I didn’t make any changes to the schema of the database for the implementation.

The application’s program files are fairly straightforward. There is a single driver cpp file that handles all of the implementation. I didn’t add any additional classes or structs but rather handle the data as it is retrieved. At the beginning of the application a Session object is constructed with the user details to connect to MySQL. From there the application queries the session to retrieve or create the database ‘Experiment’. The application then switches to using the Experiment database and creates all of the tables if they don’t already exist. This concludes the setup for the application. From there the user is prompted with a menu and the user’s input is read in. A switch statement determines what option the user selected. Each of the tasks that the user can do has been broken into a separate function. One of the important rules within the application is that when a user goes to insert an experiment or a run, it is all or nothing. The user cannot break the pieces apart and insert just a single result or parameter into the database. For inserting an experiment, the application prompts the user for the experimentId first, making sure there isn’t already an experiment in the table with that id. There isn’t any checking that needs to be done when inserting an experiment, so the user is prompted for the experiment meta data along with the parameters and results and it all gets inserted into the database. For inserting a run, it becomes more complicated since there are quite a few more checks that need to happen. First, after locating a valid experiment, the user must enter all the required parameters and then may optionally enter any parameters that aren’t required. The same goes for results of the experiment. When inserting either a run parameter or run result, the application uses regex to make sure the value the user enters, matches with the type of the parameter. To look up information about the experiment, the application prompts the user for the experimentId and, making sure that it exists in the database, queries for the meta data to print and then queries the ParameterType table and the ResultType table, matching to the experimentId, to print out all of the meta data about the parameters and results. To look up a run it is mostly the same except first the application prompts the user by printing out all possible runs that belong to the experimentId that the user provides. Then the user selects which run they’d like to display the information about. From there printing out the meta data about the run, run parameters, and run results is similar to printing the experiment meta data. For the experiment report it is nearly the same as fetching data about the experiment except the application creates an html file and prints the data along with html tags for creating tables. The aggregate report option prompts the user to pick from a list of parameters for the experiment they provided that may be aggregated, which are any that are an int or float type. Once they’ve selected the parameter the application prompts for a min and max date and then prints the aggregated values which are calculated using SQL queries. The application makes sure the user doesn’t enter the same date for the start and end date. Lastly for the parameter search the application asks the user for a parameterName and type and uses a SQL query to retrieve the meta data of each experiment with that parameter, by joining the Experiment and ParameterType tables.