**Architecture Description**

**Self-Optimizing Personalized Health Assistant**

**Intelligent System for Assisting Personalized Health Assistance**

**텍스트, 사람이(가) 표시된 사진

자동 생성된 설명**

**Jan. 2023**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Revisions Made** |
| **0.8** | **2/1(Wed), 9pm** | **진수언** | **(Interim) Refining SRS & Apply Context View** |
| **0.9** | **2/14(Tue), 9pm** | **진수언** | **(Pre-final) Designing Skeleton Architecture  & Apply Viewpoints** |
| **1.0** | **2/27(Mon), 9pm** | **진수언** | **(Final) Architecture with All Activities Applied** |

**삼성전자 첨단기술연수소**

**Table of Contents**

[1. Introduction 8](#_Toc128416279)

[1.1. Purpose of the Document 8](#_Toc128416280)

[1.2. System of Interest 8](#_Toc128416281)

[1.3. Definitions, Acronyms, and Abbreviations 9](#_Toc128416282)

[1.4. References 10](#_Toc128416283)

[1.5. Process applied to Architecture Design 11](#_Toc128416284)

[1.6. Template used for Architecture Description 12](#_Toc128416285)

[2. Activity 1. Architectural Requirement Refinement 13](#_Toc128416286)

[2.1. [Step 1] Identify Stakeholders 13](#_Toc128416287)

[2.2. [Step 2] Refining Functional Requirements 13](#_Toc128416288)

[2.3. [Step 3] Architectural Concerns 14](#_Toc128416289)

[2.4. [Step 4] Refine Non-Functional Requirements 16](#_Toc128416290)

[2.5. [Step 5] Write Refined Software Requirement Specification 16](#_Toc128416291)

[3. Activity 2. System Context Analysis 17](#_Toc128416292)

[3.1. [Step 1] System Boundary Context 17](#_Toc128416293)

[3.1.1. Context Diagram 17](#_Toc128416294)

[3.1.2. Description of Context Diagram 18](#_Toc128416295)

[3.2. [Step 2] Functional Context 18](#_Toc128416296)

[3.2.1. Use Case Diagram for the System 18](#_Toc128416297)

[3.2.2. Description of Use Case Diagram 19](#_Toc128416298)

[3.3. [Step 3] Information Context 20](#_Toc128416299)

[3.3.1. Class Diagram for the System 20](#_Toc128416300)

[3.3.2. Description of Classes 21](#_Toc128416301)

[3.4. [Step 4] Behavioral Context 21](#_Toc128416302)

[3.4.1. Allocate Functionality over Tiers 21](#_Toc128416303)

[3.4.2. Define Invocation Patterns 22](#_Toc128416304)

[3.4.3. Draw Activity Diagram for Health Assistant Mobile App 23](#_Toc128416305)

[3.4.4. Draw Activity Diagram for Health Assistant Server 24](#_Toc128416306)

[3.5. [Step 5] Additional Contexts 24](#_Toc128416307)

[4. Activity 3. Skeleton Architecture Design 25](#_Toc128416308)

[4.1. [Step 1] Observe Architectural Characteristics 25](#_Toc128416309)

[4.2. [Step 2] Candidate Architectural Styles 25](#_Toc128416310)

[4.3. [Step 3a] Applicability of ‘Candidate 1. Client-Server Architecture Style’ 25](#_Toc128416311)

[4.3.1. Matching the Applicable Situation 25](#_Toc128416312)

[4.3.2. Matching the Benefits 26](#_Toc128416313)

[4.3.3. Handling Drawbacks 26](#_Toc128416314)

[4.3.4. Evaluation Result 26](#_Toc128416315)

[4.4. [Step 3b] Applicability of ‘Candidate 2. MVC Architecture Style’ 26](#_Toc128416316)

[4.4.1. Matching the Applicable Situation 26](#_Toc128416317)

[4.4.2. Matching the Benefits 27](#_Toc128416318)

[4.4.3. Handling Drawbacks 27](#_Toc128416319)

[4.4.4. Evaluation Result 27](#_Toc128416320)

[4.5. [Step 3c] Applicability of ‘Candidate 3. Event-Driven Architecture Style’ 27](#_Toc128416321)

[4.5.1. Matching the Applicable Situation 27](#_Toc128416322)

[4.5.2. Matching the Benefits 27](#_Toc128416323)

[4.5.3. Handling Drawbacks 27](#_Toc128416324)

[4.5.4. Evaluation Result 28](#_Toc128416325)

[4.6. [Step 3d] Applicability of ‘Candidate 4. Blackboard Architecture Style’ 28](#_Toc128416326)

[4.6.1. Matching the Applicable Situation 28](#_Toc128416327)

[4.6.2. Matching the Benefits 28](#_Toc128416328)

[4.6.3. Handling Drawbacks 28](#_Toc128416329)

[4.6.4. Evaluation Result 28](#_Toc128416330)

[4.7. [Step 4] List of Selected Architecture Styles 29](#_Toc128416331)

[4.8. [Step 5] Integrating Architecture Styles 29](#_Toc128416332)

[4.8.1. Applying Client-Server Architecture Style 29](#_Toc128416333)

[4.8.2. Applying MVC Architecture Style 29](#_Toc128416334)

[4.8.3. Applying Event-Driven Architecture Style 30](#_Toc128416335)

[4.8.4. Applying Blackboard Architecture Style 30](#_Toc128416336)

[4.8.5. Resulting Skeleton Architecture 31](#_Toc128416337)

[4.8.6. Interactions of the Skeleton Architecture 32](#_Toc128416338)

[4.9. [Step 6] Strength and Limitations of the Skeleton Architecture 32](#_Toc128416339)

[4.9.1. Strengths 32](#_Toc128416340)

[4.9.2. Drawbacks 32](#_Toc128416341)

[5. Activity 4. View-specific Architecture Design 33](#_Toc128416342)

[5.1. Functional View 33](#_Toc128416343)

[5.1.1. [Step 1] Observe Functional Characteristics 33](#_Toc128416344)

[5.1.2. [Step 2] Refine Use Case Diagram 34](#_Toc128416345)

[5.1.3. [Step 3] Derive Functional Components 35](#_Toc128416346)

[5.1.4. [Step 4] Refine Functional Components for Tiers 38](#_Toc128416347)

[5.1.5. [Step 5] Allocate Functional Components 39](#_Toc128416348)

[5.1.6. [Step 6] Design Functional Components 40](#_Toc128416349)

[5.1.7. [Step 7] Define Interfaces of Functional Components 41](#_Toc128416350)

[5.2. Information View 43](#_Toc128416351)

[5.2.1. [Step 1] Observe Informational Characteristics 43](#_Toc128416352)

[5.2.2. [Step 2] Refine Persistent Object Model 43](#_Toc128416353)

[5.2.3. [Step 3] Derive Data Components 45](#_Toc128416354)

[5.2.4. [Step 4] Refine Data Components for Tiers 45](#_Toc128416355)

[5.2.5. [Step 5] Allocate Data Components 46](#_Toc128416356)

[5.2.6. [Step 6] Design Data Components 47](#_Toc128416357)

[5.2.7. [Step 7] Define Interfaces of Data Components 47](#_Toc128416358)

[5.3. Behavioral View 48](#_Toc128416359)

[5.3.1. [Step 1] Observe Behavioral Characteristics 48](#_Toc128416360)

[5.3.2. [Step 2a] Refining Control Flow of Health Assistant Mobile App 49](#_Toc128416361)

[5.3.3. [Step 2b] Refining Control Flow of Health Assistant Server 50](#_Toc128416362)

[5.3.4. [Step 3] Choosing Elements for Detailed Control Flow 50](#_Toc128416363)

[5.3.5. [Step 4a] Detailed Control Flow for ‘EK06. Optimize P\_EK’ 51](#_Toc128416364)

[5.4. Deployment View 54](#_Toc128416365)

[5.4.1. [Step 1] Observe Deployment Characteristics 54](#_Toc128416366)

[5.4.2. [Step 2] Define Nodes 54](#_Toc128416367)

[5.4.3. [Step 3] Define Network Connectivity 55](#_Toc128416368)

[5.4.4. [Step 4] Define Artifacts to Deploy 55](#_Toc128416369)

[5.4.5. [Step 5] Allocate Artifacts on Nodes 55](#_Toc128416370)

[6. Activity 5. NFR-specific Architecture Design 56](#_Toc128416371)

[6.1. Design for NFR-1. High Effectiveness of Recommended Activities 56](#_Toc128416372)

[6.1.1. [Step 1] Underlying Facts and Policies 57](#_Toc128416373)

[6.1.2. [Step 2] Criteria for Satisfying NFR 58](#_Toc128416374)

[6.1.3. [Step 3] Candidate Tactics for the Criteria 58](#_Toc128416375)

[6.1.4. [Step 4] Evaluation of the Candidate Tactics 63](#_Toc128416376)

[6.1.5. [Step 5] Impact Analysis of Tactics on Views 65](#_Toc128416377)

[6.1.6. [Step 6] Architecture with Tactics Applied 66](#_Toc128416378)

[6.1.7. [Step 7] Verifying the Traceability 66](#_Toc128416379)

[6.2. NFR-2. High Modifiability of the System 67](#_Toc128416380)

[6.2.1. [Step 1] Underlying Facts and Policies 68](#_Toc128416381)

[6.2.2. [Step 2] Criteria for Satisfying NFR 69](#_Toc128416382)

[6.2.3. [Step 3] Candidate Tactics for the Criteria 69](#_Toc128416383)

[6.2.4. [Step 4] Evaluation of the Candidate Tactics 71](#_Toc128416384)

[6.2.5. [Step 5] Impact Analysis of Tactics on Views 71](#_Toc128416385)

[6.2.6. [Step 6] Architecture with Tactics Applied 72](#_Toc128416386)

[6.2.7. [Step 7] Verifying the Traceability 73](#_Toc128416387)

[7. Activity 6. Architecture Validation 74](#_Toc128416388)

[7.1. [Step 1] Presenting ATAM 74](#_Toc128416389)

[7.2. [Step 2] Presenting Business Drivers 74](#_Toc128416390)

[7.3. [Step 3] Presenting the Architecture 74](#_Toc128416391)

[7.4. [Step 4] Identifying the Architectural Approaches 74](#_Toc128416392)

[7.5. [Step 5] Generating Quality Attribute Tree 74](#_Toc128416393)

[7.6. [Step 6] Analyzing Architectural Approaches 75](#_Toc128416394)

[7.7. [Step 7] Brainstorming and Prioritizing Scenarios 77](#_Toc128416395)

[7.8. [Step 8] Analyzing Architectural Approaches 77](#_Toc128416396)

[7.9. [Step 9] Presenting the Results 77](#_Toc128416397)

[8. Concluding Remarks 78](#_Toc128416398)

**List of Figures**

Figure 1. Examples of Health Assistant Mobile App 7

Figure 2. Tiers of the Target System 8

Figure 3. Process to design Software Architecture 10

Figure 4. Degree of Relevance of NFRs to Stakeholders 14

Figure 5. DFD Diagram for showing Boundary Context 17

Figure 6. Use Case Diagram for the Functional Context 19

Figure 7. Class Diagram for showing Information Context 20

Figure 8. Activity Diagram for showing Behavior Context of Mobile App 23

Figure 9. Activity Diagram for showing Behavior Context of Server 24

Figure 10. Applying Client-Server Architecture Style 29

Figure 11. Applying MVC Architecture Style 30

Figure 12. Applying Event-Driven Architecture Style 30

Figure 13. Applying Blackboard Architecture Style 31

Figure 14. Resulting Skeleton Architecture 31

Figure 15. Refined Use Case Diagram 35

Figure 16. Deriving Functional Components 36

Figure 17. Place Holders for Functional Components 39

Figure 18. Functional Components allocated on Tiers 40

Figure 19. Refined Class Diagram 44

Figure 20. Deriving Data Components 45

Figure 21. Places for Holding Data Components 46

Figure 22. Data Components allocated on Tiers 47

Figure 23. Refined Control Flow of Mobile App tier 49

Figure 24. Refined Control Flow of Server tier 50

Figure 26. Target Elements for Designing Detailed Control Flow 51

Figure 29. Deployment Diagram for the Target System 55

**List of Tables**

Table 1. Profiles of Stakeholders 12

Table 2. Relevance of NFRs to Stakeholders 14

Table 3. Allocation of Functionality over Tiers 22

Table 4. Invocation Patterns defined for Functional Groups 22

Table 5. Functional Components and their Use Cases 36

Table 6. Functional Components allocated on Tiers 38

Table 7. Design of Functional Components 40

Table 8. Interfaces of Functional Components 41

Table 9. Data Components allocated on Tiers 46

Table 10. Invocation Patterns defined for Functional Groups 48

**Architecture Description of  
Self-Optimizing Personalized Health Assistant**

# Introduction

## Purpose of the Document

The purpose of this document is to specify the architecture design for the target system. It describes all the essential architectural aspects of the target system including its structure, functional components, data components, their relationships, runtime behavior, and deployment.

## System of Interest

Self-Optimizing Personalized Health Assistant is a software system that acquires various health indexes, health-relevant activities, performed exercises and stress level to determine the effectiveness of the performed activities, and recommends health-promoting activities to users.

To recommend health-promoting activities, this System maintains a knowledgebase on health-related activities and their relevance to health indicators.

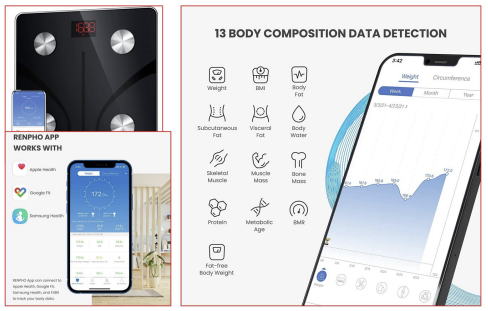


Figure 1. Examples of Health Assistant Mobile App

As shown in the Figure 1, this App provides the ability to measure the user's health status by registering health monitoring devices to measure various health indices.

As shown in Feature2, the target system consists of a mobile app for end users and a server for staff.

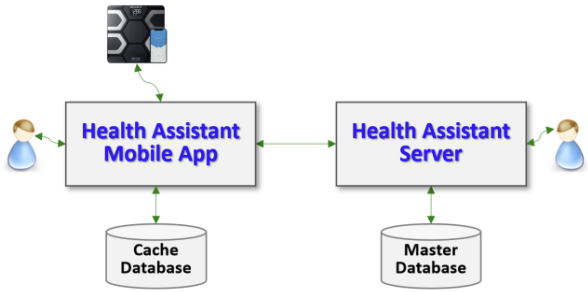


Figure 2. Tiers of the Target System

* Health Assistant Mobile App

This mobile app can connect health monitoring devices that can measure the user's health index and recommend health-promoting activities to the user.

* Health Assistant Server

The server provides the functionality with high complexity for the mobile app and the functionality for the system operators, i.e., staff members. It also maintains the ma.ster database.

## Definitions, Acronyms, and Abbreviations

* Health Index

A health index is a quantitative measurement of a human body condition. Typical Health Indexes include Weight, BMI, Body Fat Percentage, Fat-free Body Weight, Subcutaneous Fat, Visceral Fat, BMR, etc.

* BMI: A value derived from the mass and height of a person.
* Body Fat Percentage: The total (mass of fat) divided by (total body mass), multiplied by 100.
* Fat-free Body Weight: All body components except fat.
* Subcutaneous Fat: Fat that we can pinch and is found just under our skin.
* Visceral Fat: Fat that wraps around abdominal organs deep inside body.
* BMR: Basal Metabolic Rate (BMR) is the rate of energy expenditure per unit time by a person at rest.
* Health-Relevant Activity

Health-relevant activities are human activities that have a slight effect on health indices such as BMI, Body Fat, and Protein Level. Health-relevant activities include Water and Beverage Consumption, Food Consumption, Physical Excellence, Health Food Supplies, and Mental State.

* Health Monitoring Device

A Health Monitoring Device is a device that measures a user's Health Index, such as BMI, Body Fat, and Protein Level.

* Health-Promoting Activity

Health-promoting activity refers to activities deemed necessary to improve the user's health according to the user's health index.

* Personal Effectiveness Knowledgebase(P\_EK)

Personal Effectiveness Knowledgebase (P\_EK) is a user-specific personal knowledge that maintains the relevance of health-related activities on health indexes. The relevance values in this table are automatically computed by the system by observing the health-related activities performed recently and their influences on the health indexes.

* Universal Effectiveness Knowledgebase(U\_EK)

Universal Effectiveness Knowledgebase (U\_EK) is the general knowledgebase that maintains the relevance of health-related activities on health indexes. This table can initially be created by healthcare human specialists.

## References

[Kim 23a] Soo Dong Kim, Associate Architect Program, 2023-A1, CEP Specification of Self-Optimizing Personalized Health Assistant*,* Version 0.9, 삼성전자 첨단기술연수소, Jan. 2023.

[Kim 23b] Soo Dong Kim, Associate Architect Program, 2023-A1, CEP Template for Self-Optimizing Personalized Health Assistant, 삼성전자 첨단기술연수소, Jan. 2023.

[Kim 23c] Soo Dong Kim, Architecture Design Process & Instructions, 2023-A1, Lecture Note #1, 삼성전자 첨단기술연수소, Jan. 2023.

[Kim 23d] Soo Dong Kim, 2023-A1, CEP Specification of Self-Optimizing Personalized Health Assistant*,* Version 1.0, 삼성전자 첨단기술연수소, Jan. 2023.

[ISO 42010] ISO/IEC/IEEE, *Systems and software engineering - Architecture description*, pp. 46, Dec. 2011.

## Process applied to Architecture Design

The process applied to designing software architecture in this sample solution is given [KIM 23c]. It consists of the following six activities as shown in Figure 3.

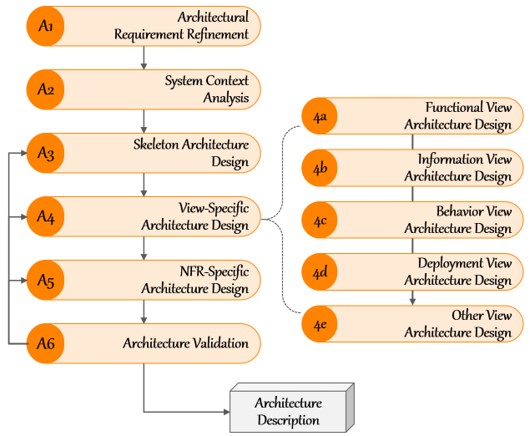


Figure 3. Process to design Software Architecture

* Activity 1. Architectural Requirement Refinement

This activity is to refine the given requirements for developing a target system.

Software Requirement Specification should be refined before designing the architecture of the target system. The principles of requirement engineering can be well applied in this activity.

* Activity 2. System Context Analysis

This activity is to analyze the given requirements for comprehending the target system before making any architectural decisions. The initial comprehension on the target system is specified in the context model of the target system.

* Activity 3. Skeleton Architecture Design

This activity is to design the initial and high-level architecture of the target system, called skeleton architecture. The skeleton architecture mainly specifies the structural aspect of the target system and becomes the stable basis for making additional architectural decisions and defining more detailed architectural elements accordingly.

The skeleton architecture can effectively be derived by utilizing architectural styles.

* Activity 4. View-specific Architecture Design

This activity is to specify more detailed architectural elements for different views. It is advantageous to separate architecture design activities by view, backed by the principle of separate of concerns. Essential Views of Software Architecture are Functional view, Information view, Behavior view, and Deployment view. Utilize viewpoints.

* Activity 5. NFR-specific Architecture Design

This activity is to refine the architecture with additional architectural decisions for each NFR item. Each NFR is thoroughly analyzed and effective architectural tactics are defined to fulfill the NFR. Then, the existing architecture is refined with the defined architectural tactics.

* Activity 6. Architecture Validation

This activity is to validate the resulting architecture design of the target system for both functional and non-functional aspects.

Architecture description becomes a concrete baseline document on which detailed system design are made for implementation. Hence, this activity is essential to confirm the fulfillment of both the functional and non-functional requirements.

## Template used for Architecture Description

The template used for writing this architect description is given in [Kim 23b].

# Activity 1. Architectural Requirement Refinement

This chapter describes the refinements made over the initial requirements of the target system.

## [Step 1] Identify Stakeholders

A stakeholder can be an individual, a group, or an organization. Stakeholders have interests on the target system and concerns that are used as key drivers for designing architecture.

* Stakeholder 1. Client (as Software Distributor)

This represents a manager of this project team in company who determines the main issue of the project.

* Stakeholder 2. App User

This represents users who upload their Health Index and receive Health-Promoting Activity recommendations from the App.

* Stakeholder 3. Healthcare Professional

This represents an expert who evaluates the user's health index, recommends suitable activities, and has extensive and in-depth knowledge about healthcare.

* Stakeholder 4. System Manager

This represents a company staff who operates the app and server system and makes reports for usage analysis.

The profile of each stakeholder is summarized in Table 1.

Table . Profiles of Stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
| **Stakeholder Group** | **Representative Name** | **Contact Information** | **Availability** |
| Client | Linda Johnson | 251-546-9442 Gulf Shores, AL | After 2pm, only on F, Phone Only |
| App User | James Brown | 415-546-4478 San Francisco, CA | Before Noon, MWF, Phone Only |
| Healthcare Professional | Susan Tayler | 949-569-4371 Santa Clara, CA | 10-Noon, T. Th, Office Visits |
| System Manager | David Harris | 408-925-1352 San Jose, CA | All Day, M-F,  Phone Only |

[Step 2] Refining Functional Requirements

Utilize the *SRS Refinement Table* to document the results of requirement refinement.

* Deficiency #1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Deficiency ID** | FR.DEF.01 | **Deficiency Type** | Ambiguity | **Location** | SRS 4.6 |
| **Original Context** | “The system should apply ***advanced software analytics methods*** to analyze the relevance and maintain the personal table of relevance.” | | | | |
| **Questioning** | What kind of advanced software analytic method? | | | | |
| **Refined Context** | “The system should apply ***ML Model*** to analyze the relevance and maintain the personal table of relevance.” | | | | |

* Deficiency #2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Deficiency ID** | FR.DEF.02 | **Deficiency Type** | Ambiguity | **Location** | SRS 4.8 |
| **Original Context** | “Another way of determining the need for optimizing U\_EK is to measure the gap between the initial recommendation made in step 1 and the refined recommendation made in step 2. ***If the average gap between the two recommendations is high***, then the U\_EK may be outdated.” | | | | |
| **Questioning** | How big is the difference between the two recommendations to update the U\_EK? | | | | |
| **Refined Context** | “U\_EK should be updated if the number of recommendations refined by Step2 is more than 10000 times and the number of inconsistencies is more than 30%.” | | | | |

## [Step 3] Architectural Concerns

An architectural concern is a feature or a characteristic of the target system that are raised and defied by stakeholder(s). Hence, architectural concerns represent the stakeholders’ view on the target system and its architecture. Consequently, architectural concerns are expressed in the application domain language, rather than technology languages.

