**Project Report**

**Customer Care Registry**

**1. INTRODUCTION**

**1.1. Project Overview:**

Customer care registry is an application to help customers in processing their complaints. Customers can raise tickets with the description of the problem. An agent will be assigned to the ticket and will be tasked to solve the customer’s problem.

**1.2. Purpose:**

Companies today wants to satisfy their customers so that customers would be more inclined to visit their company for services next. For those companies that prioritizes customer satisfaction would prefer to use a customer care registry to keep track of complaints and solve it.

**2. LITERATURE SURVEY**

**2.1 Existing problem**

Customer support has been a separate department in almost any big company. But for small companies the amount of queries that they receive can overwhelm them. An application that can keep track of customers’ requests will be helpful solve most of the requests raised by customers.

**2.2 References:**

**2.2.1.** CUSTOMER CARE AND COMPLAINT REGISTRATION MANAGEMENT SYSTEM

**2.2.2.** NEED FOR SYSTEM - COMPLAINT REGISTRATION MANAGEMENT SYSTEM

**2.2.3.** CUSTOMER CARE AND COMPLAINT REGISTRATION MANAGEMENT SYSTEM PROJECT REPORT

**2.2.4.** Online Customer Care and Service Center Project in Java

**2.2.5.** Pooria Rashvand and Muhd Zaimi Abd Majid (2014) Critical Criteria on Client and Customer. Satisfaction for the Issue of Performance Measurement.

**2.2.6.** Zeljko M. Torbica and Robert C. Stroh(2001) Customer Satisfaction in Home Building.

**2.2.7.** Natalia Yanovaa(2015) Assessment of Satisfaction with the Quality of Education: Customer Satisfaction Index.

**2.2.8.** Jam Shahzaib Khan and Salim Khoso (2014) Strategic Management for Customer Satisfaction within Construction Projects.

**2.2.9.** Samira Dezhkam, Farzaneh Ghanbarpoor, Nazanin Shabani, Forooz Pishgar and Mahboobeh Ashoori (2013) The Impact of Product Innovation on Customer Satisfaction and Customer Loyalty.

**2.2.10**. Tunde Akinola Folorunso and Oluwaseyi Alabi Awodele(2015) Assessment of Clients Needs and Satisfaction at Various Stages of Building Projects Delivery

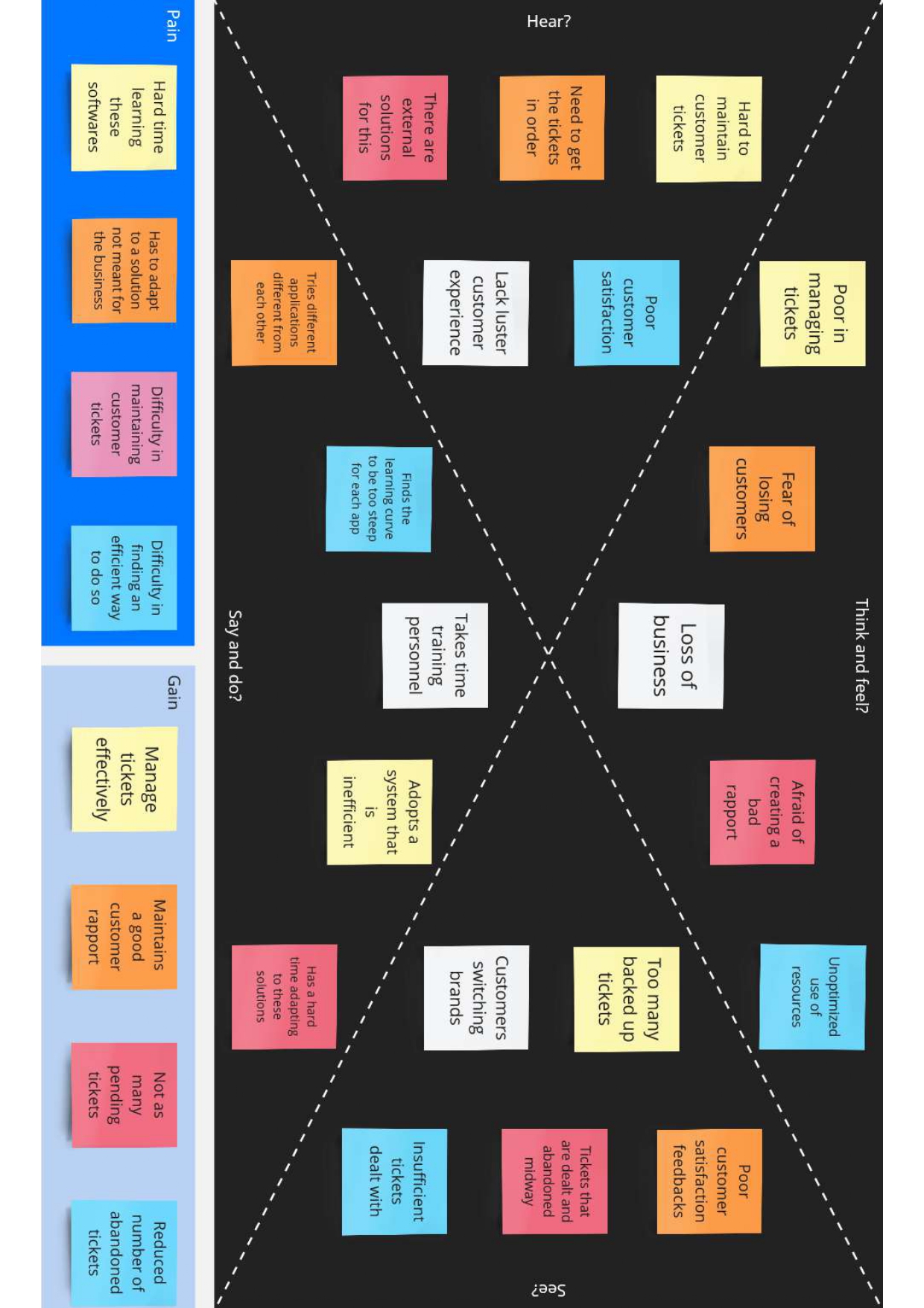
**2.2.11.** Ghaleb J. Sweis, Rana M. Imam, Ghaith M. Kassab and Rateb Sweis (2013) Customer Satisfaction in Apartment Buildings: The Case of Jordan.

