

Lebanese American University
Department of Computer Science & Mathematics
Computer Networks
CSC 430



Project: COVID@UNI Report
Instructor: Sanaa Sharafeddine Dawy
Members:

Hasan Mshawrab	-201905353-	hasan.mshawrab@lau.edu
Nahla Baalbaki	-201905353-	nahla.baalbaki@lau.edu
Ali Knayber	-201905353-	ali.knayber@lau.edu
Yara Chahine	-201905353-	yara.chahine@lau.edu

Part I : Client-Server	3
Description:	3
Server Package Classes:	4
MultipleServer:	4
CreateNewAccount:	4
UserLogin:	4
DatabaseInfo:	4
Locations:	4
InsertLocation:	4
UserStatus:	4
QuarantineDays:	4
CheckExposedUsers:	5
Framework Package (Client) :	5
Main:	5
MyFrame:	5
SignUpFrame:	5
AccountPageFrame:	5
Protocol Description:	5
Functionalities:	5
Signing up:	5
Login:	7
Check number of Active Cases:	8
Add trusted friends:	8
Check friends' status:	9
Submit PCR:	9
Get Quarantine Days:	10
Tips and Instructions:	10
Location Tracking and User Status Update:	11

Part I : Client-Server

Description:

A “Covidless” campus; that is the goal of our application. As per the requirements of the project, our application implements a client-server paradigm, where the client is a student at the university. This student can sign up or login to our application. While logged in, the server keeps track of the student’s location, as well as the time the student enters a new location. The student can retrieve information from the server, such as the number of active cases at the university. Moreover, a student can add “Trusted Friends” with whom they don’t mind sharing their health status, and they can check on the health status of other students who marked them as “trusted”. If a student is exposed to someone who is contagious, they will be marked as at “Risk”. This report will include a detailed explanation of how each requirement was implemented.

First, we will start by explaining our project structure such as how we divided our classes to later be able to explain how our functionalities were implemented.

We used two packages in our project, a Server package and a Client package called “Framework”.

Server Package Classes:

1) **MultipleServer:**

This is our Server class. We start by establishing a connection with the database since our server is the link between the client and database. Then, we create a client socket and start listening for incoming connections. We then assign the client socket, database connection, input and output stream to a thread that handles the client's requests.

2) **CreateNewAccount:**

This class's constructor takes the user's info from the signup page and adds the user's information to the database. The client is now able to log in.

3) **UserLogin:**

This class is responsible for authenticating the user logging in by checking if the username and password (hashed) (given in the constructor by the server) matches an entry stored in the database.

4) **DatabaseInfo:**

This class contains all the functions that execute queries related to the database. Whenever the server needs to insert or get information to and from the database, it does so by calling one of the functions in the DatabaseInfo class such as getPCRresult(), updateUserStatus().

5) **Locations:**

This class has a public string array containing the locations at the University. It has a function that provides a random location from the array. This location is set by the server whenever a user is online.

6) **InsertLocation:**

This class extends the Thread class, it updates the user's location every 5 seconds (only for testing purposes, this time can be elongated to meet with realistic scenarios), and saves the location in the database as well as the time in which the user entered the given location.

7) **UserStatus:**

The server uses this class to send the user instructions based on the user's health status (Safe, contagious, at risk).

8) **QuarantineDays:**

This class takes the user's username and returns the number of days a user has been in quarantine. (More details in the functionalities section).

9) CheckExposedUsers:

This class extends the Thread class. It is always running as its main role is to check if the user has been in proximity of another infected user for a given period of time, and updating the user's status accordingly.

Framework Package (Client) :

1) Main:

This class creates a new client with the port on which the server is listening, and saves the client's input and output stream to be given as parameters to the new MyFrame() class object.

2) MyFrame:

This class shows the UI frame where the user can either sign in or sign up (directs the user to SignUpFrame()).

3) SignUpFrame:

This class allows the user to sign up.

4) AccountPageFrame:

After a user signs in, the AccountPageFrame is displayed where the user has options to choose from (as buttons). These options are the following remaining classes: CheckActiveCasesFrame(), CheckFriendsFrame(), SubmitPCRFrame(), ViewTrustedListFriends(), CheckQuarantineDays().

