**Emergency Messaging Client-Server System**

**(Networking Java Project)**

Table of Contents

[Background 1](#_Toc388972915)

[1. Business Overview 1](#_Toc388972916)

[1.1. Objectives of the Proposed System 1](#_Toc388972917)

[2. E2E Architecture 1](#_Toc388972918)

[3. Ecosystem 1](#_Toc388972919)

[3.1. OPERATING ENVIRONMENT 1](#_Toc388972920)

[3.2. EXTERNAL INTERFACE 1](#_Toc388972921)

[4. Description of the functionalities 1](#_Toc388972922)

[5. Detailed Design 1](#_Toc388972923)

[6. Database Design 1](#_Toc388972924)

[7. Screens 2](#_Toc388972925)

[8. Testing 2](#_Toc388972926)

[9. GLOSSARY OF TERMS 2](#_Toc388972927)

# Background

A Java-based Emergency Messaging System has to be developed in which the emergency messaging server will send all the emergency messages to the client who have registered to this emergency messaging server.

# Business Overview

The Emergency Messaging System has a Server and at least 2 or 3 clients. The server and clients are separate machines. The clients will first register themselves to the server for receiving the emergency messages. Once registered, the client will receive all the emergency messages that the server will unicast, multicast or broadcast to the registered clients.

## Objectives of the Proposed System

The application is supposed to have a simple user interface with the following features:

* The server will be hosting a web application. The web application will have the following functionalities:-
  + List all the registered clients
  + Register a new client
  + De-register an existing client
  + Send Emergency Messages
* The Client also should be running a java based application which will receive the messages sent by the server and display the messages on a webpage.

# E2E Architecture

The given application is being implemented to send messages from server side to client side. The basic functions which needs to be implemented in the system for it to run smoothly and cover all the aspects of the application. We are providing with an E2E flowchart to check all the requirements.

**Register/De-register Client**

**Register Client**

**De-Register Client**

**List all the Client**

**Send Message**

**Unicast**

**Multicast**

**Broadcast**

**S.O.S**

**Server**

**Server Database**

**Login**

**User Authentication**

**Show Message**

**Delete Message**

**Client**

**Client Database**

# Ecosystem

## OPERATING ENVIRONMENT

## Hardware

The server here is a different machine and clients are all different machines.

## Software

* Client application – It uses Java Spring Boot application in the backend and runs Angular application for the frontend.
* Server application – It uses a separate Java Spring Boot application for backend and separate Angular application for frontend.
* Database Tier – MySQL server is being used.

# Description of the functionalities

We are using two different applications for the client and server side application. The server side application works as the master node and sends all the messages, while there can be multiple client side applications running parallel to each other. Each application has its main features and functionalities.

**On Server-side Application**

On the server side, two links are given to the user to i.e. “Register/de-register the client” and “Send a message” on the homepage. When clicking the first button for registration and deregistration, a new webpage opens. This webpage shows the list of the existing clients in the system. When we want a new user to register, they can click the button to open the form for adding details of the client. Similarly, when we want to de-register a client, the user can select the client name and de-register the user in real-time.

When we click the second button for sending the message, we are served with an option of sending a unicast, a multicast or a broadcast message to the client machine. When we choose to send unicast or multicast message, we are served with the list of all the registered clients and we can choose that where we have to send the message. While choosing to send a broadcast message, we implicitly send the message to all registered clients.

We are using one table to store all the registered and deregistered clients to save their IP address and description. When we deregister a client, the data is not lost in the database.

Functional flowcharts are given as follows:

1. Register a new Client

Database (Server)

Web Application (Server)

Sends IP Address

Sends Description

1. Display the registered Client

Database (Server)

Web Application (Server)

Sends list of

Clients

1. Delete the data of any registered client

Database (Server)

Web Application (Server)

Sends data to be deleted

1. Send emergency messages to the client(s)

Displays message

Database (Client)

Web Application (Server)**Logout**

Sends message

Web Application (Client)

**On Client-side Application**

On loading the client-side application, a log in page arrives to check for the credentials of the user with its username and password. When we give the valid credentials, the user can look into the messages that they have received. The application works in real-time and shows all the messages that have been sent by the server. A pop-up message is shown initially whenever a new message arrives from the server-side application.

The user is also given a functionality to delete the messages they don’t want anymore. The database saves all the messages in encrypted form in the database and then the messages are decrypted before accessing the value from the database.

Functional flowcharts are given as follows:

1. Authenticates client

Database (Client)

Web Application (Client)

Sends authentication request

1. Fetches messages of the client

Database (Client)

Web Application (Client)

Sends messages of clients

1. Deleting messages

Database (Client)

Web Application (Client)

Messages to be deleted

# Detailed Design

The following functionalities have been implemented as proposed previously:

**On Server-side Application**

1. List all the registered clients
   * When the user opens server side application and clicks on registering new user, they can see the existing registered users.
   * Also, when the server wants to send a new unicast or multicast message, they get an access to the list of registered users to send the required message.
2. Register a new client
   * Open the server side application.
   * Click on the required button to get the client registered.
   * The user will be taken to a new page where they can fill the form and add details about the client such as the IP address and the description of the client.
   * After submitting the details, the user will get the confirmation of registering the client successfully.
3. De-register an existing client
   * On the homepage of the server side application, click on the button available to register/de-register clients.
   * List of all the registered clients will be present after clicking the button.
   * In front of each registered clients, a checkbox is present. We can choose the clients we want to de-register and then click the de-register button.
   * A confirmation pop-up will arrive to show for a successful de-registration of the client.
4. Send emergency messages
   * On the homepage of the server-side application, click on the button available to send the messages to the clients.
   * Then we are taken to a new page where we get a dropdown list to send unicast, multicast and broadcast message.
   * On selecting unicast message, we get a list of registered clients with radio buttons to select one client.  
     On selecting multicast message, we get a list of registered clients with checkboxes to select one or more clients.  
     On selecting broadcast message, we do not get a list of registered clients, but all the registered clients are selected implicitly to get the message.
   * In the same form, we get a text-area to enter the emergency message to the respective clients.
   * After hitting the “Send message” button, we get a pop-up message on the confirmation of the action.

