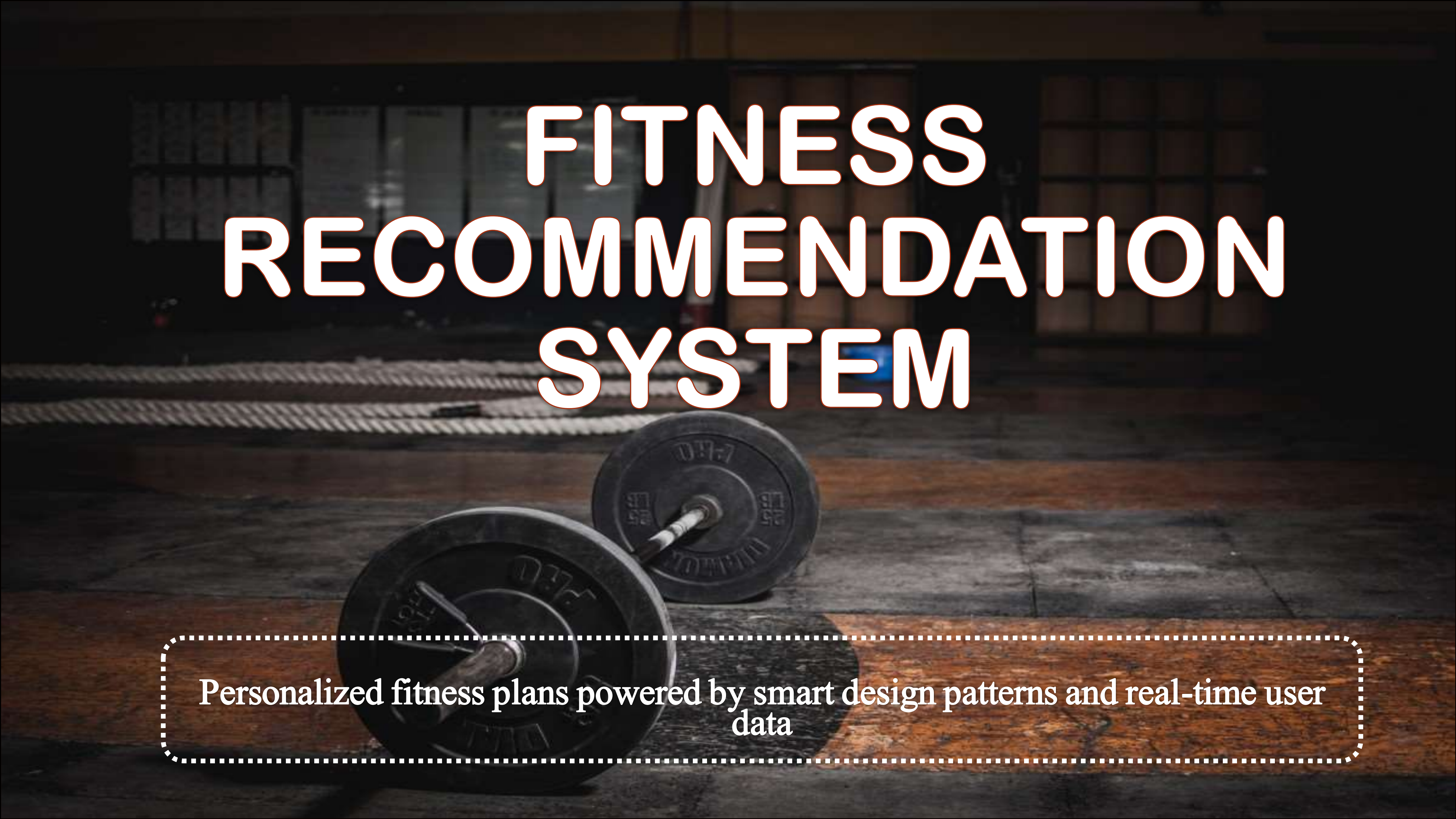


FITNESS RECOMMENDATION SYSTEM

Personalized fitness plans powered by smart design patterns and real-time user data

WHAT'S THE GOAL?

