**CUSTOMER CARE REGISTRY**

**TEAM DETAILS**

**TEAM NO :** PNT2022TMID43073

**COLLEGE NAME :** SREE SAKTHI ENGINEERING COLLEGE

**DEPARTMENT:** COMPUTER SCIENCE & ENGINEERING

**TEAM LEADER:** SAMSUSHIEKABDULLA S

**TEAM MEMBER 1:**MOUNIKA K

**TEAM MEMBER 2:**SANJAY M

**TEAM MEMBER 3:**KEERTHANA R

**TABLE CONTENT**

**1. INTRODUCTION**

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 Reference

2.3 Problem Statement Definition

1. **IDEATION & PROPOSED SOLUTION**

3.1Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

**4.REQUIREMENT ANALYSIS**

4.1 Functional requirement

4.2 Non-Functional requirements

**5.PROJECT DESIGN**

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

**6.PROJECT PLANNING & SCHEDULING**

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

**7. CODING & SOLUTIONING (Explain the featuresadded in the project along with code)**

7.1Feature 1

7.2 Feature 2

**8.TESTING**

8.1 Test Cases

8.2 User Acceptance Testing

**9.RESULTS**

Performance Metrics

**10. ADVANTAGES & DISADVANTAGES**

**11. CONCLUSION**

**12.FUTURE WORK**

**13.APPENDIX**

Source Code

Github& project demo link

**1 INTRODUTION**

Aiming to discover the most suitable service cater to the discovery request of service consumer which includes functional requirements and nonfunctional requirements, this paper proposes a service registry model named as SRC (Service Registry on Cloud) which is an extension of the keywords based service registry model and deployed as a cloud application to provide behavior-aware and QoS-aware service discovery services. SRC stores the semantic descriptors of Web Services and the feedbacks of dynamic status of QoS of Web Services as GFS files in a cloud, and uses MapReduce mechanism to process these files. The running results of an instance of SRC deployed in an experimental environment have shown that SRC is effective and feasible.

**1.1 PROJECT OVERVIEW**

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.

**1.2 PURPOSE**

The customer service team is the face of the organization and the frontline when customers require assistance. Customer service agents help customers pay bills, review or make changes to accounts, handle returns and answer frequently asked questions.

**2 LITERATURE SURVEY**

Establishing User-centric Cloud Service Registries Many potential cloud consumers are overburdened by the challenges persisting when discovering, assessing, and selecting contemporary Cloud Service offerings: the cloud market is vast and fast-moving, the selection criteria are ambiguous, service knowledge is scattered through the Internet, and features as well as prices are complex and incomparable.

**2.1 EXISTING PROBLEM**

* Product Quality affect Customer Satisfaction
* Quality of Service affect Customer Satisfaction
* Product Quality affect Complaint Level
* The Quality of Service affect the Level of Complaints
* Customer Satisfaction affect Complaint Levels

**2.2 REFERENCES**

* H., Narulita, E., & Nurmahdi, A. (2018b). The Influence of Service Quality, Brand Image and Promotion on Purchase Decision at MCU Eka Hospital. Saudi Journal of Business and Management Studies.
* <https://doi.org/10.21276/sjbms.2018.3.1.12> Anggita, R., & Ali, H. (2017). The Influence of Product Quality, Service Quality and Price to Purchase Decision of SGM Bunda Milk (Study on PT. Sarihusada Generasi Mahardika Region Jakarta, South Tangerang District). Scholars Bulletin. <https://doi.org/10.21276/sb>.
* Brata, B. H., Husani, S., & Ali, H. (2017). The Importance of Quality Products, Price, Promotion, and Location to Product Purcese Decision on Nitchi At PT. Jaya Swarasa Agung in Central Jakarta. Saudi Journal of Business and Management <https://doi.org/10.21276/sjbms>.

**2.3 Problem statement definition**

**Problem Statement:**

I am sheik 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.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement (PS)** | **Iam (Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | Regular Customer | Purchase products online | I don’t get proper reviews | They are not from real experts. | Frustrated |
| PS-2 | Regular Customer | Bought a product | I cannot get my doubts clarified | There is no proper system | Disappointed |
| PS-3 | Regular customer | Raise queries about a product | I am getting invalid answers /  replies are too late | Replies are from unauthenticated  persons | Stupid |

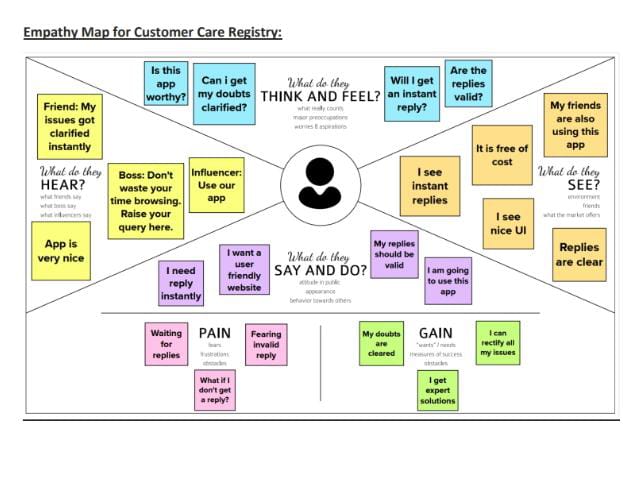
**3. IDEATION & PROPOSED SOLUTION**

**3.1 Empathy Map Canvas:**

• An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviours and attitudes.

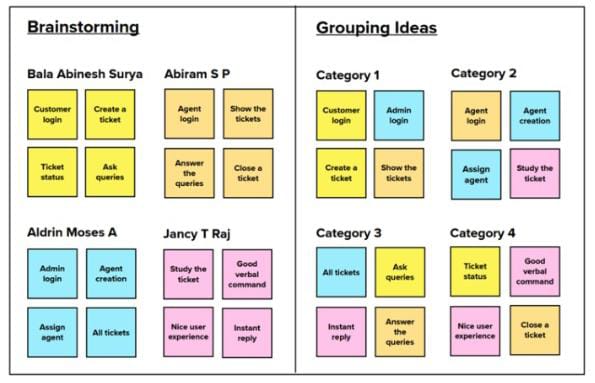
• It is a useful tool to helps teams better understand their users. • Creating an effective solution requires understanding the true problem and the person who is experiencing it.

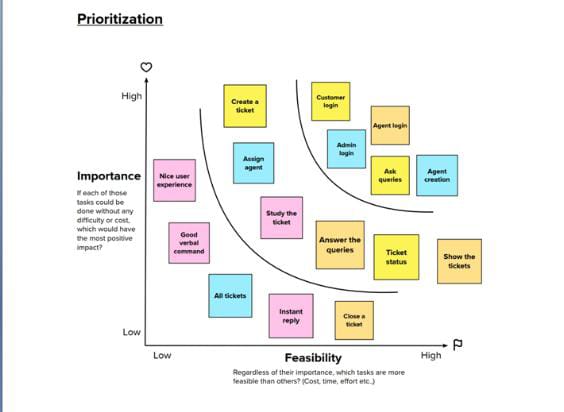
• The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.