Additional functionality

1. Sending a broadcast SOS message

* This extra feature is being added to enhance the function of server side system, so that they can send the message directly to all the registered users without getting into the details of the webpage.
* An additional bell button is being added at the end of all the webpages.
* As soon as we hit the button, it send the message to all the registered clients and a pop-up appears to show the completion of the action.

**On Client-side Application**

1. Receiving and displaying the messages
   * On logging into the system with their username and password, the client can see all the previous messages being sent to the system.
   * The system updates at real-time i.e. whenever client machine is connected to the server machine and the server sends a new message, then the messages gets displayed as soon as it arrives.
   * A pop-up message is shown initially whenever a new message arrives from the server-side application.

Additional functionality

1. Log-in and log-out for the client
   * When the user opens the client side application, he is asked to log-in to the system using its specified credentials.
   * After logging in, they can see the messages.
   * After the user is done using the client application, they can log out of the system.
   * Session management and login guards are implemented in the system to ensure data security and privacy.
2. Deletion of received messages
   * The user can opt to delete any specified message of their choice, if they no longer need it.
   * A delete icon is present in front of every message in the client system for the user to use it.

# Database Design

Two different databases are maintained using MySQL server to store the details for both client and server side application.

**On Server-side Application**

* A table named as “client-details” to store all the details of the clients that server registers or deregisters. It has three fields i.e. “ip-address” to store the IP address of the client machine, “description” to store the description of the client machine, “appear” to store Boolean value for checking if the IP address is registered or not.
* This table gets the values from the server-side application when the user clicks on the register or de-register button.
* The data of this table is accessed to show all the registered client on the webpage various links.

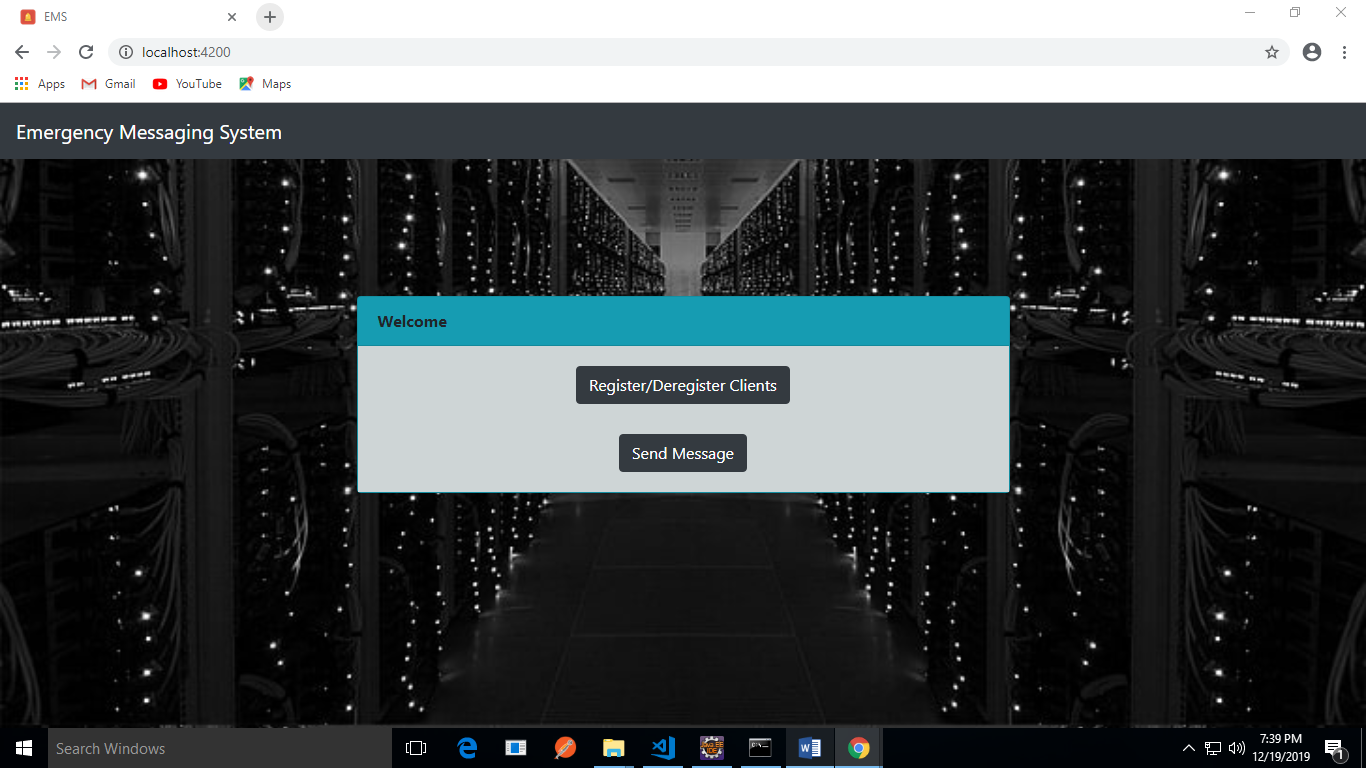
**On Client-side Application**

* A table named as “server-messages” to store the messages received from server application. This table also has three fields i.e. “time-arrived” to store the time at which message is being sent, “message” to store the message in encrypted format, “appear” to store Boolean value to check if the message is deleted or not.
* This table gets the value from the server-side application and then are sent to client-side application to get persisted in the database.
* While sending and receiving the message, the messages are being encrypted and decrypted accordingly.
* The data of this table is accessed and the messages are decrypted to show the messages to the client.

