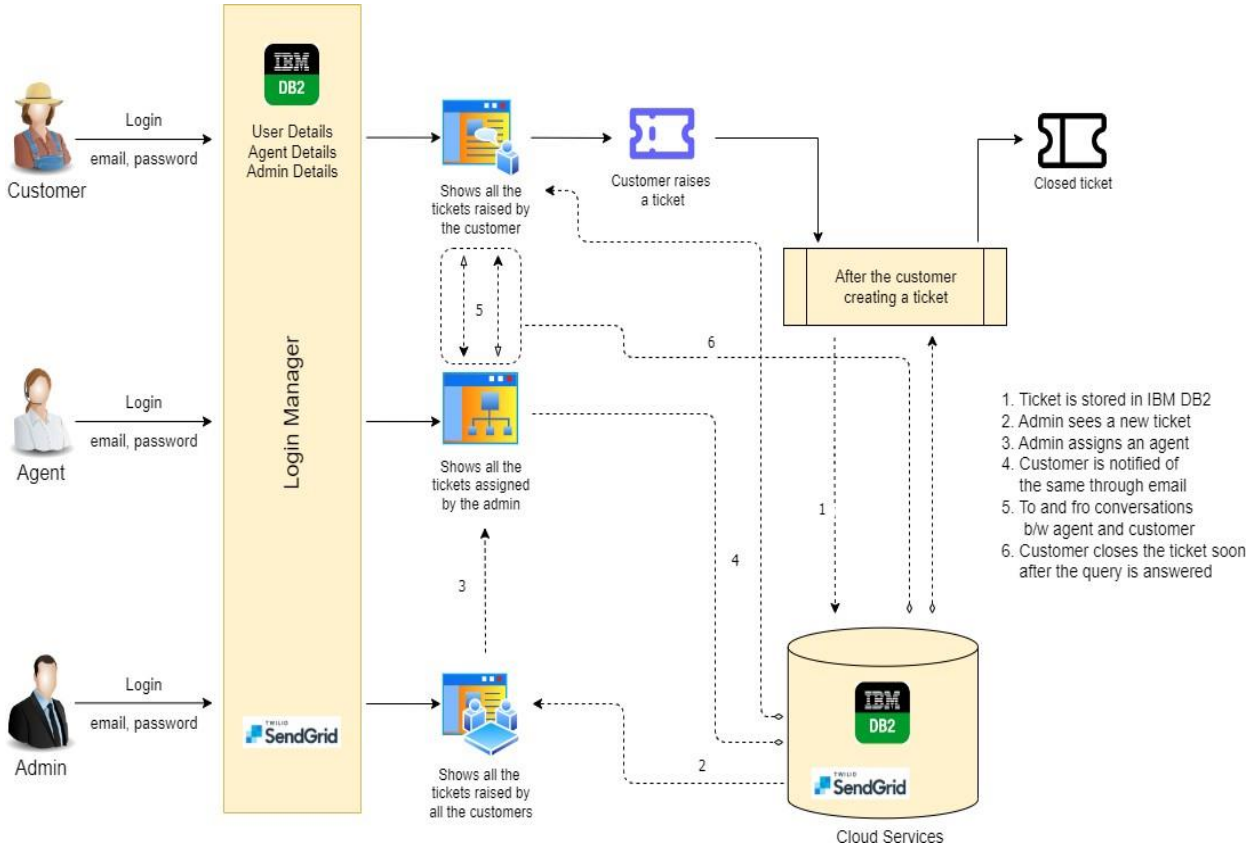
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Customers can use the application in almost all the web browsers. Application is with good looking and detailed UI, which makes it more friendly to use.
NFR-2	**Security**	Customers are asked to create an account for themselves using their email which is protected with an 8 character-long password, making it more secure.
NFR-3	**Reliability**	Customers can raise their queries and will be replied with a valid reply, as soon as possible, making the application even more reliable and trust-worthy.
NFR-4	**Performance**	Customers will have a smooth experience while using the application, as it is simple and is well optimised.
NFR-5	**Availability**	Application is available 24/7 as it is hosted on IBM Cloud
NFR-6	**Scalability**	In future, may be cross-platform mobile applications can be developed as the user base grows.