Many of the architectural concerns are requirements and expectations about the target system. And, in fact, many of the concerns in a target system may already be represented in the SRS of the target system in the forms of functional and non-functional requirement items.

The following concerns are acquired from the stakeholders.

* Concern-1. High Effectiveness of Recommended Activities

The recommendation made by the system should highly be effective on promoting the healthiness of users.

* Concern-2. High Modifiability of the System

The target system should be highly modifiable in adding measuring device devices, adding Health Index, adding health-related activities, and modifying the relationship between health-related activities and health indexes.

* Concern-3. Minimize the user's time loss for various functions.

The system has many features that take a long time, including machine learning. This should not affect the user.

The two concerns are mapped to the existing NFR items. One new NFR item is derived from the concern #3.

Merge newly derived NFR items and the NFR items of the SRS. We now have 3 NFR items.

* Concern-1. High Performance 🡪 NFR-1. High Effectiveness of Recommended Activities
* Concern-2. High Extensibility 🡪 NFR-2. High Modifiability of the System
* Concern-3. Minimal Time loss 🡪 NFR-3 High Usability for Users (Newly Added)

We define the relevance of NFR items to the identified stakeholder using the template in Table 2.

Table . Relevance of NFRs to Stakeholders

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **NFR Items** | **Relevance to Stakeholders** | | | | **Average Relevance** | **Standard Deviation** | **Selection (Y/N)** |
| Client | User | Professional | System Manager |
| NFR-1 | 2 | 2 | 1 | 1 | 1.5 | 0.58 | Y |
| NFR-2 | 2 | 0 | 0 | 2 | 1 | 1.15 | Y |
| NFR-3 | 1 | 2 | -1 | -2 | 0 | 1.83 | N |

We apply a 5-level Relevance rating scheme as shown in Figure 4 to fill in the table.



Figure 4. Degree of Relevance of NFRs to Stakeholders

We apply the following guidelines for choosing NFR items

* Case 1) High Average Relevance & Low Standard Deviation ⇒ Choose!
* Case 2) Medium Average Relevance & Low Standard Deviation ⇒ May choose with justification!
* Case 3) Medium Average Relevance & High Standard Deviation ⇒ May not choose with justification!
* Case 4) Low Average Relevance ⇒ Do not choose!
* Case 5) High Average Relevance & High Standard Deviation ⇒ Would not occur.
* Case 6) Low Average Relevance & High Standard Deviation ⇒ Would not occur.

As the result of quantitative assessment on NFR items, we choose NFR-1 and NFR-2.

## [Step 4] Refine Non-Functional Requirements

Utilize the *SRS Refinement Table* to document the results of requirement refinement.

* Deficiency #1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Deficiency ID** | NFR.DEF.01 | **Deficiency Type** | Ambiguity | **Location** | SRS 5.1 |
| **Original Context** | “This recommendation made by the system should ***highly be effective*** on promoting the healthiness of users.” | | | | |
| **Questioning** | How should I determine the criteria for "highly effective" in improving user health? | | | | |
| **Refined Context** | It can be judged effective if the deviation from the improvement that experts think is less than 10%. | | | | |

* Deficiency #2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Deficiency ID** | NFR.DEF.02 | **Deficiency Type** | Incorrectness | **Location** | SRS 5.1 |
| **Original Context** | “When P\_EK is not well optimized to reflect the results of recent user sessions, the user-specific refinement of U\_EK by referring the ***P\_EU*** will not be effective.” | | | | |
| **Questioning** | The word ‘P\_EU’ seems to be a typo of ‘P\_EK’.  The meaning of the above sentence means that U\_EK reflecting P\_EK is not effective if P\_EK is not well optimized.  Does the abbreviation P\_EU have any special meaning? | | | | |
| **Refined Context** | 'P\_EU' is a typo of 'P\_EK'. | | | | |

The resulting SRS now becomes more complete and well-aligned with stakeholders’ concerns.

## [Step 5] Write Refined Software Requirement Specification

The revised SRS is available here [KIM 23d].

# Activity 2. System Context Analysis

This chapter specifies the context of the target system in terms of;

* Target System and Its Boundary
* Functionality provided by the system
* Information manipulated in the system
* Runtime behavior of the system

Additional type of the context can be described.

[Step 1] System Boundary Context

The target system may interact with external systems or other sources in the operational environment of the system. *System Boundary Context* describes the boundary of the system and elements in the environment which interact with the target system. This helps architect and developers to clearly understand the scope of the system.

### Context Diagram

We use *Context Diagram*, i.e. Level 0 of Data Flow Diagram (DFD), which shows each tier of the target system as a process and relationships with its environment.

* Level 0 DFD for Context View

The SRS implies 2 tiers and accordingly the context diagram includes 2 processes as shown in Figure 5.

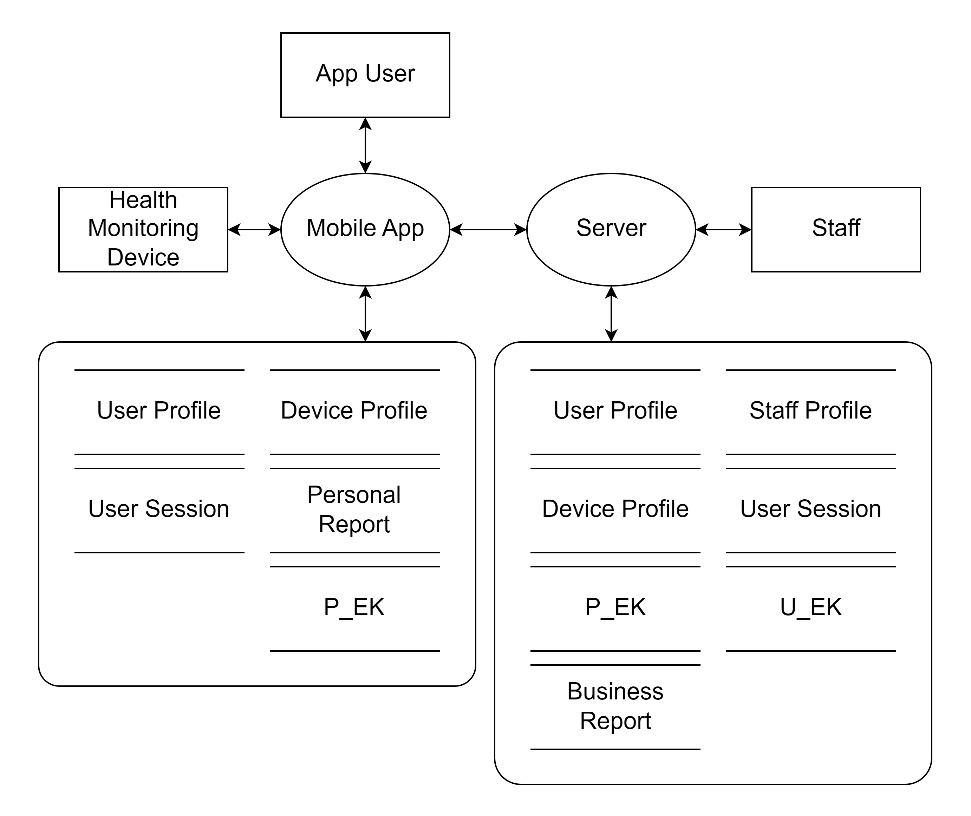


Figure 5. DFD Diagram for showing Boundary Context

### Description of Context Diagram

* Processes
* The process of Health Assistant Mobile App represents a mobile app for users who uses the system.
* The process of Health Assistant Server represents the system server that provides the whole functionality to users.
* Terminals
* Terminals of human types are *App user* and *Staff.*
* Terminals of external devices are *Monitoring Device*.
* Store
* The server maintains the ma.ster repository of all data stores.
* The mobile app maintains a cache database for efficient data access.
* Data Flow

An arrow between two elements depicts a flow of data, and the names of the data on arrows are omitted in the diagram.

[Step 2] Functional Context

### Use Case Diagram for the System

The functional context of the target system can be well described with a use case diagram and descriptions of the use cases. A use case diagram shows the whole functionality of the target system. It is specified with Include actors, use cases, and their relationships.

The following use case diagram shows the whole functionality of the target system. It is specified with Include actors, use cases, and their relationships.

We do not attempt to show the control flow in the use case diagram; rather show only the use cases and invocation relationships. We do not consider tiers; rather we consider the whole system.

* Functional Groups

We apply a scheme for numbering the use cases by considering functional groups. A functional group is a collection of *closely related* use cases. And we assign a two-character prefix to each functional group. A use case diagram with use case identification numbers becomes easier to comprehend and to manage.

Based on the given SRS, we identify the following functional groups and their prefixes.

* User Profile Management 🡪 UP
* Staff Profile Management 🡪 SP
* Health Monitoring Device Registration 🡪 HD
* Acquisition of Health Index 🡪 HI
* Acquisition of Health-Relevant Activity 🡪 HA
* Analytics of Health-Relevant Activities on Controlling Health Index 🡪 AA
* Recommendation of Health-Promoting Activity 🡪 RA
* Self-Optimizing Effectiveness Knowledgebase 🡪 SK
* Report Generation 🡪 RG
* Use Case Diagram

The use case diagram for the functional context is shown in Figure 6.

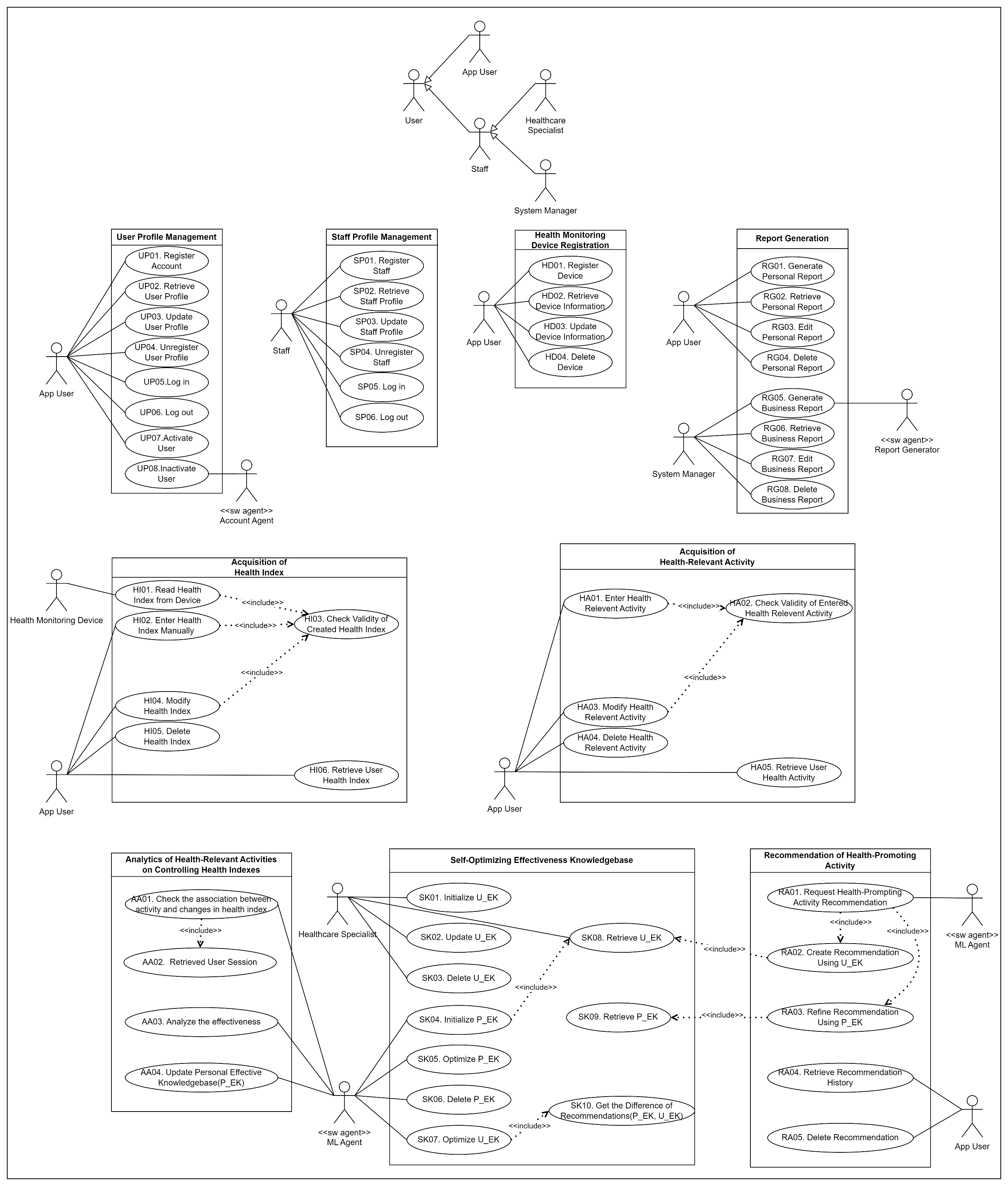


Figure 6. Use Case Diagram for the Functional Context

### Description of Use Case Diagram

* Actors
* User Types: App User, Staff (Healthcare Specialist, System Manager)
* Software Agents
* Account Agent, Report Generator, ML Agent
* External Systems
* Health Monitoring Device
* Use Cases

The use cases in the diagram are directly derived from the functional requirement of SRS. The name of each use case begins with 2 character-long prefix which indicates the functional group it belongs to.

The use cases in a functional group are placed together in the diagram. This will make easier to identify functional components during applying the functional viewpoint.

[Step 3] Information Context

### Class Diagram for the System

The information context of the system shows the datasets manipulated by the system. Class diagram can be effectively used to capture the information context.

For context modeling, show only the entity-type classes, their relationships, cardinalities on the relationships, and a few key attributes. Do not attempt to define methods for each class at this stage.

* Class Diagram

The class diagram for acquiring the information context consists of only classes and their relationships. A relationship is defined with cardinalities.

The context-level class diagram of the target system is shown in Figure 7.

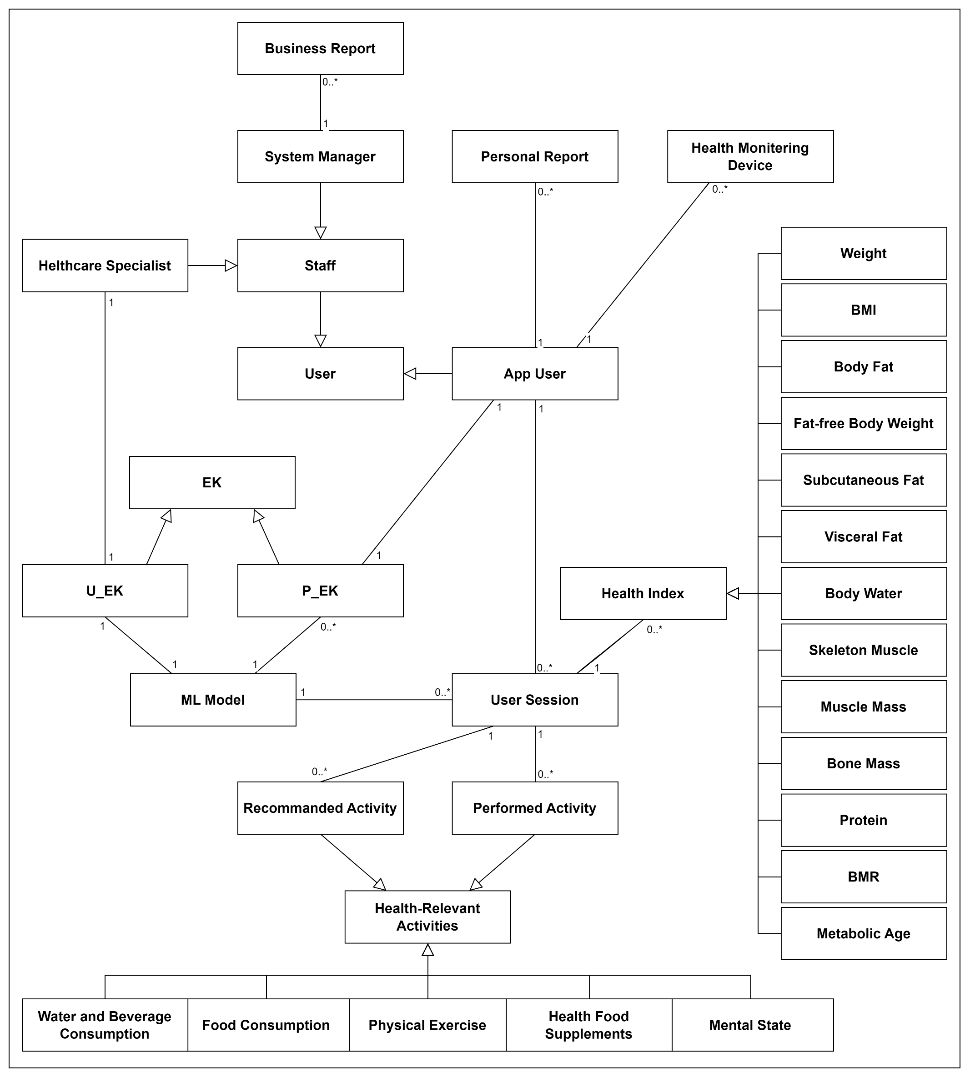


Figure 7. Class Diagram for showing Information Context

### Description of Classes

* Classes of User Types
* *User* is the superclass which captures the common property of its subclasses.
* *Staff* is the superclass which captures the common property of its subclasses.
* *App User* captures the information about users using the app.
* *Healthcare Specialist* captures the information of the staffs managing U\_EK.
* *System Manager* captures the information of the staffs managing the server-side operations.
* Effectiveness Knowledgebase
* *EK* stands for ‘Effectiveness Knowledgebase’ and is a superclass that captures the common properties of its own subclass.
* *U\_EK* stands for ‘Universal Effectiveness Knowledgebase’ and is the general knowledgebase that maintains the relevance of health-related activity on health index.
* *P\_EK* stands for ‘Personal Effectiveness Knowledgebase’ and is a user-specific personal knowledge that maintains the relevance of health-related activity on health index.
* User Session
* *User Session* is a whole object that consists of part elements: *Health Index, Recommended Activity and Performed Activity.*
* Health Index
* *Health Index* captures the quantitative measurement of a human body condition.
* Health Relevant Activity
* *Health Relevant Activity* captures the human’s activity that has some impacts on the health indexes.
* Personal Report, Business Report

This represents the generated reports by the system.

## [Step 4] Behavioral Context

Behavioral context of the system shows the execution and control flow at runtime. Behavioral context may be more important for systems with complex workflows, parallel processing, and timing constraints.

### Allocate Functionality over Tiers

This task is to allocate the system functionality over the tiers. The allocation of the identified functional groups is shown in Table 3.

Table . Allocation of Functionality over Tiers

|  |  |  |
| --- | --- | --- |
|  | **Mobile App** | **Server** |
| User Profile Management | ✓ | ✓ |
| Staff Profile Management |  | ✓ |
| Health Monitoring Device Registration | ✓ | ✓ |
| Acquisition of Health Indexes | ✓ | ✓ |
| Acquisition of Health-Relevant Activities | ✓ | ✓ |
| Analytics of Health-Relevant Activities on Controlling Health Indexes |  | ✓ |
| Recommendation of Health-Promoting Activity | ✓ | ✓ |
| Self-Optimizing Effectiveness Knowledgebase | ✓ | ✓ |
| Report Generation | ✓ | ✓ |

### Define Invocation Patterns

We define appropriate invocation patterns of the allocated functional groups. Each functional group is assigned with one or more invocation patterns. The common types of invocation patterns are Explicit Invocation, Event-driven, Timer-based, and Closed Loop.

The event-driven invocation may occur in two different ways:

* **Event I** is for handling events within a tier, i.e., intra-tier event-driven invocation.
* **Event II** is for handling events among multiple tiers, i.e., inter-tier event-driven invocation.

The invocation patterns defined on the functional groups is shown in Table 4.

Table . Invocation Patterns defined for Functional Groups

|  |  |  |
| --- | --- | --- |
|  | **Mobile App** | **Server** |
| User Profile Management | Explicit | Event II |
| Staff Profile Management |  | Explicit |
| Health Monitoring Device Registration | Explicit | Event II |
| Acquisition of Health Indexes | Explicit | Event II |
| Acquisition of Health-Relevant Activities | Explicit | Event II |
| Analytics of Health-Relevant Activities on Controlling Health Indexes |  | Event II |
| Recommendation of Health-Promoting Activity | Explicit, Event II | Event II |
| Self-Optimizing Effectiveness Knowledgebase | Event II | Explicit, Event I |
| Report Generation | Explicit | Explicit, Timer |

Now, the control flow of the target system can be well modeled based on the specified invocation patterns.

### Draw Activity Diagram for Health Assistant Mobile App

Based on the invocation patterns defined over the tiers, we draw an activity diagram for each tier. Figure 8 shows activity diagram showing the behavior context of the Mobile App.

