HelpDesk

An Online Chatting Software for Live Communication

Project: 2023 - EECS4481 - HelpDesk

Document: SOFTWARE REQUIREMENTS SPECIFICATION

Author: Andrew, Christian, Dexuan, Sidharth, Stefan

Last Update on: Feb 2023

Version: 0.0.1a Approved

Demo: https://onlinehelpdesk.top

GitHub: https://github.com/petroste/4481Project

Revision History

Version:	Date Approved:
0.0.1a	**2023/2/24**

1. Introduction

This section of the Software Requirements Specification (SRS) document provides the purpose of the SRS, scope of the SRS, acronyms and abbreviations used in the SRS, intended users, an overview of the SRS document's structure, as well as a table for reference.

1.1 Purpose

The purpose of this SRS document is to describe the requirements specifications for the development, deployment, and operation of the HelpDesk Application. It provides guidance on the function definition and priorities, the security requirements and testing methodologies as well as technology stack used in Applicants building and operation.

1.2 Product Scope

The HelpDesk SRS defines a Web based online chatting software used for customer services. It enables quick and easy access to customer service or agent inquiries over Web browsers for potential customers. The same time allows agents to communicate with multiple customers concurrently.

1.3 Document Conventions

The convention used in this document for defining the requirement is described in this section. Each requirement has a tag starts with REQ and followed by a unique identification number; these numbers are assigned when the requirement is approved. It shall not be changed and removed from any revision of the document. Each requirement is also assigned a list of attributes that well defines the status, priority, or dependencies etc. Like the following ones:

Status
Deleted: Component is no longer required
Proposed: Requirement has been requested
Approved: Requirement has been approved but not implemented
Implemented: Code that meets the requirement and tested

Priority
High: critical, required for next release
Medium: necessary, but can wait until next release
Low: can wait until resources permit

1.4 Intended Audience

Software developers, security testing engineers, marketing staff and interested end-users are the intended audience of this SRS document. The document is mainly constructed for the purpose of software development guidance, but it also highlights both important functional features and critical security practices of the HelpDesk Application, for better marketing and comparison between similar products.

1.5 Acronyms

GUI - Graphical User Interface
HTML - HyperText Markup Language
HTTP - HyperText Transfer Protocol
HTTPS - HTTP Secure
JS - JavaScript
JSON - JavaScript Object Notation
MERN – MongoDB Express.js React.js Node.js
MVC - Model-View-Controller
REST - Representational State Transfer
SRS - Software Requirements Specification
SSL - Secure Sockets Layer
TLS - Transport Layer Security
UUID - Universally Unique Identifier
XML - Extensible Markup Language

1.6 References

IEEE Software Requirements Specification Template - Dalhousie University. (n.d.). Retrieved February 20, 2023, from https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc

2. Overall Description

This section of the Software Requirements Specification (SRS) document describes the perspective, major function, intended end users and use cases of the HelpDesk Application. Also provides an outline for deployment, operation, constraints, and necessary users documentations.