**2.2.12.** Abayomi Omonori and Akinloye Lawal (2014) Understanding Customers Satisfaction in Construction Industry in Nigeria.

**2.2.1**3. Abdulhamid Shebob, Amit Mhalas, and Raj Kapur Shah (2013) A Review of Customer Satisfaction Factors in Libyan Housing Projects.

**2.3. Problem Statement Definition**

Customer care industry is a big industry, but not all problems could be solved. So to help this we create an app to help keep track of the requests until they are solved.



**3. IDEATION & PROPOSED SOLUTION**

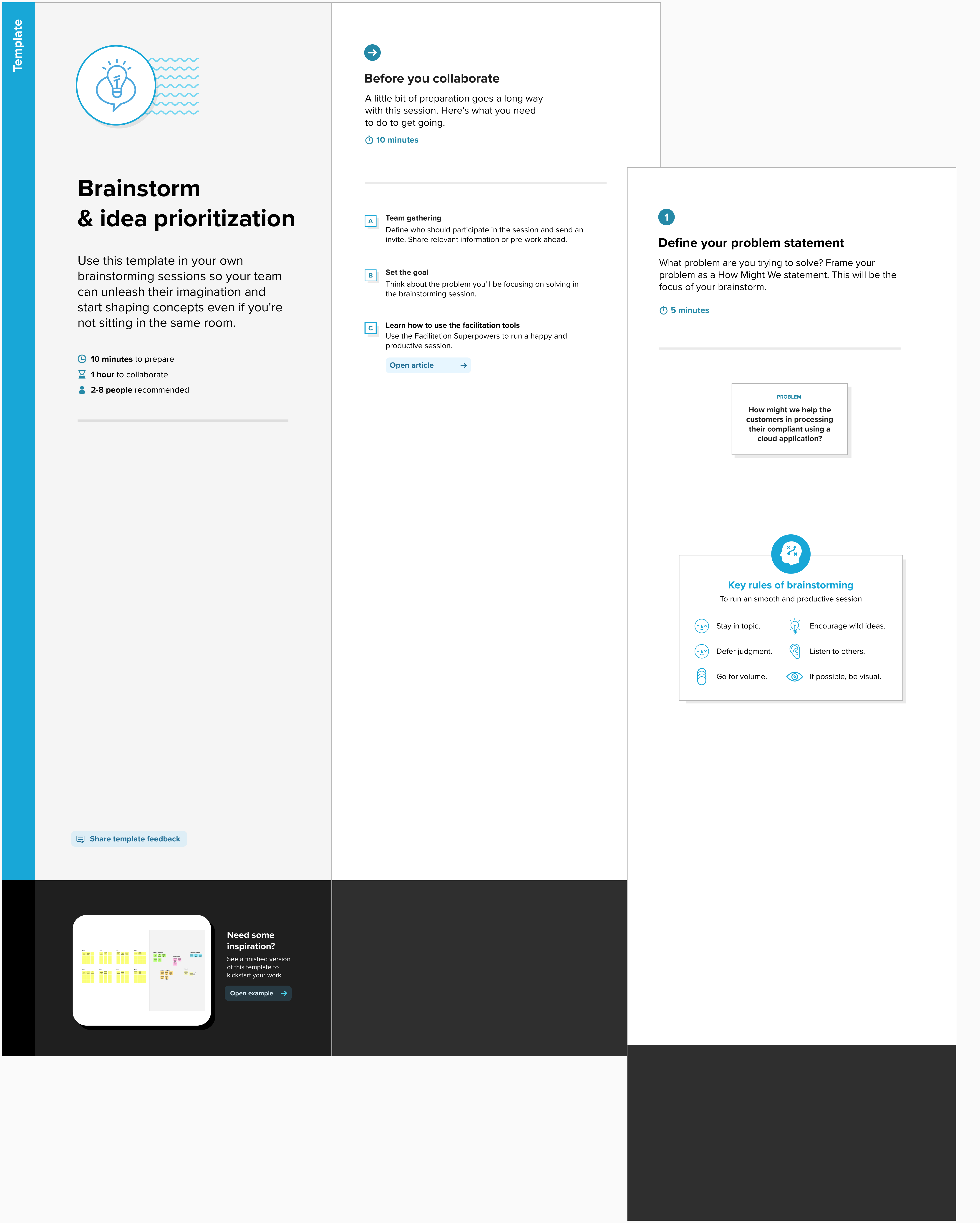
**3.1. Empathy Map Canvas**

**3.2. Ideation & Brainstorming**

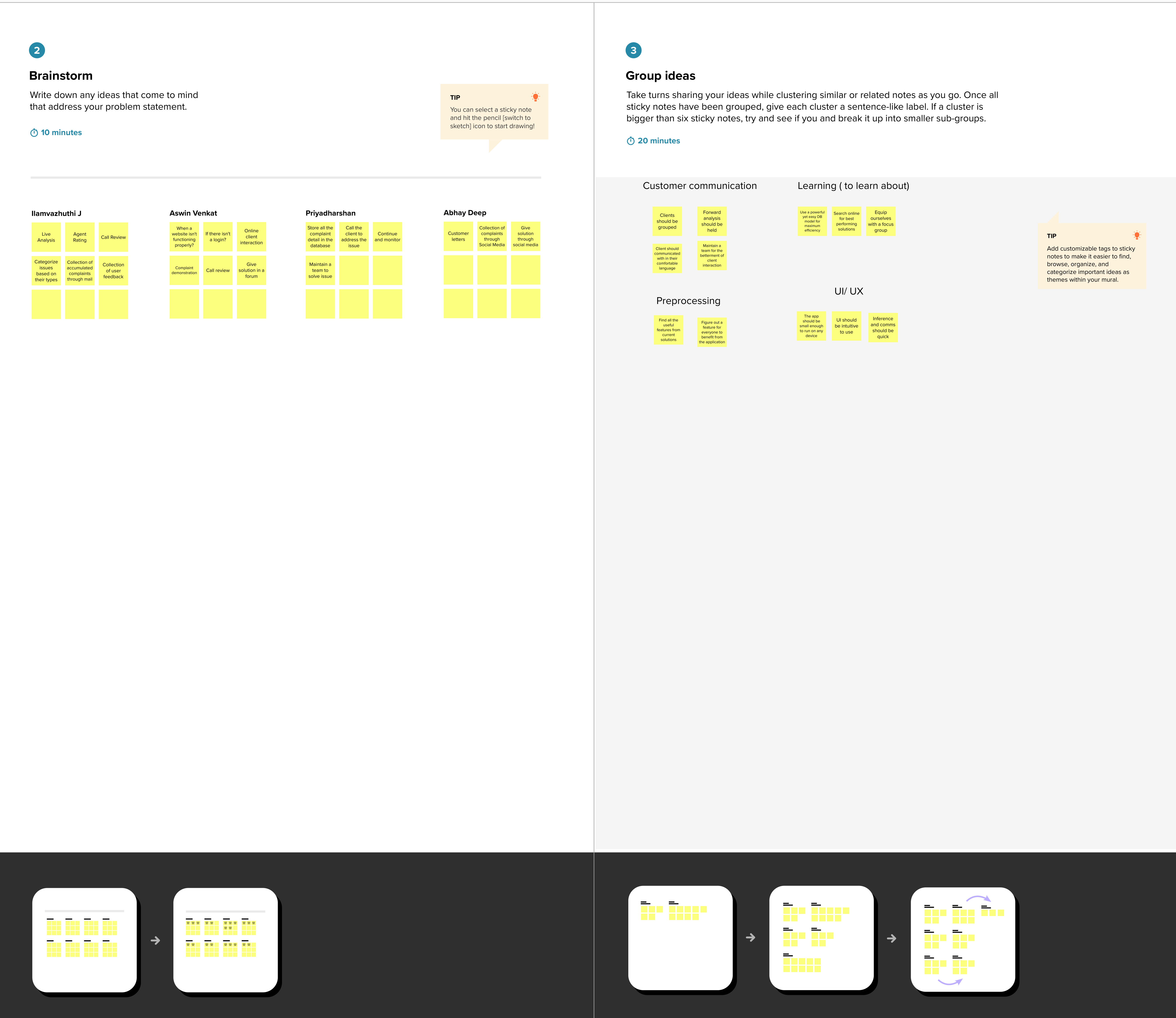
**Reference:**

<https://app.mural.co/t/kumaraguru8114/m/>[kumaraguru8114/1667246740963/7df12b9a1a7600a4e328faaa886f118f89e4339f?sender=udd2bbcd686c2cd4fefc55238](https://app.mural.co/t/kumaraguru8114/m/kumaraguru8114/1667246740963/7df12b9a1a7600a4e328faaa886f118f89e4339f?sender=udd2bbcd686c2cd4fefc55238)