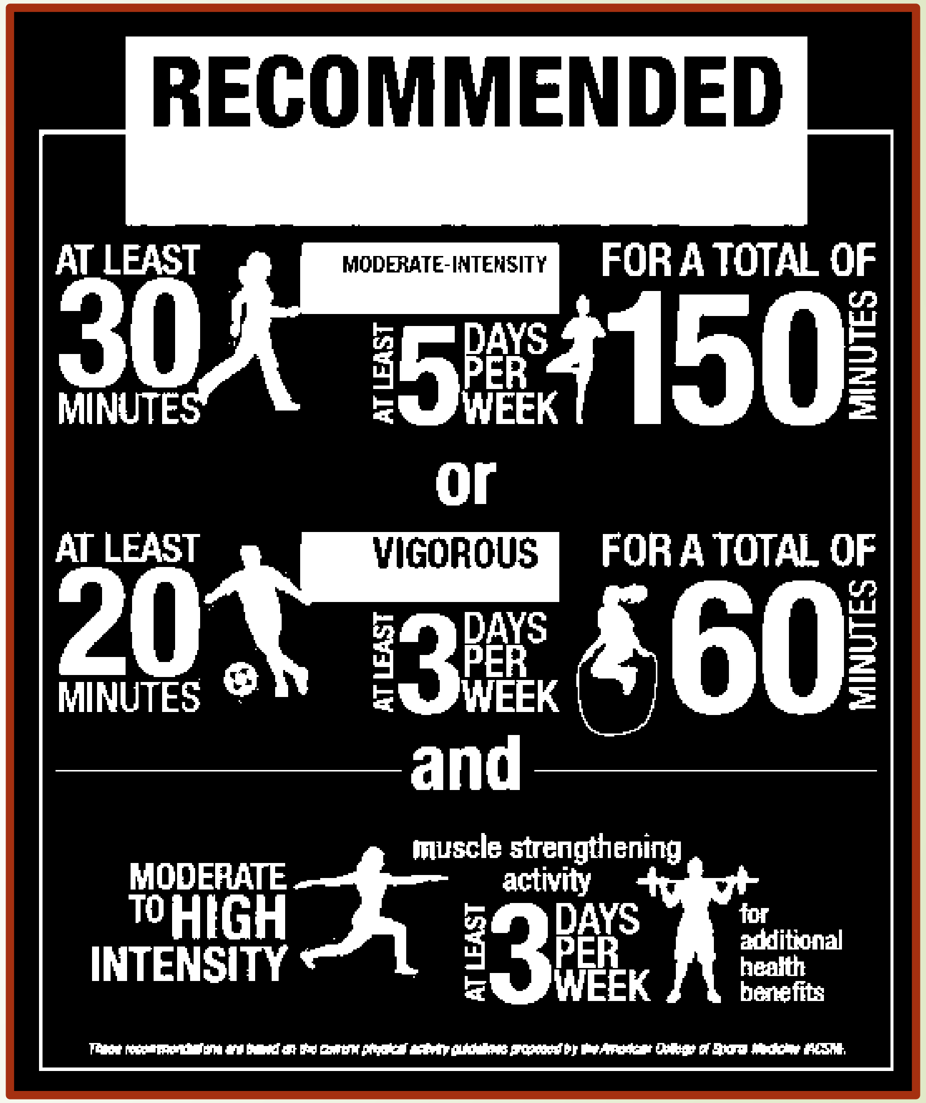
Essential Features of Today's Fitness Apps

Activity Tracking
Nutrition Planning
Workout Library
Community Support

The goal is to develop a system that leverages design patterns to generate personalized fitness and exercise plans tailored to individual user attributes such as age, weight, activity level and fitness goals, providing customized recommendations for exercises, diets, and equipment.

CHALLENGES

- *Handling complex personalization for each user's fitness goal*
- *Responding to changing user data and health metrics in real-time*
- *Implementing and integrating selected design patterns effectively*
- *Generating relevant, adaptable fitness recommendations*


RECOMMENDED
AT LEAST 30 MINUTES
MODERATE-INTENSITY
AT LEAST 5 DAYS PER WEEK
FOR A TOTAL OF 150 MINUTES
or
AT LEAST 20 MINUTES
VIGOROUS
AT LEAST 3 DAYS PER WEEK
FOR A TOTAL OF 60 MINUTES
and
MODERATE TO HIGH INTENSITY
muscle strengthening activity
AT LEAST 3 DAYS PER WEEK
for additional health benefits
These recommendations are based on the current physical activity guidelines proposed by the American College of Sports Medicine (ACSM).

