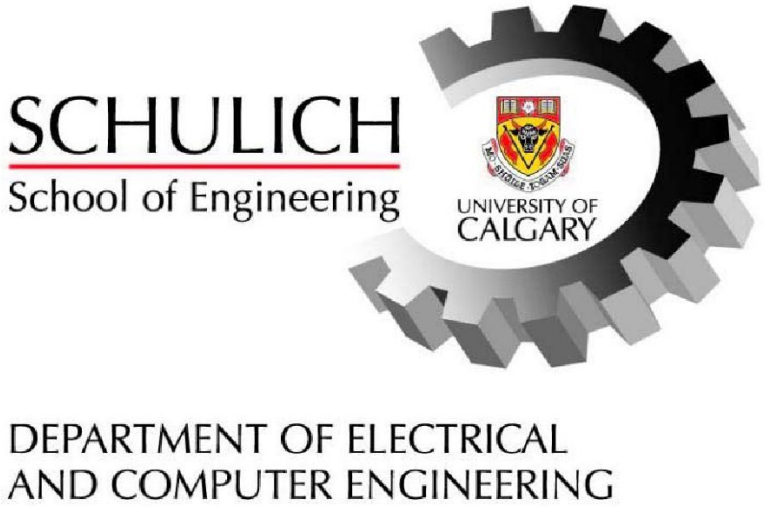
SENG 696 Agent-Based Software Engineering
Course Project



COVID-19 Alert MEDMAS (MEDical Multi-Agent System)

Project Report

Group 9

Jiayu Zhuang / Cheng Li / Xun Sun / Xinghan Chen

Table of Contents

1. Project Background

This project is to connect people with the health service and the lab. The idea of this system was motivated by the COVID-19 pandemic. The whole world has been suffering from this virus for 9 months. It is quite difficult to control the spread due to the infectiousness of the disease, and people are not aware of danger when they are close to a high-risk area since they don't have access to the latest information. Therefore, a system is to be set up to establish the interaction between residents, the government health service department, and medical labs so that the alert information could be sent to everyone timely. As a result, people could reduce exposure to risky neighbourhoods and lower the possibility of getting infected.

The system developed in this project is named COVID-19 Alert MEDMAS (MEDical Multi-Agent System).

2. Develop Environment

The system was developed in the following environment:

· Java 8 (JDK 1.8.0_271-b09)
· JADE 4.5.0 revision 6825
· MySQL Database 8.0.22 (For offline testing)
· Amazon Web Services Database, MySQL 5.7.29 (For online running)
· IDE: IntelliJ IDEA Community V2020.3

Due to the limit of time and resources, we were not able to develop a program with GUI. The system is demonstrated with text input/output in the IDE console.

3. Program Structure

This program comprises 4 agents, 3 of which (User, Lab, PHS) are running in the foreground while one (Alert) is running in the background. An online database is standby for data interchange. Ideally, there should be separate login entrances on different terminals for the three foreground agents so that they can operate simultaneously. However, that would make the development too difficult. Therefore, we start the system with one user logging in, and switch to other agents sequentially to demonstrate their functions.

The flowchart of the program is shown in Figure 1.

