Software Requirements Specification

for

EECS 4481 Chat-App

Version 2.0

Prepared by Rishab Dhamija

York University

April 6, 2020

Table of Contents

Table of Contents 1

Revision History Error! Bookmark not defined.

1. Introduction 2

1.1 Purpose 2

1.2 Intended Audience and Reading Suggestions 2

1.3 Project Scope 2

1.4 References 2

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Features 3

2.3 User Classes and Characteristics 3

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 Assumptions and Dependencies 4

3. System Features 5

3.1 Private Chat System 5

3.2 Client/Server System 5

3.3 Helpdesk Users Authentication System…………………………………………………...6

4. External Interface Requirements 7

4.1 User Interfaces 7

4.2 Hardware Interfaces 7

4.3 Software Interfaces 7

4.4 Communications Interfaces 7

5. Other Nonfunctional Requirements 8

5.1 E-R Diagram 8

5.2 Safety Requirements 8

5.3 Security Requirements 8

5.4 Software Quality Attributes 8

# Introduction

## Purpose

The purpose of this document is to build an online helpdesk chat system to provide anonymous users with one-to-one assistance with help desk staff .The app provides services in 4 major areas : Tech Support, General Enquiries, Sales associate and Cancellation/Refunding. For administrators, the application provides an interface where they can monitor the current assignments of clients to helpdesk users as well as number of rooms that are vacant / occupied.

## Intended Audience and Reading Suggestions

Clients which can anonymously get one-to-one assistance from helpdesk.

Users in a completely private environment.

## Project Scope

The purpose of this project is to build an online helpdesk application as well as to create a user-friendly, browser-independent application for regular users, which require assistance from experts in various areas. The app currently focuses on 4 major divisions listed as tech support, general enquiries, sales associate as well as cancellation/refunding. The database server hosts all the credentials for authenticating helpdesk staff members so that a complete secure environment can be maintained.

## References

There are no references required for this project to work in form of any tutorials or any reference guides.

# Overall Description

## Product Perspective

A dedicated one-to-one online helpdesk system for anonymous users. The application

consists of 3 major categories:

● **Admin Login** : This category gives/provides a helpdesk staff member to login to a specific helpdesk room , where the client has already logged in and is waiting for the admin.

● **Helpdesk Dashboard** : This category provides a dashboard view of all the

administrators a list of all the logged in clients and admins in all the currently used rooms. An unassigned helpdesk staff member can access this dashboard to see which client is still not assigned a helpdesk staff member and login to that respective room.

● **Client Login :** this category constitutes a dropdown list for the client to login to a

respective room using an alias. This could be any name client wishes to communicate with the admin (can also be used as anonymous identity by using anonymous as display name).

## Product Features

The main features of the online helpdesk application are NoSQL MongoDB for faster and efficient authentication as well as socket.io /socket.io-client for interaction between server as well as multiple instances of client.

## User Classes and Characteristics

Users of the system (helpdesk users as well as anonymous users) have a specific characteristic based on their roles. For an anonymous user, they login as client by providing their login name as well as the room they wish to login.

For helpdesk users, they are required to provide credentials to login to a specific room they have the authority to login to.

The clients should have the following functions:

● Login to any of the 16 available rooms for helpdesk service

● Get into one-on-one interaction with the admin logged into their room.

● As soon as a client occupies a room, it becomes unavailable for the other client to login. Therefore, the system currently provides only 16 room capacity.

The helpdesk users have the following functionalities:

● Login to an existing room based on the credentials they are required to use during the login screen.

● Able to check for existing clients waiting to be assigned a dedicated helpdesk user by logging into the Helpdesk Dashboard system.

## Operating Environment

The operating environment for the helpdesk application is as listed below:

● NoSQL

● client/server system

● Operating System : Windows

● Database: Mongo DB

● Platform: Express.js/Node.js/Moustache.js/Moment.js

## Design and Implementation Constraints

● The database instance consists of a single collection credentials.

● Collection consists of documents following the pattern below:

{

**“username”** : “admin01”,

**“password”** : “pass01”,

“ **room”** : “Tech Support 1”

}

The collection in total consists of 16 documents, each of them holding the

credentials for a helpdesk user to login as admin into that room.

## Assumptions and Dependencies

● **4 categories of rooms noted as Tech support, Cancellation/refund, General Enquiries and Sales Associate , each have 4 sub-rooms.**

The documents for the collection are such assigned values such that an admin can login to only a certain category of rooms.

Example : a sales associate help desk user can login to any of the 4 available sales associate rooms, but he cannot login to tech support room.

