1. Software System

User
[Person]
The user of our application.

Uses

Sends email to

Food Database
[API, e.g. Spoonacular, Nutritionix v2]
An external database containing different foods and supplements along with their nutritional values.

Uses for Diet Plans

Personal Health Coach
[Web Application]
A personal health coach app based on AI.

Uses for Login

Mail Server
[Software System]
A simple mail service for user comunication.

2. Container

User
[Person]
The user of our application.

[M1&M4] Front-End
[Software Module]
A WEB interface with the User.

[M2] Back-End
[Software Module]
Home to our Databases, facilitates access to the AI Module.

[M3] AI Module
[Software Module]
A WEB interface with the User.

Mail service
[Software System]
A simple mail service for user comunication.

Food Database
[API, e.g. Spoonacular, Nutritionix v2]
An external database containing different foods and supplements along with their nutritional values.

Personal Health Coach
[Software System]

3. Component

[M2] Middleman
[Software Module]
Our interface with the databases and the front-end.

User Data

User Profiler
[Software Component]
Updates the user's Profile according to new information about them.

User Data
User Feedback

AI API

Food Database
[API, e.g. Spoonacular, Nutritionix v2]
An external database containing different foods and supplements along with their nutritional values.

Diet Plan Generator
[Software Component]
Generates a diet plan based on user input.

Fitness Plan Generator
[Software Component]
Generates a fitness plan based on user input.