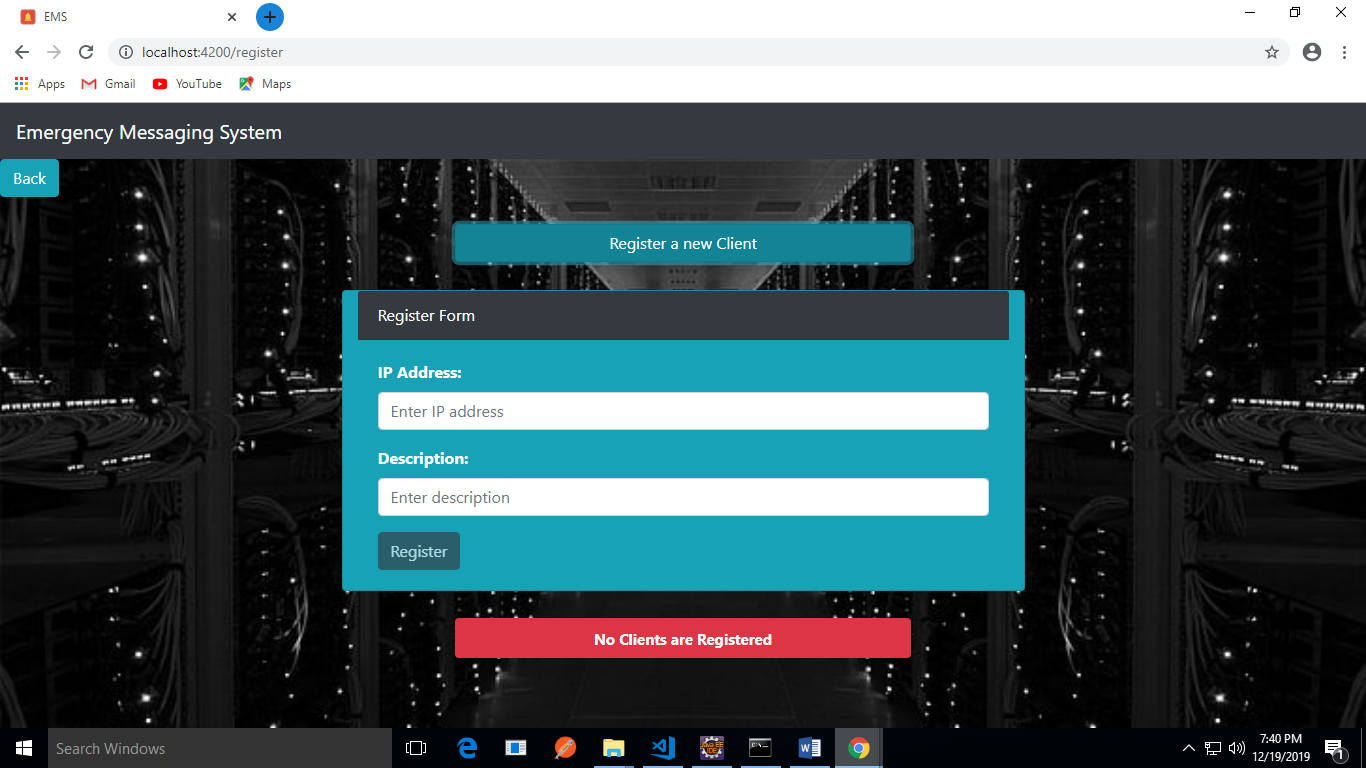
# Screens

**On Server-side Application**

1. Server home screen

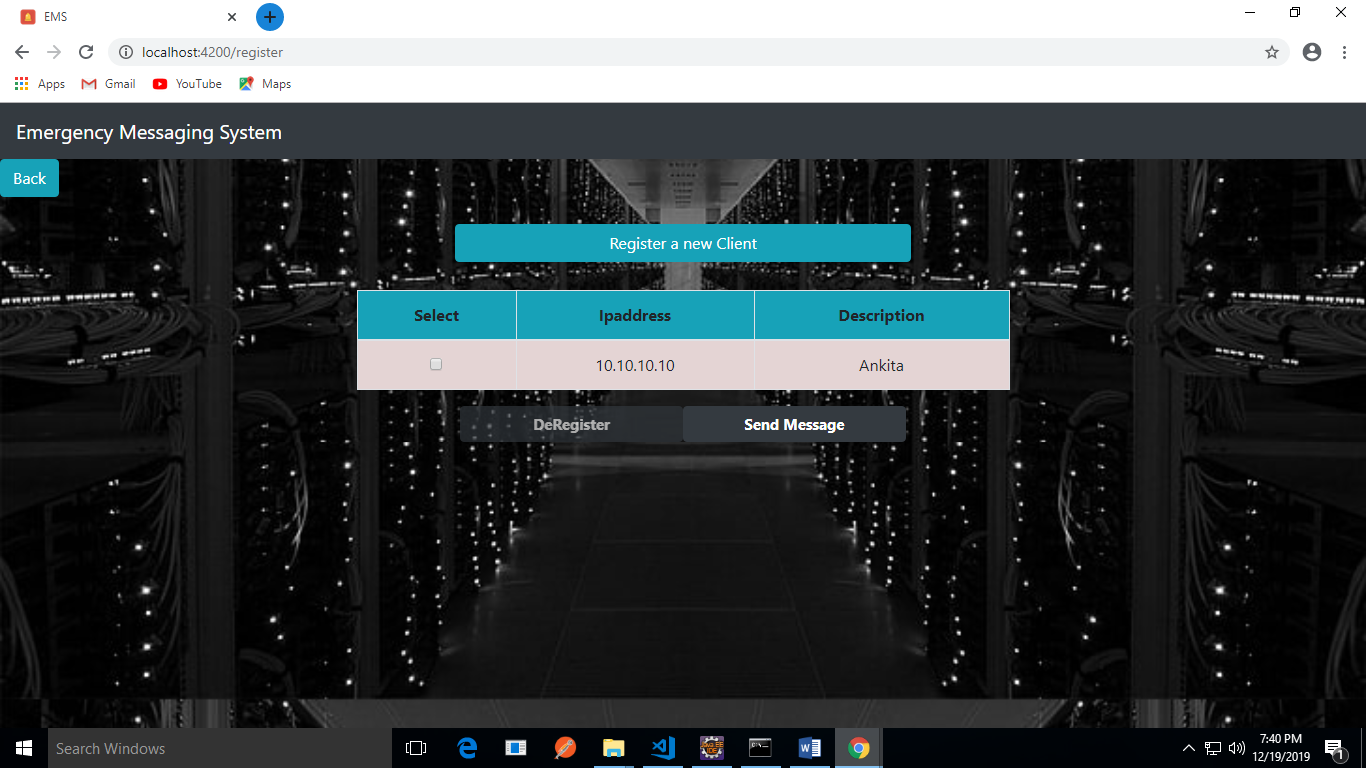


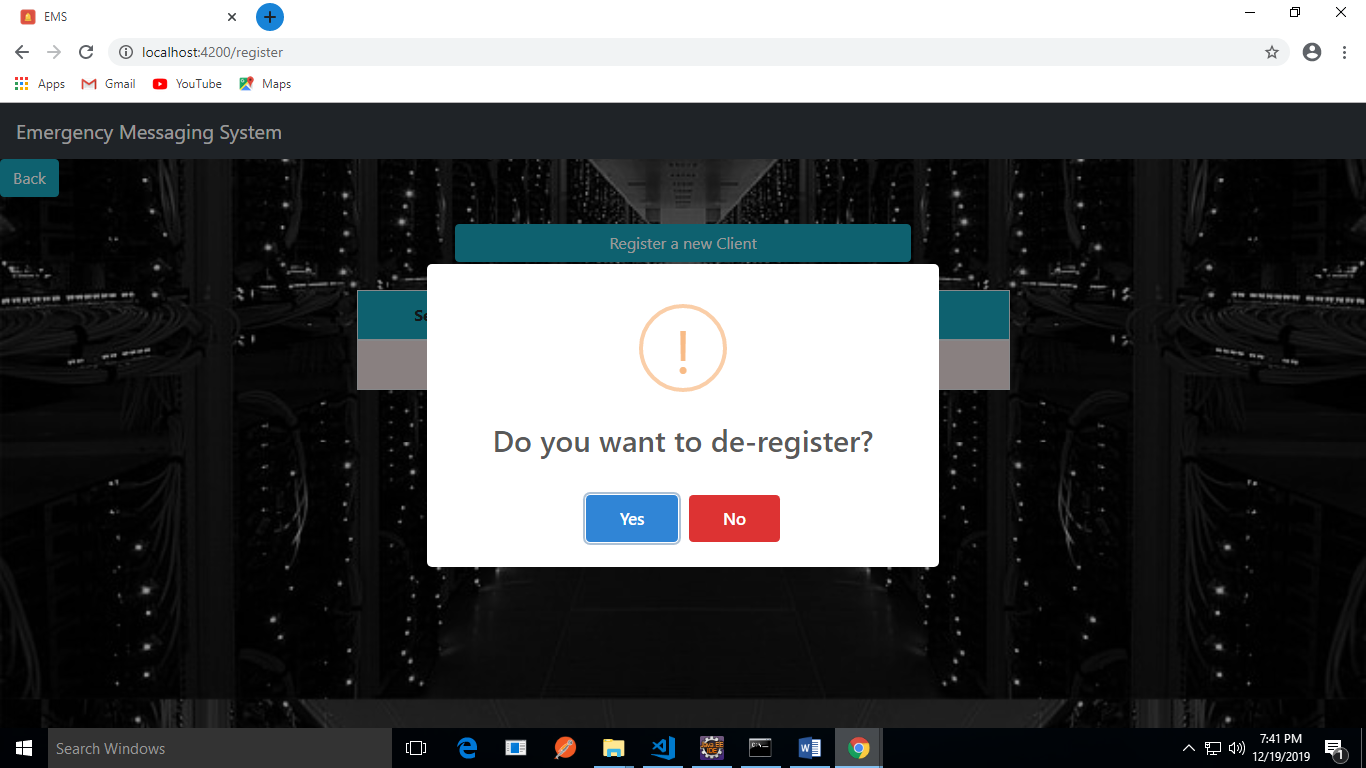
