Justification

This project has 3 different packages namely model, db and application. Application package consists of the GUI implementation and db contains database implementation. In the whole project, I have used many HashMap's using Java generics which helped to store the unique key and values. Collections such as ArrayLists are used several times.

First, the Model package it is designed in a particular way in which if the user wants to enter information through the console, they can run the Assignment class which contains the main function and implements the Menu using switch case. Then, there is a ProjectHandler class which contains all the methods for handling different operations like storing and fetching data in files. Also, serializing all the objects which makes all the classes to implement Serializable interface. There are many other classes as well like Company, Project, ProjectOwner, Student, StudentInfo(Child of Student). This ensures Encapsulation as all the members are private making them inaccessible directly. Also, this package consists of many Exception classes used for forming teams. The Project Handler class consists of all the important methods required in this project like Swap, Undo, Add and Remove etc.

The View part is in the application package which consists of the Main class, Controller class and Fxml class. The GUI part is in this package which is made using scene builder using the MVC architecture. Using the Scene Builder, I created the bar charts, buttons and HBox. And all the checkboxes and labels were created dynamically in the controller class. In the controller class I have used the lambdas to set bar chart colours.

For the database, I created another package db. It consists of classes for creating tables, inserting into tables, and updating the tables using sqlite3. I have used PreparedStatement for inserting and updating to tables. And this database functionalities are called in the Main class which is in the application package.

For the Undo functionality, I have used the Memento design pattern which is one of the GoF pattern. This pattern consists of an Originator(The class of whom we need to make a memento) and a CareTaker(which stores all the memento's in one place without looking into the Mementos). The Originator class saves its states as a Memento for which it uses the Memento class and uses the Memento class to retrieve its state as well. The CareTaker consists of a HashMap which stores all the Mementos as a Savepoint securely. The Memento design pattern makes this functionality very robust.

The Junit test class is generated using Eclipse. I have implemented all 10 test cases which are running very smoothly. I have used streams to read from files as well.

The project follows the SOLID principles as follows:

Single Responsibility Pattern: Each class has a specific responsibility like if we want to update database there is a different class for that, or we want to update GUI, or for Menu, or for state safekeeping etc.

Moreover, I have used important OOPs principles like Encapsulation, Inheritance to make my application robust.