MOBILE NETWORKS & APPLICATIONS

EECE 451

FALL 2021-2022

PROJECT

Project Report

GROUP 2

Tracy Nader (tmn06) – 201900412

Paul Sassine (prs01) – 201902675

Noura Bakhos (nrb16) – 202000413

I – Application Functionality:

This application helps manage the vaccination process of a specific population/country. Users can simultaneously connect to a certain server due to the implementation of multi-threading. The application is also connected to a database; thus, users are able to maintain accounts and log in at any time. Users are classified into 3 types: Patients, Admin, or Medical Personnel. Patients can access their vaccination information and update it; the update is successful if an email is received. In the event that a patient does not have an account on this application, it is possible to create one by switching to the registration/sign up form. This will add the new patient's information to the existing database. Admins can access both patient and medical personnel information, and the medical personnel can access patient information.

Since we have only discussed the functionalities from the client side so far, we will move on to discussing the functionalities of the server. Besides having the normal functionalities of a multi-threaded server, this server also maintains a database connection. Firstly, the server reads the user input (which could be the name, username, password, and/or email of the user) and then establishes a connection to the MySQL database. It then sends the query to the database and awaits a response (which could mean either an unsuccessful/successful login or an unsuccessful/successful registration). The user is then notified of the outcome.

II – Overview on Structure of Code:

The entire code was written in Java and the GUI was implemented using JavaSwing (and via WindowBuilder). The code is made up of one package which includes several classes. The login form, registration form, server code, and client code are all separate classes. The application also contains access classes for each type of user (admin, personnel, or patient), as each type of user can access different types of information. The same package also includes linked images (used for our GUI) and the .jar file used to connect to the MySQL database.

To run the code, first run the Server class and then the TCPClient class. The registration form will subsequently pop up. The rest of the application's functionalities are made available to the user once they successfully log in.

III – Design Decisions:

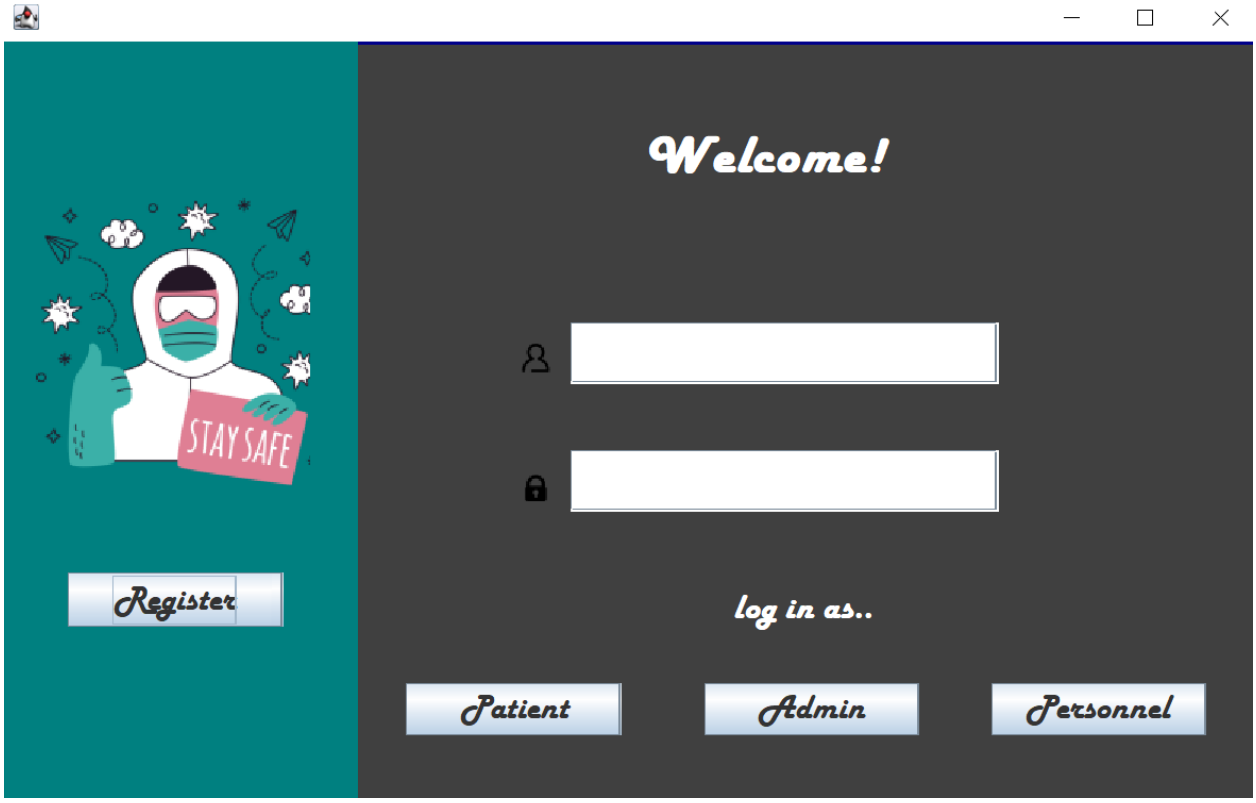
We decided to add the "log in" and "register" buttons to both the log in and registration forms. This was done so that users have the complete freedom to move between those two forms.

We also decided that queries would be sent to the database from the server (and not directly from the registration or log in forms), as the server is the entity in charge of connection establishment. Furthermore, we wanted queries to be sent from one source to make it easier to debug any database-related issue.

The theme, font, and overall GUI design were chosen to be as simplistic as possible (but also not visually boring), in order to allow the user to easily navigate through this application without feeling like this intuitiveness takes away from the design/aesthetics.