5.1 Dataflow Diagram

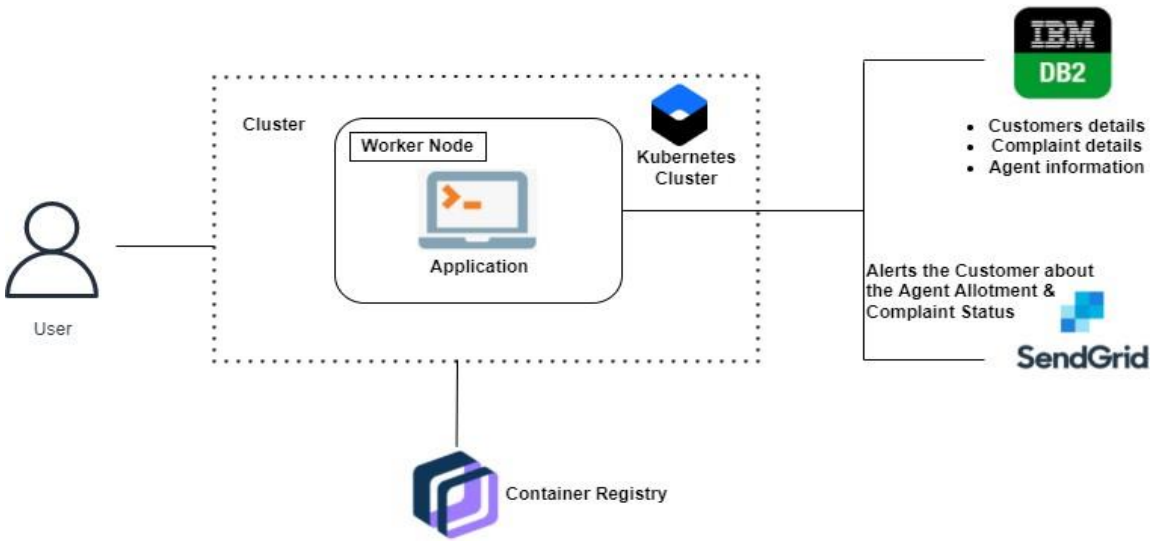


5.2 Solution and Technical Architecture

Solution Architecture



Technical Architecture



5.3 User Stories

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	login	USN-2	As a customer, I can login to the application by entering correctemail and password.	I can access my account/dashboard.	High	Sprint-1
	Dashboard	USN-3	As a customer, I can see all the orders raised by me.	I get all the info needed in my dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can place my order with the detailed description of my query	I can ask my query	Medium	Sprint-2
	Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	My queries are clarified.	High	Sprint-3
	Forgot password	USN-6	As a customer, I can reset my password by this option incase I forgot my old password.	I get access to my account again	Medium	Sprint-4
	Order details	USN-7	As a Customer ,I can see the current stats of order.	I get abetter understanding	Medium	Sprint-4
Agent (web user)	Login	USN-1	As an agent I can login to the application by entering Correct email and password.	I can access my account / dashboard.	High	Sprint-3
	Dashboard	USN-2	As an agent, I can see the order details assigned to me by admin.	I can see the tickets to which I could answer.	High	Sprint-3
	Address column	USN-3	As an agent, I get to have conversations with the customer and clear his/er dobuts	I can clarify the issues.	High	Sprint-3
	Forgot password	USN-4	As an agent I can reset my password by this option in case I forgot my old password.	I get access to my account again.	Medium	Sprint-4

Admin (Mobile user)	Login	USN-1	As a admin, I can login to the appliaction by entering Correct email and password	I can access my account/dashboard	High	Sprint-1
	Dashboard	USN-2	As an admin I can see all the orders raised in the entire system and lot more	I can assign agents by seeing those order.	High	Sprint-1
	Agent creation	USN-3	As an admin I can create an agent for clarifying the customers queries	I can create agents.	High	Sprint-2
	Assignment agent	USN-4	As an admin I can assign an agent for each order created by the customer.	Enable agent to clarify the queries.	High	Sprint-1
	Forgot password	USN-5	As an admin I can reset my password by this option in case I forgot my old password.	I get access to my account.	High	Sprint-1