2.1 Product Perspective

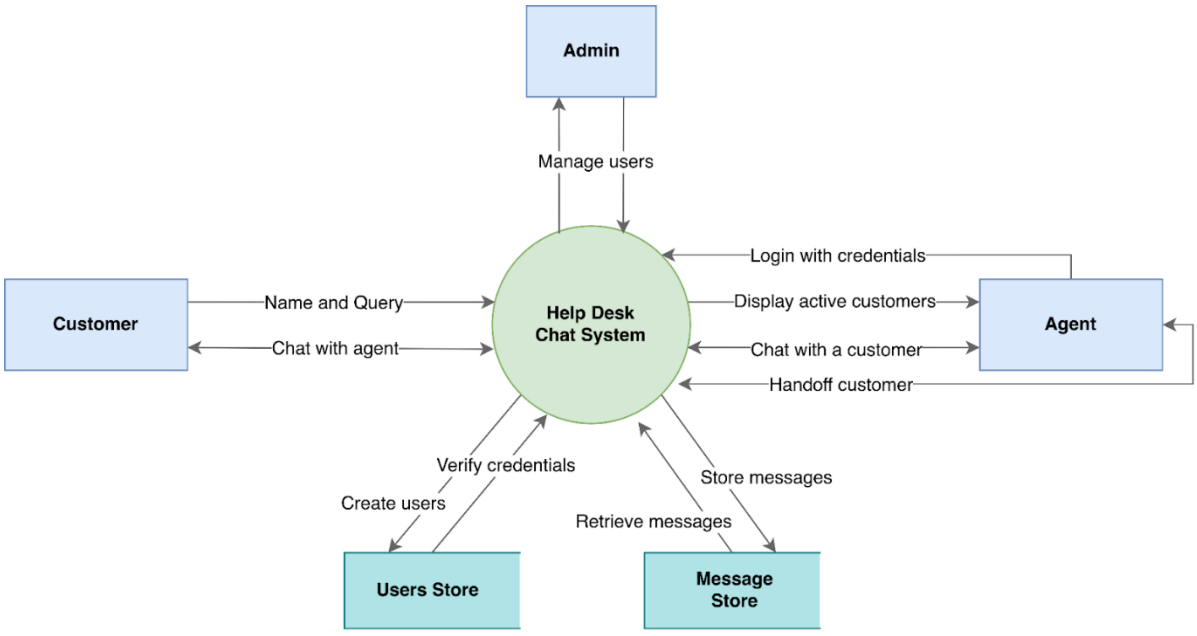
The HelpDesk Web based chatting software is a piece of software focuses on providing intercom capabilities over browser environment. Although the software and its major function is stand alone, it is not intended for standalone use, but as a supplementary to the existing Web Applications which lacks the built-in or wish to separate the functionality like live communication between service providers and their customers.

