**GUI Detail:**

To design the GUI, I used Scene Builder. Each Frame designed will have its Own Class to manage, and they have related methods to call the data. An increase in data collection will have a partial effect on results rendering due to the data processing, but it will not affect GUI Frames.

**Design Detail:**

This assignment has five different parts. Every time complete a task, I need to re-factor the entire code. Put the relevant code that reads the data in the corresponding manager for management. Because of the following assignment requirements, I will save all the data files in the database. Many collections of methods are used extensively in data structures. Because each data has different characteristics, for example, can be a single column of data, the choice is the List (data reusability) method and Set (to ignore data repetition) method.

Also, the Hash Map method has been used with a two-column data structure. Accordingly, the use of data sends away, with data content concerned. Then, some parts of JavaFX are managed using its corresponding methods. The reason for doing this is because it looks clear, and the method calls are simple.

Secondly, regarding the algorithm, Best First Search is always referred to. Priority Queue is used to sort the expanded state nodes, and only the best node is selected for expansion each time. And remember that the best status node so far is returned after the algorithm is finished.

**Algorithm Detail:**

Recommend uses the state Tree's greedy search. Here, a state is a method of composing teams, where any two students are swapped, and the state of the system changes from one state to another. The status tree is to root the current state and exchange all "reasonable" states of two students as children nodes (this is also called expand a form). Then find the best one in The Children nodes and do the same operation.

Since swapping any two students gives a new state, the children of each expansion are the 20 choose two levels of complexity, so we used a strategy to reduce the complexity by swapping the best overall team for the worst, with all students in both teams changing once. In each swap, we only try to switch one student to another team, so that the maximum number of nodes to be expanded each time is 4\*4 = 16.

Also, when searching, we stipulate that the state tree should be expanded at most five layers, which significantly reduces the complexity of the algorithm.

The algorithm refers to Best First Search, USES priority Queue to sort the expanded status nodes, and selects the best node to extend each time. And remember that the best status node so far is returned after the algorithm is finished.

To judge the quality of a state, we use a combined cost function, 0.7 \* STD dev skill fall + 0.2 \* STD dev Average Skill + 0.1 \* STD dev preference percentage. Because we think the balance of skill fall is the most important.

**Learning Detail :**

In the beginning, I put all the data in the primary method. Later, when I did the second assignment, I found some problems. Many data needed to be re-entered before the next step could be performed, which was a waste of time. In the next job, I re-factor my data and methods, and there will be a manager associated with the Class to manage the data. The advantage of doing this is that you don't have to re-enter the data, and you use the data in the file to do the calculation. The second thing That I find most interesting is the part of the Scene Builder. I am a student without any computer foundation, and I thought that the design part of the JavaFX needed code to complete. But with Scene Builder, the software is so great that I have to design it, put it directly in the project, and then connect it with a class and add some methods. In the fourth assignment, I used the knowledge of the database, and many statements are the knowledge of the database.

And finally, the part of the algorithm that I spent the longest time on. Recommend uses the state Tree's greedy search. Here, a state is a method of composing teams, where any two students are swapped, and the form of the system changes from one state to another. The status tree is to root the current state and exchange all "reasonable" states of two students as children nodes (this is also called expand a state). Then find the best one in The Children nodes and do the same operation. Since swapping any two students gives a new state, the children of each expansion are the 20 choose two levels of complexity, so we used a strategy to reduce the complexity by swapping the best overall team for the worst, with all students in both teams changing once. In each swap, we only try to switch one student to another team, so that the maximum number of nodes to be expanded each time is 4\*4 = 16. Also, when searching, we stipulate that the state tree should be expanded at most five layers, which significantly reduces the complexity of the algorithm. The algorithm refers to Best First Search, USES priority Queue to sort the expanded status nodes, and selects the best node to extend each time. And remember that the best status node so far is returned after the algorithm is finished.

**Application Demo:**

In order to run the application, you need to run the ProjectTeamFormation.jar file and you check the application.

**Conclusion:**

I have learned lot of new stuff while doing this assignment and had a good experience with this framework. And I am excited by finishing my project.