Scalability

- As the input size increases the time and space complexity increases i.e. the time taken for running the code increases. This will also increase the complexity of the program (modules).
- Another limitation is that, if all students have conflicts with one another then teams won’t be formed due to StudentConflict Exceptions.
- Since two of the SOLID Principles are implemented, adding a new similar feature would require fewer lines of code by extending the classes. However, adding new dissimilar feature may require several lines of code.
- In the future, implementing the Algorithm will indeed increase the complexities but maybe reduced to some limit if backgrounds threads are used for faster processing. If the operations are taking longer time then the GUI may go unresponsive. These issues could yet be handled if proper methodology (Using synchronized threads to avoid deadlocks OR limiting the no of threads at one time OR using priority threads) is used.
- The logic of heuristic Recommender algorithm would be to swap all the students between the Top Scoring (Skill Shortfall) team and Lower Scoring (Skill Shortfall) team. Each swap would be a new Team Combination (Set of teams) acting as a node. These swaps would produce 4*4 = 16 nodes, but necessary Conflict and other Exceptions will be validated before creating the nodes. The nodes added one by one would have some depth value and won’t be greater than 6 to avoid algorithm complexity. Until depth = 6 i.e. 6 valid nodes are added the new nodes from the last node will be created. All these valid nodes will be added in a stack. The nodes would be popped to compare with other nodes with balanced weighed metric (metrics of all teams would be added and then different metrics would have some weighing value). Finally the min cost metric would be the best team combination.