**CS3203 Iteration 2 Evaluation (Code Review)**

**Team: 05 Evaluator: Zhang Xiaoyu**

In iteration 2, code review is for feedback only and it does not affect the grade for the project. However, since code review will affect the grade for iteration 3, it is important for the team to address the outstanding issues as much as possible.

|  |  |
| --- | --- |
| **Item** | **Remarks** |
| **High-level Structure** (is it clear where new functionalities are implemented?)  **Implementation Matching the Component Diagrams and Class Diagrams** (especially the abstract APIs)  **Encapsulation and Use of API operations** (instead of direct access to class members)  **Extensibility of the Components** (are the new functionalities added organically or by hardcoding?) | Good encapsulation of API operations  Query Validation may lack extensibility: long if-else & multiple cases are used; consider Table-driven validation |
| **Coding Conventions** (same conventions should be followed for the whole project, by all team members)  **Naming Conventions** (are names easy to understand without reading the code comments? are naming symbols used (e.g., STMT, NODE, etc.)?)  **Clarity in Coding** (e.g., some if statements are easier to understand in switch-case format)  **Comments in Code** (should be not too many, but useful and in the right place) | Good coding & naming conventions  May have more comments to explain a method's "what" and "why", especially when the method names are not so straightforward  Some comments look random and not relatively informal (such as "*might not be necessary...*") |
| **Performance** (for query evaluation especially, Basic Query Evaluation should not be too inefficient even without optimization; for frontend – multiple passes of parsing / extraction?)  **Efficient Data Structures for Storing Data** (maps with keys, indexed arrays, etc.; briefly comment if you notice any problems.)  **Redundancy in Data Structures** (some info stored many times.) | Efficient Data Structures used  Redundant DS (DesignExtractor, TNode): not sure whether you have filled any content inside now |
| **Error Handling** (graceful end of execution in case of error; treat possible errors in essential places, such as function parameter errors; report useful information for debugging.)  **Use of Exceptions** (throw exceptions, and catch & handle them properly; not all exceptions should be handled at the end of the execution) | Meaning error msg and sufficient error handling block  Good use of exception |
| **Resource Leaks** (difficult to check properly; briefly check if teams have some deallocation of memory) | Same as iter1 |
| **General Unit Testing** (coverage, is the code structured to allow for easy testing? appropriate use of stubs and dummy data?) | 1. Good test coverage, but may consider have more intensive tests for PQL evaluation.  2. Good design & clear testing structure. |
| **Outstanding Issues** (any issues that have not been addressed in spite of your comments for the previous iteration?) | 1. Flat hierarchy so a bit hard to navigate -> can consider use separate packages for FE, PKB, PQL  2. Commenting convention is inconsistent  3. Many if-else statements that can be easier understood with switch-case  4. Take note when using new, remember to explicitly delete the memory after completion (may verify this about your currNode = new PNode) |