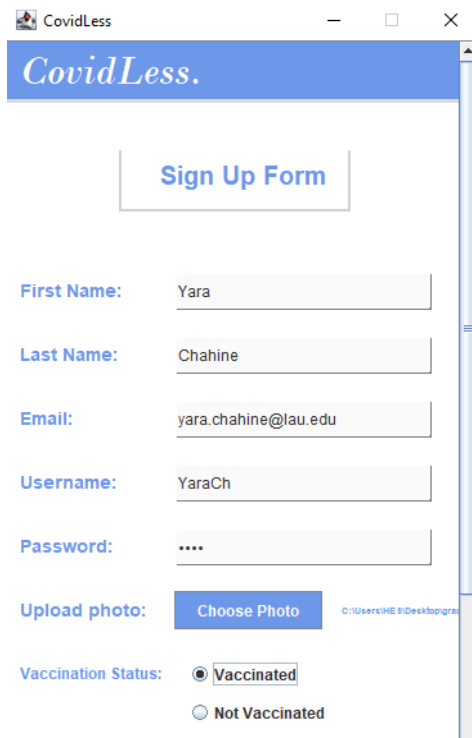
Protocol Description:

Functionalities:

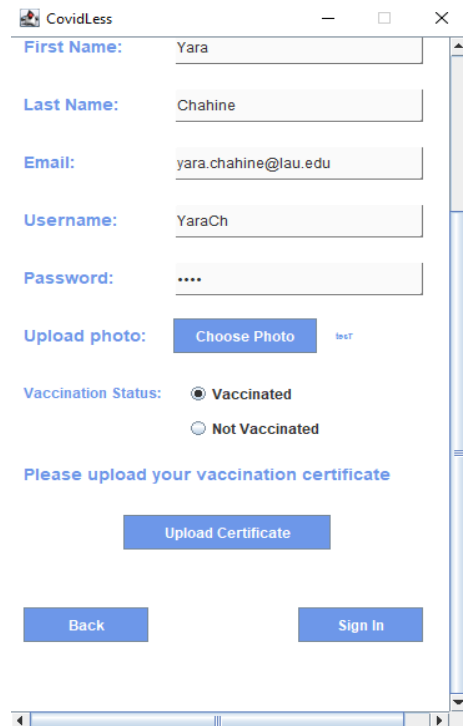
In the following, we explain the main functionalities in our application, we provide alongside each functionality, the corresponding frame snapshot of our application to better visualize the functionality.

1) Signing up:

To be able to use our application, a student (the user) has to sign up first by entering the information visible in the below frame in SignUpFrame() class. Once a user clicks on sign in button, an event listener will send the entered information through the output stream to the server. The server then creates the CreateNewAccount class passing to it the sign up info, and then calls other classes and functions to start tracking the user's location, health status and other updates. (Please refer to the code for further details)



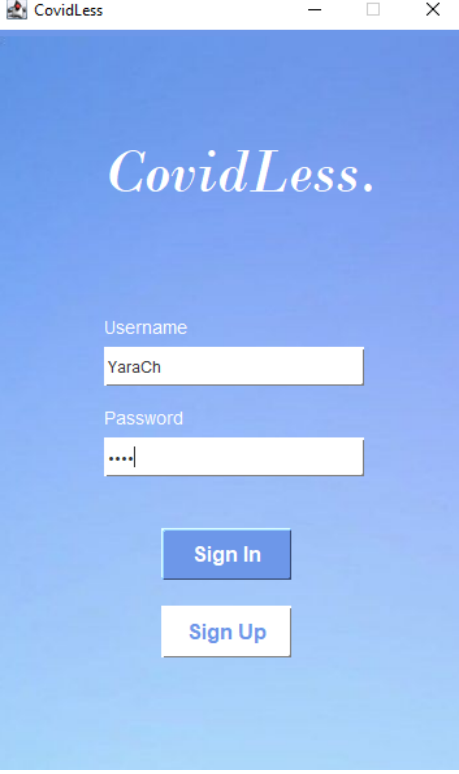
The screenshot shows a web browser window titled "CovidLess". The header is a blue bar with the text "CovidLess." in white. Below the header, there is a box labeled "Sign Up Form". The form contains several input fields: "First Name:" with the value "Yara", "Last Name:" with the value "Chahine", "Email:" with the value "yara.chahine@lau.edu", "Username:" with the value "YaraCh", and "Password:" with the value "....". There is also an "Upload photo:" section with a "Choose Photo" button. At the bottom, there is a "Vaccination Status:" section with two radio buttons: "Vaccinated" (selected) and "Not Vaccinated".



