Testing Strategy:

The main goal is to make sure things are as decoupled as possible in order to properly test all of the components with preferably automated tests. This is not easily done with databases and GUIs (at least using the tools we know). To test the database, we simply had to run queries to it. For the most part, the unit tests focused on making sure that those SQL queries being sent were the correct ones. For the GUI testing, we used manual tests, where we simply check that each component is where it should be, and unittest whatever function each button activates.

Code Review Strategy:

The main goal of the code review was to make sure that all of our code could work together. During the code review, we all had a look over the codebase, and gave each other feedback on ways to make sure that all of our code had SOLID design principles. We also had a look over some of the people's documentation to make sure another developer could easily pick up where we left off in the future. Also, we made sure to look over people's testing strategies for their components to make sure that they properly tested their own components.

Code Review Video:

https://www.youtube.com/playlist?list=PLYG6Y_flCVGeefI4RqfKvSQqFIabaas3s

Christopher's Feedback

- For PendingDatabaseEntry.addData(...), Change the documentation to state that the data is being held, as opposed to stating that the data is inserted into the data.
    - It may lead some users into thinking that you will not have to use dumpIntoDatabase(...) and causing issues
- For testing GUI
    - We can't automate it, so we do need to have a guide of how to test if it is working within a document (chances are it will go into README)
    - For our sake, we can think of possible tests and write them down as steps to perform manually
- The code in the database (DatabaseQuery.queryWithSQL(...)) might want to throw exceptions for the user
    - For high-level TEQ, where they give the SQL command, we may want to give them the error message/ an exception if they give a bad input

Alon's Feedback
- TEQ low, mid, and high level gui's should be combined into one module so that features are not added redundantly to each one.

    - That, or you can have the mid level interface inherit the low level interface, and the high level interface inherit the mid level interface. That way a change to a lower level interface will automatically be visible in the high level interface
- GUI code is split into multiple classes, which is good, but all the code in each class is in one single method. Possibly refactor it so that when when each button is pressed, it runs a method in this class. This will make it easier to update and to add functionality to each button.
-

<u>Chun Ho's Feedback</u>
- For the databaseSetup add an additional JAR file so that testing new code outside of unit tests can be done.
- For the README too little is explained about the database and how to run it.
- There is too little complete in the backend and too many holes for any front end connection to be done.

<u>Anny's Feedback</u>
- For the databaseSetup we need to move some functions out and only have the databaseSetUp to be setting up the database, since right now it is in conflict with the Single Responsibility
- For GUIs we need some more comments so that we can more easily read it.
- There are duplicate methods in databaseQuery and databaseSetup.
- Backend and frontend need to get connected.