2.2 Product Functions

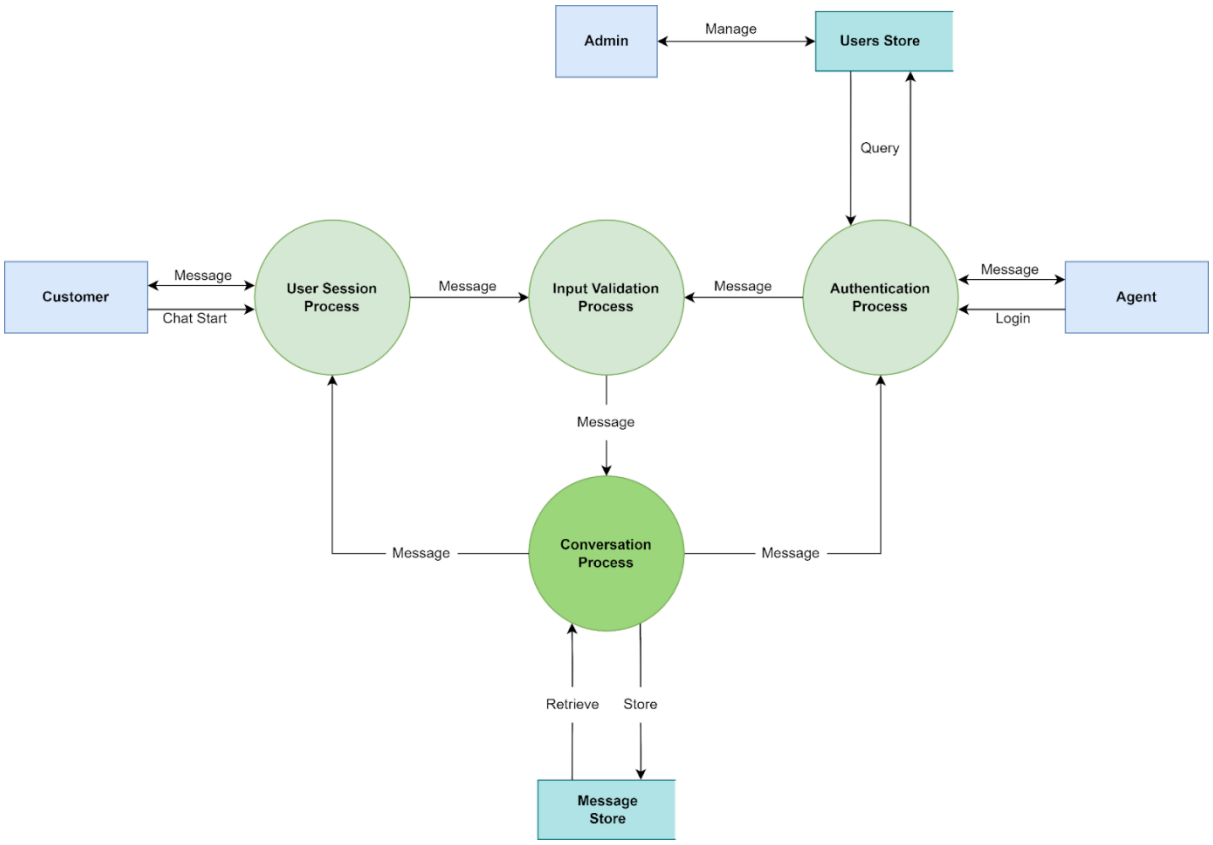
2.2.1 Function Summary

The major function of the software can be simply described as: Provides a Web interface that allow end user to connect and talk to an agent anonymously, while allow the agent to login and talk to multiple users concurrently, in addition, agents can talk to each other as well.

2.2.2 Context Data Flow Diagram (Level-0)



2.2.3 Data Flow Diagram (Level-1)



2.3 User Personas

2.3.1 Target Customers

HelpDesk application is a product designed for companies that provides any form of online communication between its user and a live agent. Such as E-commerce sites that provide online order inquiries and other customer services, or traditional industries that offer remote support for regular customers, and live quotes for potential customers.

2.3.2 End Users

The expected end user of this product is generally from two groups. The first group consists of people who offer services over conversation on the platform, which are normally called “agents”. The other group is the customers of the platform, who wish to resolve some problems or questions over the conversation with an agent. They expect to start the session with an online agent, easily and anonymously.

2.4 Use Cases

The following use case diagram will provide a broad overview of general use cases for the Helpdesk application.

2.4.1 Customer Side

A non-technical user needs assistance on a particular technical problem, and they use the Helpdesk application to connect with an agent that can help them.