6 PROJECT DESIGN AND PLANNING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and go through the services available on the webpage.	20	High	KARTHIKEYA VIVEK
Sprint-2	Admin Panel	USN-2	The role of the admin is to check out the database about the availability and have a track of all the things that the users are going to service	20	High	HARI KISHAN THARUN
Sprint-3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding the services. Get the recommendations based on information provided by the user	20	High	THARUN VIVEK
Sprint-4	Final delivery	USN-4	Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application	20	High	KARTHIKEYA HARI KISHAN

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		19 Nov 2022

6.3 **Reports from JIRA**

Sprint 1 – Burndown Chart

Sprint 2 – Burndown Chart

Sprint 3 – Burndown Chart

Sprint 4 – Burndown Chart

7. CODING AND SOLUTIONING

7.1 App.py code

```
import ibm_db
from flask import Flask, render_template, request, redirect, url_for, flash, session
from ticket.User import User
from ticket.Ticket import Ticket
import os
from sendgrid.helpers.mail import *
import sendgrid

app = Flask(__name__)
app.secret_key = b'_4#z2G"F5Q9z\n\xec]/'


@app.route("/")
def show_login():
    return redirect(url_for('login'))


@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        print("hi")
        if request.form['username'] != "" and request.form['password'] != "":
            user = User()
            user.User_Type = request.form['user_type']
            user.Email = request.form['username']
            user.Password = request.form['password']
            result = user.login()

            print("login result", result)
            if len(result) > 0:
                session['name'] = result[0]['NAME']
                session['user_id'] = result[0]['ID']
                session['user_type'] = result[0]['USER_TYPE']

            return redirect(url_for('dashboard'))

        else:
            flash(u'username or password is incorrect.', 'danger')
            return redirect(url_for('login'))

    else:
        return render_template('login.html')
```

```
@app.route("/user/signup", methods=['GET', 'POST'])
def vendor_signup():
    if request.method == 'POST':
        user = User()
        user.Id = ""
        user.Name = request.form['name']
        user.User_Type = request.form['user_type']
        user.Mobile = request.form['mobile']
        user.Email = request.form['email']
        user.Password = request.form['password']
        user.save()

        flash(
            u'User Sign up done, you login now with your username and password.',
'success')

        return redirect(url_for('login'))
    else:
        return render_template('register.html')


@app.route("/dashboard", methods=['GET'])
def dashboard():
    if session['name'] is None:
        return redirect(url_for('login'))

    # inventory = Inventory()
    # inventory = inventory.display()
    return render_template('dashboard.html')


@app.route("/ticket/create", methods=['GET', 'POST'])
def create_ticket():
    if session['name'] is None:
        return redirect(url_for('login'))

    if request.method == 'POST':
        ticket = Ticket()
        ticket.Title = request.form['title']
        ticket.Description = request.form['description']
        ticket.Priority = request.form['priority']
        id = request.form.get('id')

        old_ticket = Ticket()

        if id is not None:
            ticket.Id = id
            tickets = old_ticket.get(id)
            old_ticket = tickets[0]
```

```
        agent_id = request.form.get('agent_id')
        if agent_id is not None:
            ticket.AgentId = agent_id
        status = request.form.get('status')
        if status is not None:
            ticket.Status = status
        ticket.Status = 0
        ticket.save()

        if ticket.AgentId != 0 and ticket.AgentId != old_ticket["AGENTID"]:
            return redirect(url_for('ticketagentassigned', ticket_id=id))

        flash(u'Ticket has been saved successfully.', 'success')

        return redirect(url_for('active_tickets'))
    else:
        ticket = Ticket()
        agents = []
        return render_template('createcomplaint.html', ticket=ticket, agents=agents)


@app.route("/ticket/edit/<id1>", methods=['GET'])
def edit_ticket(id1):
    if session['name'] is None:
        return redirect(url_for('login'))

    ticket = Ticket()
    tickets = ticket.get(id1)
    ticket = tickets[0]

    user = User()
    agents = user.agents()
    return render_template('createcomplaint.html', ticket=ticket, agents=agents)


@app.route("/tickets/active", methods=['GET'])
def active_tickets():
    if session['name'] is None:
        return redirect(url_for('login'))

    ticket = Ticket()
    ticket.Status = 0
    tickets = ticket.display()

    print(tickets)
    return render_template('tickets.html', title='Active Tickets', tickets=tickets)


@app.route("/tickets/closed", methods=['GET'])
```

```
def closed_tickets():
    if session['name'] is None:
        return redirect(url_for('login'))

    ticket = Ticket()
    ticket.Status = 1
    tickets = ticket.display()
    return render_template('tickets.html', title='Closed Tickets', tickets=tickets)


@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('login'))


@app.route('/ticket/agent-assigned/<ticket_id>', methods=['GET'])
def ticketagentassigned(ticket_id):
    if session['name'] is None:
        return redirect(url_for('login'))

    id1 = ticket_id

    ticket = Ticket()
    ticket.close(id1)

    ticket = Ticket()
    tickets = ticket.get(id1)
    ticket = tickets[0]

    user = User()
    user.Id = ticket["USERID"]
    users = user.get()
    user = users[0]

    agent = User()
    agent.Id = ticket["AGENTID"]
    users = agent.get()
    agent = users[0]

    sg = sendgrid.SendGridAPIClient(
        api_key="SG.yde0mGzNTaW-
fYFGCTM0Fg.iqC5lfATXPcAYaRcicAj211yrHLqf9skvLvvQnfa6TU")
    from_email = Email("tenalikarthikeya67@gmail.com")
    to_email = To(user.Email)
    subject = "Customer Care Agent Assigned Notification"
    html_content = str(render_template(
        'email_agent_assigned.html', ticket=ticket, user=user, agent=agent))
    content = Content("text/html", html_content)
```

```
    print(html_content)
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)
    return redirect(url_for('active_tickets'))


@app.route('/ticket/close/<ticket_id>', methods=['GET'])
def ticketclose(ticket_id):
    if session['name'] is None:
        return redirect(url_for('login'))

    id1 = ticket_id

    ticket = Ticket()
    ticket.close(id1)

    ticket = Ticket()
    tickets = ticket.get(id1)
    ticket = tickets[0]

    user = User()
    user.Id = ticket["USERID"]
    users = user.get()
    user = users[0]
    sg = sendgrid.SendGridAPIClient(
        api_key="SG.yde0mGzNTaW-
fYFGCTM0Fg.iqC5lfATXPcAYaRcicAj211yrHLqf9skvLvvQnfa6TU")
    from_email = Email("tenalikarthikeya67@gmail.com")
    to_email = To(user["EMAIL"])
    subject = "Customer Care Ticket Closed Notification"
    content = Content(
        "text/html", render_template('email_ticket_closed.html', ticket=ticket,
user=user))
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)
    return redirect(url_for('active_tickets'))


if __name__ == "__main__":
    port = int(os.environ.get('PORT', 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
```