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



**Step-2:** Brainstorm, Idea Listing and Grouping



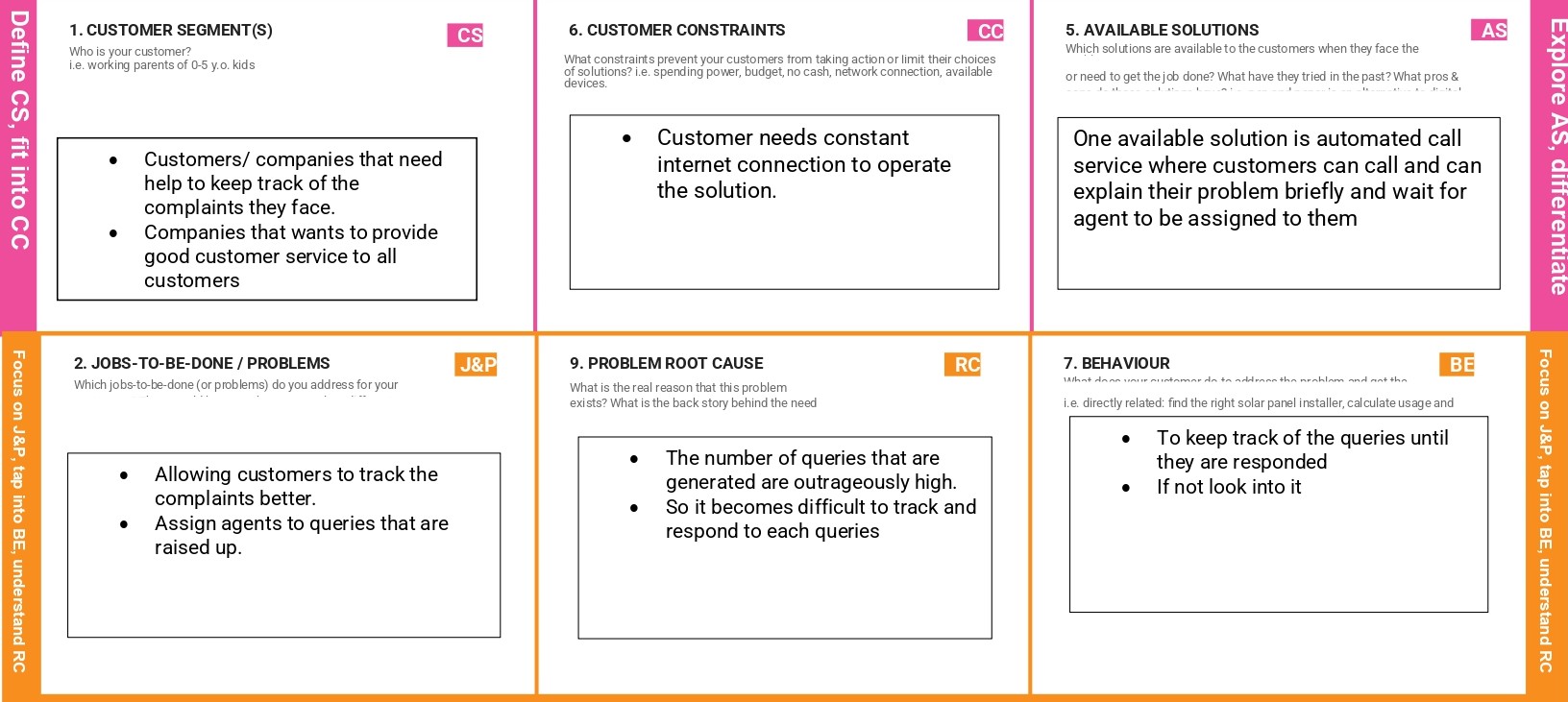
**Step-3:** Idea Prioritization:

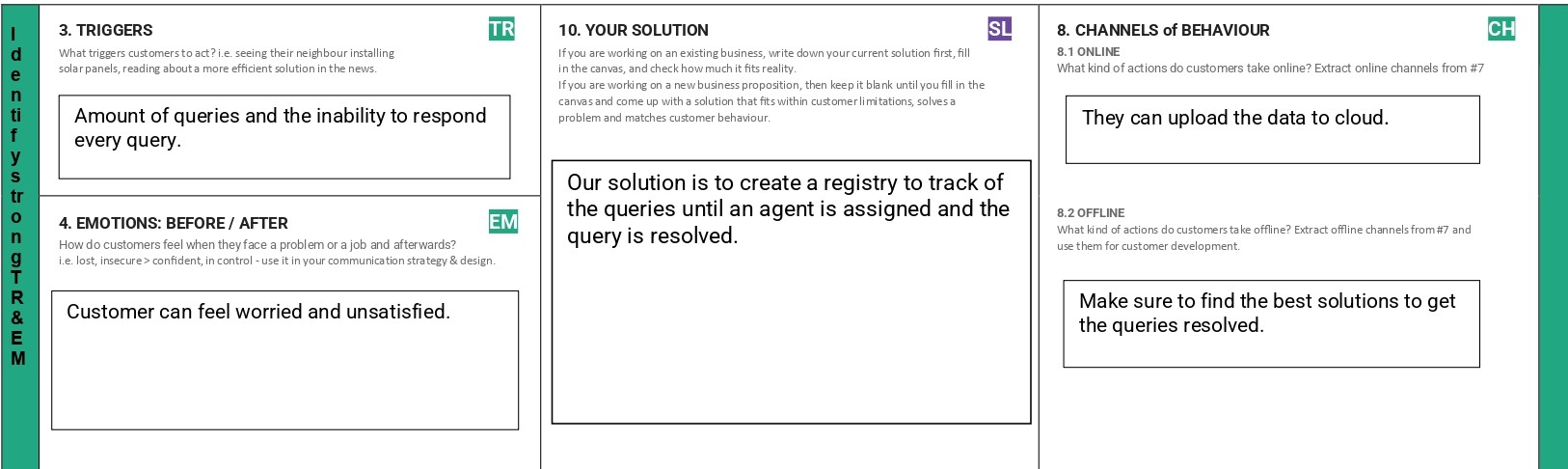


**3.3.** **Proposed Solution:**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Parameter** | **Description** |
|  | Problem Statement (Problem to be solved) | Customer care service plays a big part in any company’s business. They need to satisfy their customers by responding to their needs and queries. But the sheer number of queries generated makes it difficult for the staffs to respond to all the queries produced by several customers. |
|  | Idea / Solution description | To create a customer care registry that keeps track of very query until they are responded by a staff. |
|  | Novelty / Uniqueness | This is not a particularly unique idea as many people tried to develop it. Maybe not the exact same idea. |
|  | Social Impact / Customer Satisfaction | The company will be satisfied if they can satisfy most of their customers through their customer care service. |
|  | Business Model (Revenue Model) | We could provide this in a service based model and can charge the companies for maintaining their customer care registries. |
|  | Scalability of the Solution | The solution can be uniquely tailored to each company and can support the growing customer base of the company. |

**3.4. Problem Solution fit:**





**4. REQUIREMENT ANALYSIS**

**4.1. Functional Requirements:**

Following are the functional requirements of the proposed solution.

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

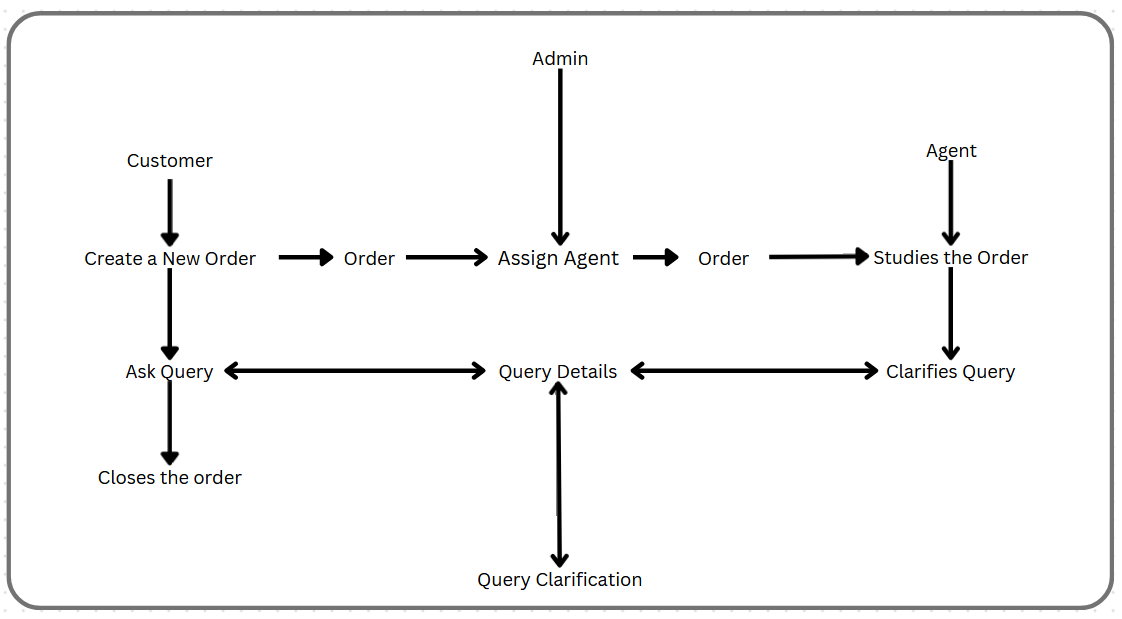
**4.2. Non-functional Requirements:**

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

|  |  |  |
| --- | --- | --- |
| **Fr NO** | **Non-Functional Requirements(Epic)** | **Sub Requirements(Story/Sub-Task)** |
| **NFR-1.** | Usability | To provide the solution to the problem |
| **NFR-2.** | Security | Track of login authentication |
| **NFR-3.** | Reliability | Tracking of decade status through Email |
| **NFR-4.** | Performance | Effective Development of Web Application |
| **NFR-5.** | Availability | 24/7 service |
| **NFR-6.** | Scalability | Agents scalability as per the number of customers |