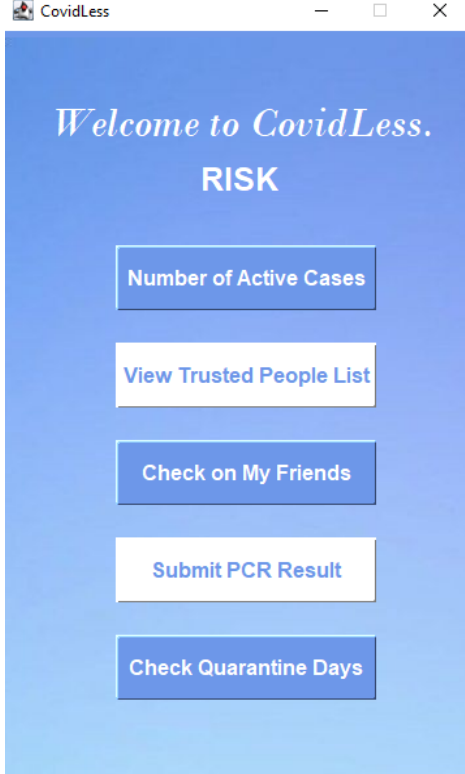
The screenshot shows the same web browser window titled "CovidLess". The form fields are now populated with the following values: "First Name:" "Yara", "Last Name:" "Chahine", "Email:" "yara.chahine@lau.edu", "Username:" "YaraCh", and "Password:" "....". The "Upload photo:" section has a "Choose Photo" button. The "Vaccination Status:" section has two radio buttons: "Vaccinated" (selected) and "Not Vaccinated". Below the vaccination status, there is a text prompt "Please upload your vaccination certificate" and an "Upload Certificate" button. At the bottom, there are two buttons: "Back" and "Sign In".

2) Login:

Our application allows the university's students to login to be able to access our other functionalities. The user sends username and password to the server through the output stream in the MyFrame() class, to the server. The server then sends reads info from the client's input stream to the UserLogin(). Once a user signs in, the AccountPageFrame() is displayed, showing our different functionalities.



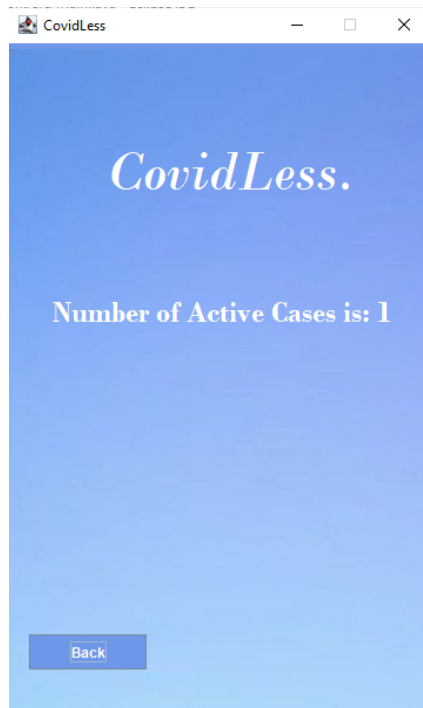
The image shows a window titled "CovidLess" with a blue gradient background. At the top, the text "CovidLess." is displayed in a white, italicized serif font. Below this, there are two input fields: "Username" with the text "YaraCh" and "Password" with masked characters "....". Under the password field, there are two buttons: "Sign In" (blue with white text) and "Sign Up" (white with blue text).



The image shows a window titled "CovidLess" with a blue gradient background. At the top, the text "Welcome to CovidLess." is displayed in a white, italicized serif font, followed by the word "RISK" in a bold, white, sans-serif font. Below this, there are five buttons stacked vertically: "Number of Active Cases", "View Trusted People List", "Check on My Friends", "Submit PCR Result", and "Check Quarantine Days". All buttons have a white background with blue text and a thin blue border.