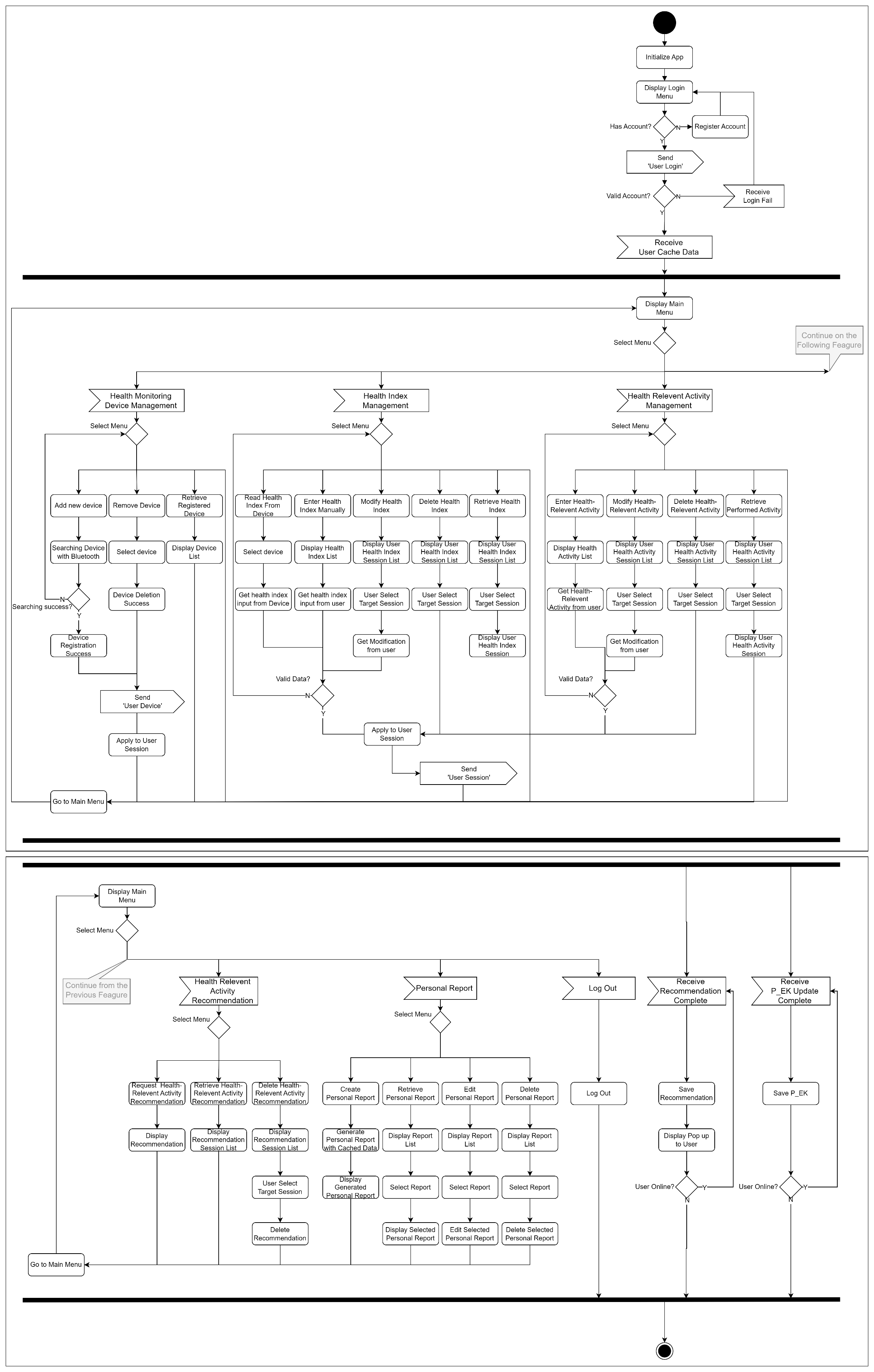


Figure 8. Activity Diagram for showing Behavior Context of Mobile App

The control flow consists of 3 parallel threads. The left most thread is to handle the functionality with explicit invocation pattern using a menu. And the remaining two threads are to handle inter-tier event-driven processing.

The behavior context shows all the functionality of the mobile app as defined in its functional context. That is, all the use cases of the mobile app are reflected in this activity diagram.

### Draw Activity Diagram for Health Assistant Server

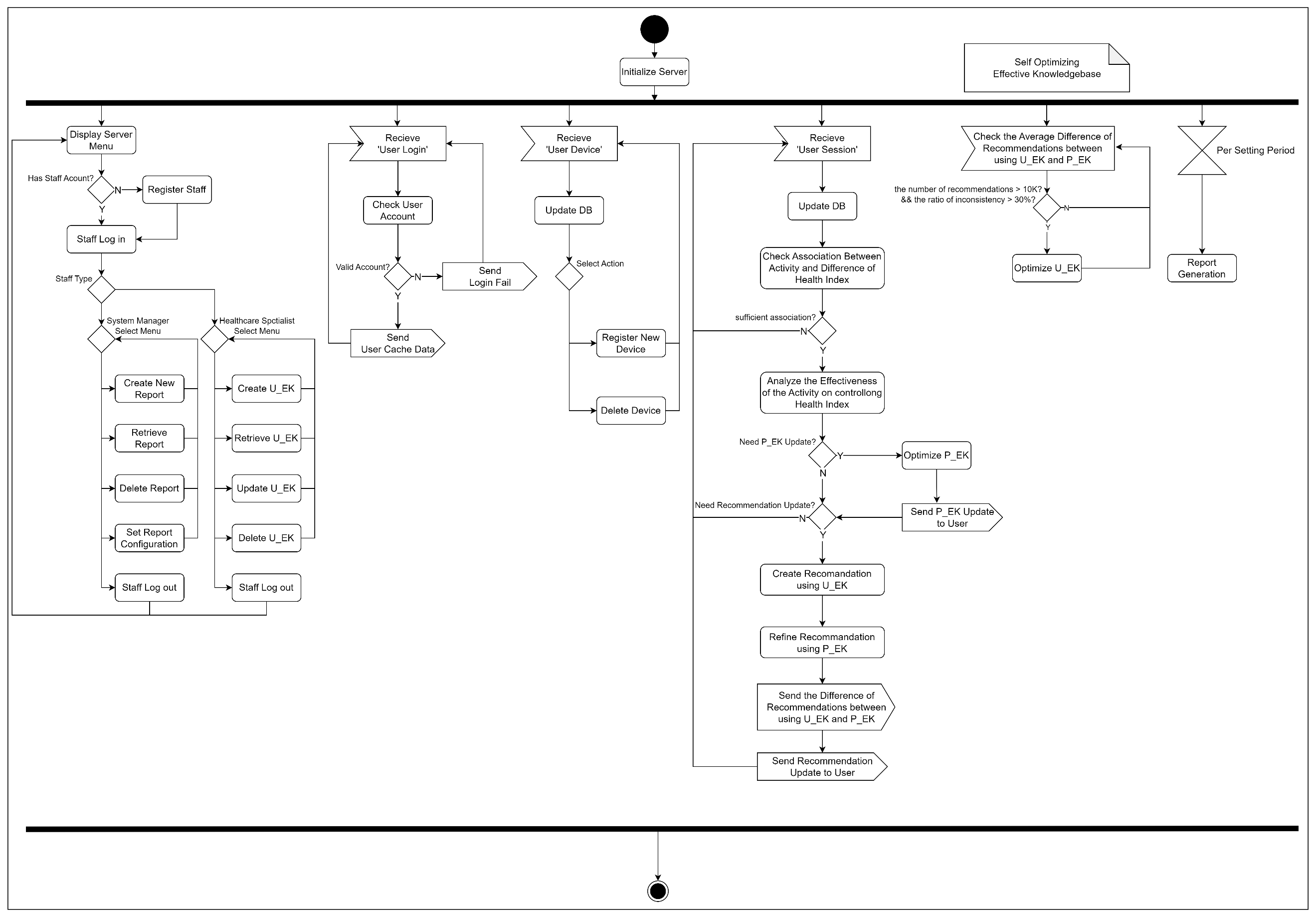


Figure 9. Activity Diagram for showing Behavior Context of Server

The control flow consists of 6 parallel threads. The left most thread is a menu-driven invocation-based behavior that includes actions for staffs: Healthcare Specialist and System Manager. The next three threads are to handle inter-tier event-driven processing. And the remaining two threads are to handle intra-tier event-driven processing.

## [Step 5] Additional Contexts

Any additional contexts of the target system can be described.

* None

# Activity 3. Skeleton Architecture Design

A skeleton architecture is a description of the structural aspect of the target system without fully describing the key components and their properties. It can be effectively derived by applying architectural style(s). Each architectural style is a named collection of architectural decisions that are applicable in a given development context.

[Step 1] Observe Architectural Characteristics

We make the following observation on the target system as a preparation for choosing candidate architecture styles.

* Consisting with Several Tiers 🡪 Client-Server Architecture Style

The number of tiers for the target system is implied by the SRS as already described in the boundary context.

* Conventional Mobile Application 🡪 MVC Architecture Style

Since the target System is a typical mobile application, it is important to maintain a stable GUI with high diversity.

* Considering Event-based Delegation on Server 🡪 Event-Driven Architecture Style

Health-Assistant server can handle asynchronous and synchronous events with different threads to make loose coupled event handling.

* High Effectiveness of Recommended Activity 🡪 Blackboard Architecture Style

In Analyzing Effectiveness function, health-related data goes through various ML algorithms to find the most effective model.

[Step 2] Candidate Architectural Styles

With the architectural observation on the target system, we propose the following architectural styles.

* Client-Server Architecture Style
* MVC Architecture Style
* Event-Driven Architecture Style
* Blackboard Architecture Style

## [Step 3a] Applicability of ‘Candidate 1. Client-Server Architecture Style’

The whole system consists of two tiers and each tier provides an independent functionality running on its dedicated function.

### Matching the Applicable Situation

|  |  |  |
| --- | --- | --- |
| **Applicable Situations** | **Match** | **Demands on the System** |
| Functions can be separated from client and server | 🔾 | The Health Assistant system can be divided into a function based on user interaction including UI and a function that requires a lot of resources such as machine learning and recommendation generation. |
| Multiple users share resources | 🔾 | Since It’s typical mobile application, server resources must be shared when many users use the app. |
| Shared data between client and server is not too much | 🔾 | The functions of this system are clearly separated into client and server, so there is not much shared data |

### Matching the Benefits

|  |  |  |
| --- | --- | --- |
| **Advantages of the Style** | **Match** | **Benefits Applicable to the System** |
| High Performance of server-side computing | 🔾 | Resource-intensive tasks using ML models can be performed on the server. |
| Complex or heavy processing can be centralized | 🔾 | It is possible to reduce the burden on users by executing heavy tasks, etc. on the server. |
| High Flexibility of UI and user device | 🔾 | This system needs to display a lot of UI to users |

### Handling Drawbacks

|  |  |  |
| --- | --- | --- |
| **Cons of the Style** | **Match** | **Handling the Drawbacks** |
| Communication overhead between client and server | 🔾 | There is not much communication between  client and server in this system. It doesn’t  need to consider. |

### Evaluation Result

Client-Server architecture style is well applicable to the target system according to the justification. There is no significant issue which prevents the application of this style.

## [Step 3b] Applicability of ‘Candidate 2. MVC Architecture Style’

The target system can be well configured with the three layers of MVC architecture style. This is applied to mobile app and server.

### Matching the Applicable Situation

|  |  |  |
| --- | --- | --- |
| **Applicable Situations** | **Match** | **Demands on the System** |
| The system can ideally be divided into 3 different roles of View, Control and Model. | 🔾 | Mobile app and server are divided by 3 parts. This division of the system is efficient in management. |
| View and Control are frequently changed. | 🔾 | Health-Assistant provides recommendation. View and Control frequently notified by changing recommendation. |
| The system is highly user-interactive | 🔾 | Many parts of the system are menu driven behavior. |

### Matching the Benefits

|  |  |  |
| --- | --- | --- |
| **Advantages of the Style** | **Match** | **Benefits Applicable to the System** |
| High Maintainability | 🔾 | In this system, since the concern of each part is clear, it is easy to divide and manage them. |
| High Reusability | 🔾 | The system will be compatible with various countries and languages in the future |

### Handling Drawbacks

|  |  |  |
| --- | --- | --- |
| **Cons of the Style** | **Match** | **Handling the Drawbacks** |
| Design complexity increases. | 🔾 | The advantages of maintenance and area expansion outweigh the difficulties in design. |

### Evaluation Result

MVC architecture style is well applicable to the target system according to the justification. There is no significant issue which prevents the application of this style.

## [Step 3c] Applicability of ‘Candidate 3. Event-Driven Architecture Style’

The target system can be well configured with the Event-driven architecture style. This is applied Health Assistant Mobile App and Server.

### Matching the Applicable Situation

|  |  |  |
| --- | --- | --- |
| **Applicable Situations** | **Match** | **Demands on the System** |
| The system consists of event emitters and event sinks. Occurrence of events invoke specific functionality of events. | 🔾 | Health Assistant system use various event signal to service specific functionality. |
| The communication between event emitter and event sinks is asynchronous. | 🔾 | Health Assistant system uses asynchronous event handling for scalability, modularity and parallelization. The communication between event emitter and event sinks is asynchronous. |

### Matching the Benefits

|  |  |  |
| --- | --- | --- |
| **Advantages of the Style** | **Match** | **Benefits Applicable to the System** |
| Decoupling between system, only interact with Event message | 🔾 | Handler is separated as a part and the part can be independently implemented. |
| Scalability is served through independent and decoupled event process. | 🔾 | System can easily add new event handler and provide event handling in parallel. |

### Handling Drawbacks

|  |  |  |
| --- | --- | --- |
| **Cons of the Style** | **Match** | **Handling the Drawbacks** |
| Hard to detect and trace error state | Δ | Health Assistant system does not have monitoring service for system. It is hard to address problem immediately. |
| Difficulty of understanding whole system flow | 🔾 | Health Assistant does not contain event complex event algorithms. The number of events in this system is not very large. |

### Evaluation Result

Event-Driven architecture style is well applicable to the target system according to the justification. There is no significant issue which prevents the application of this style.

## [Step 3d] Applicability of ‘Candidate 4. Blackboard Architecture Style’

The target system can be well configured with Blackboard architecture style. This is applied to Health Assistant server to analyze user’s health data and recommend activity.

### Matching the Applicable Situation

|  |  |  |
| --- | --- | --- |
| **Applicable Situations** | **Match** | **Demands on the System** |
| This pattern is useful for problems in which a determinable solution strategy is unknown. | 🔾 | Health Assistant system cannot determine for sure which analytic method which makes the most effective recommendations. |

### Matching the Benefits

|  |  |  |
| --- | --- | --- |
| **Advantages of the Style** | **Match** | **Benefits Applicable to the System** |
| Static or dynamic adoption of newly published knowledge expertise in solving complex problems. | 🔾 | The recommendation system should be able to dynamically adopt new methodologies to make effective recommendations to users.  NFR2 requires High Modifiability for the methods of analyzing the effectiveness of the activities on controlling health indexes. |
| Quality characteristics for accuracy of data | 🔾 | Health Assistant system should be able to make effective recommendations to users.  NFR1 requires High Effectiveness of Recommended Activities. |

### Handling Drawbacks

|  |  |  |
| --- | --- | --- |
| **Cons of the Style** | **Match** | **Handling the Drawbacks** |
| Changing the structure of the data space has the disadvantage of being difficult to do because all applications are affected.  So, it requires synchronization and access control. | 🔾 | Health Assistant system does not require much structural change in the data space. |

### Evaluation Result

Blackboard architecture style is well applicable to the target system according to the justification. There is no significant issue which prevents the application of this style.

## [Step 4] List of Selected Architecture Styles

All the candidate architecture styles are chosen for defining the skeleton architecture.

* Client-Server Architecture Style
* MVC Architecture Style
* Event-Driven Architecture Style
* Blackboard Architecture Style

## [Step 5] Integrating Architecture Styles

We apply the selected architecture styles incrementally.

### Applying Client-Server Architecture Style

We define 2 tiers for the target system by considering the system functionality as shown in Figure 10.

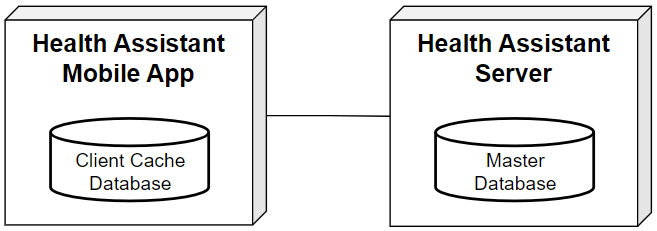


Figure 10. Applying Client-Server Architecture Style

* Health Assistant Mobile App

This mobile provides the whole functionality of end-users. It utilizes the functionality of the server in delivering the whole functionality to end-users.

* Health Assistant Server

The server provides the functionality with high complexity for the mobile app and the functionality for the system operators, i.e., staff members. It also maintains the ma.ster database.

### Applying MVC Architecture Style

Mobile app and server are each composed of MVC, and each component divides functions according to the role of typical MVC.

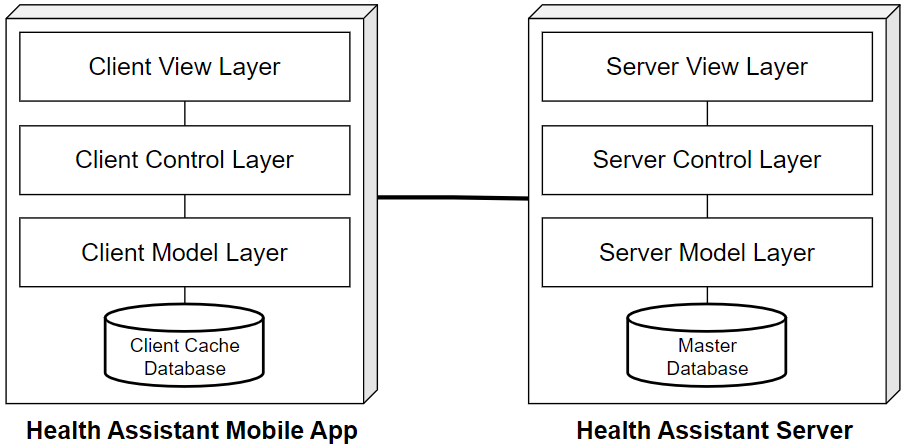


Figure . Applying MVC Architecture Style

* View Layer provides a UI and performs simple functions related to user interaction.
* The Control Layer Performs major business logic.
* The model layer provides an interface to access DB data.

### Applying Event-Driven Architecture Style

For both Health Assistant Mobile App and Server, Event-driven Architecture Style is applied. Analysis result can be modeled as asynchronous events. And user’s input event may also be treated as an event.

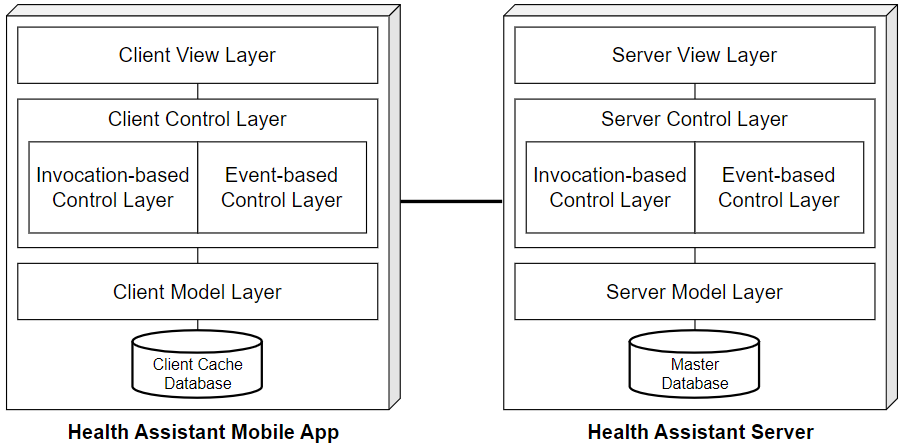


Figure . Applying Event-Driven Architecture Style

* Health Assistant Client tier

When the user enters the Health Index and Activity, the User Session is sent to the server and the core logic for personalized recommendation is used. Throughout this process, the specified set of event processing is requested to the server and the client receives a response as an event.

* Health Assistant Server tier

Server Control layer is consisted of several threads to support both synchronous and asynchronous handling of invocation. It enables parallel processing of external Client events and internal Server thread events.

### Applying Blackboard Architecture Style

The Blackboard architecture style is applied to the Model layer within the Health Assistant Server. The analysis of user data is handled by several knowledge sources, and the analysis with the highest effectiveness is selected.

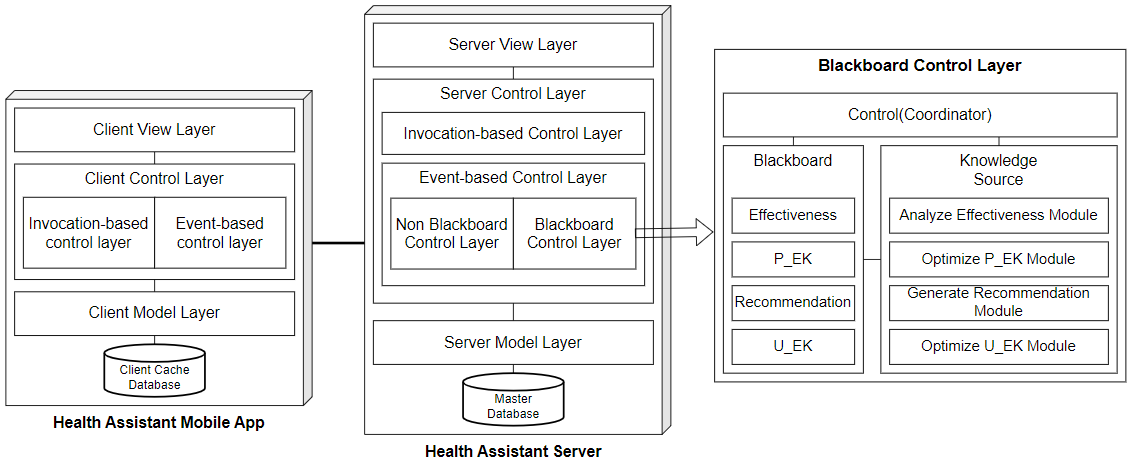


Figure . Applying Blackboard Architecture Style

* Blackboard

Blackboard is a repository of problems to solve, and partial solutions posted by knowledge sources. The problems that need to be solved in our system are analyzing effectiveness, optimizing P\_EK, generating recommendation and optimizing U\_EK.

* Knowledge source

A knowledge source is a software module providing specific expertise relevant to the problems. In our system, software module means an ML Model to solve several problems mentioned above.

* Coordinator(controller)

Coordinator controls the workflow for solving the problem.

### Resulting Skeleton Architecture

The resulting skeleton architecture of applying all the selected architecture styles is shown in Figure 14. .

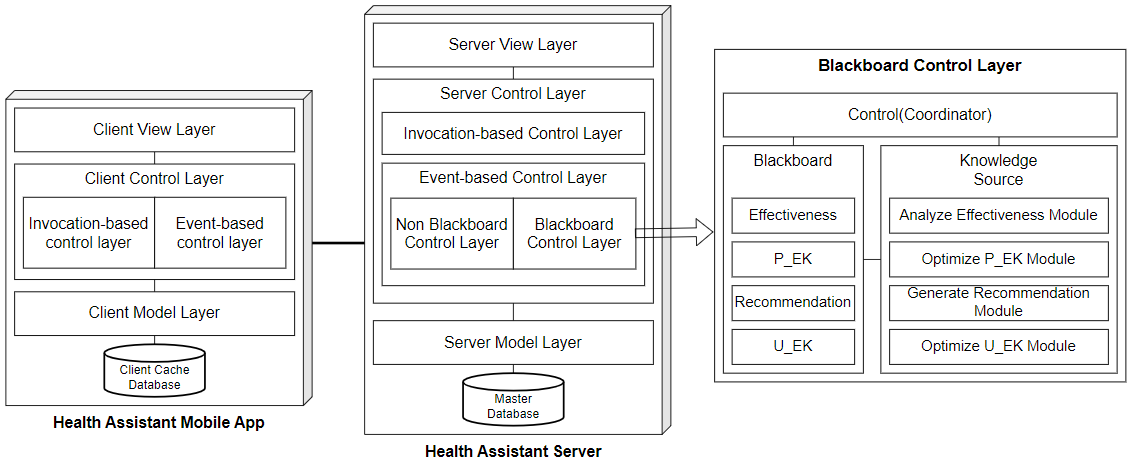


Figure . Resulting Skeleton Architecture

The resulting architecture shows the application of the selected architecture styles. And it serves the stable basis on which view-specific architectural designs can be appended.