**5. PROJECT DESIGN**

**5.1. Data Flow Diagrams**



**5.2.Solution & Technical Architecture**

**5.2.1. Solution Architecture:**

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

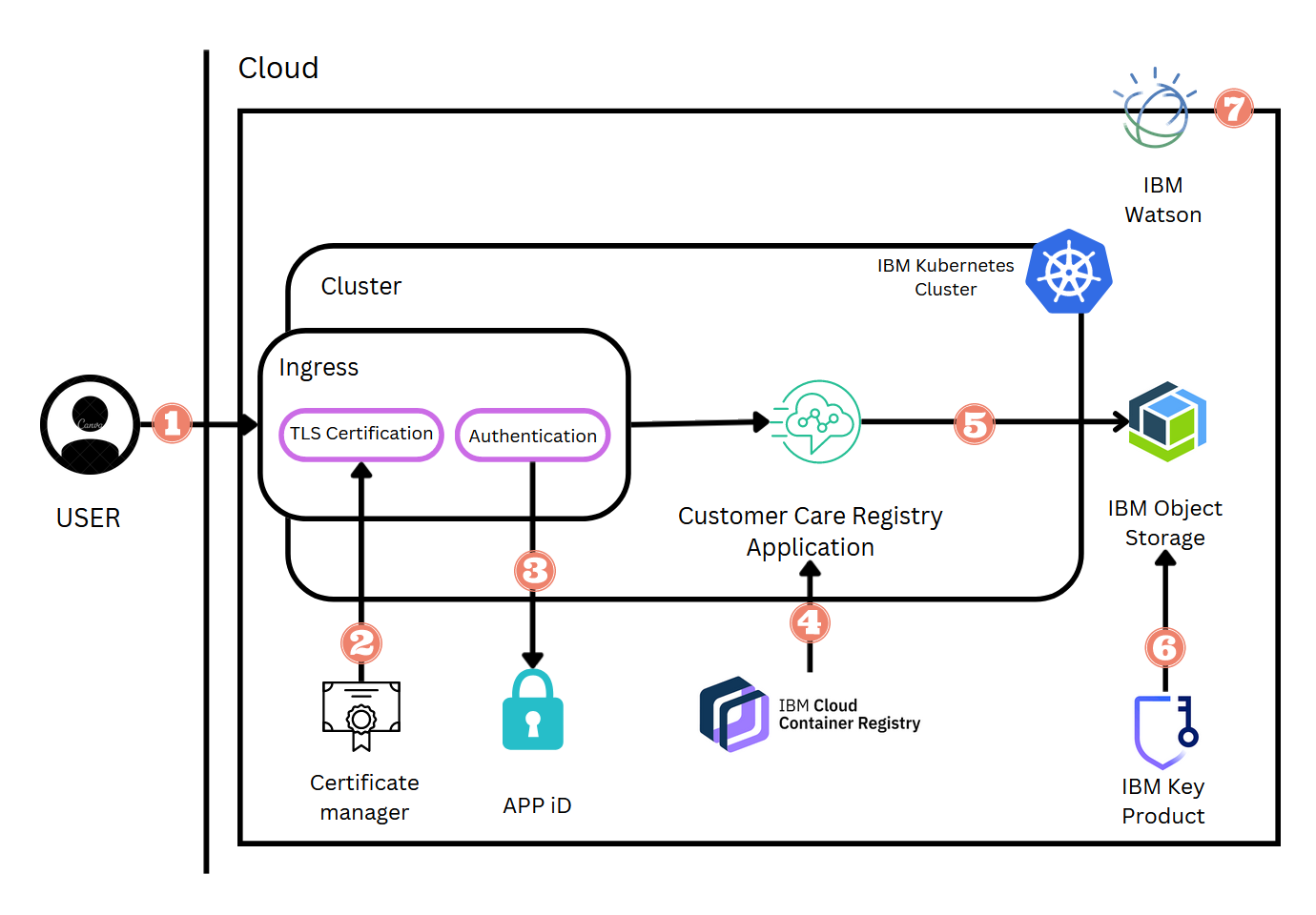
• Find the best tech solution to solve existing business problems.

• Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.

• Define features, development phases, and solution requirements.

• Provide specifications according to which the solution is defined, managed and delivered.