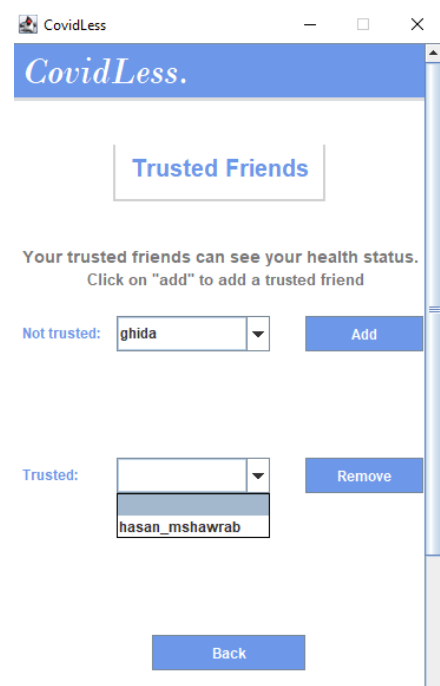
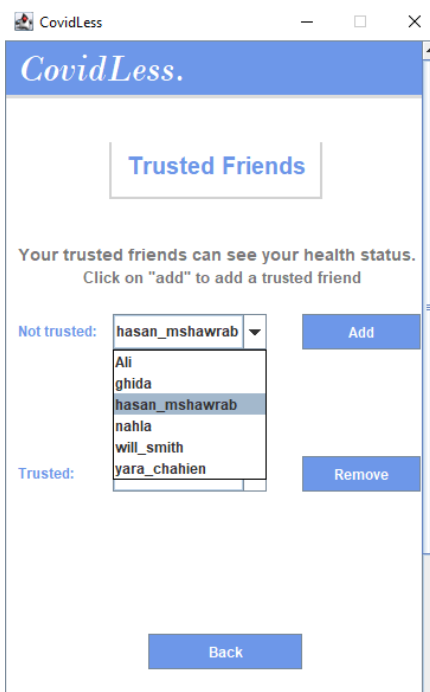
3) Check number of Active Cases:

When a user clicks on the first button, they are able to check the number of active covid cases in the university. This is done by reading from the client's input stream the number sent by the server, which in its turn received it from the DatabaseInfo.getActiveCases() method.



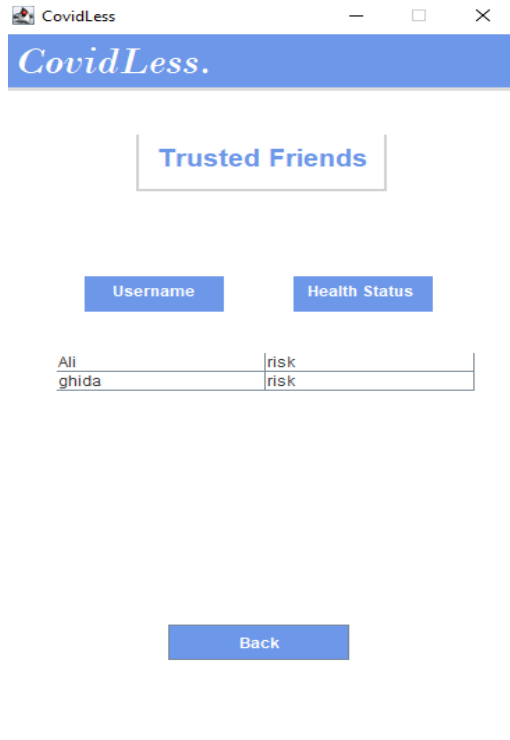
4) Add trusted friends:

A student can add trusted users to their trusted friends list. Trusted friends can see the user's health status. This is done by sending the trusted friend's name to the server when the user clicks on "add". The server then adds it to the database by calling `databaseInfo.insertFriendToMyTrustedList();` The user can also remove a trusted friend (**Additional feature**) by clicking on remove. Whenever a user adds a trusted friend, they can immediately see it in the trusted friends drop down menu as shown below.