**3.2 IDEATION & 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.

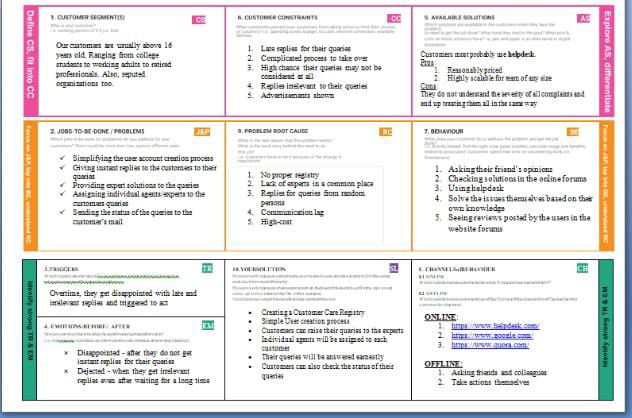




**3.3 Proposed solution**

|  |  |  |
| --- | --- | --- |
| **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. |
| 2. | 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 issues. |
| 3. | Novelty / Uniqueness | The agents are experts in the product domain and they will communicate well with the customers |
| 4. | Social Impact / Customer Satisfaction | Customers will be satisfied with the instant and valid replies. Also, it creates a doubtless society, that boosts sales. |
| 5. | 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. |
| 6. | Scalability of the Solution | This idea is so much use to the customers that the latter may refer this registry to their friends and colleagues at work. Naturally, the user base grows so does the number of queries answered.  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?”
* It is mandatory
* 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 - OPEN, AGENT ASSIGNED,  IN PROCESS, COMPLETE, CLOSED |
| FR-11 | Address Column | Agent clarifies the doubts of the customer |

**4.2 Non-functional Requirements:**

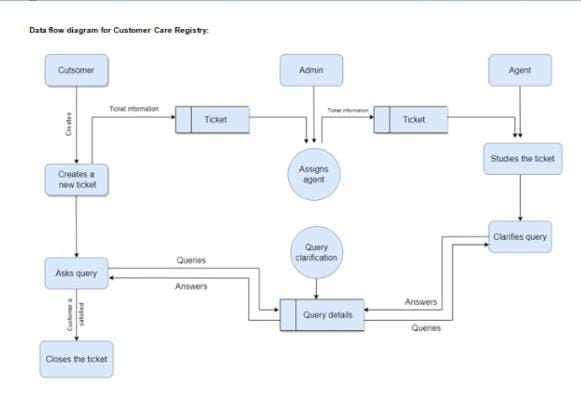
* A non-functional requirement defines the quality attribute of a software system
* It places constraint on “How should the software system fulfil the functional requirements?"
* It is not mandatory
* Applied to system as a whole
* Usually more difficult to define
* Helps you verify the performance of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

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | Customers can use the application in almost all the webbrowsers. Application is with good looking and detailed UI, whichmakes 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. |

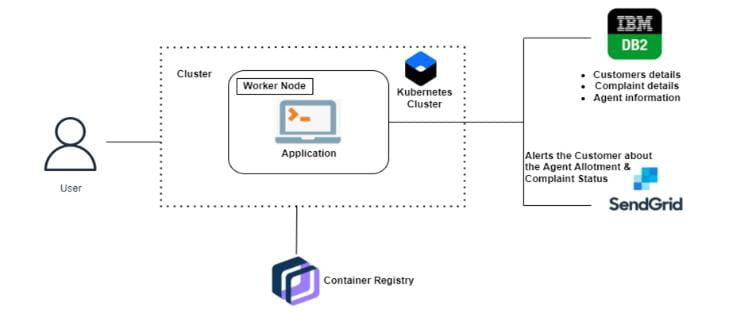
**5. PROJECT DESIGN**

**5.1 DATA FLOW DIAGRAM**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the rightamount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



**5.2 Solution & technical architecture**



**5.3 USER STORIES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **UserType** | **Functional**  **Requirement(Epic)** | **User**  **Story**  **no** | **UserStory/Task** | **Acceptance**  **criteria** | **Priority** | **Release** |
| Customer  (Webuser) | Registration | USN-1 | Asacustomer,Icanregisterfortheapplicationbyentering myemail, password, and confirmingmypassword. | I can access my account /dashboard | High | Sprint-1 |
|  | Login | USN-2 | As a customer, I can login to the application by entering correct email and password. | I can access my account / dashboard | High | Sprint-1 |
|  | Dashboard | USN-3 | Asacustomer, Ican see all the tickets raised by me and lot more. | Iget all the info needed in my dashboard | High | Sprint-1 |
|  | Ticket creation | USN-4 | Asacustomer, Icancreate a new ticket with the detailed description of my query. | I can ask my query | High | Sprint-2 |
|  | Address Column | USN-5 | Asacustomer,Icanhave 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 |
|  | Ticket details | USN-7 | As a customer, I can see the current status of my tickets. | I get 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 all the tickets  assigned to me by the 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/her queries. | 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  (Web user) | 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 tickets  raised in the entire system and lot more. | I can assign agents by seeing those tickets | High | Sprint-1 |
|  | Agent creation | USN-3 | As an admin, I can create an agent for clarifying the customer’s queries. | I can create agents | High | Sprint-2 |
|  | Assigning agent | USN-4 | As an admin, I can assign an agent for each ticket created by the customer. | Enables agent to clarify  the queries | High | Sprint-2 |
|  | Forgot password | USN-4 | As an admin, 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 |