### Interactions of the Skeleton Architecture

Define interaction paths among places in the skeleton architecture. An interaction path can be casual dependency or persistent relationship. It provides paths for making function calls or sending messages for communications among components.

* Interaction Paths derived from the Styles.

All the interaction paths defined in each style are adopted in the skeleton architecture.

* Additional Interaction Paths

None

## [Step 6] Strength and Limitations of the Skeleton Architecture

### Strengths

Specify the advantages of the proposed skeleton architecture.

* Separation of Concern

Each component or layer represent a unique and separate concern. It yields a logically well-defined architecture with high modularity.

* Complexity of the System Design and Implementation

Due to the independence of each component or layer, the complexity design is low, and effort to implement the system can be greatly reduced.

* High Maintainability

Due to the key principles applied to designing the skeleton architecture, the impact of modification would be minimal.

### Drawbacks

Specify the drawbacks and risks of the proposed skeleton architecture.

* Not Anticipated.

# Activity 4. View-specific Architecture Design

This chapter describes the results of applying essential architecture viewpoints. The skeleton architecture is now refined with additional architectural decisions made with viewpoints.

## Functional View

### [Step 1] Observe Functional Characteristics

We made the following observations on the system functionality.

* Functionality of Managing Various Profiles

The target system has 3 kinds of profiles to manage: user, monitoring device and staff.

* Functionality of Managing Health Index and Health-Relevant Activity

The target system manages user’s health index and health-relevant activity.

* Analytics of Health-Relevant Activities on Controlling Health Indexes

This functionality is to analyze the effectiveness of health-relevant activities on controlling health indexes of users. Each activity may have some influence on controlling health indexes.

* Recommendation of Health-Promoting Activity

This functionality is to recommend the health-promoting activities for each user based on the personal effectiveness of health-relevant activities performed. The recommendation of the activities to perform is made in two steps.

* Step1. Recommend using the Universal Effectiveness Knowledgebase.
* Step2. Refine the Recommendation using the Personal Effectiveness Knowledgebase
* Self-Optimizing Effectiveness Knowledgebase.

This functionality is to optimize the U-EK and P\_EK on demand.

* Optimizing Personal Effectiveness Knowledgebase

The P\_EK could be updated by the system whenever a user performs a session of entering the health indexes and performed health-related activities. The system analyzes the relevance of the performed activities on controlling each health index and accordingly update the knowledgebase.

* Optimizing Universal Effectiveness Knowledgebase

The U\_EK is meant to be stable and generically applicable to all users. However, the content of U\_EK should be updated when the U\_EK does not reflect recent trends and results of the relevance of health-related activities on health indexes.

* Report Generation

There are two ways to realize this functionality.

### [Step 2] Refine Use Case Diagram

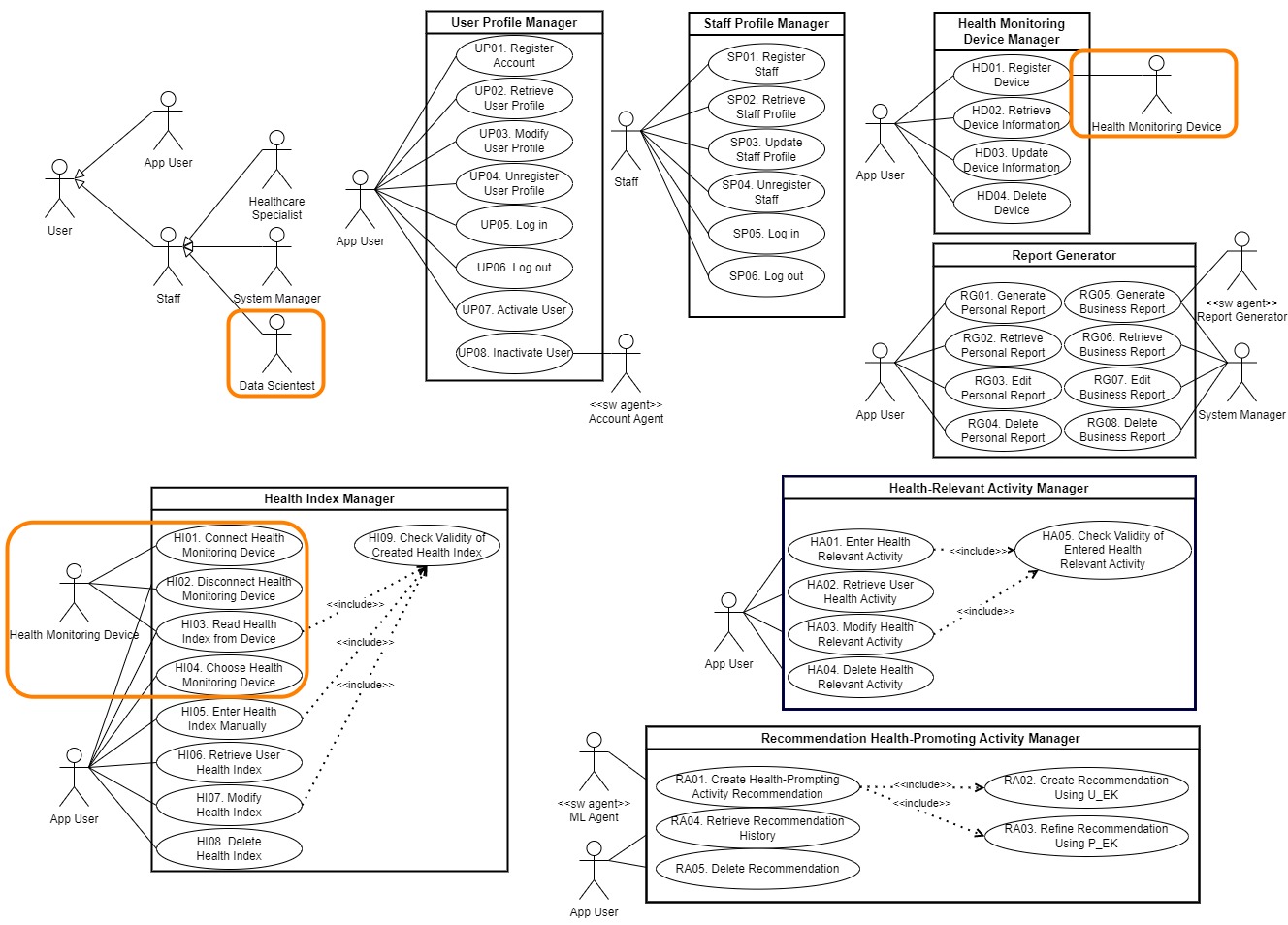
Before identifying functional components, we need to refine the context-level use case diagram with greater details and refinements. We first define the following functional groups for the target system.

* Functional Groups

We refine the functional groups.

* User Profile Management 🡪 UP
* Staff Profile Management 🡪 SP
* Health Monitoring Device Registration 🡪 HD
* Acquisition of Health Index 🡪 HI
* Acquisition of Health-Relevant Activity 🡪 HA
* Health Effectiveness Analytic Manager 🡪 EA
* Recommendation of Health-Promoting Activity 🡪 RA
* Effectiveness Knowledgebase Manager 🡪 EK
* Blackboard Management 🡪 BM
* Report Generation 🡪 RG
* Refined Use Case Diagram (Whole)

The following diagram shows the whole use case diagram. The refined use case diagram includes a number of enhancements and refinements over the context-level use case diagram as shown in



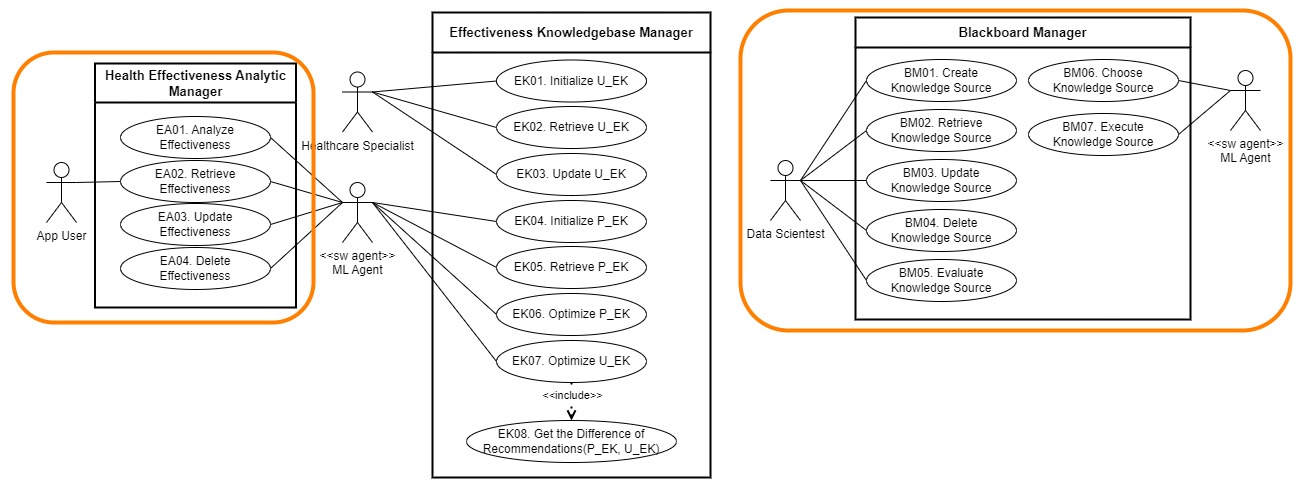


Figure 15. Refined Use Case Diagram

### [Step 3] Derive Functional Components

There are three categories of functional components to consider.

* Category 1. Functional Components derived from the SRS

The functional components are mainly derived from the system-intrinsic functionality, which is well modeled in its use case diagram. That is, the functional components can be systematically derived by clustering relevant use cases.

* Category 2. Functional Components derived from Skeleton Architecture

A skeleton architecture is typically designed by applying architecture styles. An architecture style consists of components and connectors. Some of the components and connects may need to be modeled as functional components.

* Category 3. Interface-centric Functional Components

An interface-centric functional component specifies a stable and public interface, which will be realized/implemented.

The functional components derived from Dispatcher Architecture style are the followings.

* Blackboard Management

This component is to management Blackboard and Knowledge Sources.

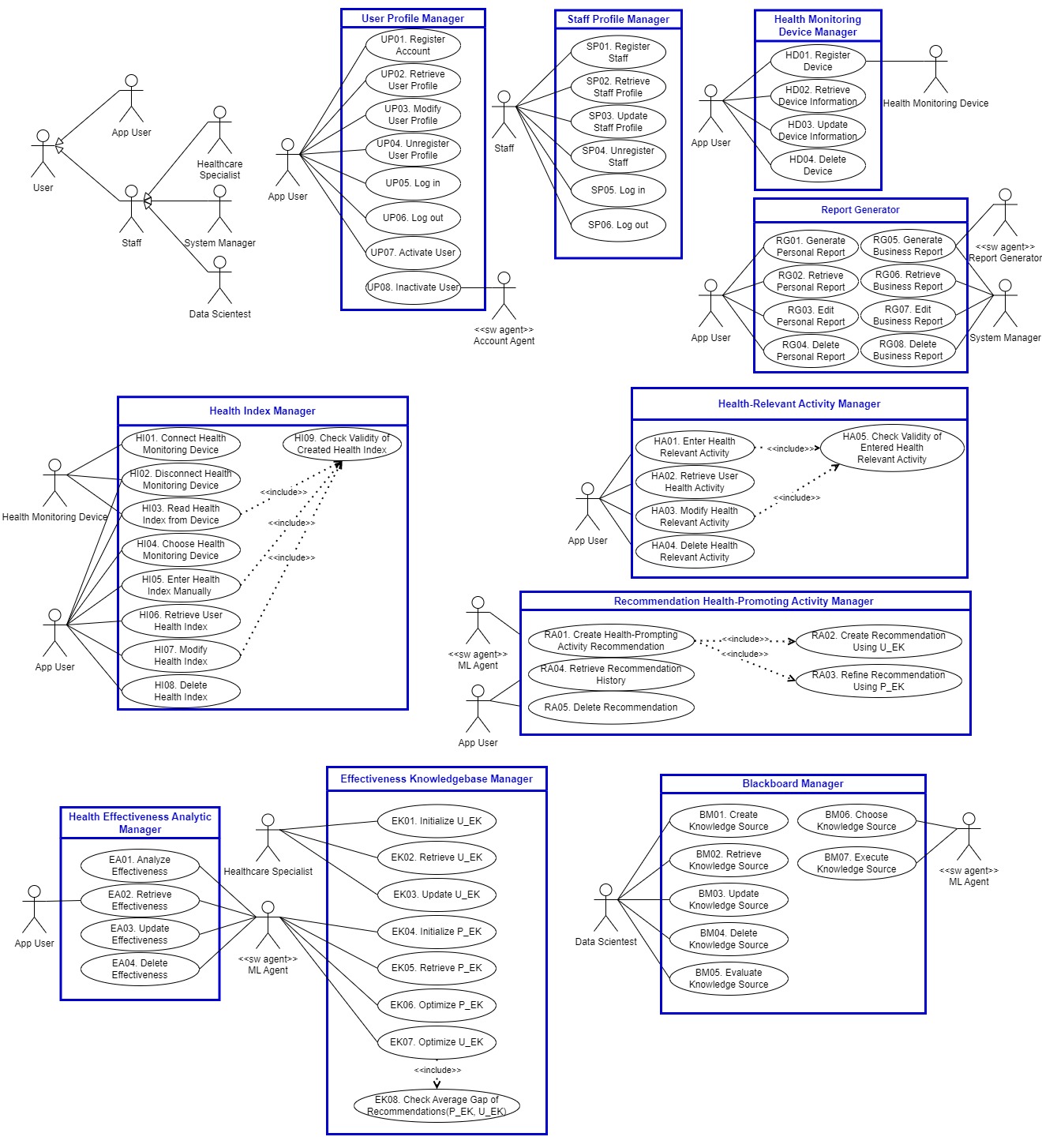


Figure 16. Deriving Functional Components

We summarize the identified functional components and their relevant use cases in a table as shown in Table 5.

Table . Functional Components and their Use Cases

|  |  |
| --- | --- |
| **Functional Components** | **Use Cases** |
| User Profile Management | UP01. Register Account  UP02. Retrieve User Profile  UP03. Modify User Profile  UP04. Unregister User Profile  UP05. Log in  UP06. Log out  UP07. Activate User  UP08. Inactivate User |
| Staff Profile Management | SP01. Register Staff  SP02. Retrieve Staff Profile  SP03. Update Staff Profile  SP04. Unregister Staff  SP05. Log in  SP06. Log out |
| Health Monitoring  Device Registration | HD01. Register Device  HD02. Retrieve Device Information  HD03. Update Device Information  HD04. Delete Device |
| Health Index Manager | HI01. Connect Health Monitoring Device  HI02. Disconnect Health Monitoring Device  HI03. Read Health Index from Device  HI04. Choose Health Monitoring Device  HI05. Enter Health Index Manually  HI06. Retrieve User Health Index  HI07. Modify Health Index  HI08. Delete Health Index  HI09. Check Validity of Created Health Index |
| Health-Relevant Activity Manager | HA01. Enter Health Relevant Activity  HA02. Retrieve User Health Activity  HA03. Modify Health Relevant Activity  HA04. Delete Health Relevant Activity  HA05. Check Validity of Entered Health Relevant Activity |
| Health Effectiveness Analytic Manager | EA01. Analyze Effectiveness  EA02. Retrieve Effectiveness  EA03. Update Effectiveness  EA04. Delete Effectiveness |
| Effectiveness Knowledgebase Manager | EK01. Initialize U\_EK  EK02. Retrieve U\_EK  EK03. Update U\_EK  EK04. Initialize P\_EK  EK05. Retrieve P\_EK  EK06. Optimize P\_EK  EK07. Optimize U\_EK  EK08. Check Average Gap of Recommendations |
| Recommendation Health-Promoting Activity Manager | RA01. Create Activity Recommendation  RA02. Create Recommendation Using U\_EK  RA03. Refine Recommendation Using P\_EK  RA04. Retrieve Recommendation History  RA05. Delete Recommendation |
| Blackboard Manager | BM01. Create Knowledge Source  BM02. Retrieve Knowledge Source  BM03. Update Knowledge Source  BM04. Delete Knowledge Source  BM05. Evaluate Knowledge Source  BM06. Choose Knowledge Source  BM07. Execute Knowledge Source |
| Report Generator | RG01. Generate Personal Report  RG02. Retrieve Personal Report  RG03. Edit Personal Report  RG04. Delete Personal Report  RG05. Generate Business Report  RG06. Retrieve Business Report  RG07. Edit Business Report  RG08. Delete Business Report |

### [Step 4] Refine Functional Components for Tiers

The skeleton architecture of the target system has the following tiers, and hence we refine functional components for the tiers as shown in Table 6.

Table . Functional Components allocated on Tiers

|  |  |  |
| --- | --- | --- |
| **Functional Components** | **Health Assistant Mobile App** | **Health Assistant Server** |
| User Profile Management | mUser Profile Manager | sUser Profile Manager |
| Staff Profile Management |  | Staff Profile Manager |
| Health Device Registration | mHealth Device Manager | sHealth Device Manager |
| Health Index Manager | mHealth Index Manager | sHealth Index Manager |
| Health Activity Manager | mHealth Activity Manager | sHealth Activity Manager |
| Effectiveness Analyze Manager | mEffectiveness Analyze Manager | sEffectiveness Analyze Manager |
| EK Manager |  | EK Manager |
| Recommendation Activity Manager | mRecommendation Manager | sRecommendation Manager |
| Blackboard Manager |  | Blackboard Manager |
| Report Generator | mReport Generator | sReport Generator |

Each functional component is described for its key functionality. Some components are allocated to both tiers, and their functionalities would partially be different.

* Components for Health Assistant Mobile App

The components for this tier provide the functionality for the mobile app users.

* Components for Health Assistant Server

The server provides the functionality with high complexity for the mobile app and the functionality for the system operators, i.e., staff members. It also maintains the ma.ster database.

### [Step 5] Allocate Functional Components

* Identifying Functionality Place Holders

Allocate functional components onto ‘functionality place holders’ of the skeleton architecture. A functional place holder is a layer, a partition, or any place which is defined to host some functionality. Often, the control layer of each tier becomes the functionality place holder.

The places for holding the functional components are shown in Figure 17.

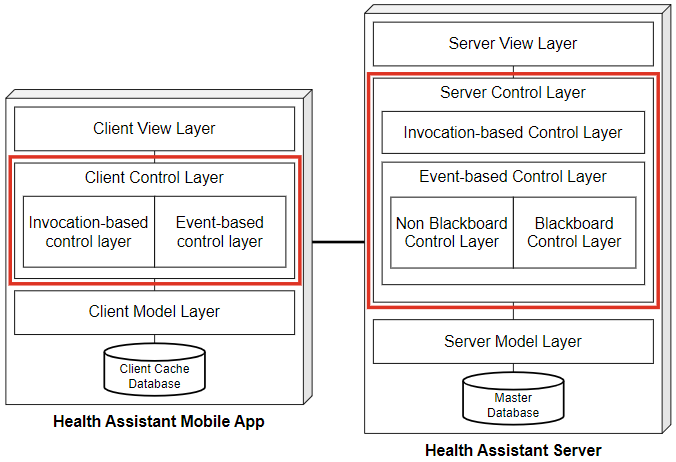


Figure 17. Place Holders for Functional Components

* Functional Components allocated

We now allocate the refined functional components on the functionality holders of each tier according to the table of refined functional components.

The functional components are allocated on the functionality place holders, as in the following figure.

테이블이(가) 표시된 사진

자동 생성된 설명

Figure 18. Functional Components allocated on Tiers

Each place holder is assigned with several functional components. The skeleton architecture is now more complete with functional components.

### [Step 6] Design Functional Components

Each functional component must be designed in greater details. There are different options for designing functional components in detail. We use the following criteria for determining the type of each functional component.

* Criterion 1. Visibility of Functional Component

Whitebox Component or Blackbox Component

* Criterion 2. Type of Interface (for Blackbox Components)

Façade-type or Mediator-type

* Criterion 3. Variability of Functional Component

Closed Component or Open Component

* Criterion 4. Variation Points (for Open Component)

Variation point where the variability occurs.

We use the table of Functional Component Design Decisions as shown in Table 7.

Table . Design of Functional Components

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Refined Functional Components** | **Visibility** | **Interface Type** | **Open/Closed** | **Variant Points** |
| mUser Profile Manager | Blackbox | Façade | Closed |  |
| mHealth Device Manager | Blackbox | Façade | Closed |  |
| mHealth Index Manager | Blackbox | Façade | Closed |  |
| mHealth Activity Manager | Blackbox | Façade | Closed |  |
| mEffectiveness Analyze Manager | Blackbox | Façade | Closed |  |
| mRecommendation Manager | Blackbox | Façade | Closed |  |
| mReport Generator | Blackbox | Façade | Closed |  |
| sUser Profile Manager | Blackbox | Façade | Closed |  |
| Staff Profile Manager | Blackbox | Façade | Closed |  |
| sHealth Device Manager | Blackbox | Façade | Closed |  |
| sHealth Index Manager | Blackbox | Façade | Closed |  |
| sHealth Activity Manager | Blackbox | Façade | Closed |  |
| sEffectiveness Analyze Manager | Blackbox | Façade | Open | Analyze Module |
| EK Manager | Blackbox | Façade | Open | EK Optimize Module |
| sRecommendation Manager | Blackbox | Façade | Open | Recommendation Module |
| Blackboard Manager | Blackbox | Façade | Open | Knowledge Source |
| sReport Generator | Blackbox | Façade | Closed |  |

* Blackbox Components

All the functional components derived from the SRS are defined as the type of blackbox component for the reusability and maintainability.

* Components with Openness

The four components show the variability and hence they are designed with open-design schemes.

### [Step 7] Define Interfaces of Functional Components

A functional component provides its functionality through a *provided* interface. A functional component with ‘open’ design may also need a *required* interface if it accepts a pluggable object as a variant. Note that the ‘Required Interface’ is only one of various ways of designing components with openness.