7.2 Login Page code

```
<html>
  <head>
    <title>Customer Care System</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"><
/script>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body class="bg-light">
    <nav class="navbar bg-light">
      <div class="container">
        <h1 style="color:Blue;">Customer </br> Care</h1>
      </div>
    </nav>
    <div class="container d-flex justify-content-center pt-5">
      <div class="card col-md-4 mb-4 mt-5">
        {% with messages = get_flashed_messages(with_categories=true) %} {% if
        messages %} {% for category, message in messages %}
        <div class="flashes alert alert-{{category}}">
          <strong>{{ message }}</strong>
        </div>
        {% endfor %} {% endif %} {% endwith %}

        <div class="card-body">
          <h4 class="card-title">Login</h4>
          <form method="post" action="/login">
            <div class="mb-3 mt-3">
              <label for="user_type" class="form-label">User Type:</label>
              <div class="dropdown">
                <select name="user_type" id="user_type" class="form-control">
                  <option value="user">User</option>
                  <option value="admin">Admin</option>
                  <option value="agent">Agent</option>
                </select>
              </div>
            </div>
```

```
            <div class="mb-3 mt-3">
              <label for="email" class="form-label">Email:</label>
              <input
                type="email"
                class="form-control"
                id="email"
                placeholder="Enter email"
                name="username"
              />
            </div>
            <div class="mb-3">
              <label for="pwd" class="form-label">Password:</label>
              <input
                type="password"
                class="form-control"
                id="pwd"
                placeholder="Enter password"
                name="password"
              />
            </div>
            <div class="form-check mb-3">
              <label class="form-check-label">
                <input
                  class="form-check-input"
                  type="checkbox"
                  name="remember"
                />
                Remember me
              </label>
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
            <a href="/user/signup" class="float-end">Register</a>
          </form>
        </div>
      </div>
    </div>
  </body>
</html>
```