Tip Generator
[Software Component]
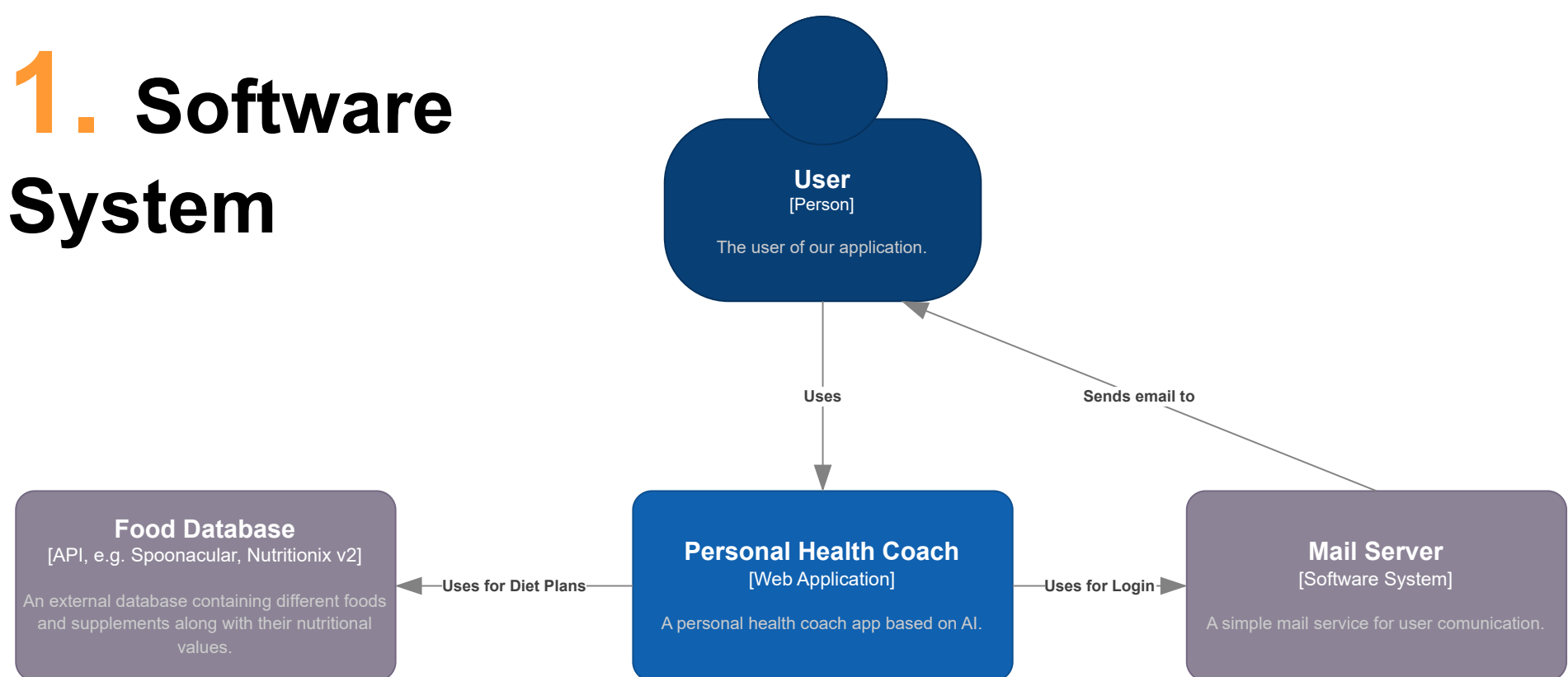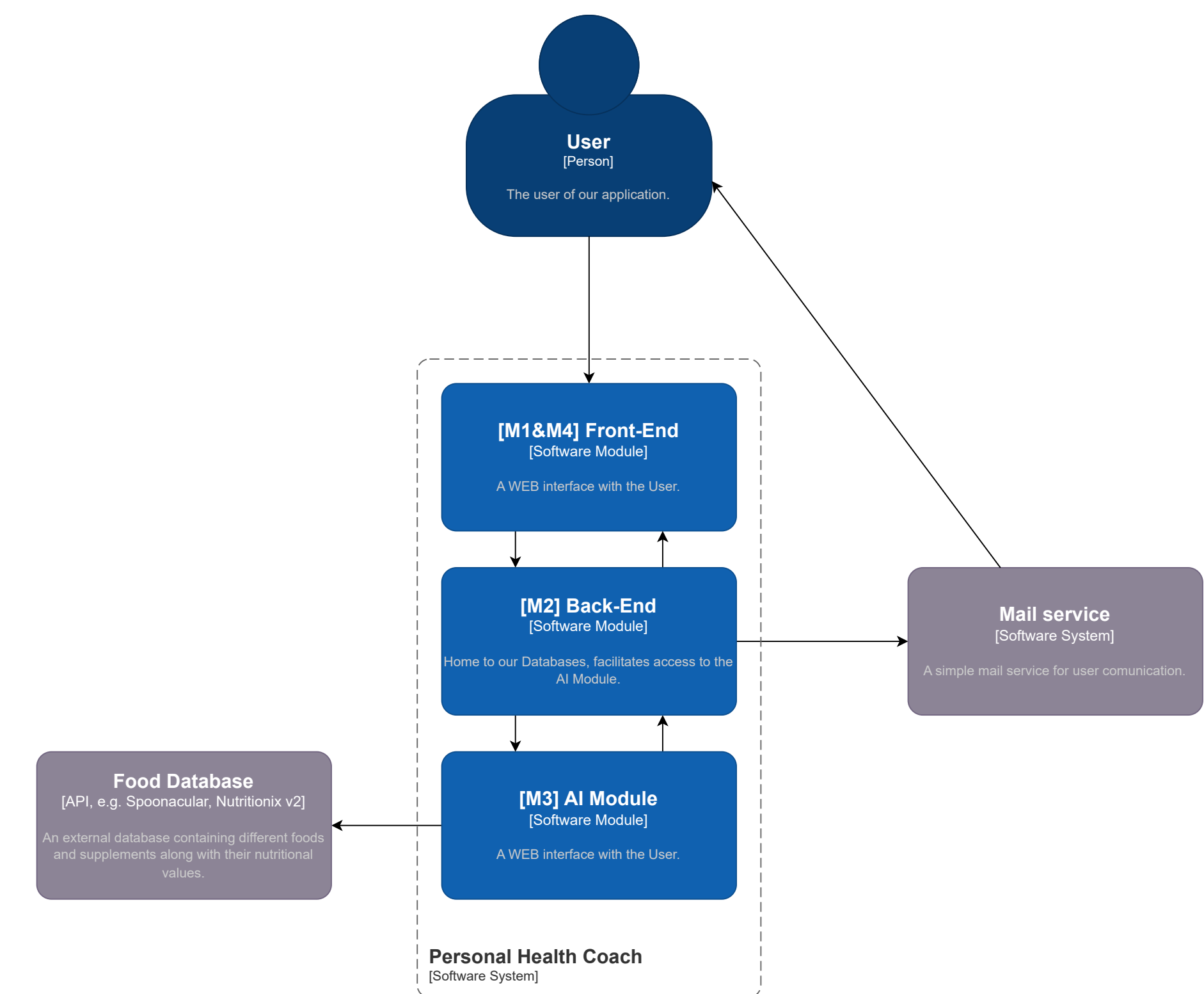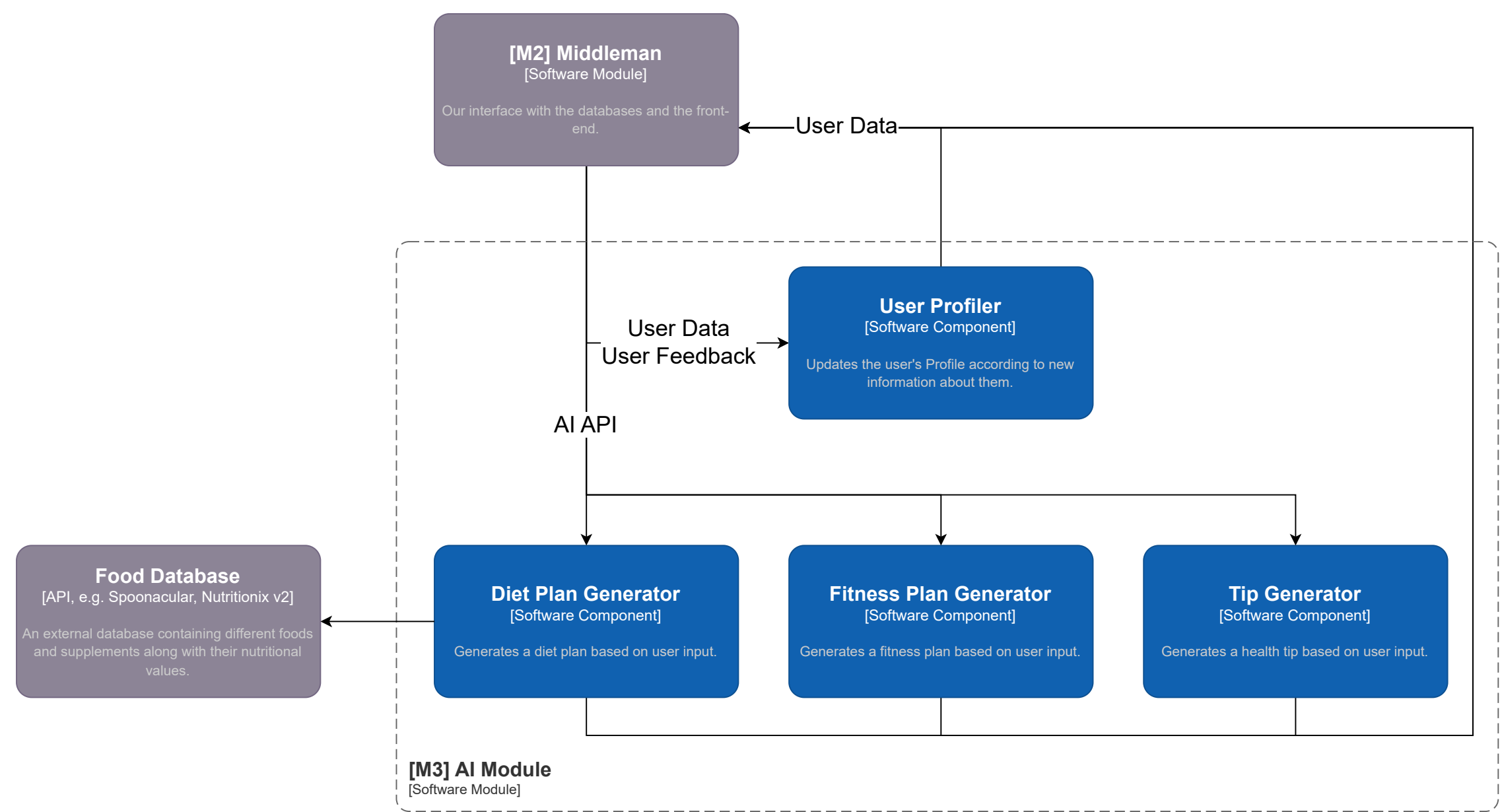Generates a health tip based on user input.

