Test Plan Description

Testing SQL Repository

For testing this repository, there are three main cases to consider. Either the query within the repository we want to execute takes parameters or it doesn't. If it doesn't take parameters then that query is a "get" query, where we are only retrieving data. If the query we want to execute does take parameters, then that query is either a "get" query or an "insert" query. Currently, the SQL Repository is defined within an XML file called sql_repo.xml and the only way to access this file is by calling the executeSelectedQuery() method within the ConnectToDB class.

No Parameters Query: In the current version of our project, there is only one query with no parameters, which is the "getSystemDateTime". Normally a "query" node within this XML file would contain two inside nodes, "sql" and "parameters". However, the sql node will always contain two attributes that help the executeSelectedQuery() find out how to proceed. The first attribute is "paramNum" which defines how many parameters is the SQL query expecting (in this case, zero parameters for SELECT NOW()) and the second attribute is "insert" set to 0 if we are retrieving data, and 1 if we are inserting.

```
<queries>
    <query name="getSystemDateTime">
        <sql paramNum="0" insert="0">SELECT NOW()</sql>
    </query>
```

(Fragment from sql_repo.xml)

For this case, it is sufficient to just call the executeSelectedQuery() method (which expects two parameters: name of the query to be executed and a String array with the parameters of the matching SQL query). Since we are using a getSystemDateTime, which has a paramNum attribute equal to zero, then there is no need to fill the second parameter for executeSelectedQuery() and we can just set it equal to null.

```
295         System.out.println(ctdb.executeSelectedQuery("getSystemDateTime", null));
296
```

```
Console ×
<terminated> ConnectToDB [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (D
2021-12-17 13:40:47
End of Program.
```

Insert Query with Parameters: There are several Insert Queries within the repository and all of them have a Parameters Node containing a list of param nodes specifying the data type of each expected field within the database. We will use as an example the InsertSurveyUserId query.

```
ctdb.executeSelectedQuery("insertSurveyUserId",new String[] { "Bruno Diaz", "2022/01/18 7:00", "234", "123" });
```

The picture above shows a call to this method using the previously mentioned query. This query expects four parameters and their data types are defined within the XML sql_repo file. For this specific query, those data types are: Varchar, date, int and int. They are also expected in that order. This call does work, but it shouldn't work if we insert things that do not correspond to the correct data type.

```
<query name="insertSurveyUserId">
    <sql paramNum="4" insert="1">INSERT INTO UserSurvey VALUES(?,?,?,?);</sql>
    <parameters>
        <param size="100">VARCHAR</param>
        <param>DATE</param>
        <param>INT</param>
        <param>INT</param>
    </parameters>
</query>
```

```
| 456~137~2022/01/18 7:00~Wed Dec 15 21:45:07 EST 2021   | 2022-01-18 07:00:00 |       137 |     456 |
| Bruno Diaz                                             | 2022-01-18 07:00:00 |       234 |     123 |
+--------------------------------------------------------+---------------------+-----------+---------+
```

(entry within database, table UserSurvey)

First of all, the first parameter should never be a person's name. Therefore, "Bruno Diaz" should not be accepted. That's why the insertSurveyUserId query is only called by the processSurveyCampaign method which is in charge of generating a proper ID for any Survey User in an specific format (as seen above the Bruno Diaz entry in the UserSurvey table). For the second, third, fourth parameters however, we can change their data types to see if they are accepted or not.

```
309
310     ctdb.executeSelectedQuery("insertSurveyUserId",new String[] { "Bruno Diaz", "salsa", "234", "123" });
Console ×
<terminated> ConnectToDB [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (Dec 17, 2021, 9:38:53 AM – 9:38:57 AM)
com.mysql.cj.jdbc.exceptions.MysqlDataTruncation: Data truncation: Incorrect datetime value: 'salsa' for column 'Expiration_date'
```

In all three cases, the program would detect which of the parameters is not matching the right data type. This is achieved by the insertParams() and executeSelectedQuery() method. Traversing the xml file is also very important and ensuring that the value of <param> is being pointed. So checking if the node is a Element node is necessary.

Retriving Query with Parameters: The testing steps would be the same as Inserting, but with the additional step of ensuring the String returned by the executeSelectedQuery() method is being updated. In the current version of the project, this method returns a String, but this creates

many limitations. Ultimately, it would be convenient for the method to be able to return an object that can represent a table within the database.

Test Plan for Checking Expiration Date, Extracting Survey, and Inserting response to DB with SurveyExtIns

ExpDateCheck

To check the Expiration date of a given survey, you will need to call the ExpDateCheck that takes a Connection, a String representing the SurveyID, and a String representing the UserID as its parameters. This method will return true if the Expiration date is not past the current date and return false if the expiration date is either not found or is past the current date. To test, create a Connection object by calling my DBConnection() method like this:

```
SurveyExtIns Survey = new SurveyExtIns(); //creates a object of my class to have access to the methods
Connection connect = Survey.DBConnection(); //creates a connection to the database
```

Next you will need to have a String for the SurveyID and UserID. In the database, there exists a test survey and you can access this test with the SurveyID = “987” and UserID = “151”. The Expiration date for this User Survey is 2022-12-15, which means that if you pass in the above parameters like so:

```
Survey.ExpDateCheck(connect, SurveyID, UserID)
```

The method should return a boolean value of true.

Extracting Survey

To have the program extract a survey, you will need to call the SurveyTest method. This will return a JSONObject, of which you will need to use org.json.JSONObject. You will need the same Parameters as stated above, a connection, SurveyId and UserID. To access the test survey created in the Database, use the same information created in the ExpDateCheck. With the Connection connect, SurveyID = “987”, and UserID = “151”. Pass in the parameters like so:

```
Survey.SurveyTest(connect, SurveyID, UserID)
```

This method will return a JSONObject so you can have a JSONObject set to receive the JSONObject returned by this method like so:

```
JSONObject Survey1 =  Survey.SurveyTest(connect, SurveyID, UserID);
```

You can check the resulting JSONObject by either writing it to a file or printing it out like so

```
System.out.println(Survey1.toString());
```

Survey Response Insert

To have the program insert responses, you must pass in a txt file containing the desired responses you want to insert. Each line should be a response to one question and each portion of the response should be separated by a "#" in the format of.

SurveyID#UserID#QID#TypeID#Response

Based on the TypeID the method will either insert the response to Responce_val or the Text column of the QResponce table of the database. Then it will input the necessary information to each column of the QResponse table. A test txt file can be created with the following information:

98#151#1#1#A
987#151#2#2#C,F,G
987#151#3#2#J,K,L
987#151#4#3#Ppppppppppppppppppppppppppp

Do note that if a response already exists inside the table with the same UserID and QID, the response will not get inserted.

MySQL Testing

MySQL can be tested by trying to access CS370 without the use of the required TLS, it will automatically be blocked and that is how we know without using the sniffer that the connection is still SSL.

Servlet Testing

In the survey retrieve and submission one can test out the survey by filling out the form at the following link:

http://52.15.52.238:8080/SurveyRetrieve/survey?usrID=151&surveyID=987

A special user and survey has been created for this test case.

You will notice that upon submission the URL changes parameters indicating the changes in HTTP. These parameters were originally destined to be hidden however we did not achieve this goal.

A user can retake the same survey however only original answers will be saved the database of an already used survey is shown below:

QRID	QID	Responce_val	Text	User_ID	Survey_ID
9871511	1	B	NULL	151	987
9871512	2	C|G	NULL	151	987
9871513	3	H|I|K	NULL	151	987
9871514	4	O	NULL	151	987