**5.2.2. Technical Architecture:**



**5.3. User Stories:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requirement (Epic)** | **User Story NO** | **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 log into the application by entering correct email 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 conversation with the assigned agent and get my query classified. | My queries are clarified. | High | Sprint-3 |
| Forgot password | USN-6 | As a customer, 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 |
| Order details | USN-7 | As a Customer, I can see the current stats of order. | I get a better 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 doubts | 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 |
| Login | USN-1 | As an admin, I can login to the application 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 PLANNING & SCHEDULING**

**6.1. Sprint Planning & Estimation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Functional**  **Requirement (Epic)** | **User Story**  **Number** | **User Story / Task** |
| Sprint-1 | User Panel | USN-1 | The user will login into the website and go through the services available on the web page |
| 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 |
| Sprint-3 | Chat Bot | USN-3 | The user can directly talk to Chat bot regarding the services. Get the recommendations based on information provided by the user. |
| Sprint-4 | final delivery | USN-4 | Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application |

**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** | **Total Story Points** |
| Sprint-1 | 20 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 20 | 20 |
| Sprint-2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 20 | 20 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 20 |
| Sprint-4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 20 | 20 |

**7. CODING & SOLUTION (Explain the features added in the project along with code)**

**7.1 Feature 1:**

Different Types of Users are able to login/register from the same webpage.

**Code:**

<div class="content-section">

        <form method="POST" action="">

            {{ form.hidden\_tag() }}

            <fieldset class="form-group">

                <legend class="border-bottom mb-4">Log In</legend>

                <div class="form-group">

                    {{ form.email.label(class="form-control-label") }}

                    {% if form.email.errors %}

                        {{ form.email(class="form-control form-control-lg is-invalid") }}

                        <div class="invalid-feedback">

                            {% for error in form.email.errors %}

                                <span>{{ error }}</span>

                            {% endfor %}

                        </div>

                    {% else %}

                        {{ form.email(class="form-control form-control-lg") }}

                    {% endif %}

                </div>

                <div class="form-group">

                    {{ form.password.label(class="form-control-label") }}

                    {% if form.password.errors %}

                        {{ form.password(class="form-control form-control-lg is-invalid") }}

                        <div class="invalid-feedback">

                            {% for error in form.password.errors %}

                                <span>{{ error }}</span>

                            {% endfor %}

                        </div>

                    {% else %}

                        {{ form.password(class="form-control form-control-lg") }}

                    {% endif %}

                </div>

                <div class="form-check">

                    {{ form.remember(class="form-check-input") }}

                    {{ form.remember.label(class="form-check-label") }}

                </div>

            </fieldset>

            <div class="form-group">

                {{ form.submit(class="btn btn-outline-info") }}

                <small class="text-muted ml-2">

                    <a href="{{url\_for('reset\_request')}}">Forgot Password?</a>

                </small>

            </div>

        </form>

    </div>

**7.2 Feature 2:**

Different Users are directed to their respective dynamic home pages base on their user role.

**Code:**

<div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav ml-auto">

                    {% if current\_user.is\_authenticated %}

                    <li class="nav-item active">

                        <a class="nav-link" href="{{url\_for('home')}}">Home</a>

                    </li>

                    <li>

                        <a class="nav-link" href="{{url\_for('account')}}">Account</a>

                    </li>

                    <li>

                        <a class="nav-link" href="{{url\_for('new\_ticket')}}">New Ticket</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="{{url\_for('logout')}}">Logout</a>

                    </li>

                    {% else %}

                    <li class="nav-item">

                        <a class="nav-link" href="{{url\_for('index')}}">About us</a>

                    </li>

                    <li>

                        <a class="nav-link" href="{{url\_for('register')}}">Register</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="{{url\_for('login')}}">Login</a>

                    </li>

                    {% endif %}

                </ul>

            </div>

**7.3 Feature 3:**

The webpages for Ticket Creation, Updating Tickets and Assigning Agents to tickets are only available to the Users with respective User roles.

**Code:**

{% if current\_user.userrole == 'Admin'%}

            <div>

                <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{url\_for('assign\_ticket', ticket\_id = ticket.id )}}">Assign Agent</a>

                <button type="button"

                    class="btn btn-danger btn-sm m-1" data-toggle='modal' data-target='#deleteModal'>Delete</button>

            </div>

            {% endif %}

            {% if current\_user.userrole == 'Agent' %}

            <div>

                <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{url\_for('update\_ticket', ticket\_id = ticket.id )}}">Update</a>

                <button type="button"

                    class="btn btn-danger btn-sm m-1" data-toggle='modal' data-target='#deleteModal'>Delete</button>

            </div>

            {% endif %}

**7.4 Feature 4:**

The Author/Customer receives a ticket whenever a ticket is updated based on its status or based on its completion.

