Report

Project Design:

The project is maintained in three folders namely Application, database, Model.

The Core part has 6 primary classes to gather and store the user inputs into the text file as well as in the Database. Six primary classes gather different values needed for organising and forming the team

A shortlisting class which uses the data of the six primary class and perform the shortlisting of projects.

Then two classes are used to form the team for the project and calculate the metrics for the formed team. Classes used are Formteam and Teammetrics.

All above mentioned class are invoked and accessed by the Main class **Assignment.java** which is a Menu Driven class. In this class, object for all the above-mentioned class is created and prompt the user for enter an option by displaying the menu.

The GUI follows the MVC pattern. The Controller **SampleController.java** by using the instances of model classes It will gather all the details and display it in the GUI.

Any changes happened in the view part either by SWAP or UNDO option will recorded automatically and make the required changes in Model class as well as stores the data in Database.

The **Database.java** class mainly does primary activity like insert and update the data into the database. The instance of this class is created in controller class and the values are passed from controller class.

Three Bargraph is created to show the different metrics of the projects also shows the standard deviation for different metrics.

If any changes happened in View, Bar graph will automatically incorporate the changed value and show the updated graph.

Data Structures:

To meet the requirement of the assignment, my project used a vast variety of data Structures.

The Primary Model Classes uses **HashMap** to store all the data.

In View Part for performing the UNDO action, I utilised **STACK** to push and pop the values.

For doing the exception checks, calculating the metrics I have used **linked list** which helped me to meet the requirement.

In some places I have used **TreeSet** to get the exact order of values from the HashMap.

HashSet which helps me to check the distinct values and I have used it in while checking the Personality Imbalance exception.

Algorithm:

1. First it considers all 20 students for each project.
2. It calculates the percentage preference for all 20 Student against each project and assign a value for each student in database
3. Next it calculates the skill gap for all students against each project and assign a value for each student in database
4. The above calculation and assigning the values will happen parallelly using the thread.
5. Once the calculation is done. Then three different methods are there to assign the students to the project.
6. First method Teamassign() which takes the most matching records for each project and assign the students.
7. Second method Teamcheck() which will check the team size if team size is less than the requirement then again my algorithm looks for the next matching students for the project and assign the students.
8. Third Method Teamfill() which is the last method, here it will again check the team size if team size is less than the requirement then my algorithm fill the gaps in the team with the unassigned students.
9. While assigning the students, my algorithm verifies all the exceptions are not violated and form the team.
10. In GUI, if the **HELP** Button is pressed then my algorithm is invoked and help the project manager to reduce the Standard Deviation.
11. My Algorithm helps to reduce the Standard deviation for project preference completely and Standard deviation of Average Skill Competency till 0.5.

Object Oriented Concepts:

I have inherited the thread class in my algorithm class to perform the actions parallelly.

Object oriented principle: I am using the OCP principle. For performing the UNDO operation I am extending the STACK class and using it as storage to push and pop.

In GOF pattern, my project follows the builder creational design pattern. In my primary core class, I am creating the instance object by using the same construction process. Also my project uses the command behavioural pattern by using the instance variable of the class I can perform multiple actions.

Scalability:

I implemented my code which will be more adoptable to all the changes done by the user.

In GUI aspects if the user is swapping the student between the projects. My code will record the changes happened in the GUI, Stores the data in database and updates the graph periodically.

If the user wants to do the UNDO operation, my code uses the STACK to record the historic data and retrieves the older version. The code is structured in a way to adopt the changes.

In Algorithm aspects even if the project is added or new students are added, my algorithm incorporates all the changed value as a new value and calculate the metrics and assign the ranking

value for each students in the database. Then performs the steps mentioned above in the algorithm section.

To adopt the future changes most of the loops is made to have the dynamic conditions rather using a static value as condition. In most of the case I have used for each loop which make my code more scalable for future extensions.

For performing major operation, I have used collection frameworks so that any changes to the data and requirement the memory spaces will allocated based on the records creating.

Lesson Learnt:

Problem 1:

For each time I must read the data from text file or database and need to store it temporarily inside the class. Which way I must store? Also, for particular key I have to store multiple values

Solution: I used HashMap instead of ArrayList because the HashMap let me store the unique keys and we map the values with the key which helped me in doing validation. For second case I used arraylist datatype to store the group of values for the particular key in HashMap. This helped me to avoid using multiple arrays and arraylists.

Problem 2:

For making the negative test cases how to expect the exception. The excepted syntax is different in each version of JUNIT

Solution: I have used Junit4 to implement this which is feasible for me as well the syntax is simple.

Problem 3.

For Shortlisting the project and for calculating the Metrics. I implemented the logic in using the for and if loop.

Solution: Later I started implemented using the **LAMBDA** function which helped me to reduce the number of lines to code. Also, the Lambda is convenient to perform the multiple operation at a time for all values.

Problem 4:

For storing the historic records and how to perform the UNDO operation

Solution: I used Stack to overcome this problem with the help of Stack the UNDO operation is achieved easily.

Suggestion to Improvise:

- In some places I am still using multiple if and for loops to meet the requirements, that can be refined with Lambda function.
- The Algorithm can be tweaked little more to reduce the standard deviation of Skill Gap