**6 PROJECT PLANNING &SCHEDULING**

**6.1 SPRINT PLANNING &ESTIMATION**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **User Type** | **Functional Requirement (Epic)** | **User Story no** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-1 | Customer  (Web User) | Registration | USN-1 | As a customer, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Samsu sheik abdulla,  Sanjay |
| Sprint-1 |  | Login | USN-2 | As a customer, I can login to the application by entering correct email and password. | 1 | High | keerthana, Mounika |
| Sprint-1 |  | Dashboard | USN-3 | As a customer, I can see all the tickets raised by me and lot more. | 3 | High | Samsu sheik abdulla |
| Sprint-2 |  | Ticket creation | USN-4 | As a customer, I can create a new ticket with the detailed description of my query. | 2 | High | Samsu sheik abdulla |
| Sprint-3 |  | Address Column | USN-5 | As a customer, I can have conversations with the assigned agent and get my queries clarified | 3 | High | Mounika, Samsu sheik abdulla |
| Sprint-4 |  | Forgot password | USN-6 | As a customer, I can reset my password by this option in case I forgot my old password | 2 | Medium | Mounika, Sanjay |
| Sprint-4 |  | Ticket details | USN-7 | As a customer, I can see the current status of my tickets | 2 | Medium | Samsu sheik abdulla, Sanjay |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-3 | Agent  (Web user) | Login | USN-1 | As an agent, I can login to the application by entering correct email and password | 2 | High | keerthana |
| Sprint-3 |  | Dashboard | USN-2 | As an agent, I can see all the tickets  assigned to me by the admin | 3 | High | Sanjay |
| Sprint-3 |  | Address Column | USN-3 | As an agent, I get to have conversations with the customer and clear his/her queries | 3 | High | Samsu sheik abdulla, mounika |
| Sprint-4 |  | Forgot password | USN-4 | As an agent, I can reset my password by this option in case I forgot my old password | 2 | Medium | keerthana, Samsu sheik abdulla |
| Sprint-1 | Admin  (Web user) | Login | USN-1 | As an admin, I can login to the application by entering correct email and password | 1 | High | Sanjay, keerthana |
| Sprint-1 |  | Dashboard | USN-2 | As an admin, I can see all the tickets  raised in the entire system and lot more | 3 | High | mounika |
| Sprint-2 |  | Agent creation | USN-3 | As an admin, I can create an agent for clarifying the customer’s queries | 2 | High | mounika |
| Sprint-2 |  | Assigning agent | USN-4 | As an admin, I can assign an agent for each ticket created by the customer | 3 | High | keerthana, Sanjay |
| Sprint-4 |  | Forgot password | USN-4 | As an admin, I can reset my password by this option in case I forgot my old password | 2 | Medium | mounika, keerthana |

**6.2 Project Tracker, Velocity & Burndown Chart:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 10 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 10 | 29 Oct 2022 |
| Sprint-2 | 7 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 7 | 05 Nov 2022 |
| Sprint-3 | 11 | 4 Days | 06 Nov 2022 | 11 Nov 2022 | 11 | 09 Nov 2022 |
| Sprint-4 | 8 | 4 Days | 10 Nov 2022 | 15 Nov 2022 | 8 | 13 Nov 2022 |

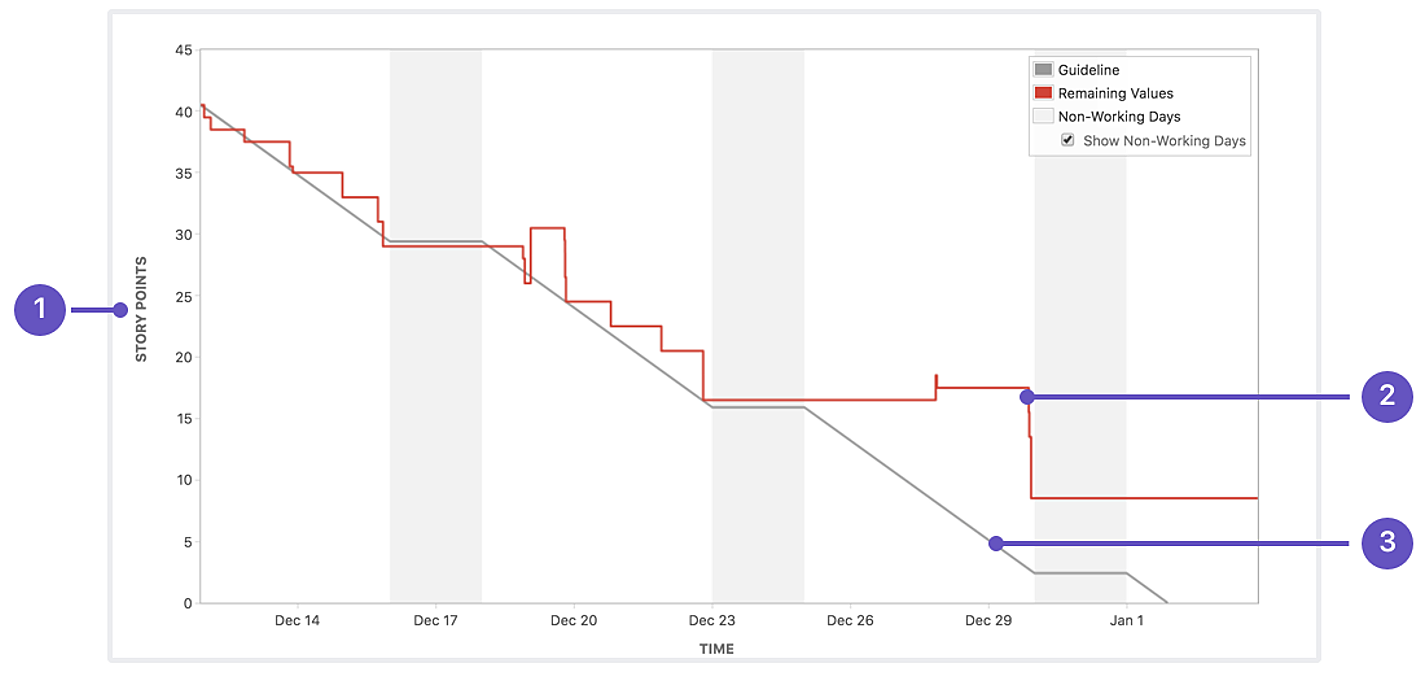
**Velocity:**

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)

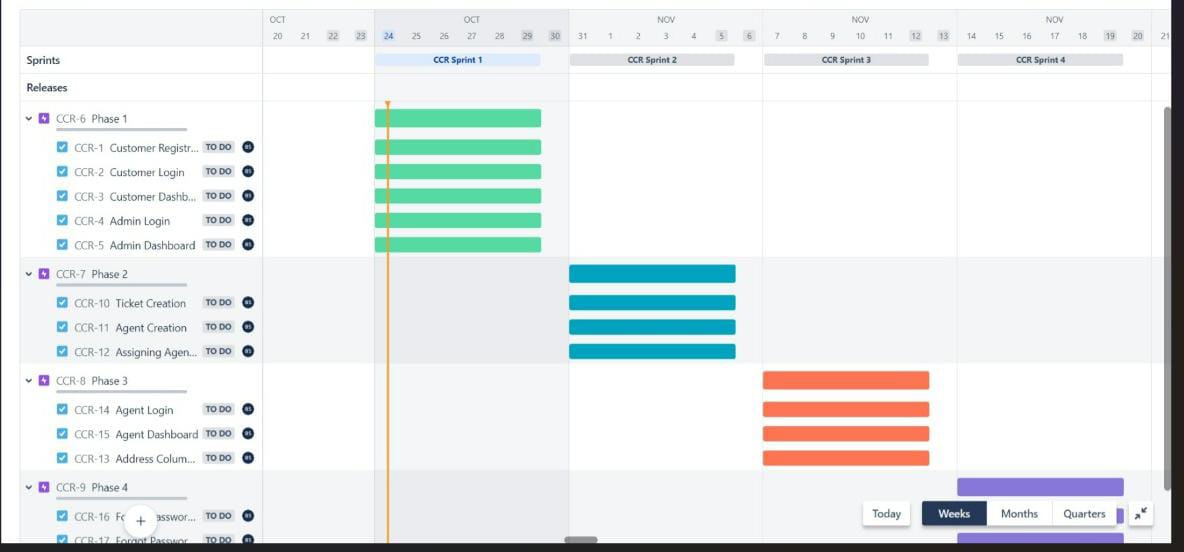
|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Total Story Points** | **Duration** | **Average Velocity** |
| Sprint 1 | 10 | 6 Days | 10/6 = 1.66 |
| Sprint 2 | 7 | 6 Days | 7/6 = 1.16 |
| Sprint 3 | 11 | 4 Days | 11/4 = 2.75 |
| Sprint 4 | 8 | 4 Days | 8/4 = 2 |

