PROJECT MILESTONE -2 DOCUMENT

Server.java :

The Server.java opens a socket on a particular port. Wherein it accepts requests from clients to start a connection. This file has a location to the SQLite database that stores details for the event passed from the command line. The database is queried in this part once request is sent from the client. Server.java accepts information from the client and checks the information from the database, and if it exists returns an "OK" or "EVENTS;<number of rows having same time>" depending on the type of EVENT_DEFINITION sent by the client.

Client.java :

The Client.java looks for a server on a particular port and connects on the port the server is listening on. It sends event information to the server using the getopt function, after receiving the information (accepted or error) it continues to listen for another request from the client.

makefile :

Makefile is used to compile the two java classes. Typing 'make' will invoke the first target entry in the makefile (the default one in this case). This target entry builds the Server class the Server.class file is dependent on the Server.java file and the rule associated with this entry gives the command to create it. To start over from scratch, type 'make clean' which Removes all .class files (although not necessary since it overwrites the existing .class files if same code is compiled), so that the next make rebuilds them.

run.sh :

It calls the makefile and runs the java classes, having the SQLite (sqlite-jdbc.jar) and get-opt (java-getopt.jar) jar files in the classpath.

test.db :

This is the database we pass as an argument in our script.

flow of the program :

1. Server opens a socket on a particular port and listens for any incoming requests.

2. Client sends a request to connect along with arguments. The arguments are passed from the client using get-opt function.

3. Server accepts the request and acknowledges the client.

4. Server inserts the event sent by the client, in the local SQLite database (test.db).

5. Once inserted, it returns a statement containing an "OK" appended to the arguments sent by the client.

6. If the client passes "EVENT_DEFINITION" as the first argument in the event variable, the server checks for the event and returns an "OK" appended to the arguments sent by the client.

7. If the client passes "GET_NEXT_EVENTS" as the first argument in the event variable, the server checks for multiple events in the database, in the same group that occur at the same time and it returns those events with the event counts.

8. Also the server checks if the event has expired, if yes, it returns a statement representing the event has expired and remove the entry from the database.