OUTCOMES

- *Automatically updates fitness plans when user activity level changes*
- *Recommends personalized workout plans: Beginner, Intermediate, or Advanced*
- *Handles invalid or unexpected input gracefully with fallback messaging*
- *Allows new recommendation strategies to be added without changing existing code*
- *Switches strategies in real-time — no need to restart or recompile*
- *Keeps logic and data management cleanly separated for better maintainability*
- *Supports future extensions like Yoga, Recovery, or Sport-specific strategies*
- *Ensures observers are always in sync with user data (real-time updates)*
- *Provides clear feedback to users based on their current health metrics*
- *Promotes scalability — the system is ready to grow with user needs*

WHAT ARE DESIGN PATTERNS?

- *Design patterns are standard, proven solutions to common software design problems.*
- *They provide a way to structure code that is reusable, flexible, and easier to maintain.*
- *Patterns capture best practices that developers have learned over time.*
- *They help simplify complex code by providing a clear template for solving design issues*
- *Design patterns provide standard solutions for recurring problems in software design — some focus on object creation, some on structure, and others on behavior and communication.*

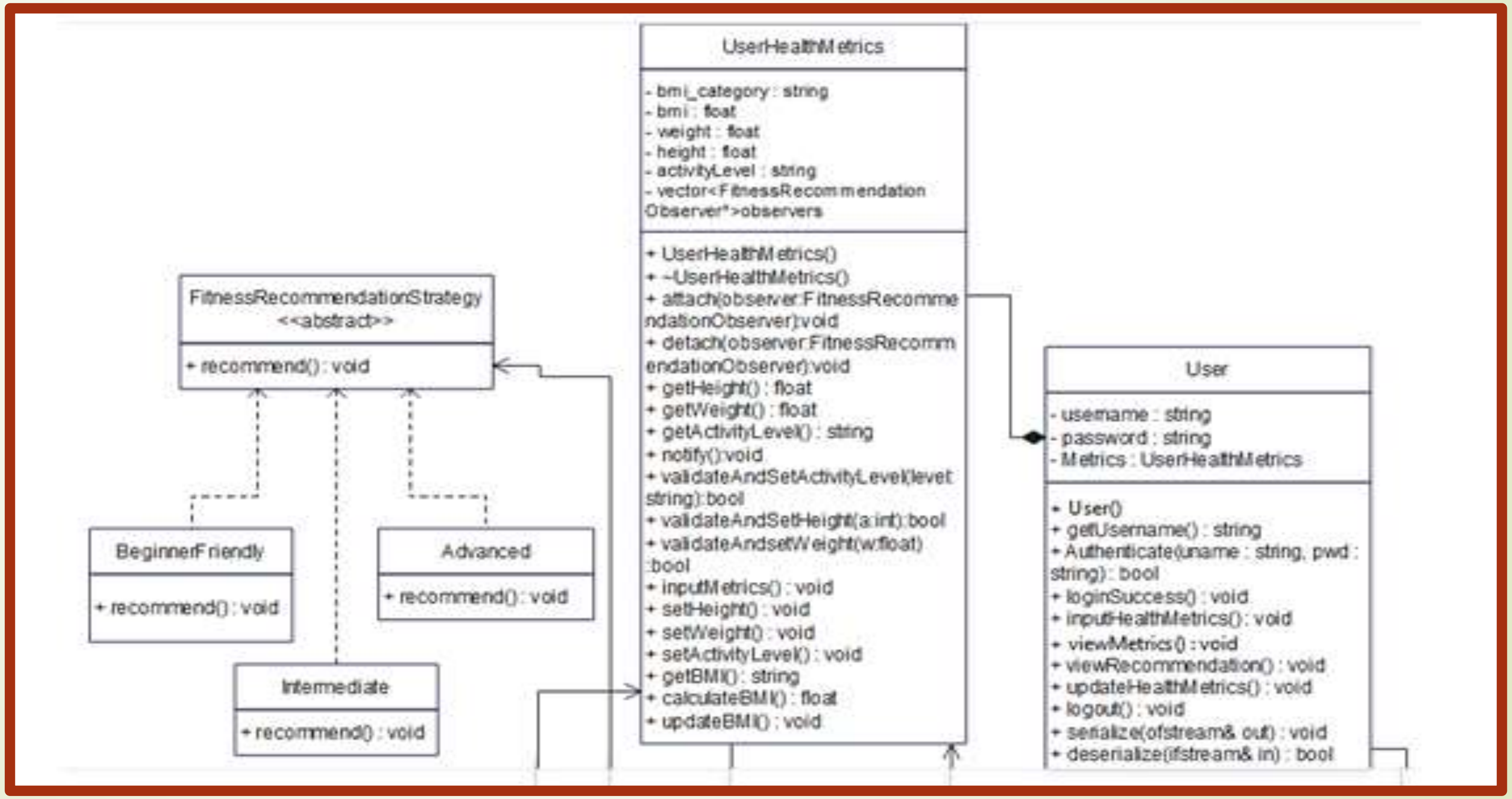
- *Behavioral design patterns, in particular, often manage communication among objects, sometimes using inheritance (e.g., Strategy Pattern), and other times using composition.*
- *Learning and applying these patterns makes our codebase more maintainable, reusable, and adaptable to change.*