**BURNDOWN CHART:**

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile [software development](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/) methodologies such as [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/). However, burn down charts can be applied to any project containing measurable progress over time.



**6.3 REPORTS FROM JIRA**



**7. CODING & SOLUTION**

**7.1 Feature 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **COMPONENT** | **DESCRIPTION** | **TECHNOLOGY** |
| **1.** | User Interface | How user interacts with application e.g. Web UI, Mobile App, Chatbot etc. | HTML, CSS, JavaScript / Angular Js / React Js etc. |
| **2.** | Developing Interface | Developing application for the task | Java / Python |
| **3.** | Voice Assistance | Voice commands instead of typing | IBM Watson STT service |
| **4.** | Chatbot Assistance | Conversational Interface | IBM Watson Assistant |
| **5.** | Database | Data Type, Configurations etc. | MySQL, NoSQL, etc. |
| **6.** | Cloud Database | Database Service on Cloud | IBM DB2, IBM Cloudant etc. |
| **7.** | File Storage | File storage requirements | IBM Block Storage or Other Storage Service or Local Filesystem |
| **8.** | Machine Learning Model | Purpose of Machine Learning Mode | Object Recognition Model, etc. |
| **9.** | Infrastructure (Server / Cloud) | Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration : | Local, Cloud Foundry, Kubernetes, etc. |

**7.2 FEATURE 2**

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| **1.** | Open-Source Frameworks | List the open-source frameworks used | Technology of Opensource framework |
| **2.** | Security Implementations | List all the security / access controls implemented, use of firewalls etc | e.g. SHA-256, Encryptions, IAM Controls, OWASP etc. |
| **3.** | Scalable Architecture | Justify the scalability of architecture (3 – tier, Micro?services | Artificial Intelligence (AI) |
| **4.** | Availability | Justify the availability of application (e.g. use of load balancers, distributed servers etc.) | RAID(redundant array of independent disks) |
| **5.** | Performance | Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc. | DRAM or flash memory |

**7.3 DATABASE SCHEMA**

**SQL**

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.SQL is not a database system, but it is a query language.Suppose you want to perform the queries of SQL language on the stored data in the database. You are required to install any database management system in your systems, for example, Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, etc.SQL is a short-form of the structured query language, and it is pronounced as S-Q-L or sometimes as See-Quell.

This database language is mainly designed for maintaining the data in relational database management systems. It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDBMS. You can easily create and manipulate the database, access and modify the table rows and columns, etc.

This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.If you want to get a job in the field of data science, then it is the most important query language to learn. Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.

**7.3. DATABASE SCHEMA:**

**SQL**

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.SQL is not a database system, but it is a query language.Suppose you want to perform the queries of SQL language on the stored data in the database. You are required to install any database management system in your systems, for example, Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, etc.SQL is a short-form of the structured query language, and it is pronounced as S-Q-L or sometimes as See-Quell.

This database language is mainly designed for maintaining the data in relational database management systems. It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDBMS. You can easily create and manipulate the database, access and modify the table rows and columns, etc. This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.If you want to get a job in the field of data science, then it is the most important query language to learn. Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.

**8.1. TEST CASES:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or finished product lt is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

There are various types of tests. Each test type addresses a specific testing requirement. Following this step, a variety of tests are conducted.

**Unit Testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application; it is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basicTests at component level and test a specific business process, application, and/or System configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration Testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied, as shown by successively unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problem that arises from the combination of components.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Function : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete,additional tests are identified and the effective value of current tests is determined.

**System Testing**

System testing ensures that the entire integrated software system meets requirements. lt tests a configuration to ensure known and predictable results. An example of system testing 1s the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It has a purpose. It is used to test areas that cannot be reached from a blackbox level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the innerworkings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, Such as specification or requirements document. It is a test in which the software under test is treated as a black box you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing:**

Unit test is usually conducted as part of a combined code and unit test and unit testing phase of the software lifecycle, although it is not uncommon for coding and unit tests to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Features to be tested

Verify that the entries are of the correct format.

No duplicate entries should be allowed.

All links should take the user to the correct page.