1. Server registration screen



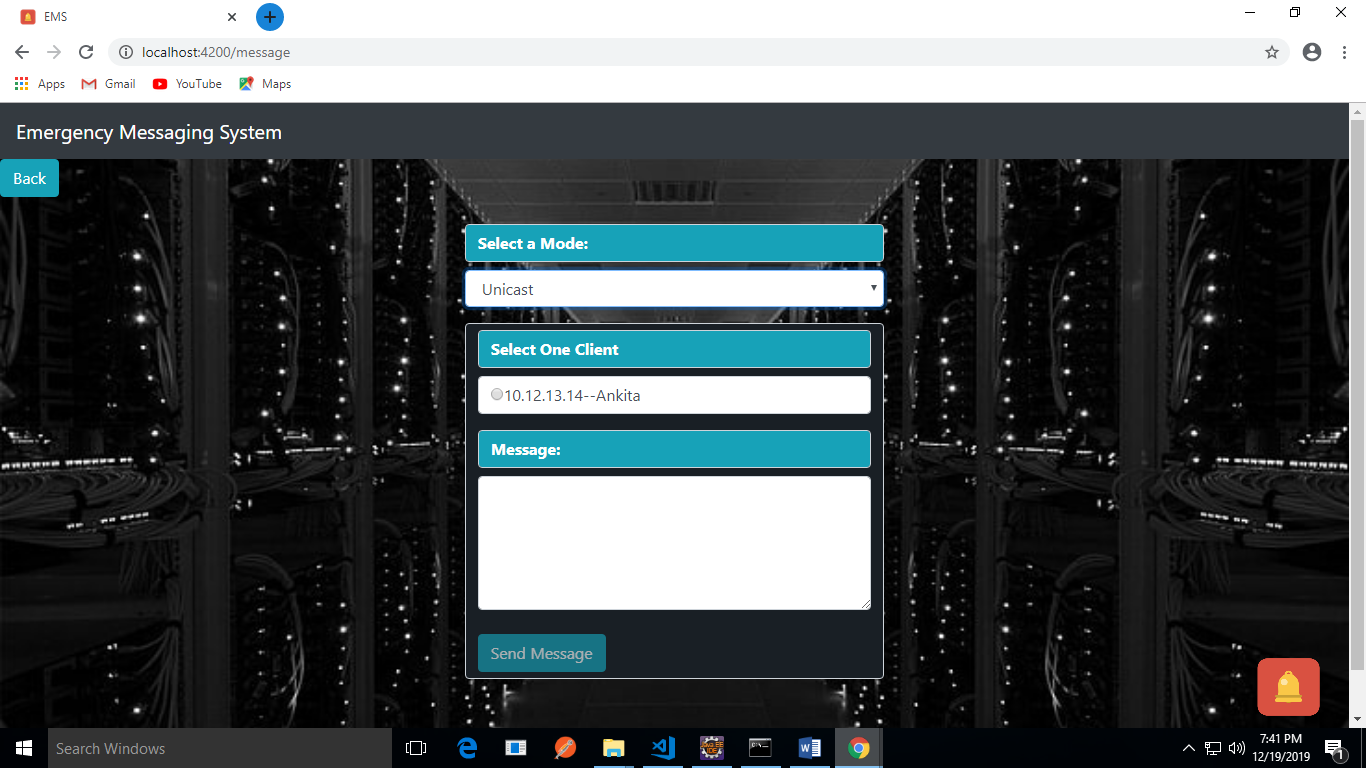
Initially the “List of all Clients” will be empty so the field-set will show that “No Clients are registered”. Once IP Address and Description is entered and clicked on “Register”. The client’s IP address should get listed in the “List of all Clients” section.