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## Private Chat System

3.1.1 Description and Priority

the helpdesk system maintains a private chat system between an anonymous client as well as assigned helpdesk user. It is a high priority (7 on the scale of 1 to 9) as client gets one-on-one interaction with the helpdesk user anonymously.

3.1.2 Stimulus/Response Sequences

a new client always gets an updated dropdown list of only the currently available sessions. A helpdesk user on logging to the dashboard, gets an updated list of currently logged in clients as well as the assigned helpdesk users to those clients.

3.1.3 Functional Requirements

A centralized database on cloud container provides administrator rights to login from any platform. Since the credentials are stored on cloud database, security is enhanced as the application itself does not contain any content worth security breach.

## Client/Server System

3.1.1 Description and Priority

the helpdesk system maintains a dedicated Client/Server System between an anonymous client as well as assigned helpdesk user. It is a high priority (7 on the scale of 1 to 9) as clients need to be assigned to the correct helpdesk user as well as the vacancies of available spots should be listed correctly by the application so the requests to 2 clients at the exact same time doesn’t crash the application.

3.1.2 Stimulus/Response Sequences

a new client always gets an updated dropdown list of only the currently available sessions. A helpdesk user on logging to the dashboard, gets an updated list of currently logged in clients as well as the assigned helpdesk users to those clients.

3.1.3 Functional Requirements

A centralized database on cloud container provides administrator rights to login from any platform. Since the credentials are stored on cloud database, security is enhanced as the application itself does not contain any content worth security breach. The dedicated dropdown system for available rooms should always work perfectly in synchronization with the backend to give most-updated information of available rooms to the client.

## Helpdesk User Authentication System

3.1.1 Description and Priority

The authentication system maintained by the Chat Application allows the helpdesk users to login the system as well as to get a global page report of all available clients waiting in the queue to be assigned to a respective helpdesk support user. This authentication system maintained is of the highest priority (9 on the scale of 1 to 9) , in comparison to the other functional requirements listed before.

3.1.2 Stimulus/Response Sequences

An existing helpdesk authentication system comes in 2 phases: the admin homepage login system for each helpdesk user to authenticate himself/herself and login into their respective domain. The second phase is for any of the helpdesk user to login into the global helpdesk dashboard to see the waiting clients in the queue.

3.1.3 Functional Requirements

Authentication system must work in two different scenarios for the helpdesk application to work effectively and properly:

1. When a helpdesk user wants to login to the helpdesk dashboard, he must enter the username-password pair. This combination can be any of the 16 combinations as the user will only able to view the assignments and not login to any other specific room that is not assigned to him.

2. When a helpdesk user wants to login to a specific room based on the knowledge from dashboard. Now, the user is required to login using the 3 -value credentials : username, password and room . They all work in a specific pattern to disallow a helpdesk user to login to another room that is not under his privileges.

# External Interface Requirements

## User Interfaces

● Front-End-Software: HTML, CSS, Moment.js, Moustache.js ,

socket.io-client , JS

● Back-End-Software: Node.js,Express.js , Socket.io , NoSQL

## Hardware Interfaces

● Windows 10 Pro

● Visual Studio Code

● Chrome Developer Edition Browser (supports developer Tools and

other required features)

## Software Interfaces

● Operating System : the operating system for the helpdesk system is completely user independent as the application will be hosted on AWS

● Database: to increase efficiency as well as to remove the complex Entity-Relationship constraints that come with SQL, the database chosen is MongoDB, with NoSQL.

● Visual Studio Code: to implement the project , the programming language chosen is JavaScript with its enhanced features for both front-end(Moustache, Moment etc.) as well as the backend(Express, Node).

## Communications Interfaces

This project supports all types of browsers. There are simple html pages using html forms as well as moustache for dynamically rendering the content on the html page (a part of it is updating the dropdown list of available rooms for the client as well as displaying logged in users in a room in the sidebar).

# Other Nonfunctional Requirements

## E-R Diagram

Since, the database is an instance of NoSQL, therefore there is no requirement of E-R diagram.

## Safety Requirements

No specific safety requirements are needed to be brought into attention considering the nature of the web application.

## Security Requirements

● Security requirements for the back-end code , especially the part that involves connection to the database is of critical value. If that back-end file is compromised, anyone can gain access to the

database and drop all the documents in the collection, or even worse, modify the credentials of administrators, or even create a super-admin superseding all privileges.

● Since, the instance of database is not stored locally on the server and hosted on the cloud, careful choices are needed to be considered when choosing a vendor for database cloud service in terms of reliability as well as security.

## Software Quality Attributes

The application in current conditions can satisfy a total of 16 concurrent clients(rooms) , each of them with their own respective helpdesk users.