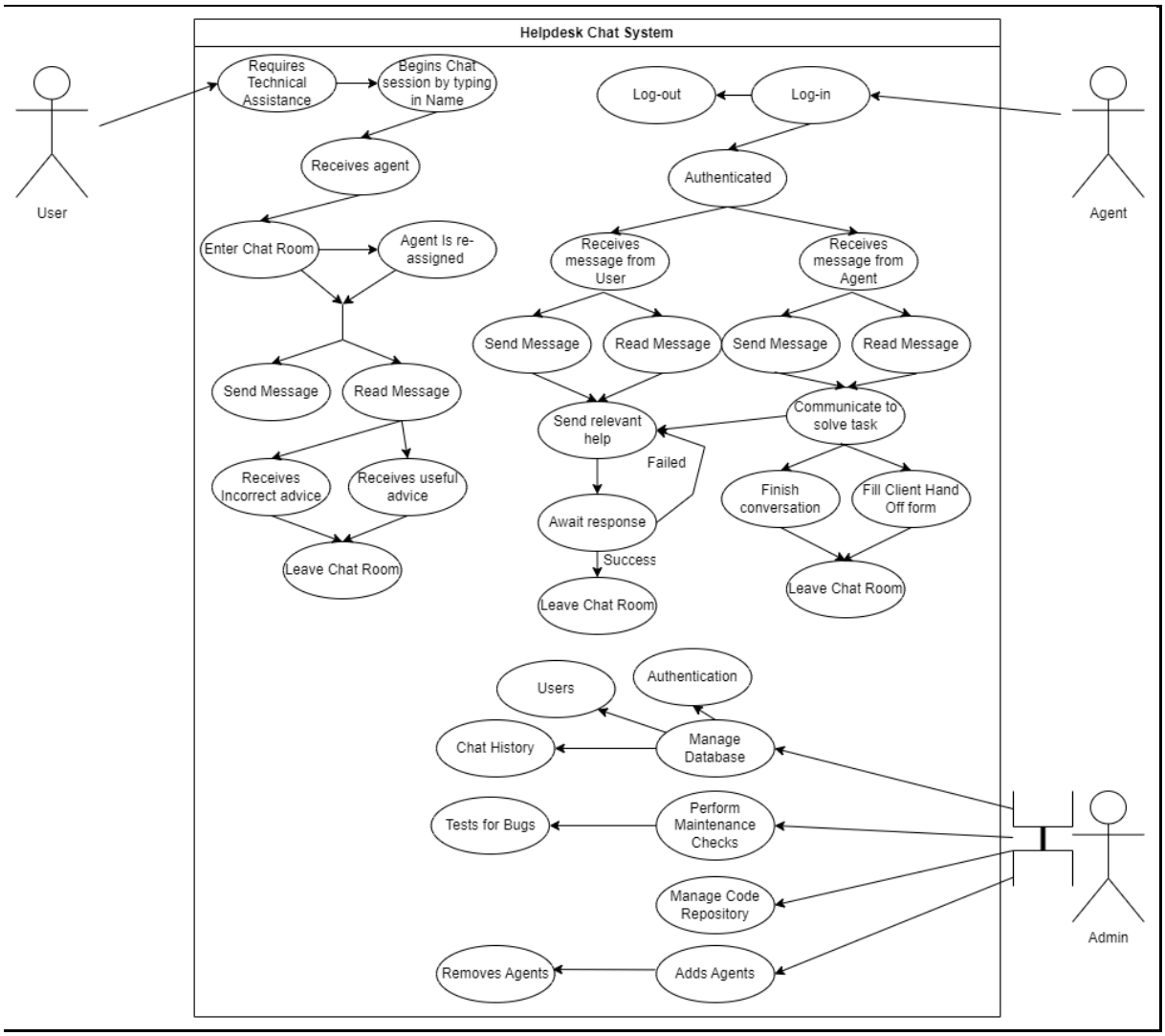
- Client enters their name in the home page to begin a session.
- They receive an automated message from the agent welcoming them to begin the conversation.
- The client clicks the chat button to the left and opens the agent chat.
- At any point the client can leave the chat by pressing the leave chat button and return to the home page.
- If the client was passed between agents, then they get an alert stating the agent switched them and a new chat button will appear in the chat bar notifying them of the new agent.

2.4.2 Agent Side

A technical expert otherwise known as an agent, will have customer(s) assigned to them, in order to provide live support.

- Agent Signs in and gains 2 new tabs (chats and switch)
- Chats will be initially empty if they are the first agent since no one is online or will contain chats of all agents in the chat bar so they can talk to them if they are not the first.
- When a client gets assigned to an agent the agent will not see the conversation until the client sends the first message in case the client changes their mind and leaves. This avoids cluttering the agent's screen. The client gets an automated welcome message offloading some work from the agents.
- In a chat session with anyone an agent can leave, but it is not advised unless the conversation is stale, or the customer stated they are leaving the conversation. Once they click the leave chat button, the conversation is removed from the chat bar once they click onto a new conversation, in case, they want to review what has been said so far.
- The client is not notified of an agent leaving so if an agent leaves they must notify the client first.
- When an agent leaves a chat it remains on their screen but when they enter another chat the chat they left will disappear.
- The agent is not notified of the client leaving the chat.
- The switch tab is where an agent can enter the agent, they wish to pass a client to and the client name. If either piece of information is wrong, then an error alert pops up.
- If the switch is successful the agent switches to the chat tab and the conversation remains in the chat body, but the chat button is removed from the chat bar.
- If a customer button disappears an agent is to assume they left
- If the agent receives a customer being switched to them their chat bar will be updated and the agent that passed the customer can message them.