1. Server de-registration screen

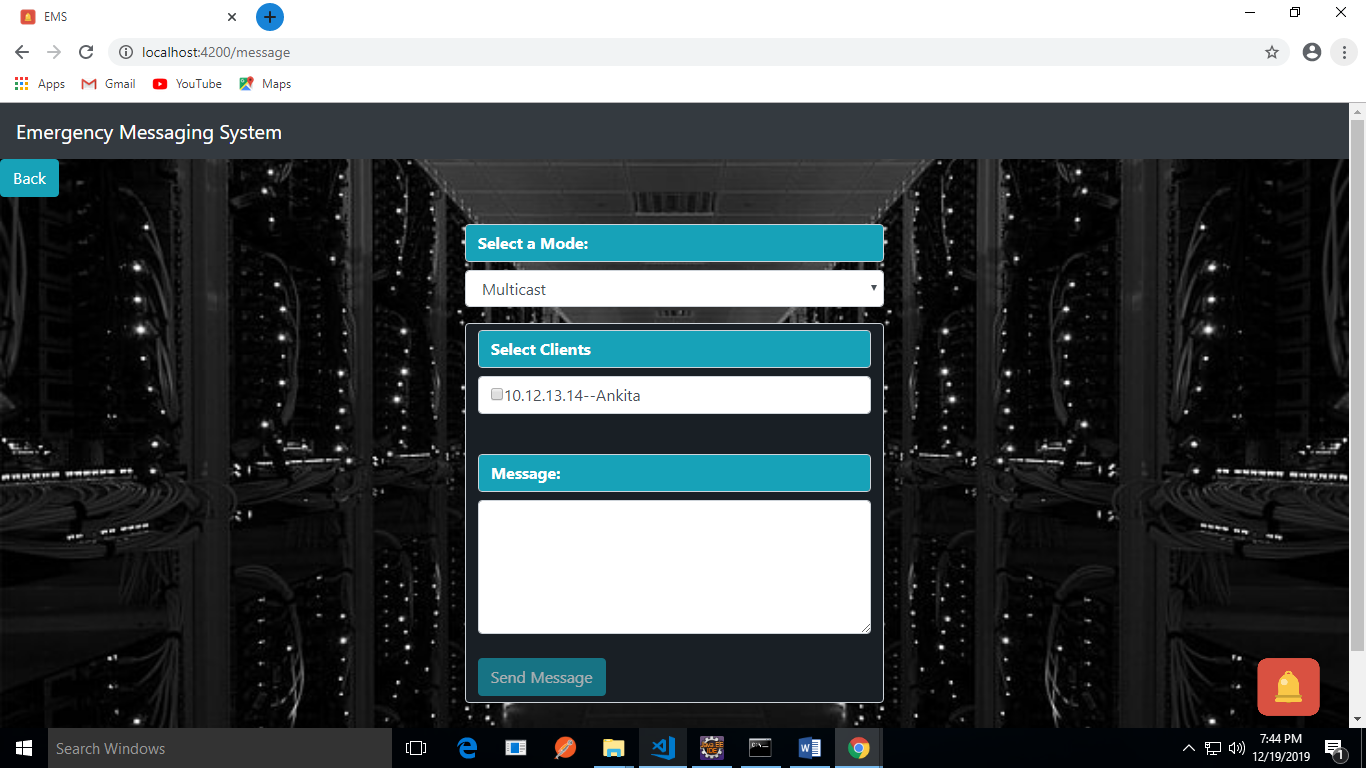




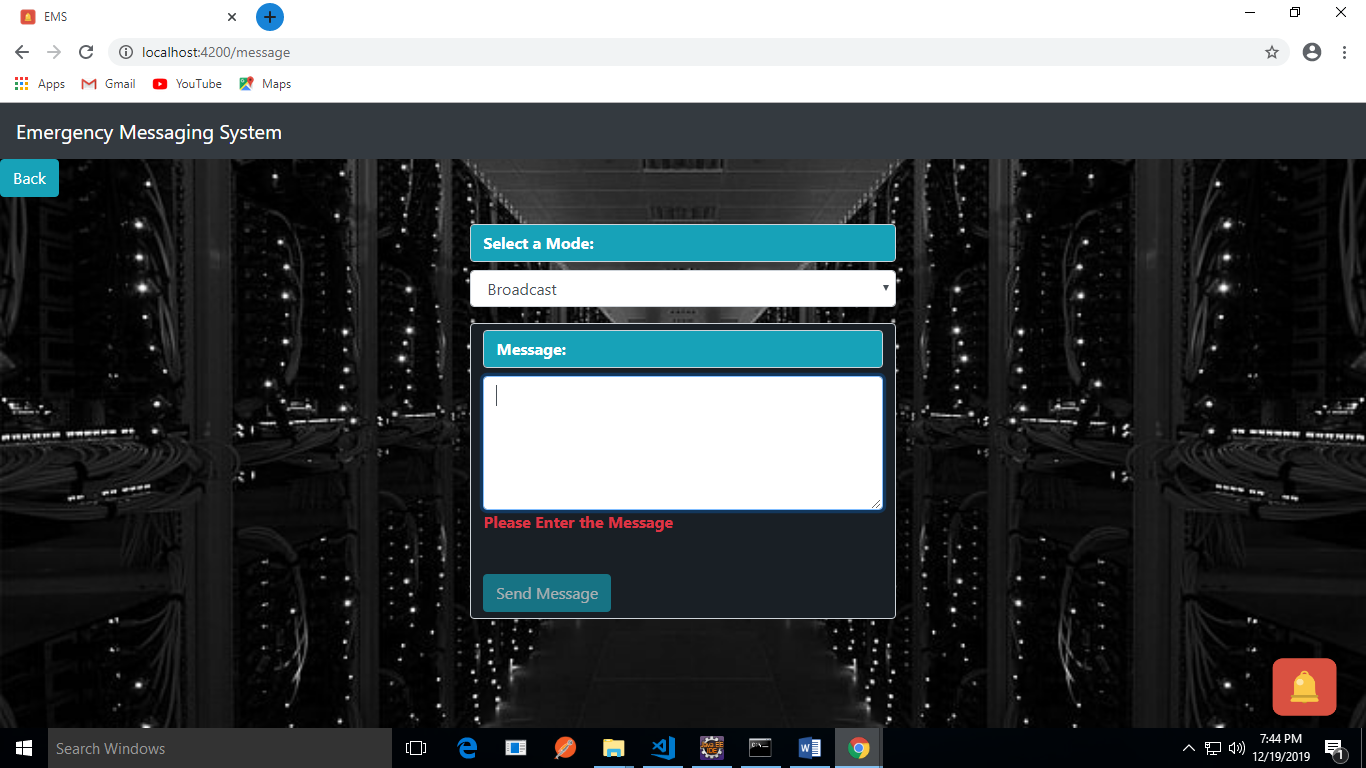
1. Server Messaging Screen
   1. For unicast