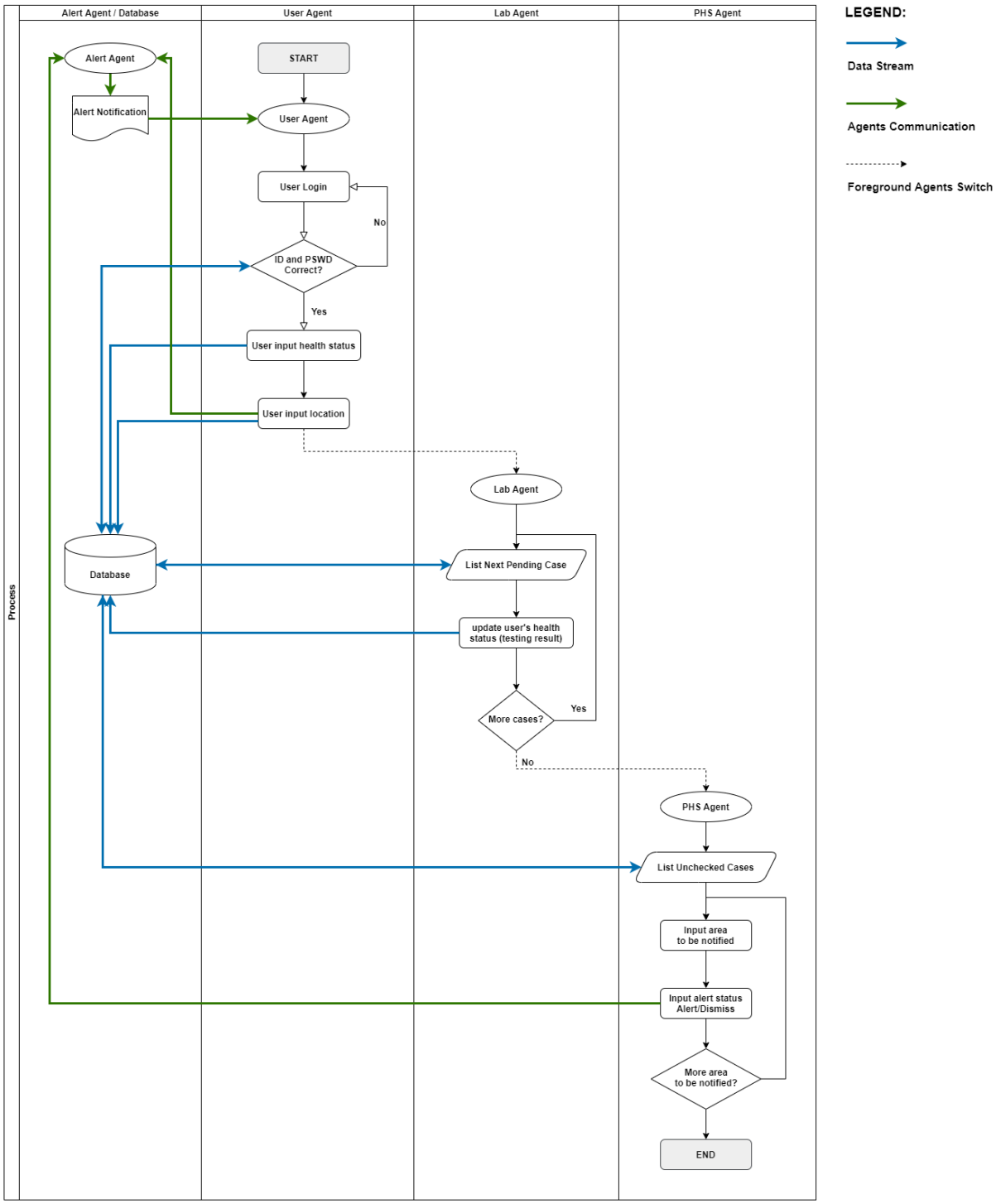


Figure 1 Execution flow of the program

4. Agents Definition

4.1 User Agent

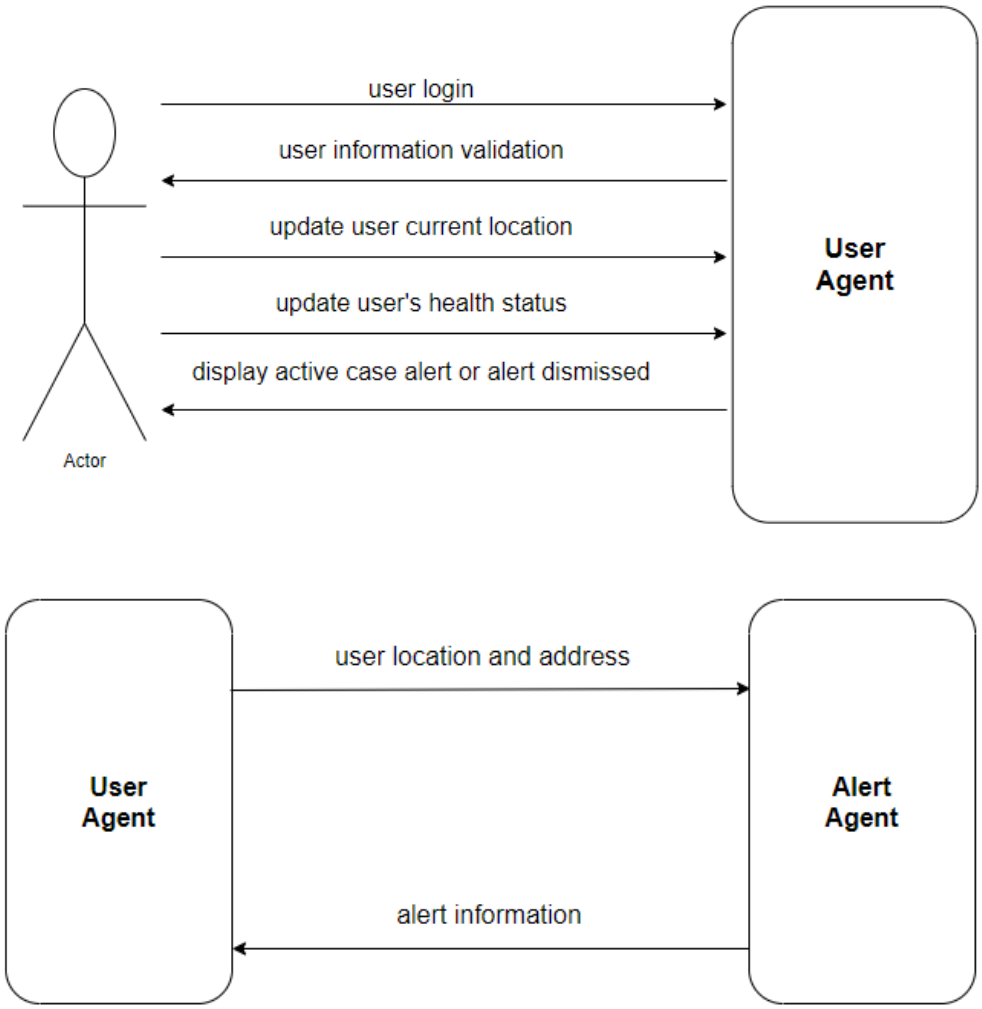


Figure 2 Process steps between Actor and User Agent

Brief Descriptions: User agent is designated for residential users who are the main users of this system. Residential users are supposed to be every resident of the city. They can update their locations, report their health status, and receive alerts.

Preconditions: User Profile is created before providing any service.

Post Conditions: If active cases appear or alerts are dismissed in specific areas, people in these areas or people living in these areas will receive the alerts.

Process Steps:

1. Actor requests logging in the system by providing username and password.
2. User agent compares the user information with the one stored in the remote database. Once they are matched, the user logins the system successfully.

3. User updates the current location and health status to the database.
4. User Agent requests from Alert Agent to get alert notifications in users' current locations and live addresses.

Exceptions:

1. The system will be terminated if unable to get access to the database.

Relationships:

- Initiating: Actor
- Collaborating: Alert Agent

Data Requirements:

- User's name and password
- User's current location and home address
- User's health status