We first define the names of *provided* Interfaces for functional components. We use a prefix of ‘i’ to indicate an interface name.

Table . Interfaces of Functional Components

|  |  |  |
| --- | --- | --- |
| **Functional Components** | **Health Assistant Mobile App** | **Health Assistant Server** |
| User Profile Management | imUser Profile Manager | isUser Profile Manager |
| Staff Profile Management |  | iStaff Profile Manager |
| Health Device Registration | imHealth Device Manager | isHealth Device Manager |
| Health Index Manager | imHealth Index Manager | isHealth Index Manager |
| Health Activity Manager | imHealth Activity Manager | isHealth Activity Manager |
| Effectiveness Analyze Manager | imEffectiveness Analyze Manager | isEffectiveness Analyze Manager |
| EK Manager |  | iEK Manager |
| Recommendation Activity Manager | imRecommendation Manager | isRecommendation Manager |
| Blackboard Manager |  | iBlackboard Manager |
| Report Generator | imReport Generator | isReport Generator |

We define the interface for each essential component. In this CEP, we define the interface of only one component, *‘EK Manager’*. The use cases included in the functional component are the followings.

* EK01. Initialize U\_EK
* EK02. Retrieve U\_EK
* EK03. Update U\_EK
* EK04. Initialize P\_EK
* EK05. Retrieve P\_EK
* EK06. Optimize P\_EK
* EK07. Optimize U\_EK
* EK08. Check Average Gap
* Provided Interface of *Convert Manager*

Referring the use cases in the component, we define the following method signatures.

* initializeUEK(uek) 🡪 void

This method is to initialize U\_EK.

* retrieveUEK() 🡪 UEK

This method is to retrieve U\_EK.

* updateUEK(uek) 🡪 void

This method is to update U\_EK.

* initializePEK(user id) 🡪 void

This method is to initialize P\_EK using U\_EK.

* retrievePEK(user id) 🡪 PEK

This method is to retrieve specific user’s P\_EK.

* optimizePEK(user id) 🡪 PEK

This method is to optimize P\_EK using specific user’s information.

* optimizeUEK(List<pek>) 🡪 UEK

This method is to optimize U\_EK using user’s P\_EK.

* checkAverageGap(List<recommendation\_pek>, List<recommendation\_uek>) 🡪 Average Gap

This method is to get average gap of recommendations using P\_EK and U\_EK.

## Information View

Architecture design for Information View is to make decisions about persistent datasets, properties, and their management. This activity includes a number of tasks including identifying data components, allocating data components, defining their data contents, ownership, data distribution, replication, migration, data security, and data timeliness.

### [Step 1] Observe Informational Characteristics

We first observe the informational characteristics of the target system.

* Blackboard and knowledge source Classes added
* Health Index Measurement Class added
* Effectiveness Class added
* Refining Relationships

### [Step 2] Refine Persistent Object Model

In object-oriented paradigm, persistent datasets are modeled as entity-type classes. Hence, we refine the context-level class diagram with greater details, which becomes the basis for deriving data components.

* Refined Class Diagram

The revised class diagram shows additional classes and refined relationships as shown in Figure 19.

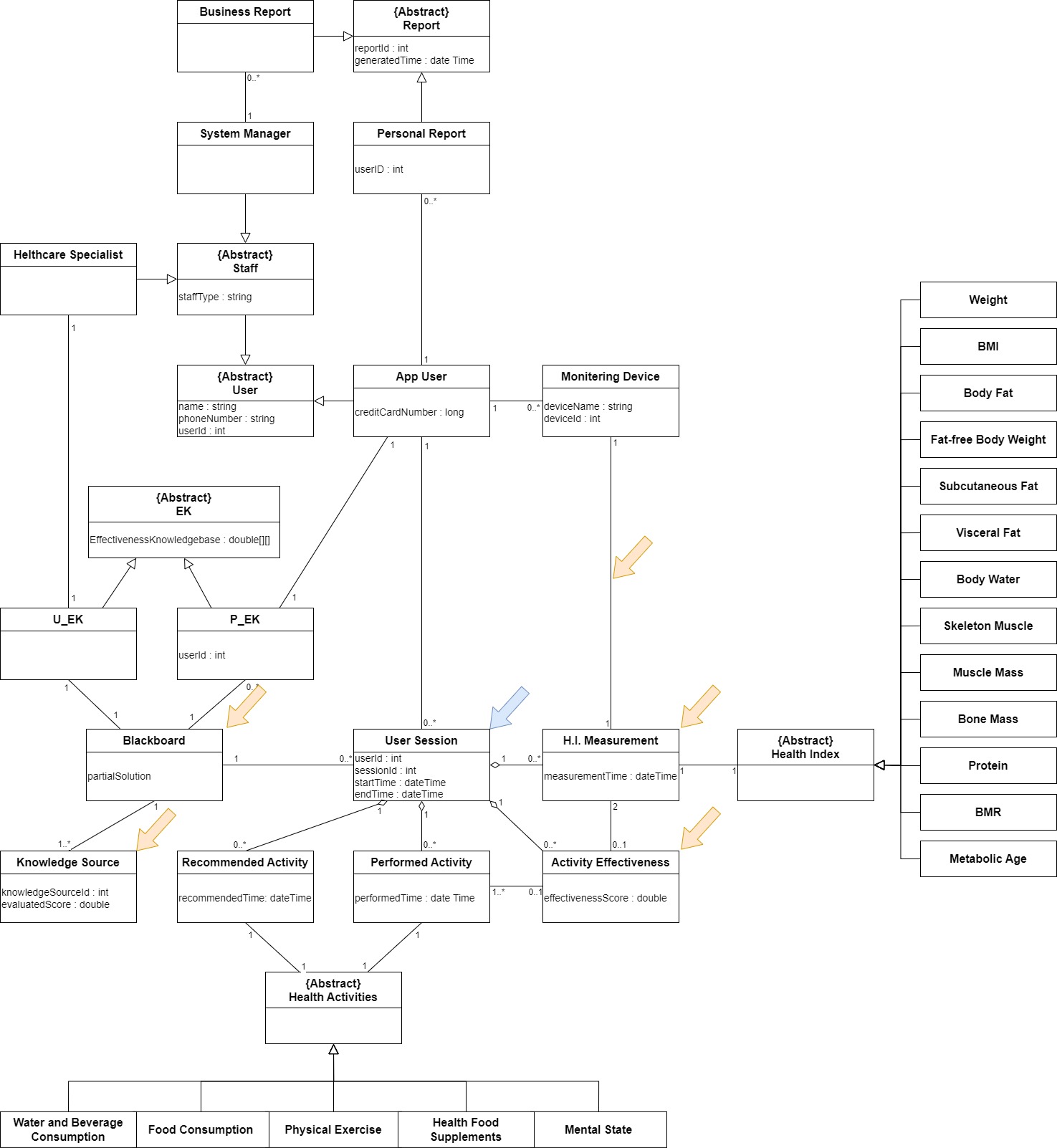


Figure 19. Refined Class Diagram

The diagram shows various refinements.

* 1 Session-related Classes
* User Session.
* Adding New Classes
* Health Index Measurement, Activity Effectiveness, Blackboard, Knowledge Source
* Adding Relationships
* Adding key Attributes to Classes

Each class is refined with key attributes and methods.

* Each class is specified with a textual description.

Omitted in this sample solution.

### [Step 3] Derive Data Components

By considering the strengths of inter-class relationships, we group a set of related classes into a data component as shown in Figure 20.

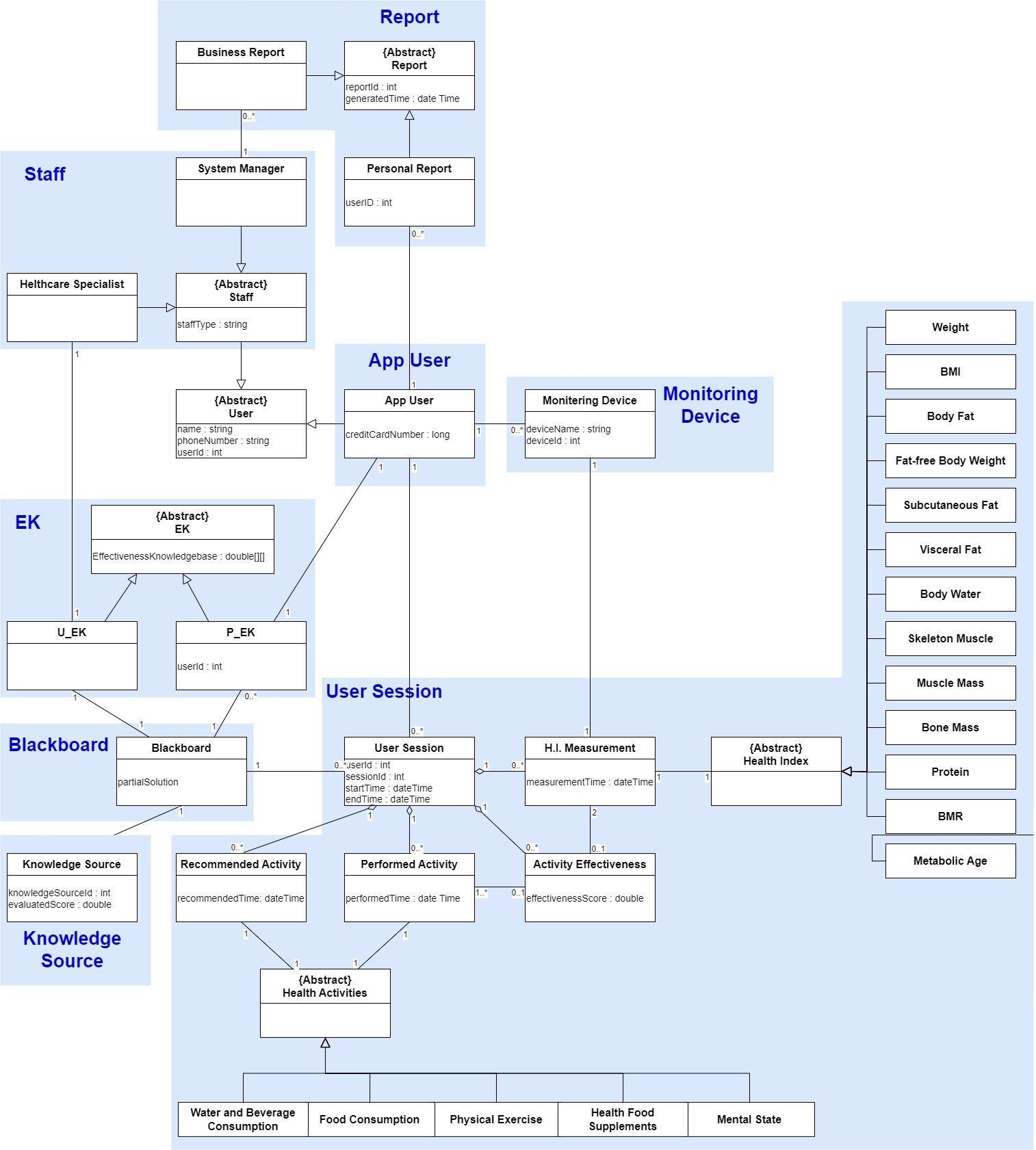


Figure 20. Deriving Data Components

As shown in the figure, classes with strong dependency such as inheritance and compositions are grouped into a same data component. The data components that are derived by considering the strengths of relationships among classes.

* Data Components derived from Architecture Style

Blackboard, Knowledge Source

### [Step 4] Refine Data Components for Tiers

The skeleton architecture of the target system has multiple tiers, and hence we need to refine them for the tiers. Table 9 shows the allocation of data components on tiers.

Table . Data Components allocated on Tiers

|  |  |  |
| --- | --- | --- |
| **Data Components** | **Health Assistant Mobile App** | **Health Assistant Server** |
| App User | mApp User | sApp User |
| Staff |  | Staff |
| Health Monitoring Device | mHealth Monitoring Device | sHealth Monitoring Device |
| User Session | mUser Session | sUser Session |
| EK | mEK | sEK |
| Report | mReport | sReport |
| Blackboard |  | Blackboard |
| Knowledge Source |  | Knowledge Source |

* Data Components on *Health Assistant Mobile App*

These components are used by the functional components of the mobile app tier.

* Data Components on *Health Assistant Server*

These components are used by the functional components of the server tier.

* Considering the Consistency with Allocation of Functional Components

The allocation of data components is well aligned with the allocation of functional components. Hence, the inter-tier access between functional components and data components is not presented.

### [Step 5] Allocate Data Components

Allocate the data components to the appropriate place holders of the architecture. The data component holders for the target system are shown in Figure 21.

* Place Holders for Data Components

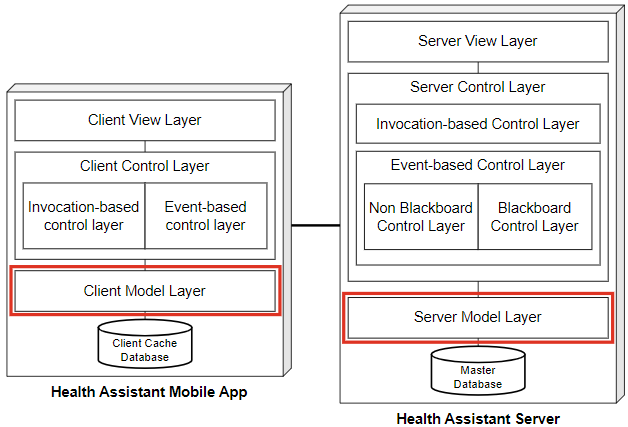


Figure . Places for Holding Data Components

Each tier includes a model layer which hosts the data components except the Dispatcher.

We now allocate the data components on the data component placeholders on each tier. We use the table of refining data components over tiers as shown in Figure 22.

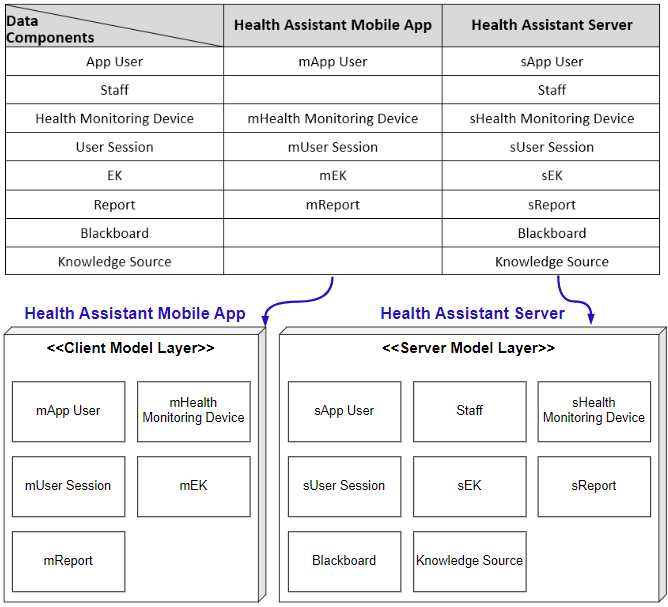


Figure . Data Components allocated on Tiers

The allocation of data components is made according to the table for ‘Data Components Refinement’. Each model layer is allocated with an appropriate set of data components.

### [Step 6] Design Data Components

This step is to design the internal details of data components. This is trivial since each data component consists of classes and each class is defined with persistent attributes.

* Omitted in Sample Solution

### [Step 7] Define Interfaces of Data Components

This step is to define the interface of each data component. The interface for data components is mostly for CRUD-type data manipulation.

* Omitted in Sample Solution

## Behavioral View

The behavioral view of the architecture describes the dynamic aspect of the system, focusing on the runtime behavior of the system.

### [Step 1] Observe Behavioral Characteristics

The observations made in the behavioral context are applicable in this view-level design. The system behavior exhibits the following invocation types.

* Explicit Invocation
* Event-driven Control Flow
* Parallel Processing
* Closed-loop Control Flow

The event-driven invocation may occur in two different ways:

* **Event I** is for handling events within a tier, i.e., intra-tier event-driven invocation.
* **Event II** is for handling events among multiple tiers, i.e., inter-tier event-driven invocation.

The invocation patterns defined on the functional groups is shown in Table 10.

Table . Invocation Patterns defined for Functional Groups

|  |  |  |
| --- | --- | --- |
|  | **Health Assistant Mobile App** | **Health Assistant Server** |
| User Profile Management | Explicit | Explicit |
| Staff Profile Management |  | Explicit |
| Health Device Registration | Explicit | Explicit (Read only) |
| Health Index Manager | Explicit | Explicit (Read only) |
| Health Activity Manager | Explicit | Explicit (Read only) |
| Effectiveness Analyze Manager | Explicit (Read only), Event II | Event II |
| Recommendation Activity Manager | Explicit, Event II | Event II |
| EK Manager | Event II | Explicit, Event II |
| Report Generator | Explicit | Explicit, Timed, C-loop |
| Blackboard Manager |  | Explicit, Event I, C-loop |

Now, the control flow of the target system can be well modeled based on the specified

### [Step 2a] Refining Control Flow of Health Assistant Mobile App

We design the overall control flow of the target system. This is done by refining the context-level activity diagram. If the target system has multiple tiers, each tier has its own control flow and the interaction between the tiers should also be designed.

All the use cases in the functional view are reflected in the activity diagram, and all the actions and activities have their corresponding use cases. Hence, the consistency between the use case diagram and the activity diagram is well-maintained.

The refined control flow of the mobile app tier is shown in Figure 23.

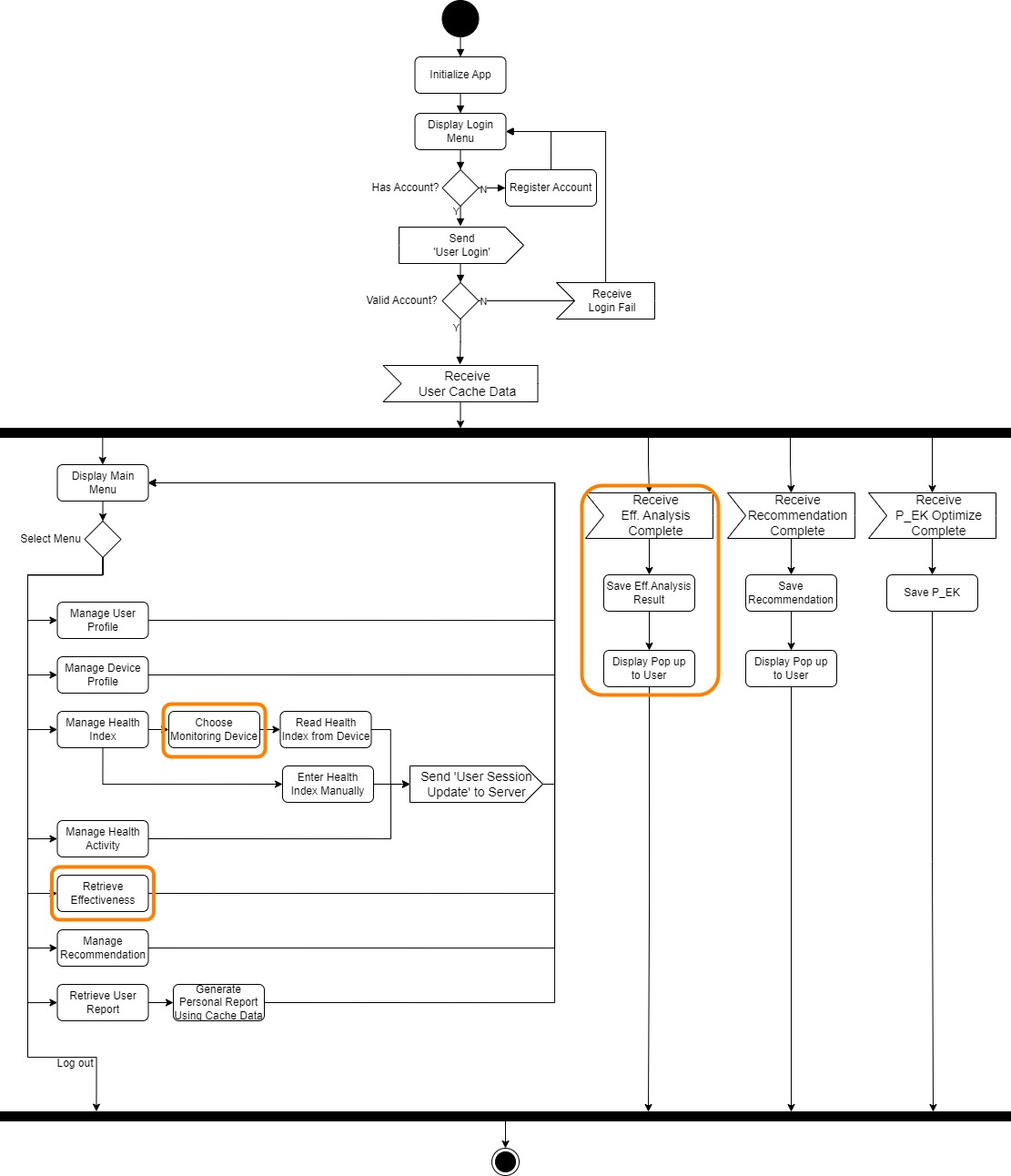


Figure . Refined Control Flow of Mobile App tier

The diagram shows various refinements.

* 2 Functionalities Added: Choose Monitoring Devices, Retrieve Effectiveness
* 1 Thread Added: Receive Effectiveness Analysis Complete from server.