5) Check friends' status:


The logged in user can see the health status of every other user that has previously added them as trusted. The users' info is sent to the logged in user by the server which called the `DatabaseInfo.getFriendsListedMeTrust()` function.



Username	Health Status
Ali	risk
ghida	risk

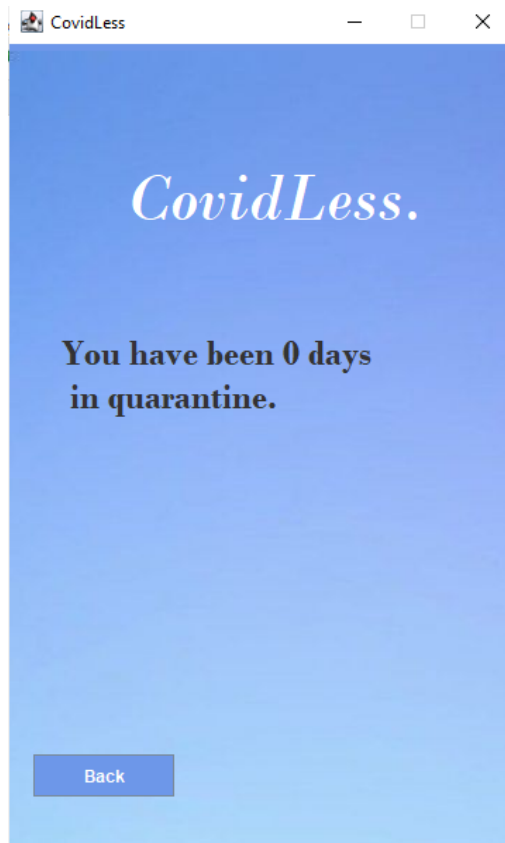
6) Submit PCR:

A user can submit their PCR result to the server which updates their health status according to the result by executing the `databaseInfo.insertPCRResult()` function.



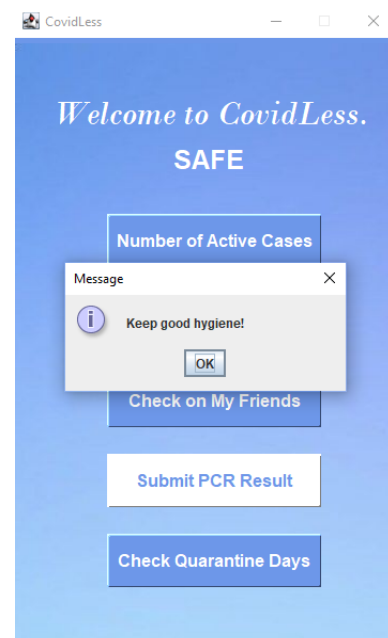
7) Get Quarantine Days:

Whenever a user submits a positive PCR result, the server also stores the time of submission (date infected) in the database. When a user clicks on “Check Quarantine days”, the number of quarantine days is displayed, previously sent from the server that got them by calling the `databaseInfo.getQuarantineDays()` function. This function subtracts the infected time from the current time and returns it to the server.



8) Tips and Instructions:

The user receives tips and instructions constantly (thread) from the server which retrieves instructions by calling `userStatus.getInstruction()` function.



9) Location Tracking and User Status Update:

We leave the best for the last. One of the most important functionalities of our application, is the ability to track the user's location and update it whenever the user is online. The application must notify the user whenever they have been in contact with another user who is contagious (has submitted a positive PCR) for more than three minutes. Whenever this occurs, the user's health status is updated to at risk (as shown in the account page capture above).

To implement this, the server has to run the `InsertLocation()` thread whenever a user is online (this thread updates the user's location and time of arrival constantly as explained above). Then, the server launches a `CheckExposedUsers()` thread class, which checks whether the user has been in the same location with an infected user for three minutes or more.

This is calculated by getting the difference between the user and the infected user's arrival times. Our algorithm takes into consideration who arrived first. If the logged in user arrives first, we start counting the minutes after the infected user arrives. Else, we count the minutes starting from our logged in user's arrival time. If the time calculated exceeds three minutes, the function `databaseInfo.setUserStatus("risk")` is executed to change the user's status to at Risk.