4.2 Alert Agent

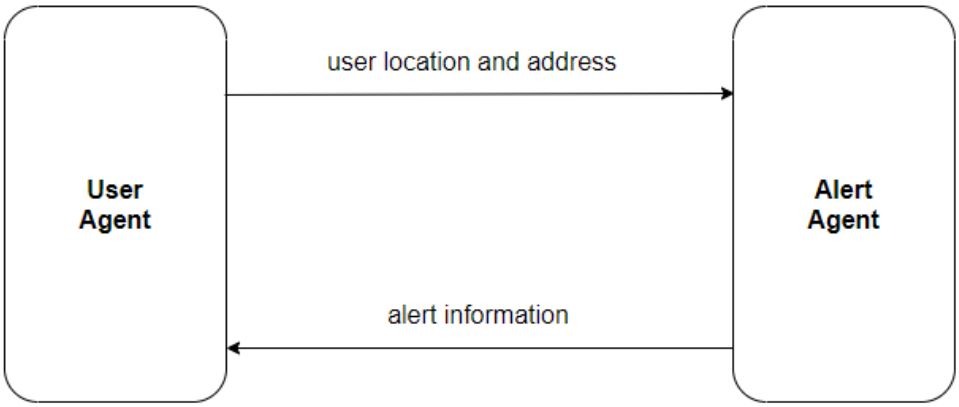


Figure 3 Process steps between User Agent and Alert Agent

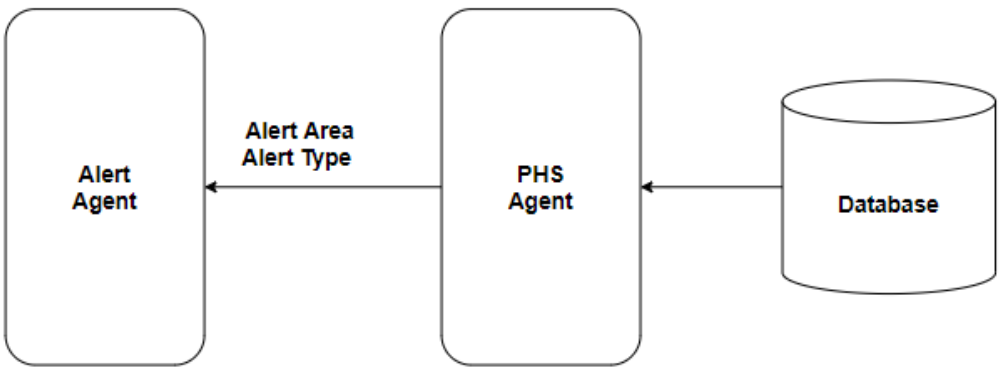


Figure 4 Process steps between Alert Agent and PHS Agent

Brief Descriptions: Alert agent is designated for the “Alert Center”, which could be a server that collects alert information and pushes notification to involved residential users.

Post Conditions: If users’ health statuses change in specific areas, Alert Agents will send alerts information to User Agent.

Process Steps:

1. Alert Agent requests users’ locations and addresses.
2. Alert Agent requests from PHS Agent to get alert information from PHS Web Service.
3. Alert Agent sends Alert information to User Agent matched with location and address.

Relationships:

- Collaborating: UserAgent, PHS Agent

Data Requirements:

- User’s current location and home address
- Alert area and alert type

4.3 Lab Agent

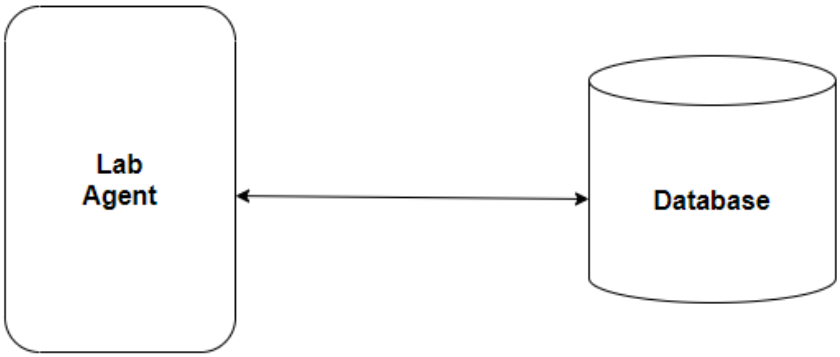


Figure 5 Process steps related to Lab Agent

Brief Descriptions: User agent is designated for medical lab users, who are responsible to update test results of COVID-suspected residents.

Post Conditions: If users’ health statuses are pending, then the Lab users need to update these statuses to positive or negative from the database.

Process Steps:

1. Lab Agent requests list of users whose health condition is “pending” from the database.
2. Lab Agent updates users’ health statuses in the database.