### [Step 2b] Refining Control Flow of Health Assistant Server

The refined control flow of the Health Assistant server tier is shown in Figure 24

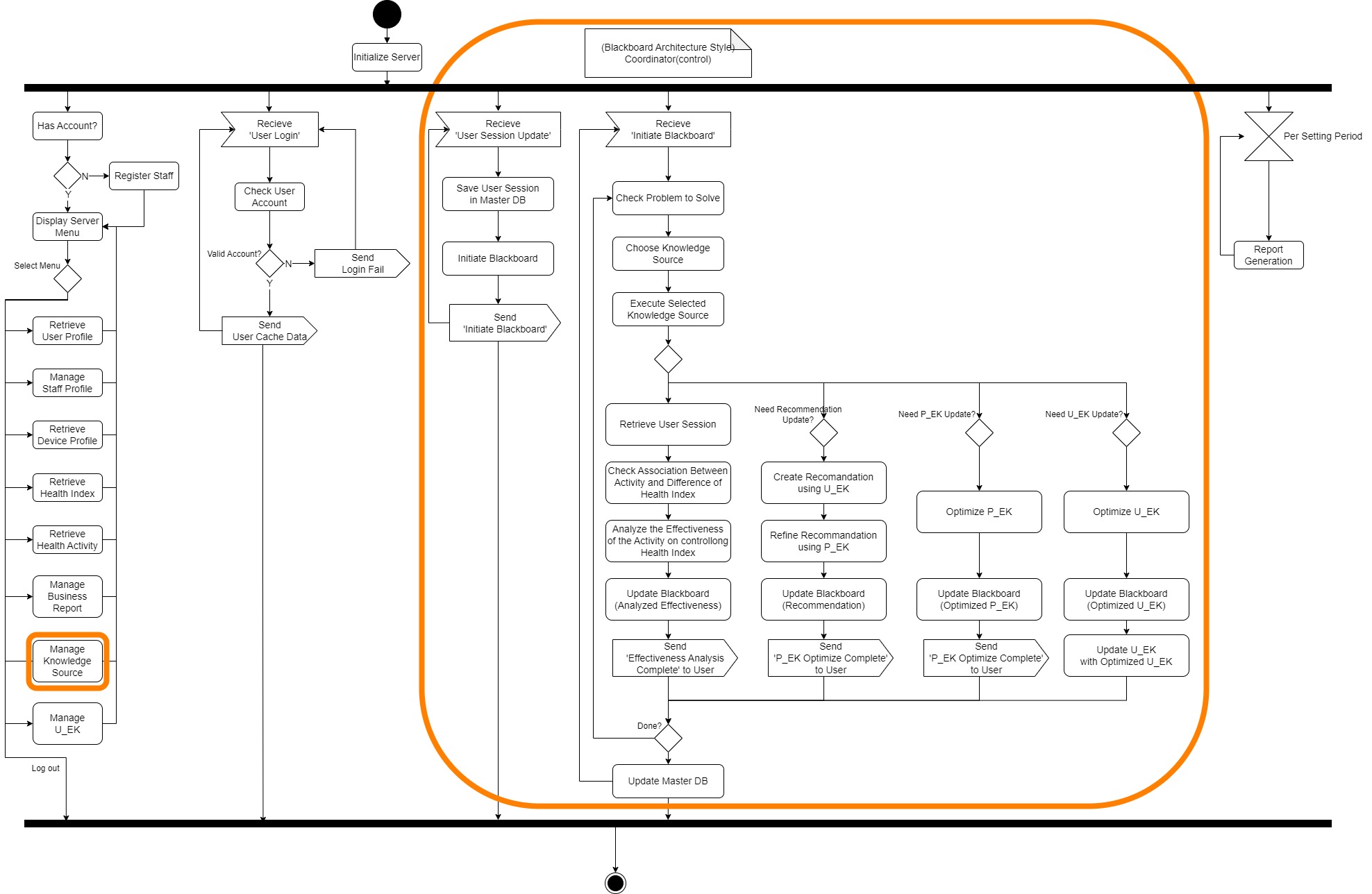


Figure . Refined Control Flow of Server tier

The diagram shows various refinements.

* 1 Functionalities Added: Manage Knowledge Source
* 2 Thread Added: Initiate Blackboard, Blackboard Coordinator(controller)

### [Step 3] Choosing Elements for Detailed Control Flow

In this step, we choose the functionality with complex control flows. That is, we chose use cases in Use Case Diagram, functional components, or actions and/or activities in Activity Diagram. Then, we perform a detailed behavior design for each element chosen.

For the target system, we choose the following elements for detailed control flow as shown in Figure 26.

* Use Case, EK06. Optimize P\_EK, Use Case, EK07. Optimize U\_EK

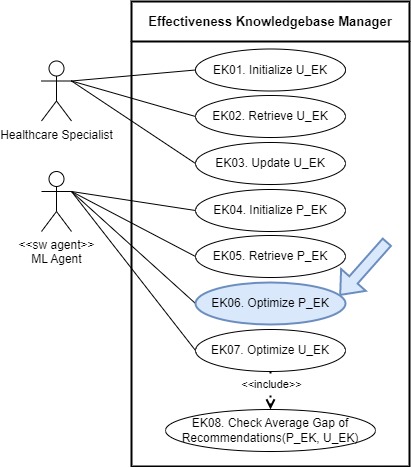




Figure 26. Target Elements for Designing Detailed Control Flow

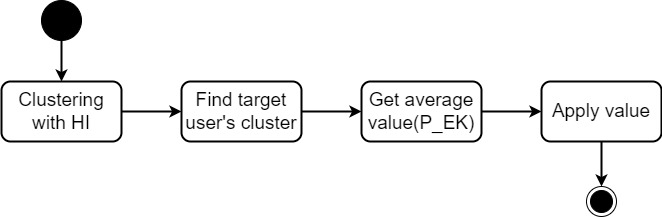
### [Step 4a] Detailed Control Flow for ‘EK06. Optimize P\_EK’

* Functionality

Personal Effectiveness Knowledgebase (P\_EK) is a user-specific personal knowledge that maintains the relevance of health-related activities on health indexes. In this system, the Effectiveness knowledgebase (EK) defines the effect of each health activity (HA) on the health index (HI) as a numeric value in the form of a double between -1 and 1, where 1 is a 100% positive effect and -1 is a 100% negative effect.

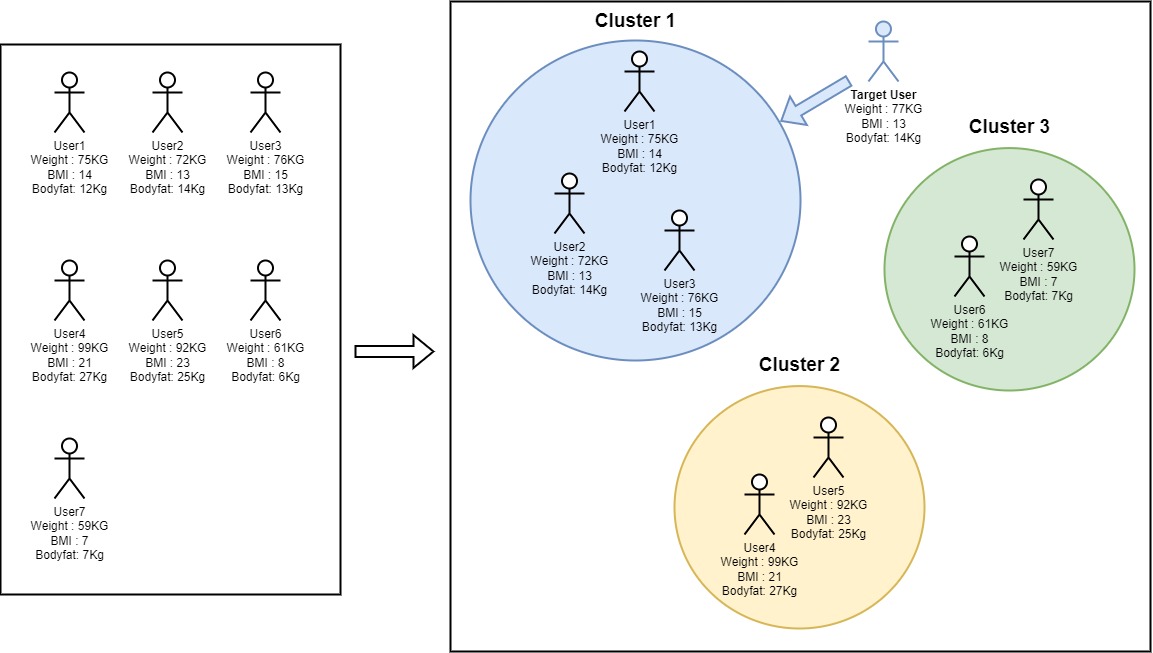
This use case is to optimize Personal EK for individual user attributes in the methodology described below. Each methodology might be implemented as a Knowledge Source.

* #1. Using ‘similar body shape clustering’
* The system uses ‘clustering algorithms’ to find groups of users with ‘similar body shapes’ and optimizes the target user's P\_EK with the P\_EK of that group.
* In this case, the ‘Reliability’ (personalized ratio) of each P\_EK item is 0.   
  ‘Reliability’ is used to create a Recommendation.
* This method is used when there is no information about a specific EK item in the target user (with reliability == 0).
* This method can be used when the P\_EK is outdated or at a specific time interval (ex. 1 week or 1 month).
* A ‘similar body shape’ clustering is to cluster with the user's HA value.
* #2. Update with user input
* This system applies the effectiveness of performed activities on controlling health indexes of target users into a predefined P\_EK.
* In this case, the P\_EK reliability for the updated item increases.
* #3. Using ‘similar constitution clustering’
* If the user cluster does not have data for a specific EK or is significantly lacking, this system uses clustering to find groups of users with ‘similar constitution’
* And sampling in entire cluster to estimate the value as weighted average.
* A "similar constitution" clustering is to cluster the effects of a specific HA on other HIs as a vector.
* Detailed Control Flow
* #1. Using ‘similar body shape clustering’



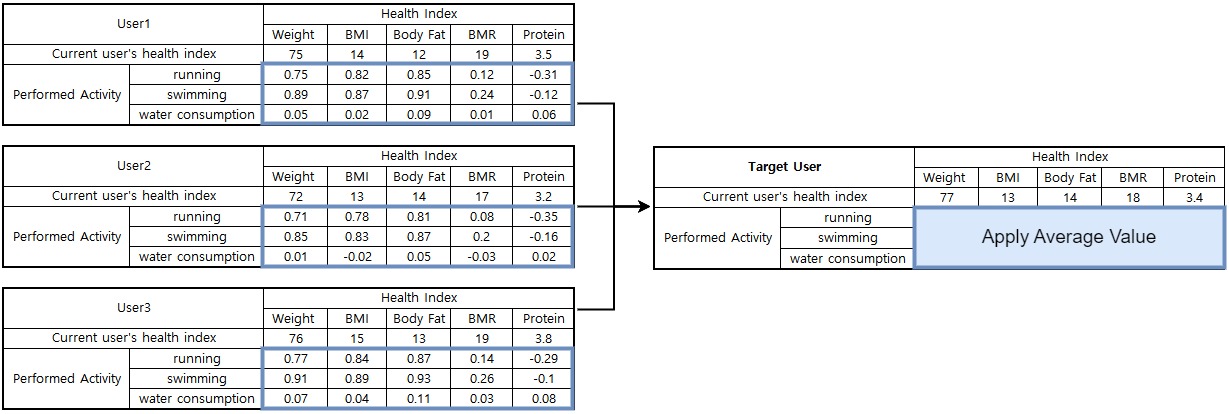
Step1. Clustering user with ‘similar body shape (HI)’

Step2. Find the target user’s cluster.



Step3. Get the average value of the users in target cluster.

Step4. Apply average value in target user’s P\_EK



테이블이(가) 표시된 사진

자동 생성된 설명

* Precondition
* Sufficient Health Index Data of Whole users.
* Sufficient Performed Activity Data of Whole users.
* Input
* HI of all (or sampled) Users.
* P\_EK of all (or sampled) Users.
* HI of target user.
* P\_EK of target user.
* Output
* P\_EK & Reliability Table
* Pseudocode

optimizePEK(user id) {

BEGIN

//Step1. Clustering user with ‘similar body shape (HI)’

HIList = getHIList(sampling number); // Health Index List of sampled users

clusteringResult := clusteringWithUserHI(List<HI>);

//Step2. Find the target user’s cluster.

HI = HI(user id);

targetCluster := findCluster(clusteringResult, HI);

//Step3. Get the average value of the users in target cluster.

for (tmpUser ∈ targetCluster) {

tmpPEK += getPEK(tmpUser);

}

averagePEK = tmpPEK / targetCluster.getNumOfUser( );

//Step4. Apply average value in target user’s P\_EK

targetPEK = getPEK(user id);

for(tmpHA ∈ P\_EK, tmpHI ∈ P\_EK) {

averageEK = getEK(averagePEK, tmpHA, tmpHI);

targetEK = getEK(targetPEK, tmpHA, tmpHI);

if (targetEK.checkReliability == 0) {

targetEK.setValue(averageEK)

}

else {

tmpEK = getWeightedAverage(targetEK, averageEK);

targetEK.setValue(tmpEK);

}

}

return targetPEK;

END

## Deployment View

Deployment view of the architecture is concerned with the topology of software components on the physical layer, as well as the physical connections between these components.

### [Step 1] Observe Deployment Characteristics

* The mobile client will be implemented for Android and iOS for now.
* The Health Assistant Server will be realized with a high-end server with sufficient computing resources including multiple GPUs for running ML Model.

### [Step 2] Define Nodes

The skeleton architecture of the target systemconsists of two nodes. Each node is configured with a hardware specification and its execution environment.

* Node 1. Health Assistant Mobile App
* Hardware Specification
* Smartphone or Tablet PCs running Android or iOS
* No Minimal Amount of Computing Power and Resource
* Execution Environment
* Operating System: Android and iOS
* Node 2. Health Assistant Server
* Hardware Specification
* High-end CPU with a minimum of 10 cores
* High Capacity of Main Memory for Training
* Multiple Units of High-end GPU for Training
* Execution Environment
* Operating System: Window 11
* Django Framework, PyTorch

### [Step 3] Define Network Connectivity

* Between Health Assistant Mobile App and Health Assistant Server: HTTP-based network with LTE or 5G

### [Step 4] Define Artifacts to Deploy

* Artifacts for Health Assistant Mobile App
* Functional Components specified in Functional View-design
* Implementation of Behavior View-design
* Artifacts for Health Assistant Server
* Functional Components specified in Functional View-design
* Data Components specified in Information View-design
* Implementation of Behavior View-design

### [Step 5] Allocate Artifacts on Nodes

This step is to allocate all the deployable artifacts and show the network connections.

We use a Deployment Diagram to represent the nodes and artifacts as shown in Figure 29.

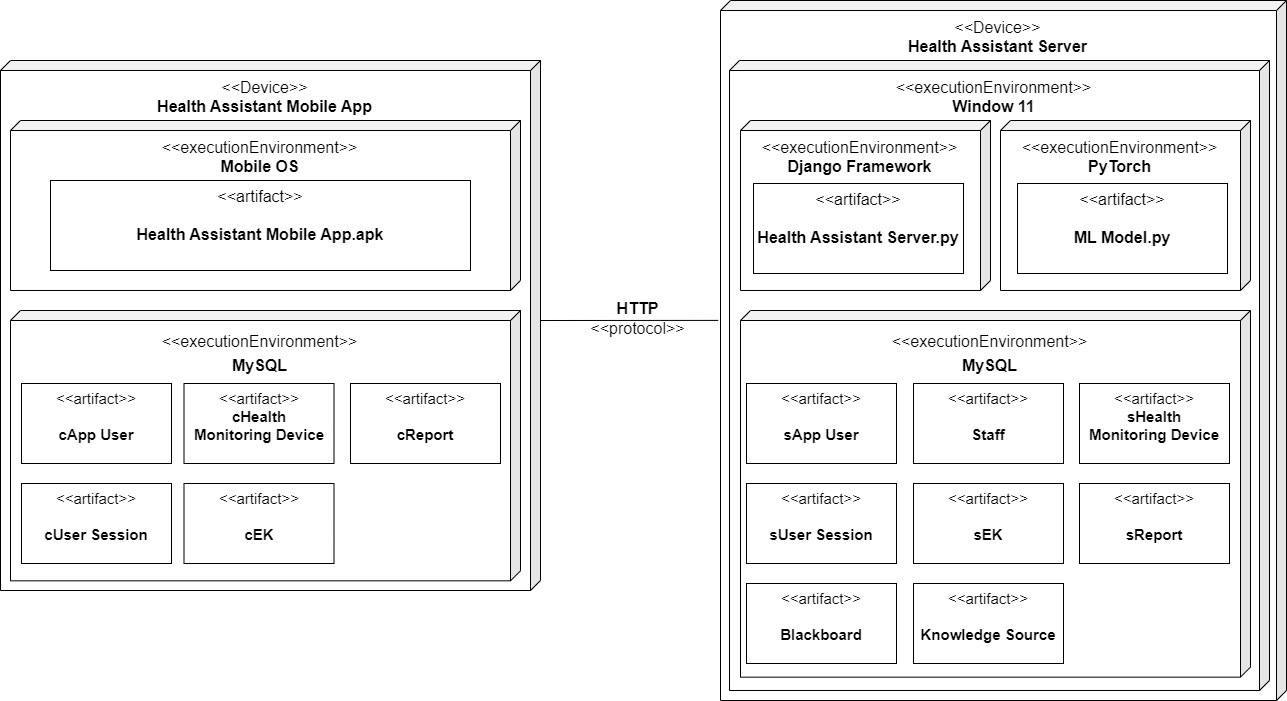


Figure 29. Deployment Diagram for the Target System

# Activity 5. NFR-specific Architecture Design

This chapter shows the architectural design for the given NFRs. The following process is used for applying NFR-based design.



## Design for NFR-1. High Effectiveness of Recommended Activities

A key functionality of the target system is to recommend users with health-promoting activities. This recommendation made by the system should highly be effective on promoting the healthiness of users.

When the recommended activities are not sufficiently effective in promoting health condition, users will experience the waste of valuable time and effort spent for performing the activities.

* Recommendation and Effectiveness Knowledgebases
* Both U\_EK and P\_EK become the basis for deriving the recommendation of health-related activities
* To make effective recommendations, both U\_EK and P\_EK should be well defined and optimized by reflecting recent user sessions of performing the actions and improving the health conditions. That is, U\_EK and P\_EK should be is optimized correctly by the system.
* When U\_EK is not generic enough for all the users, it will result in less effective recommendation. When P\_EK is not well optimized to reflect the results of recent user sessions, the user-specific refinement of U\_EK by referring the P\_EU will not be effective.
* Accuracy and Validity of Health Index Measurements

The accuracy and validity of the measurements become a factor in determining the effectiveness of recommendations made. Hence, the health index measurements entered from a device or by a user should be checked before being processed.

### [Step 1] Underlying Facts and Policies

We define the following facts and policies regarding the NFR.

* (F1) High accuracy of recommendation algorithms

A key function of the target system is to recommend users who engage in health promotion activities. This recommendation made by the system is very effective in improving the health of users in a good way. Clustering algorithm is widely used in various recommendation systems.

* (F2) Large volume of dataset

Running an effective recommendation algorithm requires a large amount of data. This system needs different types of data sets that can be classified by user body type, constitution, and age group. Various data from different users is essential for clustering.

* (F3) Generalized U\_EK

U\_EK refers to a knowledge base that is suitable for general users. If U\_EK is not common enough for all users, the recommendation will be ineffective.

* (F4) Personalized P\_EK with reflecting recent user session.

This system defines and optimizes P\_EK by reflecting recent user sessions. Old data from users may not be appropriate for user’s current body type. This system provides the correct recommendation for your current situation. When P\_EK is not well optimized to reflect the results of recent user sessions, the user-specific refinement of U\_EK by referring the P\_EK will not be effective.

* (F5) Recommendations using U\_EK and P\_EK

Both U\_EK and P\_EK are the basis for deriving recommendations for health-related activities. U\_EK is a knowledge base for the effectiveness of all users, and P\_EK is a knowledge base personalized to a specific user. Therefore, providing recommendation to users by utilizing both U\_EK and P\_EK can have a more positive effect on users.

* (F6) Self-optimizing P\_EK and U\_EK

When users enter performed health-relevant activities and their health indexes, each user wants to receive personalized recommendations in real time. Automated analysis and recommendation systems are needed to make effective recommendations to users in real time.

* (F7) Valid health index measurements.

The accuracy and validity of the measurements become a factor in determining the effectiveness of recommendations made.

### [Step 2] Criteria for Satisfying NFR

We derive the criteria for devising architecture tactics, from the facts and policies.

* (C1) High accuracy of recommendation algorithm (Relevant to F1)

The recommendation made by this system should highly be effective on promoting the healthiness of users. Therefore, this system should select a recommendation model to generate highly effective recommendations.

* (C2) Collecting various dataset (Relevant to F2)

Various datasets must be collected for clustering in various ways. The collected data should not be biased against a particular class. The collected data should include data of various age groups, body types, and constitution.

* (C3) Generalized U\_EK (Relevant to F3)

The U\_EK should contain information about the effectiveness of all users. If the number of users is too large to reflect all users' data, the data of as many users as possible should be reflected.

* (C4) Reflecting recent user sessions on personalized P\_EK (Relevant to F4)

This system should optimize P\_EK by reflecting recent user sessions. Recent data should be treated more heavily than old data.

* (C5) Recommendation using both U\_EK and P\_EK (Relevant to F5)

This system must generate a recommendation using U\_EK, which is generated from all users' data, and then refine the recommendation using personalized P\_EK.

* (C6) Self-optimizing P\_EK and U\_EK (Relevant to F6)

Optimization of P\_EK and U\_EK should be automatically performed to provide effective recommendation to users in real time.

* (C7) Validity of health index measurements. (Relevant to F7)

This system should validate the health index measurements entered from a device or by a user before processing.

### [Step 3] Candidate Tactics for the Criteria

We define the following candidate architecture tactics that satisfy the given criteria.

* (T1) Select the appropriate recommendation method (Relevant to C1)

The ability to make effective recommendations to users is a high level of technology. Recommendation algorithms using clustering techniques are widely used in E-commerce or OTT service. Three possible options were identified by searching the paper.

We evaluate the proposed candidate clustering tactics in terms of their benefit and cost.

|  |  |  |
| --- | --- | --- |
| **Clustering Algorithm** | **Justification** | **Selected** |
| K-Means | Strengths: Short execution time, Intuitive algorithm, Various vector applicability | O |
| Weakness: Hard to define K, Nondeterministic Result |
| Mean-Shift Clustering | Strengths: K is automatically determined by r, |  |
| Weakness: Hard to find r, Difficult to handle various vectors |
| DBSCAN | Strengths: K is automatically determined by ε, |  |
| Weakness: Hard to find ε, Effectiveness determined by ε Vulnerable to outlier |

Since our system plans to cluster not only with the user's Age and gender but also with various HI combinations, we finally selected the K-means algorithm, which has a small change in implementing the clustering algorithm according to the vector change.

* (T2) Applying clustering method on Optimize U\_EK and P\_EK (Relevant to C1)

In our system, EK stands for a table showing the effects of each HA on HI in numbers from -1 to 1.