[M3] AI Module
[Software Module]

4. UML Diagram

Regarding the difficulty

if the difficulty chosen is:
Easy   - 0.1 - 0.25 set at first at 0.1
Intermediate - 0.25 - 0.65 set at first at 0.4
Advanced   - 0.65 - 1   set at first at 0.7

<>DailyPlan

- userId : int
- difficulty : float
- planType : PlanType
- currentDay : int

+ fromMacroPlan() : DailyPlan
+ getDifficulty() : float
+ alterPlan() : void
+ generateDailyPlan() : DailyPlan

enum : UserGoal
loseFat
gainMuscleWeight
cardio
strength
endurance
maintainHealth
etc.

<>MacroPlanner

- userId : int
- restrictions : List<PlannerRestriction>
- plannedDaily : List<(planType, difficulty) : float>
- timePeriod : int
- difficulty : float
- goalList : List<UserGoal>

+ generateMacroPlan() : void
+ addRestriction(PlannerRestriction)
+ alterPlan() : void
+ getTimePeriod() : int
+ setTimePeriod(int) : void
+ getDifficulty() : float
+ setDifficulty(float) : void

<> PlannerRestriction

- priority : float

+ getPriority() : float
+ setPriority(float) : void
+ apply() : void

UserFeedback

+ satisfaction : float
+ newRestrictions : List<PlannerRestriction>

Extends

FitnessDailyPlan

-exerciseList : List<Exercise>

+generateExerciseList : void
+getExerciseList : List<Exercise>

Extends

Extends

FitnessMacroPlanner

+ generateMacroPlan() : void

FitnessMacroPlanner

+ generateMacroPlan() : void

UserProfiler

- userId : int
- userMacroPlans : List<MacroPlanner>
- restrictions : List<PlannerRestriction>

+ calibrateUser(UserFeedback) : void
+ addRestriction(PlannerRestriction)
+ fetchUserData() : void
+ postUserData() : void
+ setUserId(userId) : void
+ getUserId : int

Exercise

- type : ExerciseType
- difficulty : float

+ getType() : ExerciseType
+ getDifficulty() : float

enum : ExerciseType
arms
legs
fullBody
etc.

TipGenerator

+ generateTip(userId, type) : String