Documentation

KISS (Keep It Simple, Stupid)

Summary: The KISS principle promotes simplicity in both design and code.

Example: In week 2, My code initializes a `Skill` class that directly interacts with a SQLite database connection. In week 4, I refactored it into a `SkillRepository` class.

Implementation: The change from a single `Skill` class to a `SkillRepository` class simplifies the design by separating concerns. It follows the KISS principle by making the code easier to understand and manage.

DRY (Don't Repeat Yourself)

Summary: Avoid duplicating code.

In both versions, I respect the DRY rule.

Implementation: I encapsulated this shared logic in the constructor of the `SkillRepository` class. This follows the DRY principle, as it eliminates duplication and makes the code more maintainable.

YAGNI (You Aren't Gonna Need It)

Summary: Only incorporate functionalities or write code that is presently necessary or foreseen for the immediate future.

Example: In week 2, I had a single class handling both database connections and skill-related operations. In week 4, I refactored it to separate concerns.

Implementation: The refactoring in week 4 demonstrates the YAGNI principle. By creating a `SkillRepository` class, I ensured that I focused on what was necessary at that stage without unnecessary complexities.

Single Responsibility Principle (SRP)

Summary: Each function, class, or module should have only one reason to change.

Example: In week 2, my `Skill` class handled both database connection setup and skill operations. In week 4, I separated these concerns into a `SkillRepository` class.

Implementation: By creating the `SkillRepository` class, I followed SRP more effectively. It ensures that each class has a singular purpose, making my codebase more maintainable.

Open/Closed Principle

Summary: We can add new functionality without changing existing code.

Example: My code doesn't demonstrate the Open/Closed Principle.

Use meaningful names

Summary: The goal is to give an explicit name to a variable; that is, to clearly explain what it represents.

Example: In my code, the names of my variables or methods were not entirely clear. For the CRUD method, I simply named them `Read`, `Update`, etc., in the code from week 2, and then I changed them to `ReadSkill`, `UpdateSkill`, etc., in week 4.

In my code, I removed the comments that explained what each function in the CRUD method was doing.