**Integration Testing**

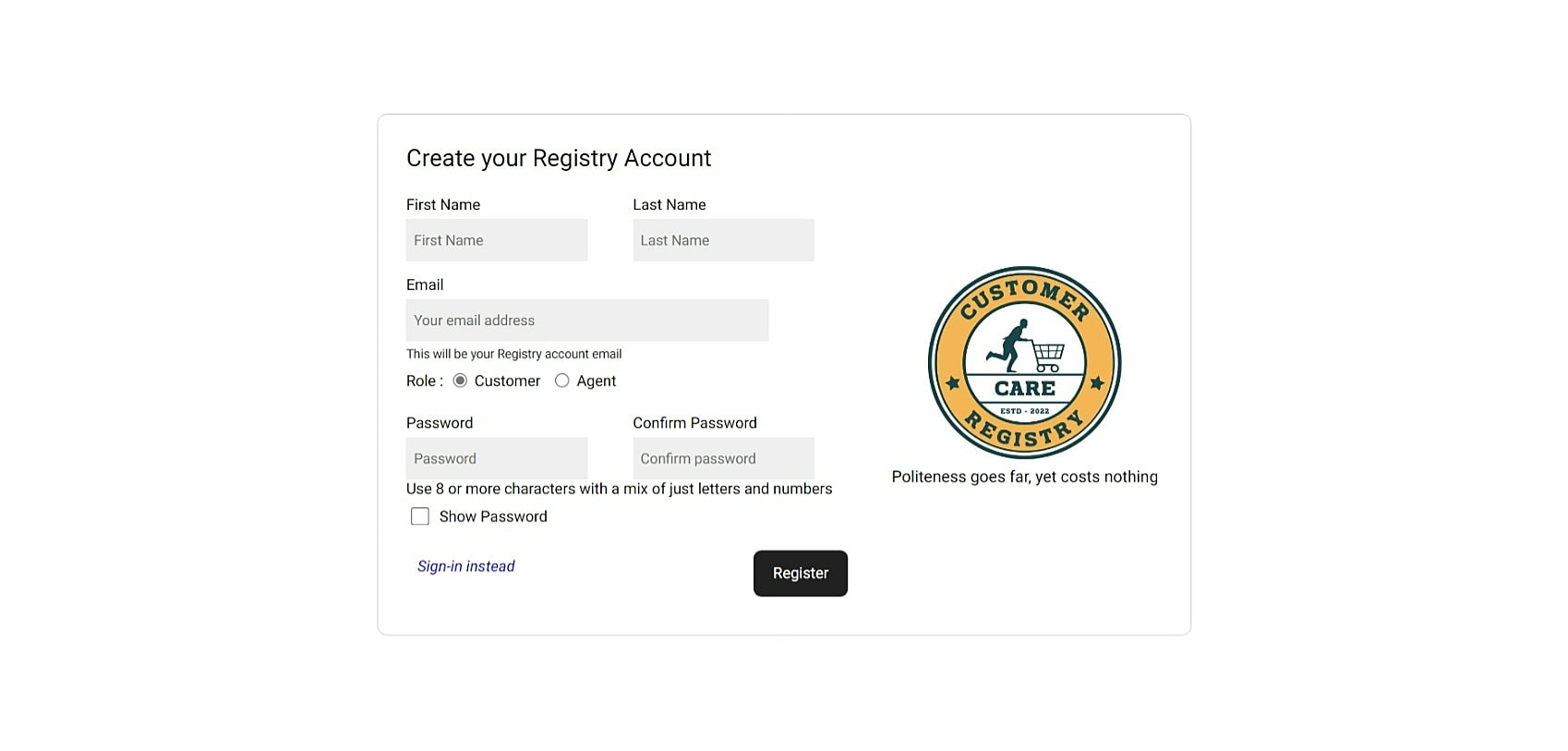
Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.The task of the integration test is to check that components or software applications, e.g. components in a software system or-one step up- software applications at the company level - interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