7.3 Registration Page Code

```
<html>
<head>
    <title>Signup</title>
<meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"><
/script>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body class=" bg-light">
<nav class="navbar bg-light">
  <div class="container">
    <h1 style="color:Blue;">Customer </br> Care</h1>
  </div>
</nav>
<div class="container d-flex justify-content-center pt-5">
    <div class="card col-md-4 mb-4 mt-5">
        <div class="card-body">
            <h4 class="card-title">Register</h4>

             <form method="post" action="/user/signup">

                <div class="mb-3 mt-3">
                   <label for="user_type" class="form-label">User Type:</label>
                    <div class="dropdown">
                          <select name="user_type" id="user_type" class="form-
control" required>
                           <option value="user">User</option>
                           <option value="admin">Admin</option>
                           <option value="agent">Agent</option>
                           </select>
                       </div>
                 </div>
                <div class="mb-3 mt-3">
                   <label for="name" class="form-label">Name:</label>
```

```
                    <input type="name" class="form-control" id="name"
placeholder="Enter name" name="name" required>
                  </div>
                 <div class="mb-3 mt-3">
                    <label for="mobile" class="form-label">Mobile:</label>
                    <input type="text" class="form-control" id="mobile"
placeholder="Enter mobile" name="mobile" required>
                  </div>
                <div class="mb-3 mt-3">
                    <label for="email" class="form-label">Email:</label>
                    <input type="email" class="form-control" id="email"
placeholder="Enter email" name="email" required>
                  </div>
                 <div class="mb-3 mt-3">
                    <label for="password" class="form-label">Password:</label>
                    <input type="password" class="form-control" id="password"
placeholder="Enter password" name="password" required>
                  </div>

<button type="submit" class="btn btn-danger">Cancel</button>
<button type="submit" class="btn btn-success float-end"">Save</button>
                </form>
  </div>


  </div>

</body>
</html>
```

7.4 IBM Cloud Connection code

```
import ibm_db
import ibm_db_dbi
import pandas

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_hostname = "8e359033-a1c9-4643-82ef-
8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_port = "30120"
dsn_protocol = "TCPIP"
dsn_uid = "hjr98238"
dsn_pwd = "lodgcIj11yMvWl47"

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


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


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

    except:
        return 0


def view(query):
    conn = ibm_db.connect(dsn, "", "")
```