2.4.3 Interaction Diagram



2.5 Deploy and Operate

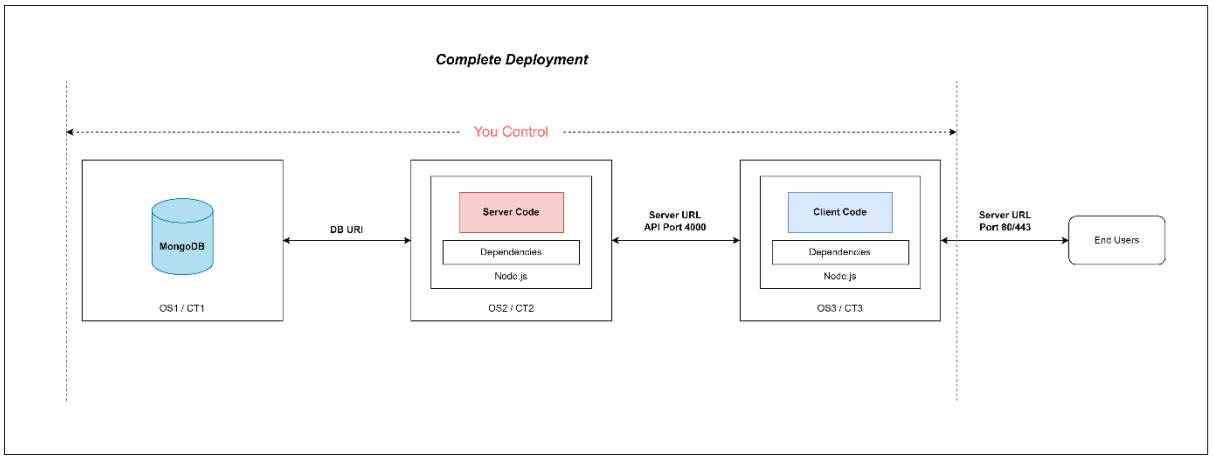
This section of the SRS provides the possible deploy methods and the expected routine during software operation.

2.5.1 Deployment

The application should be capable of three methods of deployment, including Complete Deployment, PaaS, and SaaS deployment.

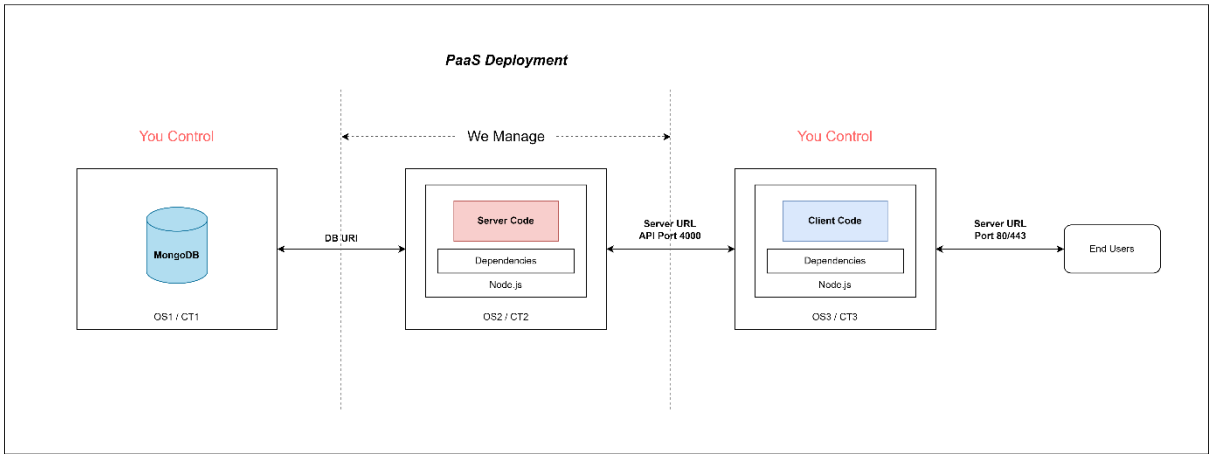
Complete Deployment:
In a Complete Deployment of this application, the user is required to use a compatible Linux based OS and install all dependencies using “npm” commands from Node.js, this deployment method offers the

maximum control of data and software. The user can decide freely on software/dependence update and customization. The following diagram shows the structure of a complete deployment.



