Design Structure

- There are total 6 Model classes implemented: Company, Owner, Project, Student, StudProjPreferences and Team.
- Each of these classes store different types of data according to the requirements. The fields and methods of these classes are Encapsulated. These fields can only be accessed or modified using those methods.
- Lambda expressions are used implementing Serialization for reading through files and BufferedWriter for writing to files.
- For File handling, where FileHandler is abstract super classes and rest Handlers are sub classes. This ensures File Manipulation available to only when needed by extending the functionality. For Team Metrics, TeamHandler sub class is extended which acts as an intermediate for Team Metrics file.
- HashMap is used for key|Value way of storing different contents. HashSet is used to easy validate for duplication of values.
- Positive and Negative test cases are implemented. 5 Negative cases are the Exceptions such as "InvalidMember", "RepeatedMember", "NoLeader", etc. 4 Positive cases are the team fitness metrics such as "Skill Shortfall", "Average Skill Competency", "1st and 2nd Preferences" and Standard Deviation of "Average Skill Competency".
- Refactoring is done by using Reusability concept i.e. functions used to avoid code repetition. MVC (Model View Controller) format is used for proper structuring of the classes hierarchy. Also lambda expressions for some operations and thus reducing the complexity of the code. Code indentation and Comments are added focusing on the code quality and readability.
- FXML i.e. design of GUI is designed using SceneBuilder by using some Horizontal Box and Vertical Box layers added over a stage container. It also includes 3 Bar Graphs with X axis as Category and Y axis as Number. The elements are added over it and they are styled using properties like position, font, position, etc.

  The logic behind the GUI i.e. Controller classes (backend interaction with GUI) are implemented. These classes include action button listeners as triggers for executing some code if a user interacts with GUI element. As the interaction happens the contents are manipulated in a file and the changes are updated visually.
- There is a separate java file for storing file contents or retrieving those contents from SQL Lite Database. SQL methods like PreparedStatement and ExecuteQuery are used for executing DDL statements (such as CREATE or DROP or INSERT). ~~The retrieved contents are parsed before using it in further operations.~~
- Dependency Injection is used by losing the coupling of the module classes which works where only the needed classes are loaded when required. Also Single Responsibility is also considered i.e. different classes for GUI, Menu Handling, File Handling, Database Management.