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

    return result

except:
    return ''
```

8. Testing

8.1 Test Cases

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

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

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

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

8.2 User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the **Customer Care Registry** project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	0	0	2	7
External	0	2	0	0	2
Fixed	12	11	35	45	103
Not Reproduced	0	5	0	0	5
Skipped	0	0	0	0	0
Totals	17	18	35	47	117

3. Test Case Analysis

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

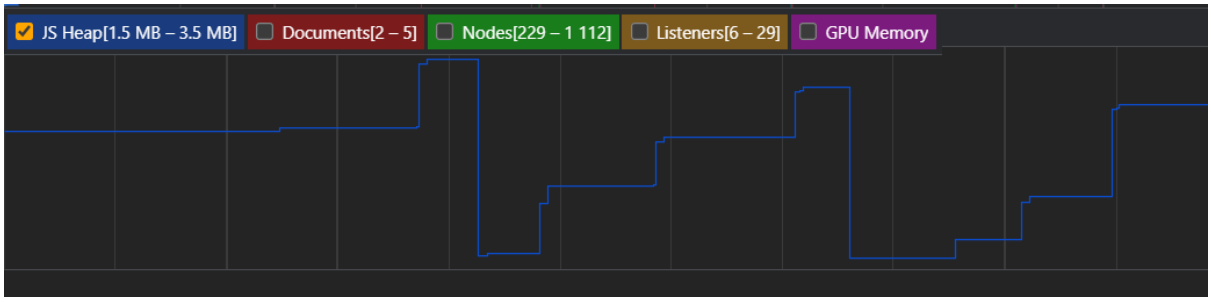
Section	Total Cases	Not Tested	Fail	Pass
Client Application	72	0	0	72
Security	7	0	0	7
Exception Reporting	5	0	0	5
Final Report Output	4	0	0	4

9. RESULTS

9.1 Performance Metrics

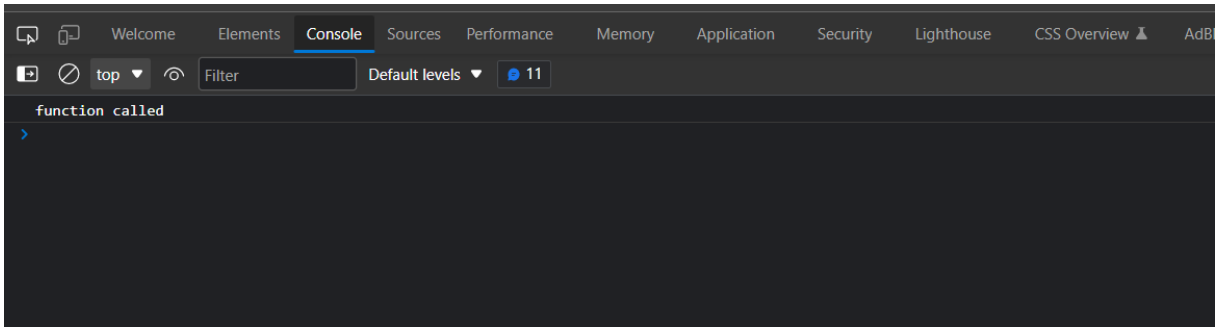
CPU usage:

- ✓ Since all the operations run using Flask is in server-side, the client (browser) need not worry about the CPU usage. Just rendering the page, static contents take place in the client-side.
- ✓ Memory for client-side functions (Javascript) is allocated using heap. It can be either increased based upon the requirement or removed from the heap.



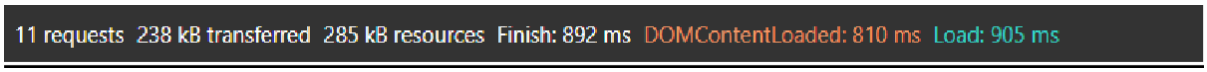
Errors:

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



Latency and Response time:

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



10. ADVANTAGES AND DISADVANTAGES

Advantages:

- ✓ Customers can clarify their doubts just by creating a new ticket
- ✓ Customer gets replies as soon as possible
- ✓ Not only the replies are faster, the replies are more authentic and practical
- ✓ Customers are provided with a unique account, to which the latter can login at any time
- ✓ Very minimal account creation process
- ✓ Customers can raise as many tickets as they want
- ✓ Application is very simple to use, with well-known UI elements
- ✓ Customers are given clear notifications through email, of all the processes related lo login, ticket creation etc.,
- ✓ Customers' feedbacks are always listened
- ✓ Free of cost

Disadvantages:

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

11. CONCLUSION

Thus, there are many customer service applications available on the internet. Noting down the structural components of those applications and we built a customer care registry application.

It will be a web application build with Flask (Python micro-web framework), HTML, JavaScript. It will be a ticket-based customer service registry.

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

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

12. FUTURE SCOPE

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

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

13. APPENDIX

Flask:

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

JavaScript:

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

IBM Cloud:

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

Kubernetes:

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

Docker:

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