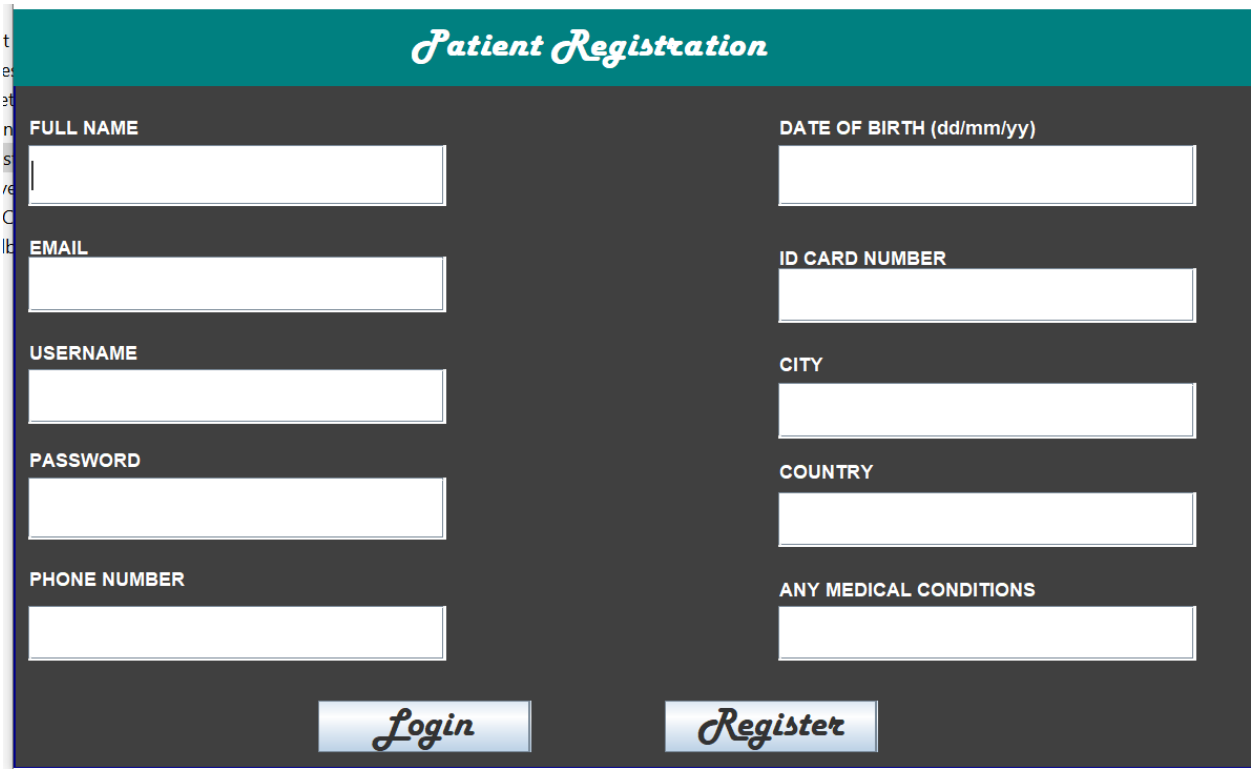
IV – GUI Design:

Upon running the application from the client side, the login page pops up and prompts the user to login as shown below:



Once the user enters their credentials, they must press the appropriate button (i.e: logging in as a patient or admin or personnel). If a user presses the button that doesn't match their type (i.e: patient presses admin button), they will not be able to log in.

Only patients can create an account; medical personnel and admin(s) can only log in. This can be done by clicking the "register" button which redirects patients to the registration page as shown below:



If the registration is successful, the patient is then prompted to log in.

Also, if the patient already has account and mistakenly pressed on the registration button, clicking on the login button will redirect them to the login page.

Upon logging in successful, all users (regardless of type) will receive this notification:

Through our connections to the database, we'll make sure if the username and password are valid or not (i.e: if they match or not). If they're valid, then that means the user has successfully logged in, and they can now proceed to view or edit information based on their user type.

V – Errors Faced/Error Handling:

We faced several errors in connecting to the database. Most of our errors were silly ones related to spelling and syntax which we fixed using the documents provided on Moodle.

We also faced errors with the login and registration forms; those errors were related to sending the query to the server side. We were able to fix it through debugging/trial and error.

Finally, we also had issues with implementing the "search" functionality which varies based on the different type of user (admin, personnel, patient).

VII – References:

1. Fadatare, R. (2020, September 14). *Registration Form using Java Swing + JDBC + MySQL Example Tutorial*. Java Guides. https://www.javaguides.net/2019/07/registration-form-using-java-swing-jdbc-mysql-example-tutorial.html **(used as template to create the registration form)**

2. *Java Save File in Oracle Database - javatpoint*. www.javatpoint.com. (n.d.). https://www.javatpoint.com/storing-file-in-oracle-database **(used for the creating documents functionality)**

3. Singh, C. (2015, July 5). *Java Swing Tutorial for beginners*. https://beginnersbook.com/2015/07/java-swing-tutorial/ **(used as template to create the login form)**

4. Eclipse Foundation. (n.d). *WindowBuilder*. https://eclipse.org/windowbuilder/ **(used to learn the WindowBuilder tool in order to freely design our application)**

5. Java Create and Write To Files. (n.d.). https://www.w3schools.com/java/java_files_create.asp **(used for the accessing documents functionality)**

6. *JDBC Tutorial: What is Java Database Connectivity(JDBC) - javatpoint*. www.javatpoint.com. (n.d.). https://www.javatpoint.com/java-jdbc **(used to establish database connection)**