SOLID principles

S states that every class should have only one responsibility, which is in our case true. For example, RecipeRepository class is responsible to only fetch data for recipes.

O states that every class should be open for extension and closed for modification. In our case, every class can be extended with new methods, but other methods should not be modified.

L states "you should be able to use any derived class instead of a parent class and have it behave in the same manner without modification". In our case, we do not have any derived classes because they are not needed and would only make things more complicated without any benefits.

I states "clients should not be forced to implement interfaces they don't use. Instead of one fat interface, many small interfaces are preferred based on groups of methods, each one serving one submodule.". IN our case, we do not have any interfaces, so we do not need to worry about this principle.

D states that high-level modules/classes should not depend on low-level modules/classes. Both should depend upon abstractions. Secondly, abstractions should not depend upon details. Details should depend upon abstractions. In our case, low-level classes are basically model and repository classes that are losely coupled because they are responsible for handling information about certain objects. Controller classes are losely coupled with repository classes because they handle communication with user and if an user asks for something, controller must be able to provide them with needed information. Recipe/feedback/user classes cannot implement or be derived from abstract classes because that does not make any sense. Repository classes also cannot implement an interface or be derived from an abstract class because different models ask for different SQL queries which cannot be generalized.