* Optimize U\_EK

U\_EK is a universal EK value obtained through clustering, and there can be various U\_EKs depending on the shape of the vector used for clustering. The figure below shows the results of clustering in two example ways:

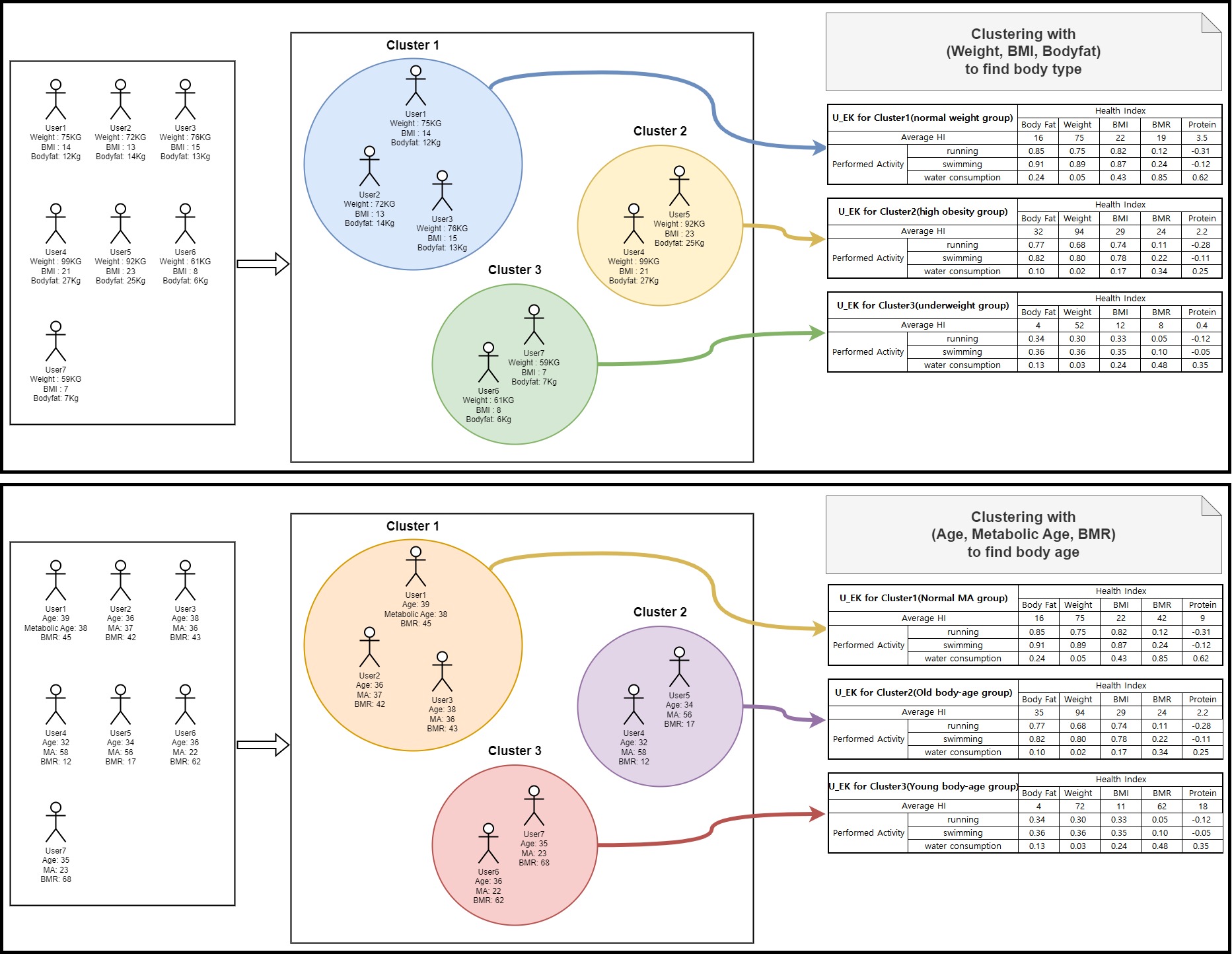
- Clustering with (Weight, BMI, Bodyfat) to find body type

From the results obtained by clustering in this way, we can know the body type group of the user. For example, you can recommend weight training to increase muscle mass for users in the skinny obese group, and aerobic exercise to reduce body fat for users in the highly obese group.

- Clustering with (Age, Metabolic Age, BMR) to find body age

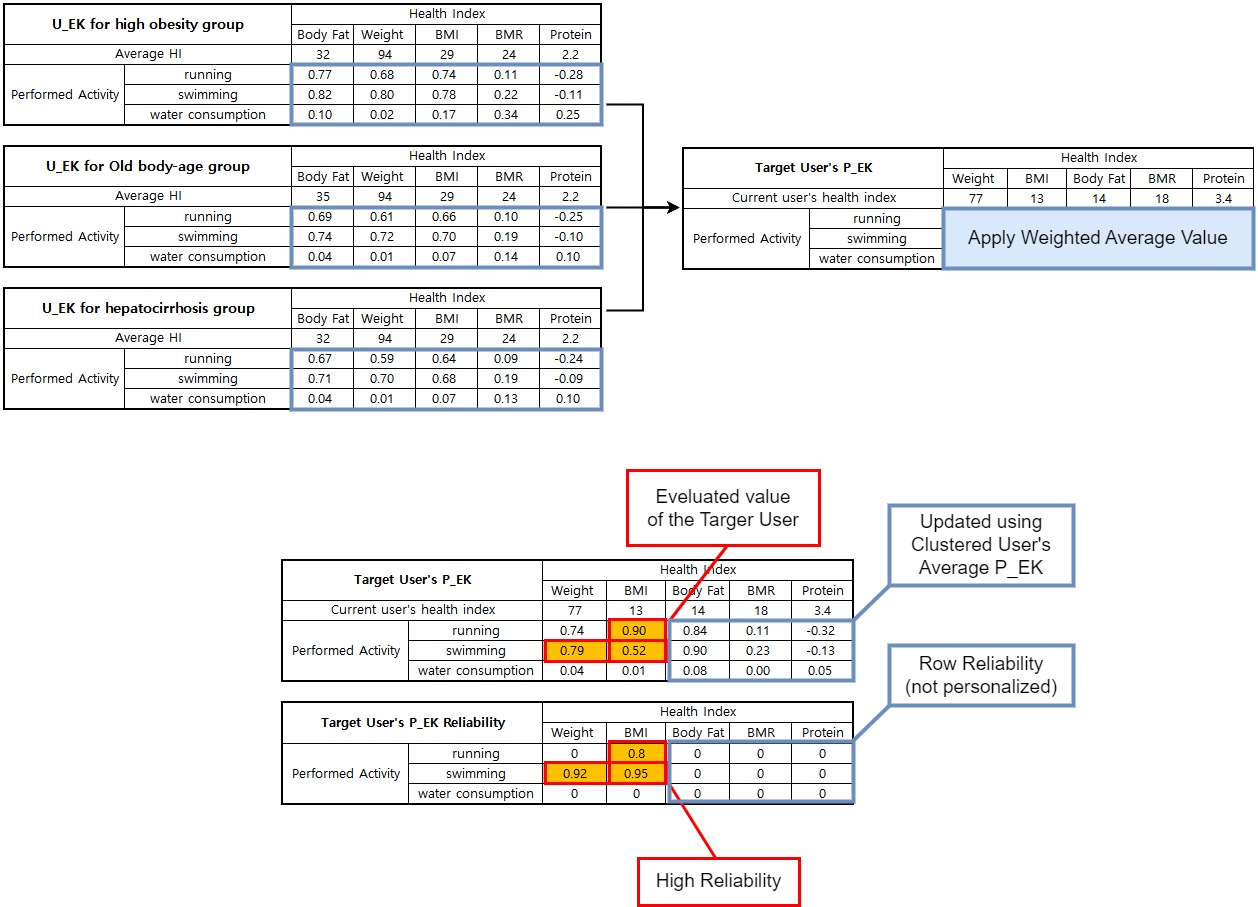
We can find the user's body(physical) age group by clustering in this way. For example, a group of users (healthy users) who are very low in body age and have a higher metabolic rate than their actual age can be advised to increase muscle mass by encouraging them to consume more protein.

The structure (combination) of the vector to be used for clustering in our system is determined by the Healthcare specialist. Clustering in a meaningful combination allows users to be better grouped and yields a U\_EK that is more suitable for users in the group. As a result, our system can give more meaningful advice for users.



* Optimize P\_EK

P\_EK is an EK that a specific user has personally. P\_EK can be obtained as the U\_EK of the cluster to which a particular user belongs. If the user's information is insufficient and only one cluster in the representative clustering is included, the representative cluster's U\_EK is copied. If a user is included in multiple clusters, the weighted average value with the user's similarity is applied from that cluster. As the individual's HI and HA are measured and the effectiveness is analyzed, the P\_EK is personalized for the user. P\_EK has reliability for each item, and if the value is close to 1, it means that it is completely personalized, and if it is close to 0, it means that there is no change in the value obtained from U\_EK. Reliability increases when the user-entered changes in HA and HI and effectiveness are reflected in the P\_EK. If the cluster to which the user belongs needs to be retrieved again due to a large HI change in the user, only the value with zero reliability is reflected.



* (T3) Evaluation of clustering algorithms (Relevant to C1)

In K-means clustering, it is difficult to determine how many clusters should be divided into (setting K values). The elbow method is used as a methodology for determining whether clustering has been done well. The vectors used in the elbow method are as follows.

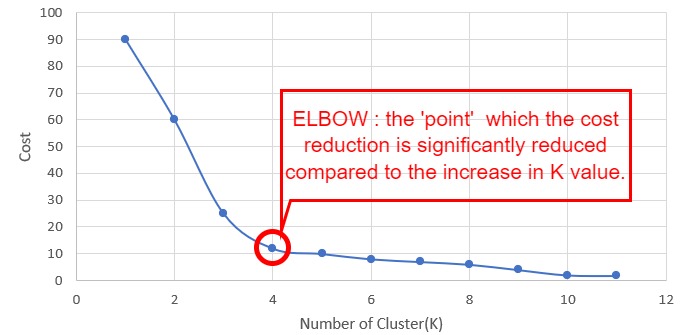
K: Number of clusters

N: Number of users

Cost: the sum of the average distance from the center of K clusters to all vectors in the cluster

M: Number of clustering performed

While increasing K from 1 to N, clustering is performed M times per each K, and the clustering result with the lowest cost is stored. As K increases from 1 to N, the cost decreases as shown in the following figure.



As the K value increases, the speed at which the cost decreases significantly is elbow, and K is the optimal K value at the point at which the cost decreases significantly. Since k-means clustering is very fast (O(n)) compared to other clustering methods, there is no major difficulty in finding the optimal K in the same way as above.

* (T4) Collect various datasets through beta testing (Relevant to C2)

Our system optimizes the U\_EK of general users through clustering. To obtain meaningful clustering results, a beta test of at least 3 months is required for at least 1000 users. With data obtained through beta testing, our system can obtain significant cluster-specific U\_EK. Even when starting the service for the first time, the data entered by the user at the time of registration can determine which cluster the user belongs to, and our system can provide meaningful advice.

* (T5) Receive essential information when initial user registration (Relevant to C2)

To set the user's initial P\_EK value, there must be data that is essentially input at the user's initial registration. The basic information that users should enter for determining which cluster they belong to is age, gender, height, and weight. Our system has a U\_EK clustered with the above four information input as initial values.

* (T6) Sampling datasets (Relevant to C3)

Applying the K-means clustering technique using the Elbow method takes O(NMK) (N = number of people, K = number of clusters, and M = number of clustering).In addition, Optimize U\_EK can take a lot of time because our system will perform various vector-specific clustering several times. As the number of users increases, the time it takes to perform clustering will increase linearly, so our system utilizes sampling when the number of users exceeds the threshold (1 million). When sampling, it will be randomly sampled to generate general U\_EK.

* (T7) Weighted value for recent user session (Relevant to C4)

The user may have a dynamic change in HI over time. In addition, if a user has used the app for a long time, the age range may also have changed. Users who were in their 20s with a skinny obesity body in the early stages of use may now be in their 30s with an athlete-type body. Therefore, the system weights and reflects the HA and effectiveness performed by the user when optimizing the user's P\_EK over time. In the case of a recently performed session, the largest weight is placed, and in the case of a session that has been performed for a long time, the weight is reduced so that the response session can be better reflected in the user's P\_EK.

* (T8) Generate Recommendation in two steps (Relevant to C5)
* Step 1. Recommend using the U\_EK

Use vectors such as the user's age group, gender, and HI to find which cluster corresponds to. Our system finds a recommendation that can improve the user's HI in a positive way by utilizing the U\_EK of the found cluster. In U\_EK, we recommend HA, which is the most effective for HI that users need to improve.

* Step 2. Refine the Recommendation using the P\_EK.

The user has P\_EK showing the effectiveness of HA for HI and the reliability for each item. Recommended HA generated by U\_EK is refined through effectiveness and reliability in P\_EK items.

* (T9) Using SW Agent (Relevant to C6)

The system evaluates the effectiveness and creates a recommendation whenever there is an input from the user's HI or HA. Since Recommendation is generated by U\_EK and P\_EK, optimizing P\_EK and U\_EK is automatically performed at the required time through SW agent. The server receives a user session event from the Client App, analyzes the effectiveness, and operates a thread that checks whether to optimize P\_EK and U\_EK. When the difference of User HI and HA performed in between them can be measured, SW agent automatically analyzes the Effectiveness and updates the P\_EK. In the second step of creating Recommendation, if there is a significant difference between the recommendation made with U\_EK and the refine with P\_EK, optimize U\_EK.

* (T10) Check validity of Health data Measurements (Relevant to C7)
* Check the health index from a device or user

In the case of health index measurement by device, the device may malfunction and a problem may occur in measurement due to external intervention during measurement. Therefore, even in the case of data measured by the device, the user should check it once again. Even in the case of data entered manually by the user, if it is determined to be abnormal, warn the user and check again to see if it is normal data.

* Outlier Detection using ESD(Extreme Studentized Deviation)

The system checks the validity of data entered by the user or once checked by the user using the Extreme Studentized Deviation (ESD) method. For example, for inputs beyond the common sense of being 10 meters tall or weighing 10 kilograms, the system detects themselves and judges them as an outlier and does not use them for analysis and recommendation. The outlier detection method utilizes the ESD method to determine data whose absolute deviation from observations and sample means exceeds the threshold.

### [Step 4] Evaluation of the Candidate Tactics

We evaluate the proposed candidate tactics in terms of their benefit and cost.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Tactics** | **Y/N** | **Justification** |
| T1 | Select the appropriate recommendation method: K-means | Y | Benefit) Suitable for U\_EK and P\_EK optimization.  Cost) It requires relatively small resources during operation.  Decision) It is an essential tactic to achieve the quality of the system |
| T2 | Applying clustering method on Optimize U\_EK and P\_EK | Y | Benefit) It is a suitable idea to achieve the quality requirements of the system.  Cost) Implementation is relatively easy and requires relatively small resources during operation.  Decision) It is an important tactic to achieve the quality of the system |
| T3 | Evaluation of clustering algorithms | Y | Benefit) It is a suitable idea to get the effectiveness of our recommendation system.  Cost) The cost is higher than performing with one specific K value.  Decision) The ability to find the optimal number of clusters (K) is essential. |
| T4 | Collect various datasets through beta testing | Y | Benefit) Various and important datasets can be obtained.  Cost) It costs a lot to recruit beta-testing users and run them before publishing.  Decision) It is necessary to derive U\_EK through clustering. |
| T5 | Receive essential information when initial user registration | Y | Benefit) It is possible to obtain a basic vector to be clustered in common with users.  Cost) It is not expensive to require information when a user first register this system.  Decision) It is essential to operate this recommendation system. |
| T6 | Sampling datasets | Y | Benefit) It is possible to reduce the time it takes to make clusters.  Cost) Sampling users costs very little. However, there is a possibility that you will not get a general U\_EK.  Decision) Available optionally if there are too many users. Sampling does not have to be used if the time to optimize U\_EK or computing power is sufficient, but if not, the quality does not decrease. |
| T7 | Weighted value for recent user session | Y | Benefit) It can improve user satisfaction.  Cost) Implementation can be a little complicated, but it's not a big cost.  Decision) It is essential to satisfy the quality of our products. |
| T8 | Generate Recommendation in two steps | Y | Benefit) It can get better recommendation results.  Cost) It takes longer than using one step.  Decision) It is essential to create an effective recommendation. |
| T9 | Using SW Agent | Y | Benefit) It increases usability because it does not require explicit invocation.  Cost) More implementations than explicit optimization and server thread usage can be bigger.  Decision) It is an essential tactic to achieve the quality of the system |
| T10 | Check validity of Health data Measurements | Y | Benefit) It is possible to prevent the outlier from dropping the effectiveness of clustering. As a result, better recommendations can be created and help achieve the quality of this system.  Cost) Additional logic needs to be implemented to judge the outlier.  Decision) It is necessary to improve the quality of the recommendation system. |

### [Step 5] Impact Analysis of Tactics on Views

We analyze the impacts of each selected tactic.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Tactics** | **Functional View** | **Information View** | **Behavior View** | **Deployment View** |
| T1 | Select the appropriate recommendation method: K-means |  |  | Add behavior view about K-means library |  |
| T2 | Applying clustering method on Optimize U\_EK and P\_EK |  | Add new classes about saving clustering result.  Increase cardinality of U\_EK | Add a control flow of using K-means clustering on optimizing U\_EK and P\_EK |  |
| T3 | Evaluation of clustering algorithms |  |  | Add a control flow of using elbow method |  |
| T4 | Collect various datasets through beta testing |  |  |  | Add datasets obtained from beta testing. |
| T5 | Receive essential information when initial user registration |  | Increase minimal cardinality of HI | Edit the behavior view about getting information from newly registered user |  |
| T6 | Sampling datasets |  |  | Add a control flow to sample users in optimizing U\_EK process |  |
| T7 | Weighted value for recent user session |  |  | Edit a control flow of optimizing P\_EK to use weighted value of recent user session |  |
| T8 | Generate Recommendation in two steps |  |  | Add a control flow of generating and refining recommendation |  |
| T9 | Using SW Agent | Remove use case of explicit optimizing U\_EK and P\_EK  Add SW agent about optimizing EKs |  | Add control flows about automatic optimizing U\_EK and P\_EK |  |
| T10 | Check validity of Health data Measurements | Add use case of checking validity of HA and HI |  | Add control flow about checking validity of HA and HI |  |

### [Step 6] Architecture with Tactics Applied

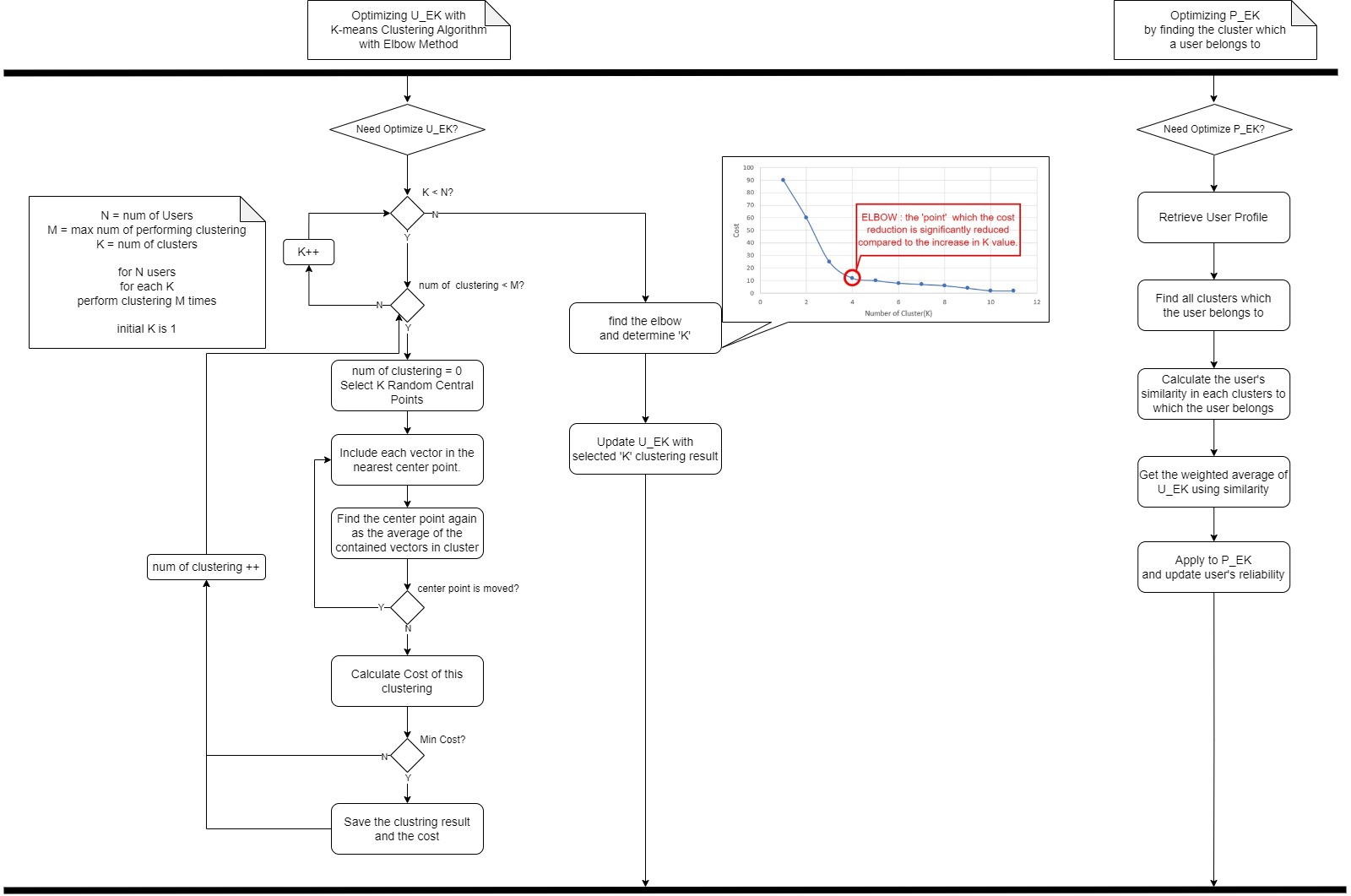
We design the detailed control flow for selected tactics. In CEP, we choose only one tactic. We choose the tactic, “(T2) Applying K-means clustering method on Optimize U\_EK and P\_EK”.

* Applying T2 on Information View

Since our system have U\_EK per cluster, increasing cardinality is needed.