**Code:**

def ticket\_status\_email(ticket, user):

    message = Mail(

    from\_email='ilamvazhuthi.j@gmail.com',

    to\_emails= user.email,

    subject = 'Ticket Status Update',

    html\_content='<p>The status of your ticket has been updated to<br>{}<br>This is an automated messeage.<br><br><br>For any queries contact our customer support</p>'.format(ticket.status))

    try:

        sg = SendGridAPIClient("SG.Oyc8TVoqRKKdZEg\_hbEI3A.\_bX8I1Wklmlhw1Vkw62KACRu2kkEYePEJBNvM1FZJy8")

        response = sg.send(message)

        print(response.status\_code)

        print(response.body)

        print(response.headers)

    except Exception as e:

        print(e)

**7.5 Feature 5:**

Users are able to reset their password only with their registered Mail\_ID and this mail is verified using Serializers.

**Code:**

def send\_reset\_email(user):

    token = user.get\_reset\_token()

    message = Mail(

    from\_email='ilamvazhuthi.j@gmail.com',

    to\_emails= user.email,

    subject='Password Reset Request',

    html\_content=f'''To reset your password, visit the following link:

    {url\_for('reset\_token', token = token, \_external = True)}

    If you did not make this request then simply ignore ths email and no changes will be made

    ''')

    try:

        sg = SendGridAPIClient("SG.Oyc8TVoqRKKdZEg\_hbEI3A.\_bX8I1Wklmlhw1Vkw62KACRu2kkEYePEJBNvM1FZJy8")

        response = sg.send(message)

        print(response.status\_code)

        print(response.body)

        print(response.headers)

    except Exception as e:

        print(e)

**7.6 Feature 6:**

The entire application is base out of a single layout.html such that new pages can be created with ease.

**Code:**

{% extends "layout.html" %}

{% block body %}

{Content for the Current Page}

{%endblock body%}

**7.7 Feature 7:**

The Application is featured in such a way that even if the browser is able to cache any page of the application no data of significance is actually revealed.

**Code:**

@app.route("/index")

def index():

    if current\_user.is\_authenticated:

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

    return render\_template('index.html')

@app.route("/logout")

def logout():

    logout\_user()

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

**8. TESTING**

**8.1 Test Cases**

**8.2 User Acceptance Testing**

**9. RESULTS**

**9.1 Performance Metrics**

# **9.1.1. 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 | 0 | 0 | 1 | 0 | 1 |
| Duplicate | 0 | 1 | 2 | 1 | 5 |
| External | 2 | 3 | 0 | 1 | 6 |
| Fixed | 1 | 3 | 3 | 3 | 10 |
| Not Reproduced | 0 | 0 | 0 | 1 | 1 |
| Skipped | 1 | 0 | 0 | 1 | 2 |
| Won't Fix | 0 | 1 | 2 | 1 | 4 |
| Totals | 3 | 8 | 8 | 8 | 29 |

# **9.1.2. Test Case Analysis**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Total Cases** | **Not Tested** | **Fail** | **Pass** |
| Print Engine | 1 | 0 | 0 | 1 |
| Client Application | 20 | 0 | 0 | 20 |
| Security | 2 | 0 | 0 | 2 |
| Outsource Shipping | 1 | 0 | 0 | 1 |
| Exception Reporting | 3 | 0 | 0 | 3 |
| Final Report Output | 4 | 0 | 0 | 4 |
| Version Control | 2 | 0 | 0 | 2 |

**10. ADVANTAGES & DISADVANTAGES**

**Advantages:**

1. The UI has been developed in such a way that layers of updates and features can be added on without having to mess around too much with previous iterations of the code.
2. The Back-end of the application is pretty much optimized to have the best performance regardless of the limitations of the device being used to access it.
3. The application has been secured using the liberalizers and several authentication factors such that no unauthorized personnel could actually tamper with the data.
4. The application also has this minimalist theme with more or less an industrial base to make it easier to use.
5. The application also has been created with the intent of not missing any of the customer requests nor the agent request.

**Disadvantages:**

1. To the untrained eye the UI could feel a bit lacklustre.
2. Some people might find the application boring to interact with while others would prefer that the application adheres to the simple goal of Customer Care.
3. Due to multiple authentication on every request, the application might feel sluggish occasionally
4. The application might be overwhelmed with a wave of tickets and users which can cause the application to be bottlenecked sooner than later.

**11. CONCLUSION**

We have studied the problem statement carefully and looked at existing solution and decided to proceed with the application navigation in such a way that a User Generated Ticket cannot be flushed out of the system unless it is marked as completed. Our application has met all the criteria for the app we intended to create and it has passed all the Test and exceeded expectations for meeting their match with the existing application. The application has an integrated Chat Bot

**12. FUTURE SCOPE**

Some Variation of Applicable can be tried to be actually implemented if the aforementioned disadvantages are fine-tuned. We can also add more features and extend the application from a simple Customer Care Registry and made into an enterprise level application. The user interface of the application can also be improved by adding more design elements based on user feedback which we were not able to do due to the fact of it being such a short sprint.

**13. APPENDIX**

Due to the sheer size of the code and the amount of files in it we have attached the GitHub repository link down below along with the link for the Video demo/presentation.

**GitHub:**

**Project Demo Link:**