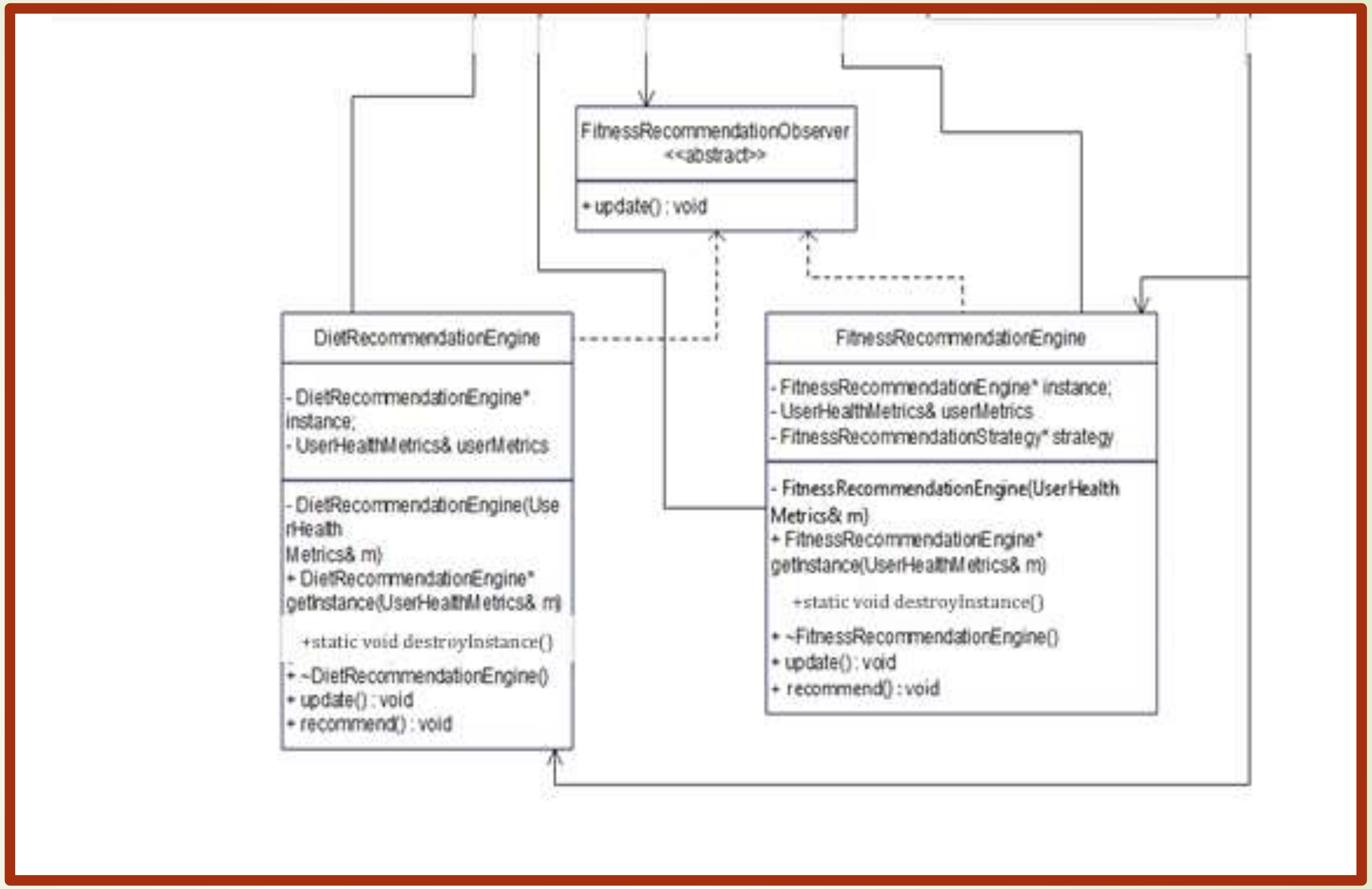
WHICH DESIGN PATTERNS ARE IMLEMENTED?