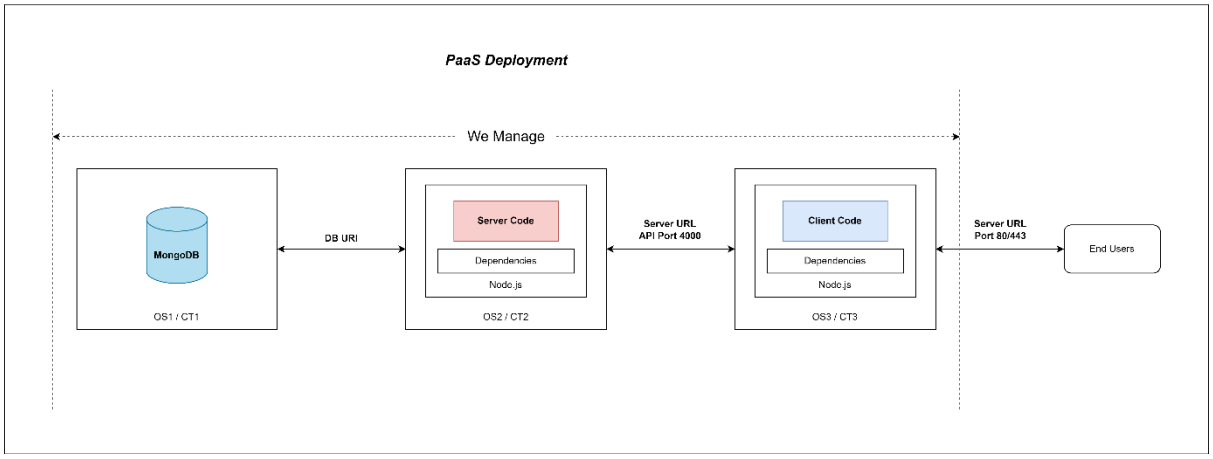
PaaS Deployment:

The user can control valuable data asset for most efficient use, and allow customized interface to be deployed as long as it is compatible with server APIs. Ease the pain of software updates and security concerns. The backend is updated and managed by the software team for best security and reliability.



SaaS Deployment:

The easiest deployment used for time critical and low-cost projects. User can simply integrate the Web page as a floating div or a popup window on their website, allow complete isolation between functions.



2.5.2 Operation

During Operation, the software should receive continuous secure testing and patching. Both the dependencies and the server-side software should maintain API backwards compatibility to ensure overall compatibility with customized client-side code. The dependencies should be maintained under the instruction of software release note to prevent unexpected behavior or potential bugs.

2.6 Assumption and Dependencies

The project assumes the user data for privileged users is generated before the use of this application. Thus, the application does not offer user registration function. The user should import existing data into database to admin or privileged user login.

The Development stack used for this project is mainly MERN, which stands for MongoDB, Express.js, React.js and Node.js. Thus these 4 elements are necessary component to run the software. The components are for the following function correspondingly: Store and provide query function over the data, creating APIs, creating GUI, run JS codes.

2.7 User Documentations

The software comes with deployment manual for detailed instruction through dependence library setup, database setup, networking requirements and update instructions. In addition, it would provide necessary examples for each function.

2.8 Constrains

- The project development stack might heavily relay on JavaScript. There could be performance impact compared to most compiled languages, such as Java and C#.
- The project does not take HA (High Availability) function into consider, which means the software would need other load balancing or fail-over method to achieve additional HA requirements.
- The scalability of the software is not specified during design, thus its not developed and tested.

3. External Interface Requirements

3.1 User Interfaces

This part provides the GUI requirements, in which the end user directly interact with. It has been divided into “Pre-Conversation Interface” which includes everything other than the page for dialogue and “Conversation Interface” which is the page responsible for actual conversations.

3.1.1 Pre-Conversation Interface

- It must offer a portal on home page for anonymous user to start conversation with simple input like “preferred name” and “describe your problem.”
- It shall also offer login button on the top of the home page, allow privileged users to quickly login.
- It should provide a about page that describes the application and the services provider.
- It should include privacy policy pop-up or page to satisfy regulations.

3.1.2 Conversation Interface

- It should be consistent for both the customers and agents, in terms of layout, color and design.
- It should offer menu for selection of different customers.
- It should provide an intuitive layout inf the message area, allow user to quickly identify the message sender, sequence, and time.

