Team 3: Register Grade

1. Improper Exception Handling

Description:

Exception handling is crucial in any application, and poor handling can lead to unexpected behavior, such as data loss. A common mistake is to catch general exceptions and not act on them appropriately.

Recommendation:

Avoid catching generic exceptions (Exception). It is better to catch specific exceptions and handle each one appropriately. Also, make sure that critical exceptions are handled or logged correctly to make debugging easier.

2. Dependency on Concrete Implementations

Description:

A common pitfall is to code directly against concrete implementations instead of interfaces. This can make code less flexible and harder to maintain.

Recommendation:

Use interfaces whenever possible, especially for classes that implement critical services or behaviors such as note management or user authentication.

3. Lack of Input Data Validation

Description:

Failing to properly validate input data (such as student names, grades, etc.) can lead to errors in your application, such as storing incorrect data or corrupting the database.

Recommendation:

Implement strict validations on all user input. For example, make sure that grades are within a valid range and that student/teacher names are correct.

4. Strong Coupling Between Classes

Description:

Having strong coupling between classes can make it difficult to implement new functionality or changes in the future. For example, if the Teacher class has too many dependencies on the Student class, any changes to Student could affect Teacher.

Recommendation:

Apply the single responsibility principle (SRP) and dependency injection to reduce coupling between classes. This will make the project easier to maintain and scale.

5. Lack of Unit Testing

Description:

Not including unit tests can lead to lower code quality and the introduction of bugs during changes. This is especially important in a logging system, where accuracy is crucial.

Recommendation:

Develop unit tests for all critical system functionality. Use frameworks like JUnit to ensure that the code works as expected.