Project 1 Team 35 Report

Team 33 Library Evaluation

Communication: 5

At the beginning of the project, we created a joint GroupMe between our team and team 33. This proved to be a clear and viable method of communication because it allowed us to quickly clear up any confusion regarding code/testing. Additionally, we maintained regular email correspondence between the two teams with regards to sharing of library files. The instructions provided with team 33's code were straightforward and easy to follow. Overall, there were no major issues here.

API Stage: 4

The provided headers were fairly logically laid out with regards to class design (arguments & return values, private data members), but inconsistent naming (snake_case for some functions and CamelCase for others, and sometimes a mix of the two styles) and sparse comments/documentation proved to be a moderate setback with respect to implementing the final application. Some design choices made the API tricky to work with at times (records were not implemented using key-value pairs; rather, an integer index was used to set/get values). In summary, while the API was by no means unorganized or deficient, minor naming issues/organization & documentation issues made working with the API less smooth than anticipated.

Test Cases: 5

Team 33's API passed all of the provided test cases from our end with minor difficulty. Any tests which were unclear or incorrect on our part were corrected. Overall, there were no significant issues here, although the test cases we created were admittedly not thorough enough.

Final Database Library: 3

Unfortunately, some of the core functionality of the API was incomplete or left a lot to be desired. min and max value comparison functions did not work properly for numeric values, likely because the data for record attributes were stored as strings. Furthermore, the query function was quite fragile/rigid with regards to parsing input - any invalid input (e.g. passing in an invalid attribute/operator/other typos) would crash the program without throwing any errors/exceptions. On the note of error throwing/handling, no exceptions were implemented or thrown, making debugging of many core issues a very complex undertaking. As a whole, functionality issues and poor error handling/robustness limited the overall functionality of our final application.

Problem Handling: 4

As mentioned above, while the code did have some functionality issues and lack of error handling, any issues we raised with the team were handled fairly quickly and to a good extent.

Any compilation issues were handled quickly by the other team. Unfortunately, we were not able to inform team 33 of some code related issues due to time constraints.

Team Evaluation Summary

Organization Plan

At the beginning of the project, it was decided that each member of the team would be fully responsible for implementing one of the 3 classes. As we had 3 members, this felt like a fairly logical decision.

Actual Organization/Workload

For the most part, we stuck to the original plan; Matt implemented all of the Database class, James implemented all of the Table class, and Brian implemented all of the Record class. This proved to have some issues with regards to individual workload/commit due to the varying complexity of the classes (Database proved to be the most significant undertaking). On the plus side, individual implementation of each class made the API overall very clear and defined (style/naming conventions were agreed upon before checkpoint 1). This also made debugging straightforward - the hierarchical structure/work breakdown allowed us to fully implement the simplest classes first and test them thoroughly before moving on to any classes which depended on the current class (e.g. Table depends heavily on Record).

Towards the end, we became fairly lax with regards to organization. The late implementation of the JSON parsing and GUI classes added some unexpected workload to all members. Additionally, we often worked as a group on a single group mate's computer using a projector in order to more clearly debug/understand some of the more complex parts of the assignment, meaning some of the git commits may be skewed or not accurately reflect each group member’s contribution.

Major Problems

By and large, the most significant problem we encountered was the **unannounced decision** for the team which was to provide us test cases to unexpectedly drop our API without informing us. This could be chalked up to a lack of communication on both teams’ parts, but the lack of information caused us to have to **implement our own test cases at the end of checkpoint 3.** This setback burned a lot of valuable time, thus compromising the functionality of our API.

Grade Division

Because Matt did the most daunting portion of the project (Database), and hosted the git repository, we feel that he deserves the largest multiplier (1.02), leaving James and Brian with 0.99 each. Note that these multipliers were discussed among the group and unanimously agreed upon.