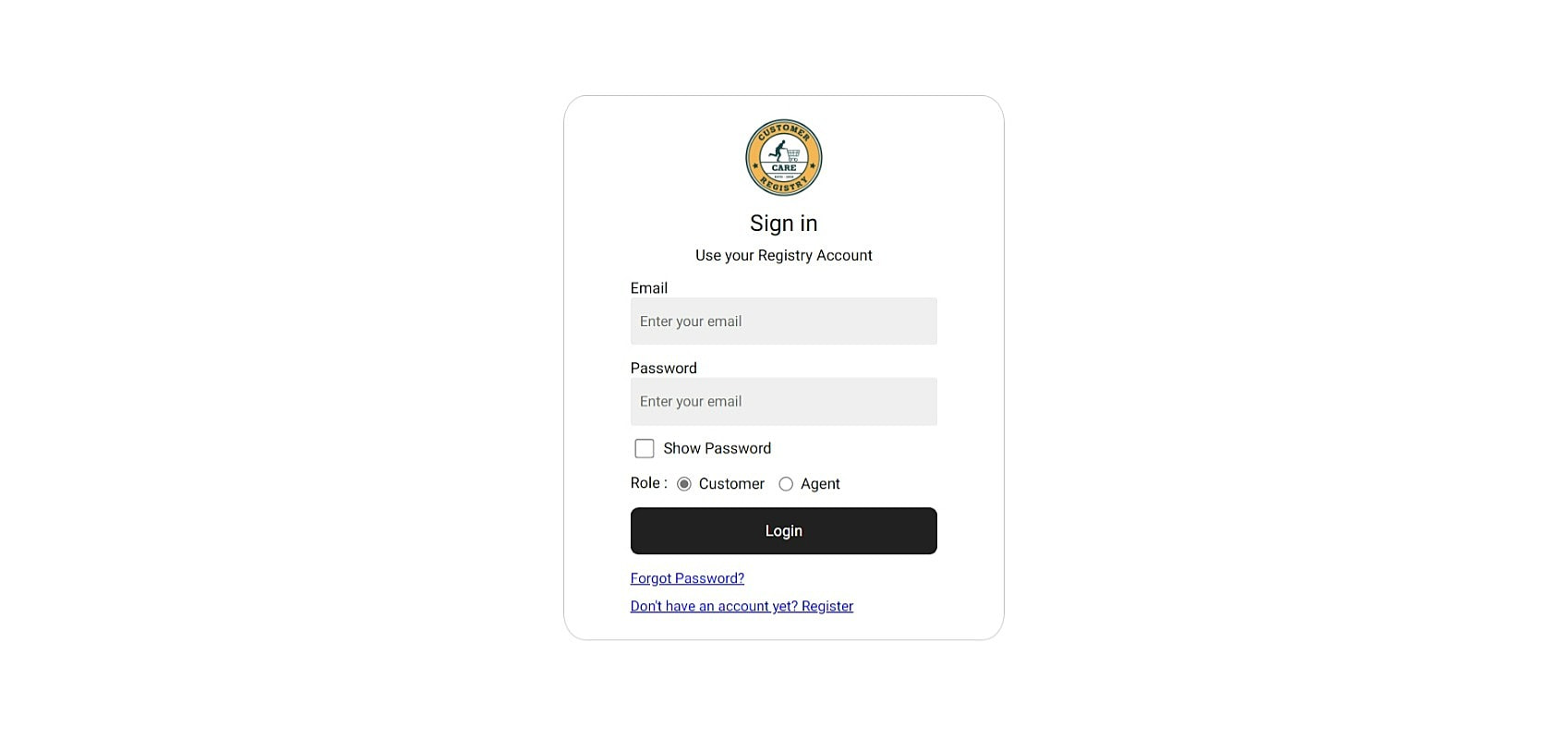
**9.Result**

**9.1 Performance Matrics**

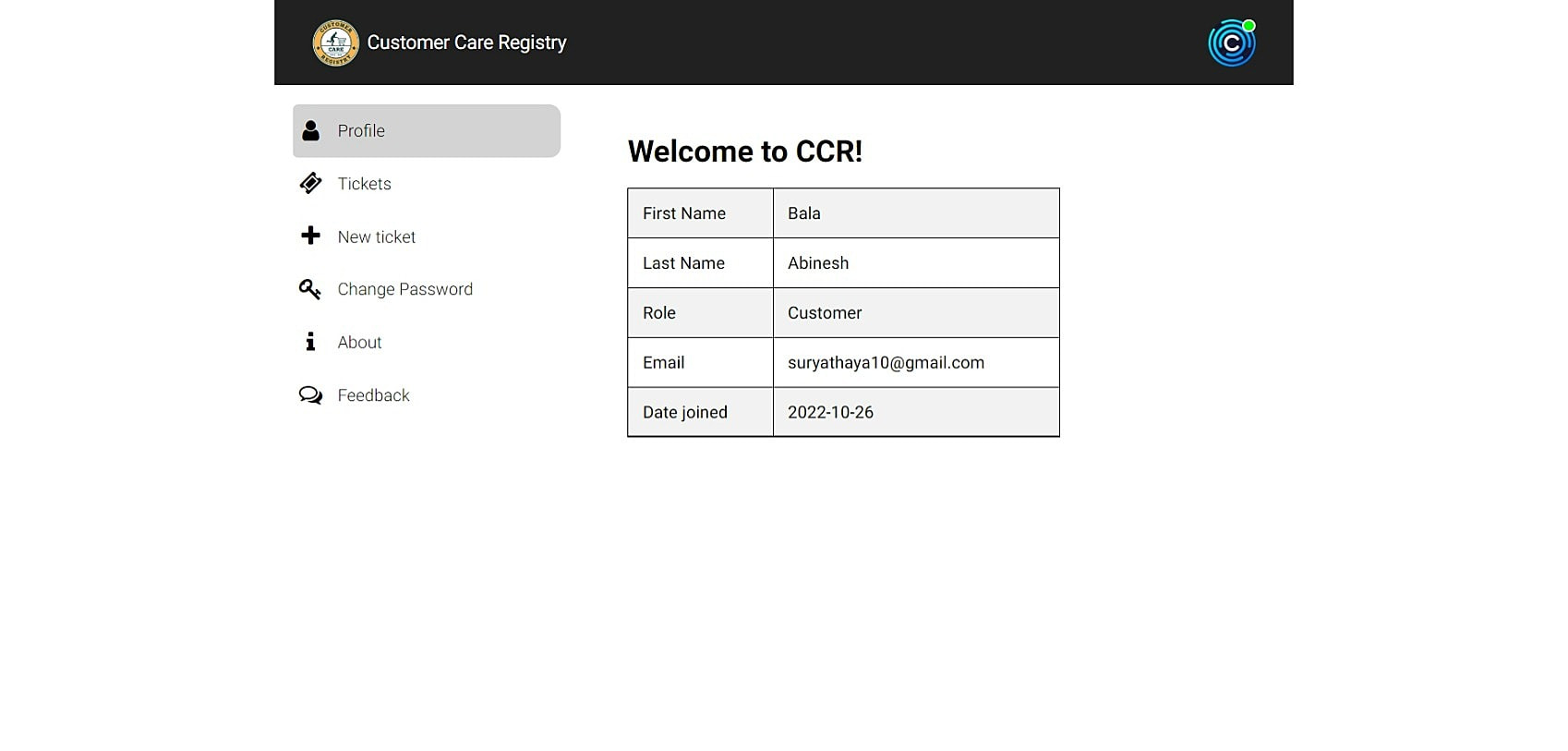
**1)Home page**

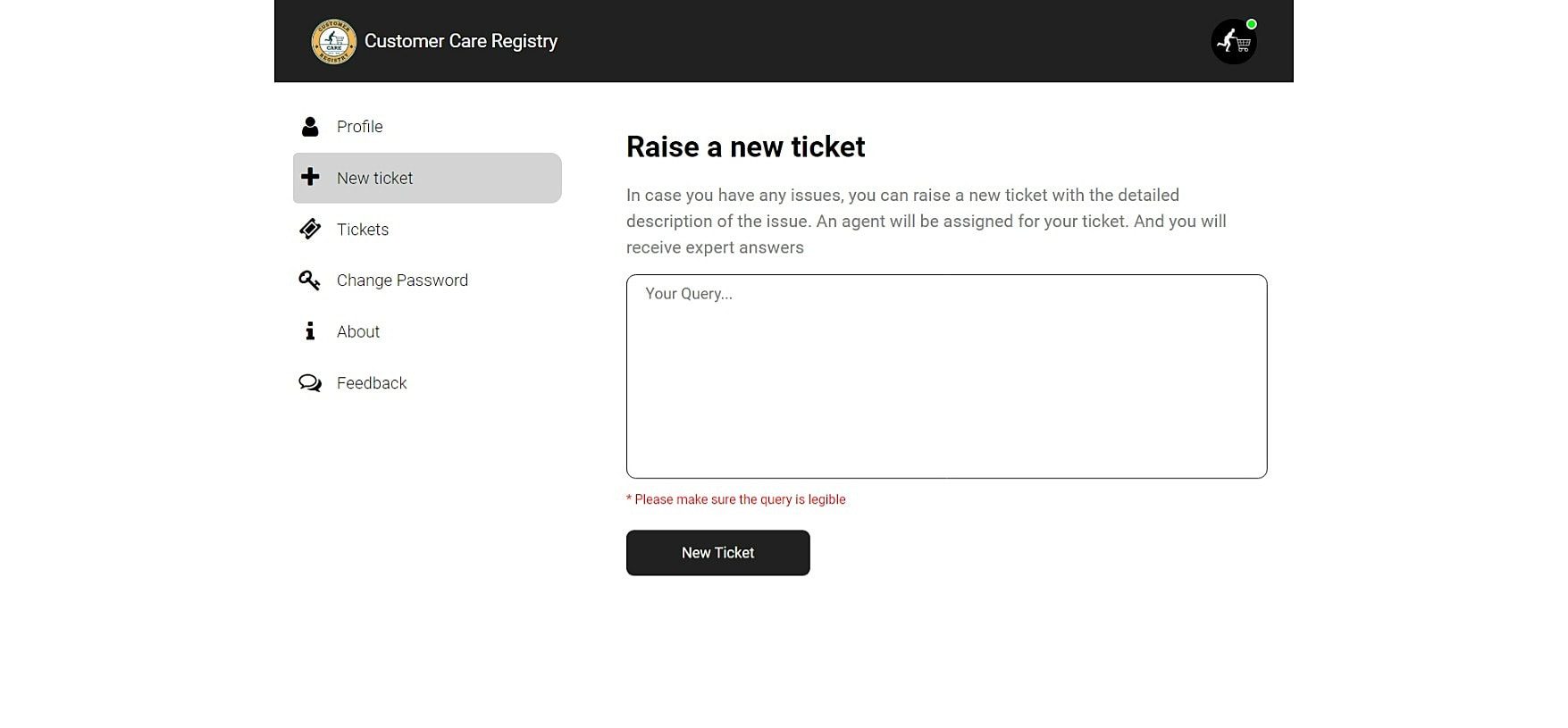


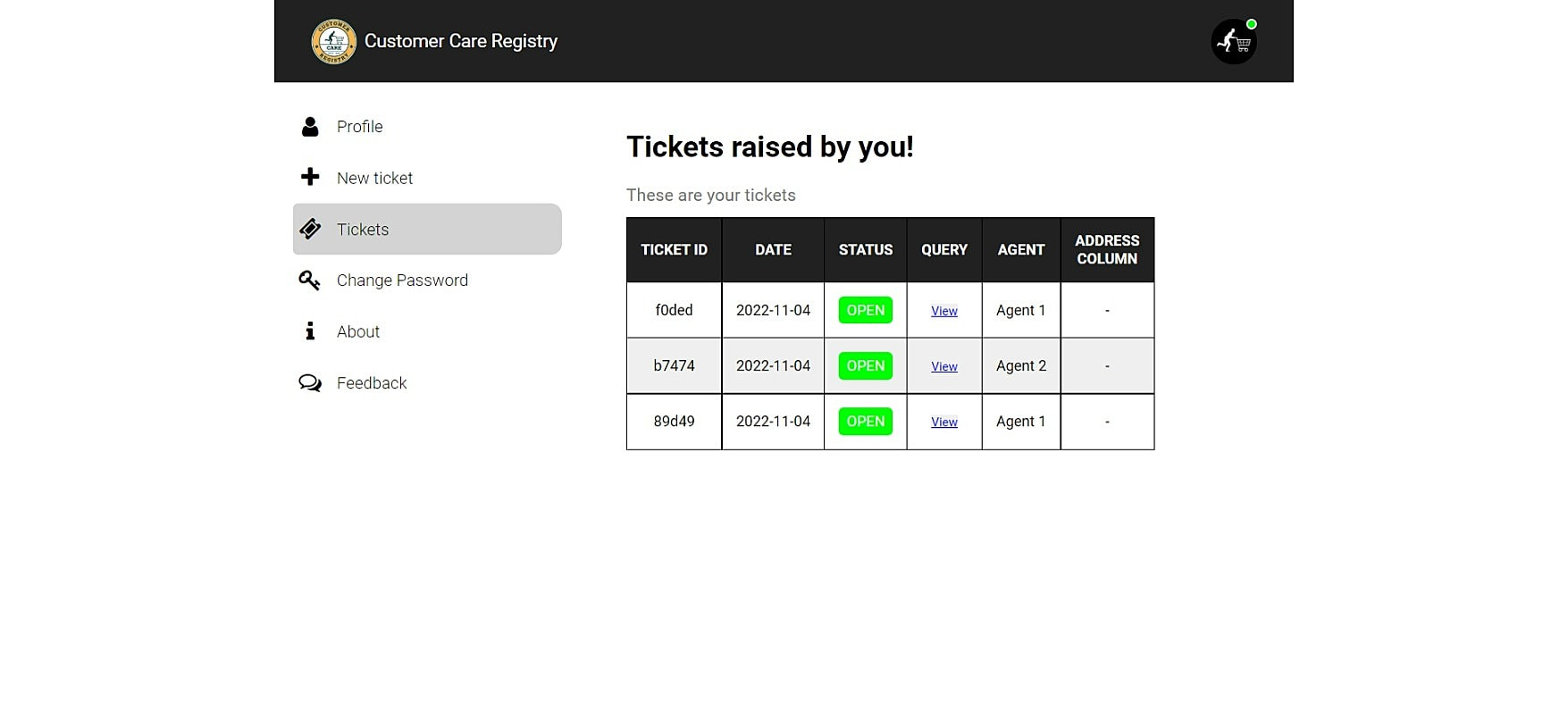
**2)Sign in**



**3)Profile**







**10. ADVANTAGES AND DISADVANTAGES**

**Advantages**

It allows for more effective sales and marketing.

It can speed up the sales conversion process.

It increases staff productivity, lowers time costs and boosts

morale

Can improve customer loyalty through exceptional experience.

**Disadvantages**

### Staff over-reliance on CRM may diminish customer loyalty through a bad experience

### .Security concerns associated with centralised data

### The excess initial time and productivity cost of implementation

**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

**source code**

base,html

<!DOCTYPE html>

<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">

<title>{% block title %}{% endblock %}</title>

<link rel="icon" type="image" href="{{ url\_for('static', filename='images/cart logo white-modified.png') }}">

<!-- Linking css, js, Google fonts -->

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}"/>

<link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400

;1,500;1,700;1,900&display=swap" rel="stylesheet">

<script src="{{ url\_for('static', filename='js/pass.js') }}"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<!-- Linking Watson Assistant -->

{% block watson %}

{% endblock %}

</head>

<body>

{% block alert %}

{% if to\_show %}

<script>

alert('{{ message }}')

</script>

{% endif %}

{% endblock %}

{% block main %}

{% endblock %}

</body>

</html>

login.html:

{% extends 'base.html' %}

{% block title %} Login

{% endblock %}

{% block main %}

<div class="bg-main-div">