Relationships:

- Collaborating: Database

Data Requirements:

- User's current location

4.4 PHS Agent

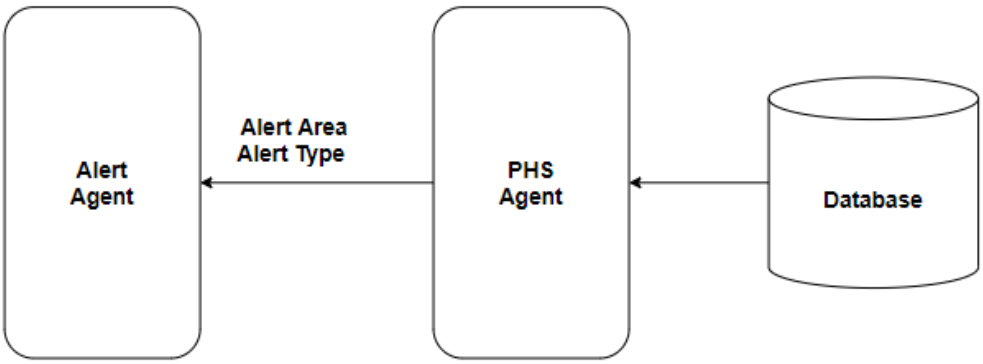


Figure 6 Process steps related to PHS Agent

Brief Descriptions: PHS is for "Public Health Service". This agent is designated for the government health department, which is authorized to check the latest case information and test results, and responsible to issue alerts to residents.

Process Steps:

1. PHS Agent requests from User Agent to get users' information whose health statuses are changed.
2. PHS Agent sends the alert information to the Alert Agent.

Relationships:

- Collaborating: Alert Agent, Database

Data Requirements:

- Users' health statuses and locations which are changed and unread.

5. Classes Definition

5.1 Config

Class 'Config' defines all the constants for other classes to make use of. It works as a configuration class and provides the programme users an easier access to the important variables, which can be edited when the global setting changes and hence become more robust in terms of other classes.

a. Start: As the run configurations in IDE says, class 'Start' is the main entrance of this java project. It's a subclass extending Agent, setting the four agents up via the 'createAgent' method, which is designed to create an agent with a corresponding AID.

5.2 User

Class User defines a number of methods for the user agent object to use.

a. Method action() of the inner class UserBehaviour defines the initialized actions for the user agent to login and build connections with other alert distributing agents.
b. Method login() helps a user to login the agent client with its health ID and password, after the successful authentication, a user can update its health condition and current location.
c. register() helps the user agent register in the system.
d. showAlert() prints the alert notification when available.

5.3 PHS

Similar to Class User, PHS extends Agent containing methods for its object to fulfill the jobs of Public Health Service. Therefore, some of the methods function in a similar way as those of User, such as register(), action() and setup(). However, PHS also has many unique methods.

a. getUpdatedRecords() connects the PHS agent to the database and constantly queries the information of unchecked users.
b. sendAlert(), defined inside the inner class PHSBehaviour, can set alert messages and phs agent's receivers, and send the messages to the latter.

5.4 Lab

As another subclass of Agent, Lab has its specialized methods apart from the similar ones described above.

a. retrieveUserInfo() returns the records of suspicious users.
b. processUser() helps the employees from the medical lab to update the health condition of suspicious cases manually.

5.5 Alert

Alert class is responsible for sending alert notifications from PHS agents to involved users.

6. Database Structure

There are 7 keys in the database and we prepared 5 records for testing and demonstration.

Id	name	loc	addr	healthCon	checked	pwd
1	Mike	Dalhousie	Brentwood	0	1	aaa
2	Tom	Brentwood	Downtown	1	1	aaa
3	Jerry	Downtown	Downtown	1	1	aaa
4	Haley	Varsity	Varsity	2	0	aaa
5	Alex	Varsity	Tuscany	0	0	aaa

Figure 7 Database contents

For the current version, all the records refer to residential users.

Field	Description	Type
Id (Primary)	User's health card ID	int
name	User's legal name	varchar
loc	User's current location	varchar
addr	User's home address	varchar
healthCon	User's health condition (COVID test result) 0 = healthy (negative), 1 = suspected (result pending), 2 = positive confirmed	int
checked	PHS unread flag 0 = unread record, 1 = checked record	int
pwd	User's login password	varchar

The program interacts with the database by MySQL commands.

7. Communication Specification

This part is for future versions of this project (Online deployment). For the communication between the agent and the database, we are using the protocol and format provided by JDBC and MySQL database.