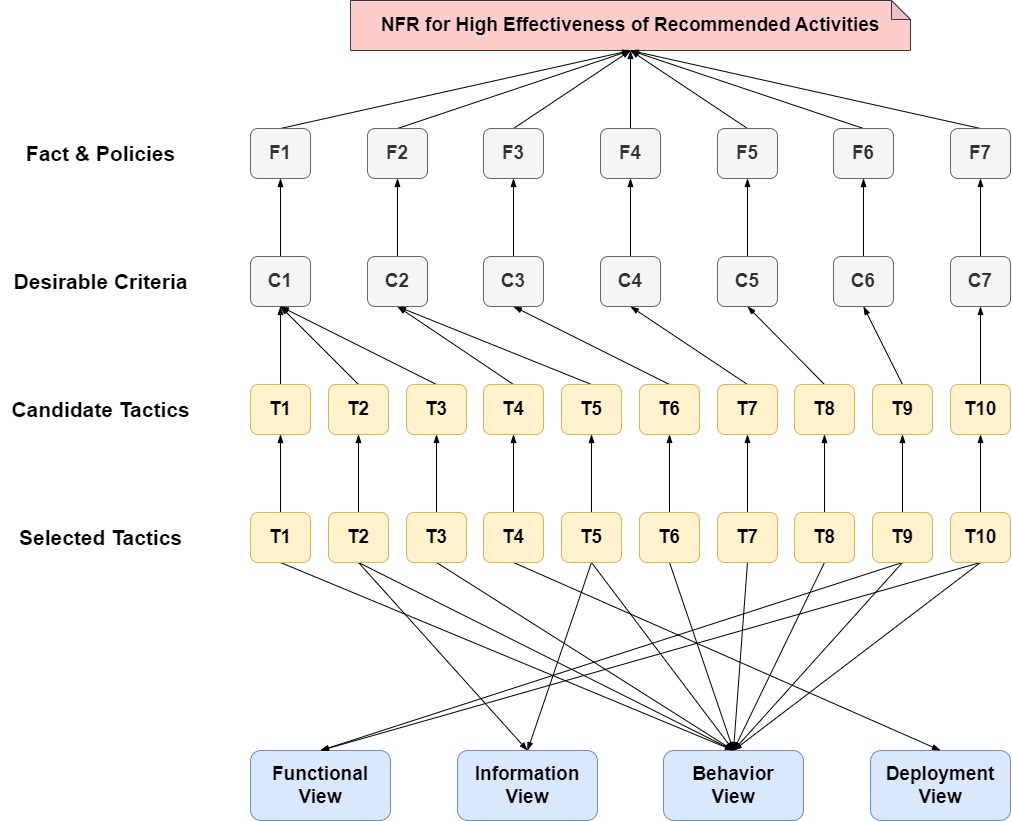
* Applying T2 on Behavior View

Details of updating U\_EK, P\_EK shown below.



### [Step 7] Verifying the Traceability

It is important to enforce the traceability among the facts/policies, criteria, tactics and impacts on views. The following figure shows the trace links among facts, criteria, tactics and their impacts on views.



As shown in the figure, all the elements defined with ‘conforms-to’ relationships, which yields a high traceability and consistency.

## NFR-2. High Modifiability of the System

The target system has variabilities on several places including the followings.

* Heterogeneity on Hardware Devices
* Adding new Weight Scale Devices measuring health indexes
* Adopting the API of newly added devices
* List of Health Indexes managed by the system.
* Adding new health indexes
* Deleting some of the current health indexes
* List of Health-relevant activities
* Adding new health-relevant activities
* Deleting some of the current health-relevant activities
* Methods of analyzing the effectiveness of the activities on controlling health indexes
* Modifying the Relevance Relationships between Activities and Health Indexes
* Modifying the Default Values of the Effectiveness
* Representations of U\_EK and P\_EK
* Representation using a Table.
* Representation using a Graph.
* Representation using a Machine Learning Model.
* Others

There can be other types of modifications that can be made to the system.

When a system reveals variability on various variation points, the system should be designed with ‘Openness’ design techniques. If a target system is development by considering the variability and applying the Openness design of Open-Closed Principle (OCP), the system can efficiently be modified.

### [Step 1] Underlying Facts and Policies

* (F1) Adding new Weight Scale Devices measuring health index.

Various new devices will be released. And if users can use the new device without difficulty in the system, users will be more satisfied.

* (F2) Adopting the API of newly added device.

APIs used by various newly added devices may also be developed and changed over time.

* (F3) Adding and Deleting health index.

New health indicators could emerge. As trends change, users will want to manage the new health index when it comes to attention. In addition, some health indicators may be judged to be meaningless over time, and users may lose interest in specific management indicators.

* (F4) Adding and Deleting health-relevant activity.

In the case of health functional food intake or physical activity, it is very sensitive to trends. As trends change, new health-related activities performed by users may appear. Some of the current health-related activities can be deleted when no one does.

* (F5) Modifying the Relevance Relationships between Activities and Health Index.

The effectiveness of HA on HI can be changed as a result of system analysis. Default values set by healthcare professionals are subject to change as a result of system analysis.

* (F6) modifying Methods of analyzing the effectiveness.

If better analytical methods for HA controlling HI are developed, the system may adopt new methods to improve performance.

* (F7) Modifying Representations of U\_EK and P\_EK

The effectiveness analysis method may be changed, and the generation method of recommendation may be changed. Accordingly, if a better representation method of U\_EK and P\_EK is developed, it may be necessary to change the representation of U\_EK and P\_EK.

### [Step 2] Criteria for Satisfying NFR

* (C1) Adding new Device (Relevant to F1)

The system manager should be able to add a new device with the same API as before.

* (C2) Adopting new API (Relevant to F2)

For devices that use the new API, the system administrator cannot add them on runtime, but they must be able to modify the program at a low cost.

* (C3) Management Health Index (Relevant to F3)

For new health index, system administrators can easily add and delete them and functions about them should be supported by the system.

* (C4) Management Health-Relevant Activity (Relevant to F4)

For new health relevant activity, system administrators can easily add and delete them and functions about them should be supported by the system.

* (C5) Modifying relevance between the activities and the health indexes (Relevant to F5)

As the results of the system analysis change, the relationship between activity and health index should be able to be modified.

* (C6) modifying methods of analyzing the effectiveness (Relevant to F6)

The system should be able to adopt new methods of analyzing the effectiveness.

* (C7) modifying representation of U\_EK and P\_EK (Relevant to F7)

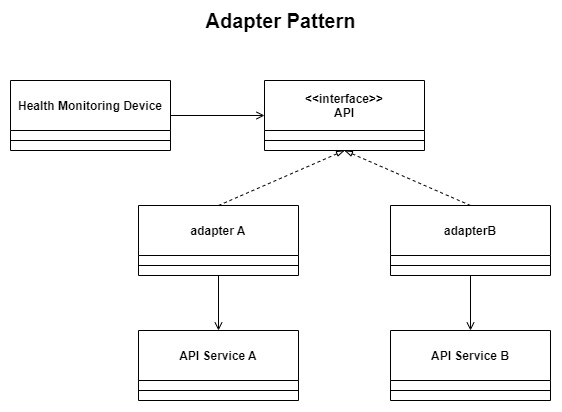
When changing the representation of U\_EK and P\_EK, it should be possible to modify the program at a low cost.

### [Step 3] Candidate Tactics for the Criteria

* (T1) Using Adapter Pattern (Relevant to C1, C2)

For the system administrators to manage devices easily, they should depend on various devices as one abstracted interface. The strategy pattern can be applied, but it is generally not applicable because the code in the API cannot be modified. As a result, adapter patterns are applied to support a variety of APIs.

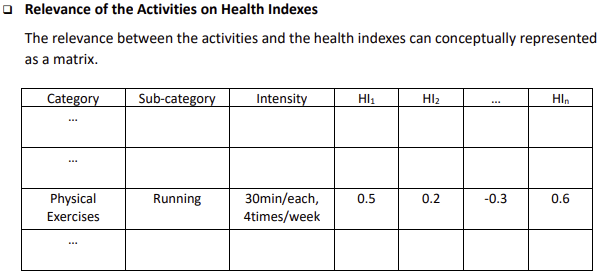
As shown in the figure below, devices can be managed without distinction even if the methods are different for each API, and even if a device using a new API appears, only Adapter C can be newly implemented and supported. Since this is a slow binding, the API of the device may be changed at runtime.



* (T2) Using Inheritance (Relevant to C3, C4)

Ensure that all HI and HA inherit the abstract class and that other components have dependency on the abstract class. This ensures that other components are not affected by the addition and deletion of HI and HA.

* (T3) Using Matrix (Relevant to C3, C4, C5)



In this system, the relationship between HA and HI can be expressed as a matrix. To add and delete HA and HI, the system must be able to modify rows and columns into a matrix at runtime. The system can make it with a map that has the combination of activity and health index as a key and the relevance as a value.

* (T4) Using Blackboard Architecture Style (Relevant to C6)

We will use the blackboard architecture style. Each knowledge source includes a method of analyzing the effect. Because it is easy to add and change knowledge sources, the system can easily change how you analyze effects.

* (T5) Using Encapsulation for Information hiding (Relevant to C7)

In early versions of the system, we utilize the matrix to express U\_EK and P\_EK. However, U\_EK and P\_EK are well encapsulated, so it is difficult to know which expressions are used by other components. Therefore, changes in the presentation of U\_EK and P\_EK do not affect the outside system.

### [Step 4] Evaluation of the Candidate Tactics

We evaluate the proposed candidate tactics in terms of their benefit and cost.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Tactics** | **Y/N** | **Justification** |
| T1 | Using Adapter Pattern | Y | Benefit) Various APIs are available.  Cost) The most of Adapter are not reusable.  Decision) The tactic can be implemented with a minimal cost while the benefit is high. |
| T2 | Using Inheritance | Y | Benefit) Using Inheritance will provide Extensibility.  Cost) Complexity can increase.  Decision) The tactic can be implemented with a minimal cost while the benefit is high. |
| T3 | Using matrix | Y | Benefit) Matrix is intuitive and easy to understand.  Cost) There may be information that cannot be expressed.  Decision) The tactic can be implemented with a minimal cost while the benefit is high. |
| T4 | Using Blackboard Architecture Style | Y | Benefit) easily add and change knowledge sources.  Cost) Changing the structure of the data is difficult because it affects all knowledge sources.  Decision) The tactic can be implemented with a minimal cost while the benefit is high. |
| T5 | Using Encapsulation for Information hiding | Y | Benefit) Data changes do not affect other objects.  Cost) The interface should be provided.  Decision) The tactic can be implemented with a minimal cost while the benefit is high. |

### [Step 5] Impact Analysis of Tactics on Views

We analyze the impacts of each selected tactic.

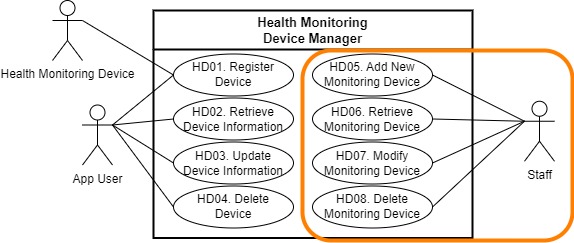
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Tactics** | **Functional View** | **Information View** | **Behavior View** | **Deployment View** | **Development View** |
| T1 | Using Adapter Pattern | Add New Use Cases (management devices by Staff) | Add Abstract class and Adapters. | Add behavior view about New Use cases |  |  |
| T2 | Using Inheritance | Add New Use Cases (management HA, HI by Staff) | Add Abstract class | Add behavior view about New Use cases |  |  |
| T3 | Using matrix |  |  | Use java collection framework |  |  |
| T4 | Using Blackboard Architecture Style |  | Add class about blackboard | Choose Knowledge Source by Controller |  |  |
| T5 | Using Encapsulation for Information hiding |  | Add Public methods on U\_EK, P\_EK. | Use Public Methods of U\_EK, PEK. |  |  |

### [Step 6] Architecture with Tactics Applied

We design the detailed control flow for selected tactics. In CEP, we choose only one tactic. We choose the tactic, “T1. Using Adapter Pattern”.

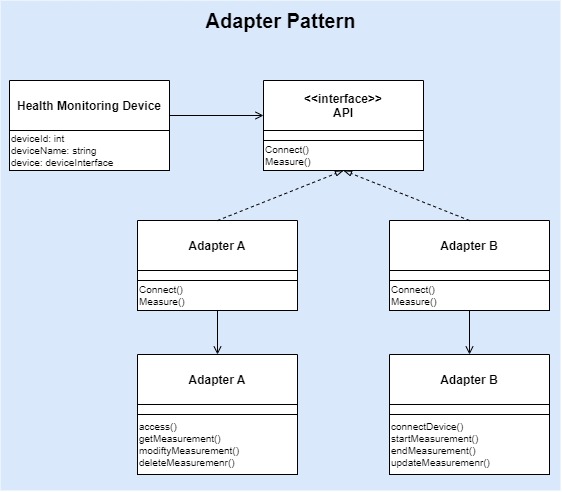
* Applying T1 on Functional View

Four use cases is added for management devices by staff



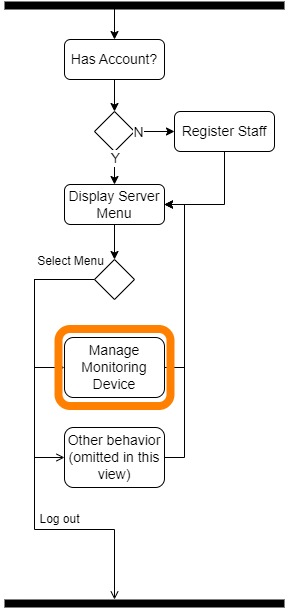
* Applying T1 on Information View

Applying adapter pattern on Health Monitoring Device Class

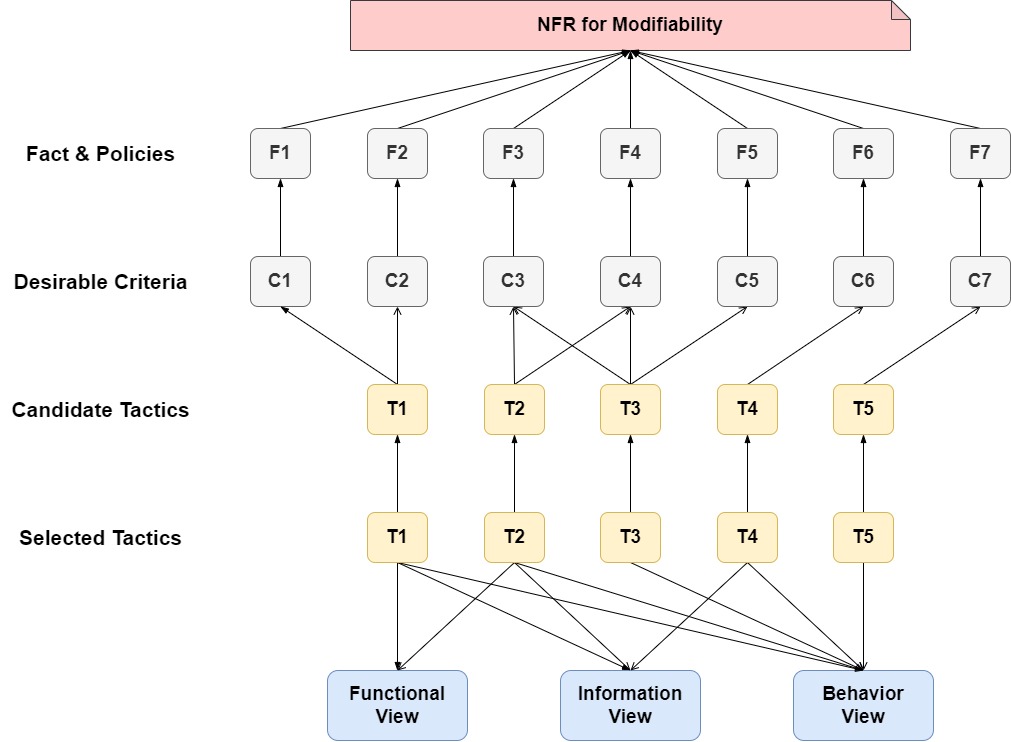


* Applying T1 on Behavior View

Adding “Manage Health Monitoring Device” on the staff explicit menu.



### [Step 7] Verifying the Traceability



# Activity 6. Architecture Validation

* Steps of ATAM



## [Step 1] Presenting ATAM

* Omitted

## [Step 2] Presenting Business Drivers

* Omitted

## [Step 3] Presenting the Architecture

* Omitted

## [Step 4] Identifying the Architectural Approaches

The architectural approaches applied to *Health Assistant System* are described in chapters 5 and 6. The architectural approaches are reflected in the following view:

* Functional View
* Information View
* Behavior View
* Deployment View

## [Step 5] Generating Quality Attribute Tree

The following table describes the quality attribute utility tree used for evaluating the architecture of *Health Assistant System*. We define 6 scenarios for evaluating functionality (i.e. High Effectiveness of Recommended Activities) and for 6 scenarios for evaluating reliability (i.e. High Modifiability of the System) of the *Health Assistant System*. Each scenario is described with a different degree of importance and difficulty.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Quality** | **Refinements** | **Scenarios** | **Importance** | **Difficulty** |
| Functionality | NFR-1. High Effectiveness of Recommended Activities | NFR-1-1. Evaluate how effective this system makes recommendations to users. | High | High |
| NFR-1-2. Evaluate how well U\_EK has been optimized with insufficient number of user Datasets | High | High |
| NFR-1-3. Evaluate how well U\_EK is optimized with sufficient number of user Datasets | High | Medium |
| NFR-1-4. Evaluate how well the effectiveness of Health-Relevant Activity on an individual's Health Index is reflected in the P\_EK. | High | High |
| NFR-1-5. Evaluate how recent user sessions reflect better in P\_EK | High | High |
| NFR-1-6. Evaluate how well this system detect outlier | High | High |
| Reliability | NFR-2. High Reliability of Server | NFR-2-1. Evaluate how small it costs to add devices | High | High |
| NFR-2-2. Evaluate how easily System Manager can add/delete Health Index. | High | High |
| NFR-2-3. Evaluate how easily System Manager can add/delete Health-Relevant Activity | High | High |
| NFR-2-4. Evaluate how much Cost is incurred in developing adapters to add new APIs | High | Medium |
| NFR-2-5. Evaluate how much Cost occurs to change the methods of analyzing | High | High |
| NFR-2-6. Evaluate how much Cost is generated to change the modifying presentation of U\_EK and P\_EK. | High | Medium |

## [Step 6] Analyzing Architectural Approaches

The following table describes an analysis result of architectural approaches addressing a scenario, NFR-1-1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Analysis of Architectural Approach** | | | | |
| **Scenario #** | NFR-1-1. Evaluate how effective this system makes recommendations to users. | | | |
| **Attribute** | Functionality | | | |
| **Environment** | Normal Operation. | | | |
| **Stimulus** | A system is run under normal condition. | | | |
| **Architectural Decision** | **Sensitivity** | **Trade-Off** | **Risk** | **Nonrisk** |
| D1-1. Applying K-means clustering method | S1-1-1 | T1-1-1 | R1-1-1 |  |
| D1-2. Clustering Users in various vector | S1-1-2 | T1-1-2 | R1-1-2 |  |
| D1-3. Collect various datasets from users | S1-1-3 | T1-1-3 | R1-1-3 |  |
| D1-4. Evaluation of clustering algorithms |  | T1-1-4 |  |  |
| D1-5. Outlier Detection using ESD(Extreme Studentized Deviation) |  |  |  | N1-1-1 |
| **Reasoning** | The decisions are made for meeting this scenario since the chosen decisions are commonly used for improving accuracy of the results of running machine learning algorithms. | | | |
| **Architectural Diagram** | Refer to refined component diagram in 6.1. | | | |

* Sensitivity Points
* S1-1-1. As the number of users increases, it takes a lot of time to optimize the model.
* S1-1-1. As the number of vector combinations increases, it takes a lot of time to optimize the model.
* S1-1-3. There is a lot of pressure on the user, and the increasing number of HA and HI should be managed.
* Trade-off
* T1-1-1. Accuracy (+) vs. Performance (-): The more complex the methodology for generating effective recommendations is, the more complex computation is involved. Even though current computer hardware architecture is designed in a way that high complicated computation is executed in an efficient manner, there still would be a negative impact on the performance if the metrics are not well-defined.
* T1-1-2. Accuracy (+) vs. Resource (-): ML training requires a lot of time and resources, so increasing accuracy requires a lot of resource consumption.
* T1-1-3. Accuracy (+) vs. Useability (-): Obtaining a large variety of datasets increases the accuracy of the model, but there is a burden on the user to provide more.
* T1-1-4. Accuracy (+) vs. Performance (-): The more complex the logic of determining whether clustering has been successful, the more complex computation is involved.
* Risk
* R1-1-1. If the combination of vectors selected by the Healthcare Specialist is not appropriate, effective clustering may not be achieved.
* R1-1-2. Acquiring data from a device or user poses a risk because it depends on how willing the user is to provide such data to the system. Without detailed instructions and useful tools, it is very difficult to obtain various data to perform well on the target system.
* Nonrisk
* N1-1-1. Outlier detection is essential in clustering algorithm.

## [Step 7] Brainstorming and Prioritizing Scenarios

* List of scenarios collected by all stakeholders (i.e. users, staffs, clients, and data scientists)
* About 3 scenarios are additionally acquired for evaluating functionality (i.e. effectiveness of recommended activities) of *Health Assistant* S*ystem*. Most scenarios are gathered from users, managers, and data scientists.
* Need a method that can effectively secure the user’s health dataset.
* Additional resources that can be used for recommendation system as much as possible.
* Find a way to reduce the resources required for recommendation system.
* About 2 scenarios are additionally acquired for evaluating modifiability for Health Assistant System. Most scenarios are gathered from clients and managers.
* Get the type of HA and HI that the user wants to add at this system.
* Supported not only by mobile apps but also by web apps.

## [Step 8] Analyzing Architectural Approaches

Since the result of this step is same as the one of step 6, we do not include the table.

## [Step 9] Presenting the Results

* All evaluation team concludes the following results;
* Concerns on the functionality, specifically on the accuracy of recommendation, are overall well considered except for finding the combination of vectors utilized in clustering. Hence, it is necessary to reconsider finding an effective combination of vectors.
* Classifying users into meaningful groups is an important but difficult topic. The Health Assistant system can initially start working with small Health Index and Health Related Activity datasets because users may not provide enough data for the system. The Healthcare specialist must find a significant combination of data sets in the information of user.
* Currently, the method of extracting meaningful datasets relies on healthcare specialists. Additional design may be required for how the system selects the valid vector.
* Concerns on the modifiability need to be re-considered.
* Current scenarios for guaranteeing high modifiability are most related to reduce the cost of responding to changes in the situation. More scenarios for modifiability need to be identified and re-evaluated regarding whether the current design fully support these new scenarios.

# Concluding Remarks

The architecture description in this document is to meet both the functional and non-functional requirements for the system. It is the result of applying the proposed core process of designing software architecture.

It is believed that this architecture description is practically implementable with current technologies and such implementation would yield a high level of quality-in-use.

**⮚ END OF ARCHITECTURE DESCRIPTION ⮘**