3.2 Software Interfaces

3.2.1 Client-Side Interfaces

- Client should offer HTTP/HTTPS interface to provide GUI for end users.
- Client should be able to connect the Web socket through HTTPS to allow message transfer.

3.2.2 Server-Side Interfaces

- Server should provide the Web Socket API used for message transmission.
- Server Should be able to connect to MongoDB through proper driver.

3.3 Communication Interfaces

4. System Features (Functional Requirements)

4.1 REQ 01-Secure Authentication

Status: Implemented
Priority: High

Description:
Allow user with privileges login and logoff securely.

Functional Requirements:

- Both Web interface and Web socket should only authorize privileged user with proper credential to access the resource.
- The login interface should have input validation and allow cookie to be used for login.
- User with privileges should only establish a Web socket after they have been correctly authenticated through Web Interface.

4.2 REQ 02-Session Management

Status: Implemented
Priority: High

Description:
Allow user sessions to be maintained on a browser with proper timeout.

Functional Requirements:

- Users should be able to maintain sessions without a second login
- Refreshing the Web page does not logoff the user.
- A proper timeout is set to ensure security.

4.3 REQ 03-Message Transmission

Status: Implemented
Priority: High

Description:
Send and receive message between client-side and server-side through Web Socket.

Functional Requirements:

- Message can be transfer though a Web socket between Client and Server.
- Each conversation is uniquely identified, to allow multiple conversation on single client.

4.4 REQ 04-Anonymous Conversation

Status: Implemented
Priority: High

Description:
Allow non-privileged user to start conversation with an agent anonymously.

Functional Requirements:

- User can start conversation without login.

- Anonymous user can only start conversation with a logged in user (Agent)
- The user can terminate the conversation and end the current session.
- Once a session is terminated, the user should restart a new conversation without previous records.

4.5 REQ 05-Conversation Restore

Status: Implemented
Priority: High

Description:
During a conversation session, the interfaces should be able to restore previous record of the session.

Functional Requirements:
- Both the Web interface and the Web Socket must correctly identify an unterminated session and allow it to be continued
- Once the session is resumed, the client should restore message history by retransmitting the messages through Web socket.

4.6 REQ 06-Input Validation

Status: Implemented
Priority: High

Description:
Every message sent by the client should be properly checked by the server, before its handled to next process.

Functional Requirements:
- The validation must be implemented on the server side.
- The message should be gone through validation first once received by the server.
- The validation should be using mature libraries for better performance and reliability.

4.7 REQ 07-User Management

Status: Proposed
Priority: High

Description:
The Web interface should offer simple user management for Admin users.

Functional Requirements:
- Admin user can add user, with username and password.

- Admin user can modify password of other non-admin users.
- Admin user can delete non-admin users.

5. Other Nonfunctional Requirements

5.1 Security Requirements

- All communication should be encrypted using SSL/TLS protocol.
- Login function shall implement function that prevent brute force password cracking.
- Validation on message input should prevent JavaScript code injection.
- User password should be encrypted or hashed with salt value in storage.
- Database shall be correctly configured with corresponding privileges to enforce access control.

5.2 Performance Requirements

- The client code is expected to load the GUI on browser within one second.
- The server code is expected to release RAM allocation for some less active conversation while under load.
- The server code shall optimize database transactions and use periodic transaction grouping to prevent over frequent query and bandwidth waste.

5.3 Customization Requirements

- Since the software uses a Client-Server model, it should allow certain degrees of customization over the client-side code. The server side is expected to provide long term compatibility over the APIs, it shall not remove support of old functions.
- The client code should be completely customizable if the code uses the server API in a standard way. The user should be able to develop and deploy preferred GUI in client codes.
- The client-code shall work a template that allow simple modification to change the branding and interface layout and color etc.

5.4 Capacity Requirements

- The server API is expected to handle at least 1000 concurrent conversation at a time.
- Each agent should be able to run maximum 10 conversation concurrently.
- The API should be able to handle at least 100 messages every second.
- Server should automatically maintain conversation history up to a week or longer for customer training.

5.5 Usability

- The GUI is straight forward and easy to navigate for both agents and customers.
- The GUI only displays necessary buttons for the corresponding user.