For the communication between agents, the protocol and format is provided by jade. SOAP is used here. For this demo, this is not implemented.

```
<user>
  <AID>User's AID</AID>
  <String>User's location and address</String>
</user>
```

Figure 8 Message between user and alert

```
<alert>
  <String>location</String>
  <Boolean>status</Boolean>
</alert>
```

Figure 9 Message between PHS and alert

```
<alert>
  <String>location</String>
  <Boolean>status</Boolean>
</alert>
```

Figure 10 Message between alert and user.

```
<Result>
  <Boolean>Was the message received(true: yes, false: no)</Boolean>
</Result>
```

Figure 11 Confirmation for the result of the message delivery

8. System Demonstration

Program Start:

```
INFO: ----------------------------------------
Agent container Main-Container@192.168.1.101 is ready.
----------------------------------------
--- Agent Created: ( agent-identifier :name User1@192.168.1.101:1099/JADE )
--- Agent Created: ( agent-identifier :name PHS@192.168.1.101:1099/JADE )
--- Agent Created: ( agent-identifier :name Alert@192.168.1.101:1099/JADE )
--- Agent Created: ( agent-identifier :name Lab@192.168.1.101:1099/JADE )
--- ALERT AGENT STARTED ---
```

User login:

```
--- SWITCH TO USER AGENT ---
 *** User Agent: Enter your Health ID:
1
 *** User Agent: Enter your Password:
abc
User Agent: Log in FAILED!, Try again!
 *** User Agent: Enter your Health ID:
1
 *** User Agent: Enter your Password:
aaa
```

User COVID status and current location update:

```
 *** User Agent: Enter your COVID status: (0-Negative, 1-Suspected/Test Result Pending, 2-Positive Confirmed)
1
 *** User Agent: Enter your current location:
Dalhousie
User Agent: COVID information updated!
```

Lab updates user's test result:

```
--- SWITCH TO LAB AGENT ---
Current Case: ID: 1 , Name: Mike
 *** Lab Agent: Please update the user's test result, 0-Negative, 2-Positive, 3-Skip, 4-Quit
0
Lab Agent: COVID test result updated!
Current Case: ID: 3 , Name: Jerry
 *** Lab Agent: Please update the user's test result, 0-Negative, 2-Positive, 3-Skip, 4-Quit
2
Lab Agent: COVID test result updated!
Current Case: ID: 4 , Name: Haley
 *** Lab Agent: Please update the user's test result, 0-Negative, 2-Positive, 3-Skip, 4-Quit
2
Lab Agent: COVID test result updated!
No more pending cases.
```

```
--- SWITCH TO PHS (Public Health Service) AGENT ---
List of unchecked cases:
Current Location / Home Address / Case Result (0-Negative 2-Positive)
Dalhousie / Brentwood / 0
Downtown / Downtown / 2
Varsity / Varsity / 2
Varsity / Tuscany / 0
 *** PHS Agent: Enter the area to be notified. Enter 'q' to exit:
Downtown
 *** PHS Agent: Enter the notification type of this area. 1-ALERT, 0-DISMISS. Enter 'q' to exit:
1
Alert Agent: Information from other agents received.
 *** PHS Agent: Enter the area to be notified. Enter 'q' to exit:
Dalhousie
 *** PHS Agent: Enter the notification type of this area. 1-ALERT, 0-DISMISS. Enter 'q' to exit:
0
Alert Agent: Information from other agents received.
```

Alert Agent receives communication from other agents:

```
Alert Agent: Information from other agents received.
```

Alert received by involved users:

```
!!! User Agent: Notification Received: Brentwood area is now on COVID ACTIVE CASE ALERT!
```

```
!!! User Agent: Notification Received: The COVID active case alert of area Dalhousie is now DISMISSED!
```

9. Important Developing Event Logs

Nov 6, 2020
Project started.

Nov 15, 2020
JADE develop environment confirmed.

Nov 19, 2020
GUI development cancelled.
Multi-terminal operation cancelled.

Nov 25, 2020
First version tested.

Dec 9, 2020
Full-function version pushed on Github and tested.

Dec 11, 2020
Online database implemented.

Dec 15, 2020
Optimization and bug fixes completed.
Final version released.

Project GitHub: https://github.com/AceOfSpadeszzz/SENG696