**DS Lab 3 Execution Instructions**

**Executing the project:**

* This a java project and was developed using eclipse IDE and its best suited to be executed on Eclipse IDE.
* The project can be broadly divided into four parts MQServer, Student, Advisor and Notification and has the packages with the same name in src folder, please import src folder to your eclipse IDE and run it as java application
* Each package of MQServer, Student, Advisor and Notification has two java class files \*ProcessName\*Main, and \*ProcessName\*GUI respectively.
* To execute the project the files should be exported into Eclipse as java project and run the ServerMain file first all other \*ProcessName\*Main class files.
* You must run the ServerMain file only once and can run \*ProcessName\*Main file as many times as you want.

**Execution process of the Application:**

Start the server before you run the application, you can start the server by running ServerMain class file.

**MQ Server: -**

As specified in the lab 3 requirement you must start the MQ Server first for the application to run without any hindrance though even if at any given time MQ Server goes offline the data which it has will not be lost which makes this a persistent model.

* To start the server run MQSMain java class file
* Once you run the file a gui will open which will open the network for other process to communicate
* You can close the GUI MQ Server when ever you want however when MQS is offline process won’t be able to communicate until it comes back online

**Student Process: -**

As specified in the lab 3 requirement, in student process the user will be able to enter the a string which is presumed to be username and the class he want to join. User can send as many string as he want at will unit he chose to stop or close the student process by either clicking on the red cross button on the top or the entering “end” into GUI. If MQS is offline student will not be able to send the message but already sent message will not be lost.

* To start the Student process run the StudentMain java class file
* Once its start it will show if its connected to network
* Enter the string containing student name and subject
* You can send as many strings as you want
* If MQS is offline your message will not be sent, and you will be should a message stating the same
* Student process can be closed at anything the user wants to.

**Advisor Process: -**

As specified in the lab 3 requirement, in advisor process the user need not do anything once the process is started it will check if the MQS is online and if online will starting pulling the data from it (data from student process requiring action from advisor) if these any data advisor will act on it and push back to server and if there isn’t any data it will sleep for 3 seconds before pulling data again

* To start the Advisor process run the AdvisorMain java class file
* Then the process will check if MQS is online, if offline it will show you the message stating that its offline and will try again in 3 Second
* If MQS is online, it will pull the data from MQS on which it has to act
* After pulling data it will apply random probability on it give a verdict to the data
* After giving a verdict it will display the verdict as action taken
* And then again push the data into MQS
* It will sleep for 3 Second if it doesn’t have any data to work on before polling the MQS again.

**Notification Process: -**

As specified in the lab 3 requirement the notification process pulls the data from MQS after the data is modified by Advisor process with a verdict. Once there is not data to pull it will sleep for 7 seconds,

* To start the Notification process run the NotificationMain java class file
* Then the process will check if MQS is online, if offline it will show you the message stating that its offline and will try again in 7 Second
* If MQS is online, it will pull the data from MQS and display the message
* Once displayed it will sleep for & seconds and then again it will check the data in MQS if available and repeat the same process.

**References:**

The program has been written from scratch and may not find any code repeated from the references, so the references are not commented in the code however project does gets inspired from the below links for specific functionalities implementations.

**Client and Server architecture:**

<https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html>

**Server Multithreading**: -

https://github.com/abhi195/Chat-Server

**GUI: -**

https://www.youtube.com/watch?v=9gDErZCtdzM