* 1. For multicast



* 1. For broadcast

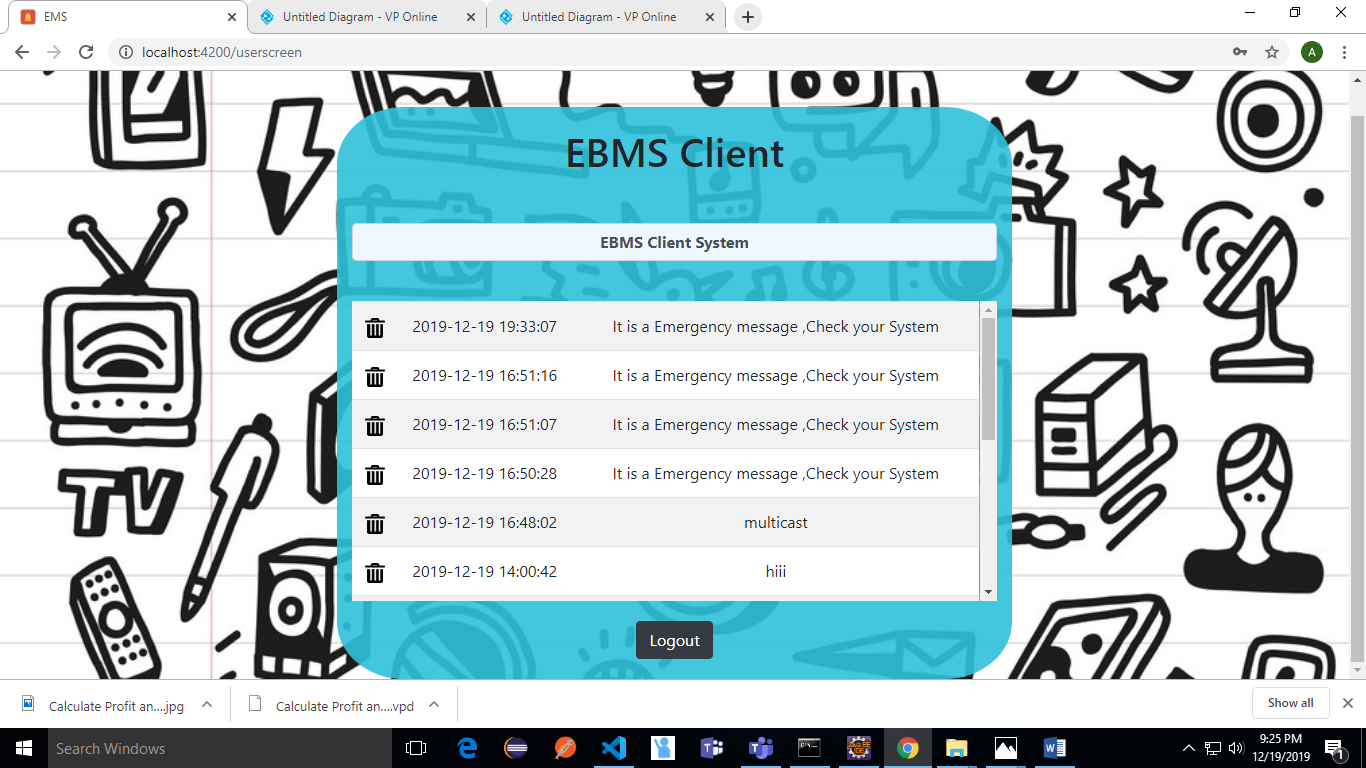


**On Client-side Application**

1. Client login screen



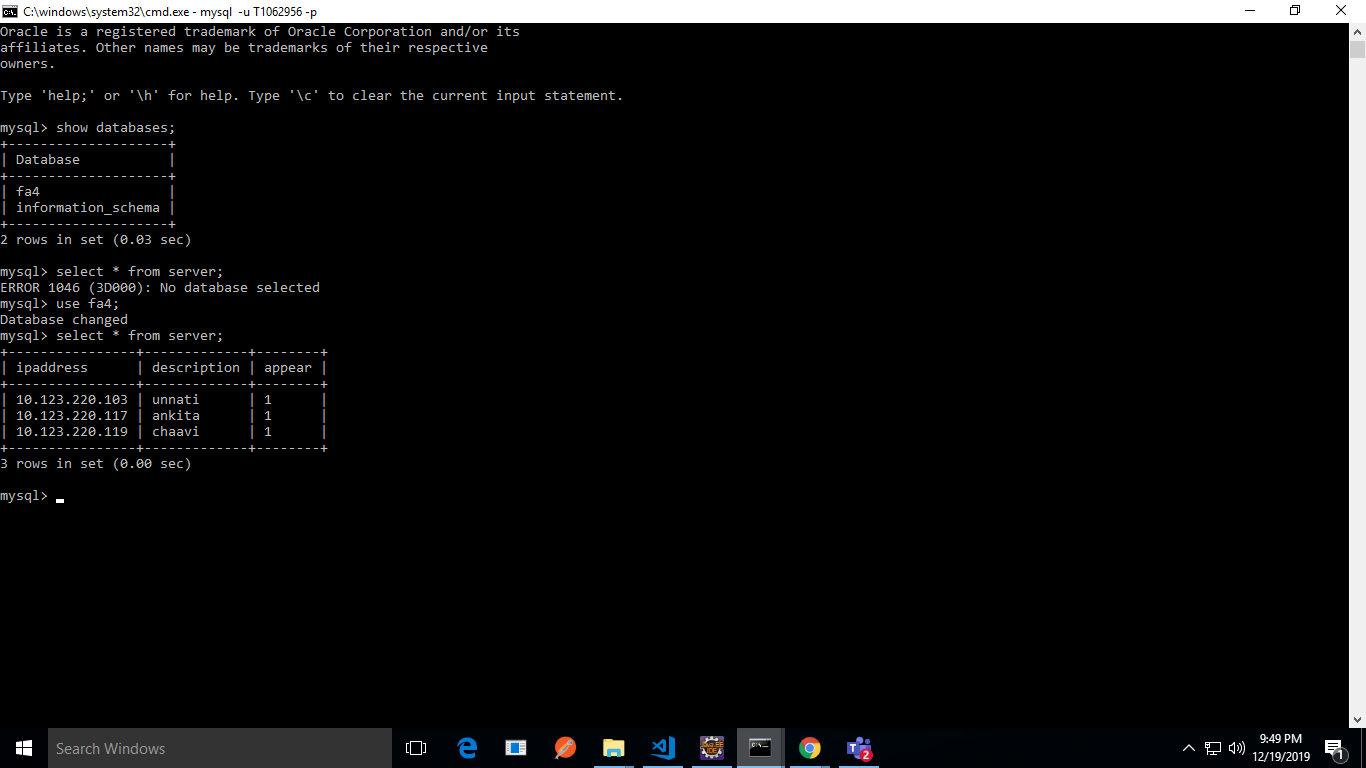
1. Client messages screen



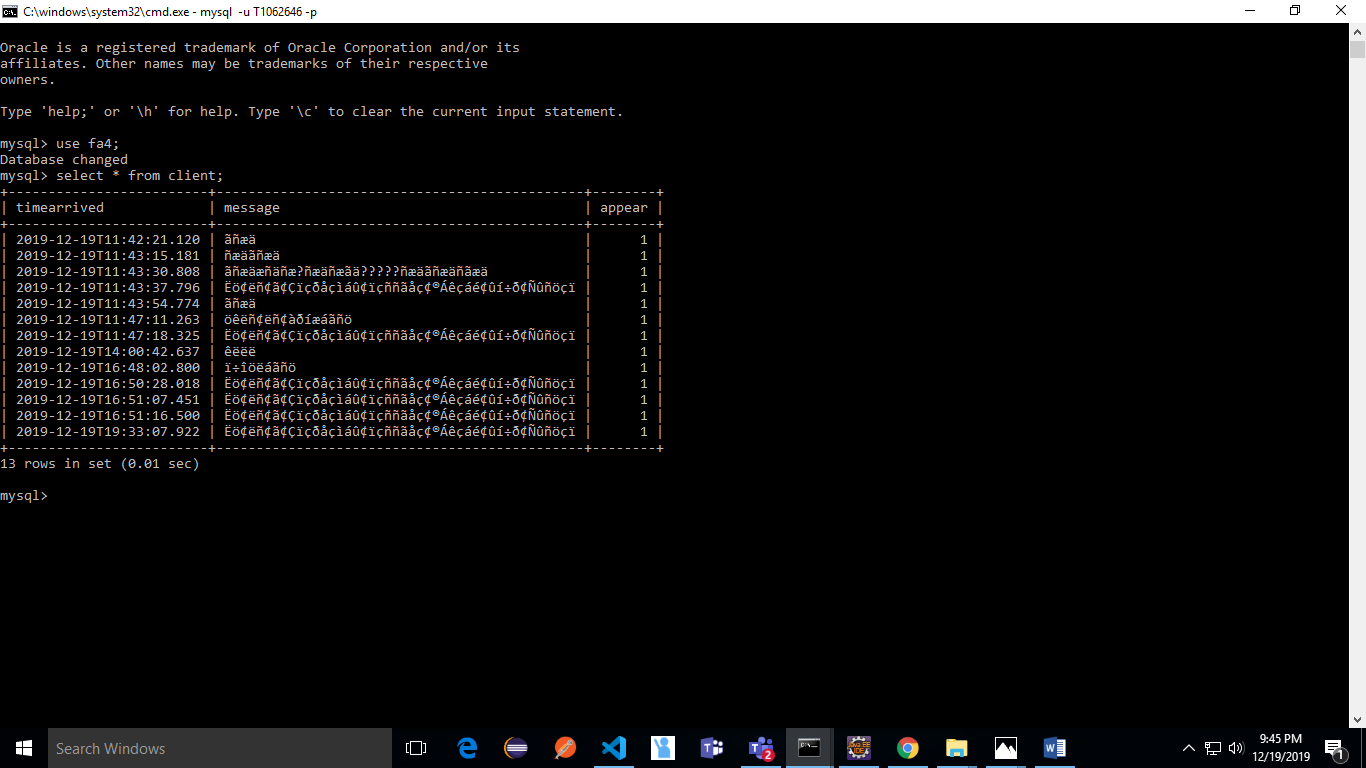
1. Client log-out screen



**Server-side tables**



**Client-side table**



# Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No** | **Condition to be tested** | **Input** | **Expected output** | **Actual o/p** | **Testcase pass/fail** |
| **1** | **All Screens should display in the format shown above** | **Input as mentioned above** | **As mentioned above** | **All screens are as shown in the figures** | **Pass** |
| **2** | **Admin should be able to register new client** | **Enter the new client information** | **The client should get added and displayed in the list below** | **Admin can successfully register client IP address with description** | **Pass** |
| **3.** | **Admin should be able to de-register the client** | **Select the client from the list and click on the De-register** | **The client should get de-registered successfully.** | **Admin can successfully de-register the existing client** | **Pass** |
| **4** | **Send messages.** | **The messages sent from the Broadcast Message Screen.** | **The messages should be received at all the clients** | **The admin can successfully send unicast, multicast and broadcast messages to the existing clients** | **Pass** |