**Real Time Chat Application (Backend)**

**Introduction**.

This documents tries to articulate the business process of the real-time chat application mainly focusing on the backend server running the business requirements of the system. The purpose, the business rules and necessary business processes have been captured to help the user understand the system and the business domain of the Real-Time Chat Application. The documentation mainly focusses on the backend side of the system as instructed by the assignment given.

**Purpose**.

The purpose of the Real-time Chat Application is to provide the Customers(Users) with a general chat platform where they can interact with the system and its functionality. The communication needs to happen in real time.

**Model of the System.**

The below diagram represent a visual business process of the system along with its actors, roles, actions, artifacts and classes.

message

|  |
| --- |
| **Client** |
|  |
| +getId(): Object  +getName(): String  +getHost(): String |

|  |
| --- |
| **Message** |
|  |
| +getSourceName(): String  +getContent(): String |

|  |
| --- |
| **StandardServer** |
|  |
| +attach(client:Client):int  +detach(client:Client)  +send(message:Message)  +initialize(maxuser:int)  +run()  +shutdown() |

**m\_client**

|  |
| --- |
| **StandardClient** |
|  |
| +transmit(message:Message  +initialize(server:String, nickname: String  +attach()  +detach()  +send(message:String)  +abort |

callback

IClientUI

IClientUI IServerUI

client server

**ServerUI**

**ClinetUI**

As Illustrated, in the diagram above users(actors) are able to interact with the system(environment) by use of various artifacts. A client makes a request to the client server which later makes a request to the Server that later sends a call back to the client server which updates the client with the feedback to facilitate bi-directional communication in real-time.

**Features of Real-Time Application.**

* Chat capabilities
* Real time communication using Socket Io
* Authentication using of JSON WEB Token
* Session Logs
* Interaction Logs
* Mongo DB as the database storage
* Server built on Express.js

**Business Rules of the System.**

* User must sign up.
* To interact with the system, the user must be logged in and authenticated.
* User must select the chat room that he or she wants to communicate with.
* Each message belongs only to one room
* A user can communicate to multiple rooms but they have to select which room.
* System chats are real-time.

**Non-functional overview.**

* The Real-time chat application allows users (customers the ability to communicate with one another in real-time.
* The system allows customers to communicate only in one room at a time or one channel at a time.
* System makes previous messages available to the user of a particular channel.
* System allows the user to login with an email and a password.
* System collects session and interaction logs.

**Conclusion.**

The design model used for the system implementation gives necessary boundaries and context under which messages and users interact with different channels in the system. This helps to prevent a user’s conversation from overlapping to another channel thus keeping the integrity of the data available in the system. The system works in a context that only logged in and authorized users are allowed to communicate in the system or access the system.