<section class="login-section">

<div class="login-div">

<div class="login-header">

<img src="{{ url\_for('static', filename='images/cart logo white.png') }}" class="login-img" alt="logo" />

<h2>Sign in</h2>

<p>Use your Registry Account</p>

</div>

<div class="login-remind">

<form action="{{ url\_for('blue\_print.login') }}" method="POST" class="login-form">

<label>Email</label>

<input type="email" required value="{{ email }}" name="email" placeholder="Enter your email"/>

<label>Password</label>

<input type="password" required value="{{ password }}" name="password" id="password-input" placeholder="Enter your password"/>

<div class="show-pass-div">

<input type="checkbox" onclick="showPassword()" style="height: 20px;"/>

<p>Show Password</p>

</div>

<div class="role-div">

<p>Role : </p>

<div>

<div>

<input type="radio" style="height: 20px;" value="Customer" checked name="role-check"/>

<p>Customer</p>

</div>

<div>

<input type="radio" style="height: 20px;" value="Agent" name="role-check"/>

<p>Agent</p>

</div>

<div class="dash-nav">

<div>

<div class="dash-img-text">

{% if user == "AGENT" %}

<a href="{{ url\_for('agent.assigned') }}">

<i class="fa fa-arrow-left" aria-hidden="true"></i>

</a>

<img src="{{ url\_for('static', filename='images/cust profile.png') }}" class="img-in-nav" alt="logo"/>

{% else %}

<a href="{{ url\_for('customer.tickets') }}">

<i class="fa fa-arrow-left" aria-hidden="true"></i>

</a>

<img src="{{ url\_for('static', filename='images/agent.png') }}" class="img-in-nav" alt="logo"/>

{% endif %}

<h3>{{ name }}</h3>

</div>

</div>

<div>

<div style="align-items: center;">

{% if value == "True" %}

{% if user == "CUSTOMER" %}

<a href="/customer/close/{{ id }}"><button class="logout-btn">CLOSE TICKET</button></a>