- *The **Observer Pattern** is helpful when one object needs to tell other objects that something has changed. For example, in a messaging app, when a new message arrives, all the chat windows or notifications can update automatically. The object that changes is called the "subject," and the ones that react to the change are "observers."*

- *The **Strategy Pattern** is useful when an object needs to do something in different ways, depending on the situation. Instead of putting all the different ways inside one class, we separate each one into its own class. Then, we can choose which one to use while the program is running. This makes the code easier to manage and change in the future*

UML CLASS DIAGRAM




FitnessRecommendationObserver
<<abstract>>
+ update() : void
DietRecommendationEngine
- DietRecommendationEngine* instance;
- UserHealthMetrics& userMetrics
- DietRecommendationEngine(UserHealth Metrics& m)
+ DietRecommendationEngine* getInstance(UserHealthMetrics& m)
+static void destroyInstance()
+ ~DietRecommendationEngine()
+ update() : void
+ recommend() : void
FitnessRecommendationEngine
- FitnessRecommendationEngine* instance;
- UserHealthMetrics& userMetrics
- FitnessRecommendationStrategy* strategy
- FitnessRecommendationEngine(UserHealth Metrics& m)
+ FitnessRecommendationEngine* getInstance(UserHealthMetrics& m)
+static void destroyInstance()
+ ~FitnessRecommendationEngine()
+ update() : void
+ recommend() : void

STRATEGY PATTERN

Purpose:

- ❑ *To encapsulate various fitness recommendation strategies based on user activity levels (e.g., Beginner, Intermediate, Advanced).*

Why Chosen:

- ❑ *Allows dynamic switching between different recommendation strategies at runtime depending on the user's activity level.*

Implementation:

- ❑ *The FitnessRecommendationStrategy interface defines a common method recommend().*
- ❑ *Concrete classes like BeginnerFitnessRecommendation, IntermediateFitnessRecommendation, and AdvancedFitnessRecommendation implement this interface, each providing a specific workout plan.*
- ❑ *The FitnessRecommendationContext class holds a reference to a strategy object and calls its recommend() method to provide the appropriate plan.*

OBSERVER PATTERN

Purpose:

- *To automatically react to changes in user health metrics (such as activity level) and update the recommendation strategy accordingly.*

Why Chosen:

- *Ensures real-time and automatic updates of recommendations whenever user health data changes, enhancing responsiveness and decoupling data from logic.*

Implementation:

- *UserHealthMetrics acts as the Subject. It maintains a list of observers (FitnessRecommendationObserver interface).*
- *FitnessRecommendationContext implements this observer interface and registers itself with UserHealthMetrics.*
- *When the activity level changes via setActivityLevel(), UserHealthMetrics calls notify(), which triggers update() on all observers — in this case, prompting the context to select and invoke the appropriate strategy*

CONCLUSION

This project delivers a smart and interactive Fitness Recommendation System that adapts to individual user health metrics. By analyzing inputs like activity level and health data, the system provides tailored fitness suggestions to promote healthier lifestyles. It's designed to be flexible, user-friendly, and capable of growing with future wellness-focused features such as diet tracking or sleep guidance.