{% endif %}

{% endif %}

</div>

</div>

</div>

</nav>

<div class="chat-body">

<div class="chat-contents" id="content">

{% if msgs\_to\_show %}

{% for chat in chats %}

{% if chat['SENDER\_ID'] == sender\_id %}

<div class="message-sent">{{ chat['MESSAGE'] }}</div>

{% else %}

<div class="message-sent received">{{ chat['MESSAGE'] }}</div>

{% endif %}

ustomer / agent myMessage = request.form.get('message-box')

if len(myMessage) == 0: to\_show = True{% endfor %}

{% endif %}

</div>

<div class="chat-input-div">

{% if value == "True" %}

<form method="POST" action="{{ post\_url }}">

<input name="message-box" class="chat-input" type="text" placeholder="Type something" required/>

<button type="submit" class="chat-send">

<i class="fa fa-paper-plane-o" aria-hidden="true"></i>

</button>

</form>

{% else %}

<div>

{% if user == "CUSTOMER" %}

<h4>You closed this ticket. Chats are disabled</h4>

{% else %}

<h4>{{ name }} closed this ticket. Chats are disabled</h4>

{% endif %}

</div>

{% endif %}

</div>

</div>

</div>

{% endblock %}

chat.py:

from flask import render\_template, Blueprint, request, session, redirect, url\_for import ibm\_db

from datetime import datetime import time

chat = Blueprint("chat\_bp", name )

@chat.route('/chat/<ticket\_id>/<receiver\_name>/', methods = ['GET', 'POST']) def address(ticket\_id, receiver\_name):

'''

Address Column - Agent and Customer chats with one another

'''

: param ticket\_id ID of the ticket for which the chat is being opened

: param receiver\_name Name of the one who receives the texts, may be Agent / Customer

# common page for both the customer and the agent # so cannot use login\_required annotation

# so to know who signed in, we have to use the session user = ""

sender\_id = "" value = "" can\_trust = False

post\_url = f'/chat/{ticket\_id}/{receiver\_name}/'

if session['LOGGED\_IN\_AS'] is not None:

if session['LOGGED\_IN\_AS'] == "CUSTOMER":

# checking if the customer is really logged in

# by checking, if the customer has uuid attribute

from .views import customer

if(hasattr(customer, 'uuid')):

user = "CUSTOMER"

sender\_id = customer.uuid can\_trust = True

else:

# logging out the so called customer return redirect(url\_for('blue\_print.logout'))

elif session['LOGGED\_IN\_AS'] == "AGENT":

# checking if the agent is really logged in

# by checking, if the agent has uuid aatribute from .views import agent

if (hasattr(agent, 'uuid')):

user = "AGENT" sender\_id = agent.uuid can\_trust = True

else:

# Admin is the one who logged in

# admin should not see the chats, sp directly logging the admin out return redirect(url\_for('blue\_print.logout'))

to\_show = False message = ""

if can\_trust:

# importing the connection string from .views import conn

if request.method == 'POST':

# chats are enabled, only if the ticket is OPEN

# getting the data collected from the c

message = "Type something!"

else:

# inserting the message in the database

# query to insert the message in the database message\_insert\_query = '''

INSERT INTO chat

(chat\_id, sender\_id, message, sent\_at) VALUES

(?, ?, ?, ?)

'''

try:

stmt = ibm\_db.prepare(conn, message\_insert\_query) ibm\_db.bind\_param(stmt, 1, ticket\_id)

ibm\_db.bind\_param(stmt, 2, sender\_id)

ibm\_db.bind\_param(stmt, 3, myMessage)

ibm\_db.bind\_param(stmt, 4, datetime.now())

ibm\_db.execute(stmt)

except:

to\_show = True

message = "Please send again!"

return redirect(post\_url)

else:

# method is GET

# retrieving all the messages, if exist from the database msgs\_to\_show = False

# query to get all the messages for this ticket get\_messages\_query = '''

SELECT \* FROM chat WHERE chat\_id = ?

ORDER BY sent\_at ASC

'''

# query to check if the ticket is still OPEN query\_status\_check = '''

SELECT query\_status FROM tickets WHERE ticket\_id = ?

'''

try:

# first checking if the ticket is OPEN

check = ibm\_db.prepare(conn, query\_status\_check) ibm\_db.bind\_param(check, 1, ticket\_id) ibm\_db.execute(check)

value = "True" if ibm\_db.fetch\_assoc(check)['QUERY\_STATUS'] == "OPEN" else "False"

# getting all the messages concerned with this ticket stmt = ibm\_db.prepare(conn, get\_messages\_query) ibm\_db.bind\_param(stmt, 1, ticket\_id) ibm\_db.execute(stmt)

messages = ibm\_db.fetch\_assoc(stmt) messages\_list = []

while messages != False: messages\_list.append(messages) print(messages)

messages = ibm\_db.fetch\_assoc(stmt)

# then some messages exist in this chat if len(messages\_list) > 0:

msgs\_to\_show = True

elif len(messages\_list) == 0 and value == "True":

# ticket is OPEN

# but no messages are sent b/w the customer and the agent msgs\_to\_show = False

to\_show = True

message = f'Start the conversation with the {"Customer" if user == "AGENT" else "Agent"}'

except:

to\_show = True

message = "Something happened! Try Again"

return render\_template(

'address.html', to\_show = to\_show, message = message, id = ticket\_id,

chats = messages\_list, msgs\_to\_show = msgs\_to\_show, sender\_id = sender\_id,

name = receiver\_name, user = user,

post\_url = post\_url, value = value

)

else:

# logging out whoever came inside the link

return redirect(url\_for('blue\_print.logout'), user = user)

init .py:

from flask import Flask, session

from flask\_login import LoginManager

def create\_app():

app = Flask( name )

app.config['SECRET\_KEY'] = "PHqtYfAN2v@CCR2022"

# registering the blue prints with the app from .routes.views import views app.register\_blueprint(views, appendix='/')

from .routes.cust import cust app.register\_blueprint(cust, appendix='/customer/')

from .routes.admin import admin app.register\_blueprint(admin, appendix='/admin/')

from .routes.agent import agent app.register\_blueprint(agent, appendix='/agent/')

from .routes.chat import chat app.register\_blueprint(chat, appendix='/chat/')

# setting up the login manager login\_manager = LoginManager() login\_manager.login\_view = "blue\_print.login" login\_manager.init\_app(app)

@login\_manager.user\_loader def load\_user(id):

if session.get('LOGGED\_IN\_AS') is not None:

if session['LOGGED\_IN\_AS'] == "CUSTOMER":

from .routes.views import customer

if hasattr(customer, 'first\_name'):

return customer

elif session['LOGGED\_IN\_AS'] == "AGENT":

from .routes.views import agent

if hasattr(agent, 'first\_name'):

return agent

elif session['LOGGED\_IN\_AS'] == "ADMIN":

from .routes.views import admin

if hasattr(admin, 'email'):

return admin

else:

return None

return app

**GITHUB LINK**

https://github.com/IBM-EPBL/IBM-Project-49004-1660814977

**PROJECT DEMO LINK**

### 