**Architecture Description**

**GPT-Assisted Coding Trainer**

**Software System for Training Coding with GPT Assistance**

A picture containing text, screenshot, font, software

Description automatically generated

**May 2023**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Revisions Made** |
| 0.8 | 6/07(Wed), 9pm | 박종안 | (Interim) Refining SRS & Context View |
| 0.9 | 6/20(Tue), 9pm | 박종안 | (Pre-final) Schematic Architecture and Viewpoints |
| 1.0 | 7/03(Mon), 9pm | 박종안 | (Final) Design for NFRs and Validation |

**삼성전자 첨단기술아카데미**

**Table of Contents**

[1. Introduction 7](#_Toc138110192)

[1.1. Purpose of the Document 7](#_Toc138110193)

[1.2. System of Interest 7](#_Toc138110194)

[1.3. Definitions, Acronyms, and Abbreviations 7](#_Toc138110195)

[1.4. References 9](#_Toc138110196)

[1.5. Process applied to Architecture Design 9](#_Toc138110197)

[1.6. Template used for Architecture Description 11](#_Toc138110198)

[2. Activity 1. Architectural Requirement Refinement 11](#_Toc138110199)

[2.1. [Step 1] Identify Stakeholders 11](#_Toc138110200)

[2.2. [Step 2] Refining Functional Requirements 12](#_Toc138110201)

[2.3. [Step 3] Architectural Concerns 12](#_Toc138110202)

[2.4. [Step 4] Refine Non-Functional Requirements 14](#_Toc138110203)

[2.5. [Step 5] Write Refined Software Requirement Specification 15](#_Toc138110204)

[3. Activity 2. System Context Analysis 15](#_Toc138110205)

[3.1. [Step 1] System Boundary Context 15](#_Toc138110206)

[3.1.1. Level 0 DFD for the Boundary Context 15](#_Toc138110207)

[3.1.2. Description of the Context Diagram 16](#_Toc138110208)

[3.2. [Step 2] Functional Context 17](#_Toc138110209)

[3.2.1. Representing the Functional Context 17](#_Toc138110210)

[3.2.2. Defining Functional Groups 18](#_Toc138110211)

[3.2.3. Defining Actors 18](#_Toc138110212)

[3.2.4. Defining Use Cases 19](#_Toc138110213)

[3.2.5. Context-level Use Case Diagram 19](#_Toc138110214)

[3.3. [Step 3] Information Context 19](#_Toc138110215)

[3.3.1. Representing the Information Context 19](#_Toc138110216)

[3.3.2. Identifying Persistent Object Classes 20](#_Toc138110217)

[3.3.3. Context-level Class Diagram 20](#_Toc138110218)

[3.4. [Step 4] Behavioral Context 21](#_Toc138110219)

[3.4.1. Representing the Behavior Context 21](#_Toc138110220)

[3.4.2. Allocate Functionality over Tiers 21](#_Toc138110221)

[3.4.3. Define Invocation Patterns 21](#_Toc138110222)

[3.4.4. Context-level Activity Diagram 23](#_Toc138110223)

[3.5. [Step 5] Additional Contexts 24](#_Toc138110224)

[4. Activity 3. Schematic Architecture Design 25](#_Toc138110225)

[4.1. [Step 1] Observe Architectural Characteristics 25](#_Toc138110226)

[4.2. [Step 2] Candidate Architecture Styles 25](#_Toc138110227)

[4.3. [Step 3a] Evaluating ‘Candidate 1. MVC Architecture Style’ 25](#_Toc138110228)

[4.3.1. Evaluate the Applicable Situations 25](#_Toc138110229)

[4.3.2. Evaluate the Benefits 26](#_Toc138110230)

[4.3.3. Evaluate the Drawbacks 26](#_Toc138110231)

[4.3.4. Result of the Evaluation 26](#_Toc138110232)

[4.4. [Step 2b] Evaluating ‘Candidate 2. Event-driven Architecture Style’ 26](#_Toc138110233)

[4.4.1. Evaluate the Applicable Situations 26](#_Toc138110234)

[4.4.2. Evaluate the Benefits 27](#_Toc138110235)

[4.4.3. Evaluate the Drawbacks 27](#_Toc138110236)

[4.4.4. Result of the Evaluation 27](#_Toc138110237)

[4.5. [Step 2c] Evaluating ‘Candidate 3. Dispatcher Architecture Style’ 27](#_Toc138110238)

[4.5.1. Evaluate the Applicable Situations 28](#_Toc138110239)

[4.5.2. Evaluate the Benefits 28](#_Toc138110240)

[4.5.3. Evaluate the Drawbacks 28](#_Toc138110241)

[4.5.4. Result of the Evaluation 28](#_Toc138110242)

[4.6. [Step 2e] List of Selected Architecture Styles 28](#_Toc138110243)

[4.7. [Step 3] Applying Architecture Styles 29](#_Toc138110244)

[4.7.1. Applying MVC Architecture Style 29](#_Toc138110245)

[4.7.2. Applying Event-driven Architecture Style 29](#_Toc138110246)

[4.7.3. Applying Dispatcher Architecture Style 30](#_Toc138110247)

[4.7.4. Resulting Schematic Architecture 31](#_Toc138110248)

[4.8. [Step 4] Refining Interaction Paths 31](#_Toc138110249)

[4.8.1. Interaction Paths derived from the Styles 32](#_Toc138110250)

[4.8.2. Refinements on the Default Interaction Paths 32](#_Toc138110251)

[4.9. [Step 4] Elaborating the Schematic Architecture 32](#_Toc138110252)

[4.9.1. Strengths 32](#_Toc138110253)

[4.9.2. Drawbacks 32](#_Toc138110254)

[5. Activity 4. View-specific Architecture Design 33](#_Toc138110255)

[5.1. Functional View 33](#_Toc138110256)

[5.1.1. [Step 1] Observe Functional Characteristics 33](#_Toc138110257)

[5.1.2. [Step 2] Refine Use Case Diagram 33](#_Toc138110258)

[5.1.3. [Step 3] Derive Functional Components 35](#_Toc138110259)

[5.1.4. [Step 4] Refine Functional Components for Tiers 36](#_Toc138110260)

[5.1.5. [Step 5] Allocate Functional Components 37](#_Toc138110261)

[5.1.6. [Step 6] Design Functional Components 39](#_Toc138110262)

[5.1.7. [Step 7] Define Interfaces of Functional Components 40](#_Toc138110263)

[5.2. Information View 41](#_Toc138110264)

[5.2.1. [Step 1] Observe Informational Characteristics 41](#_Toc138110265)

[5.2.2. [Step 2] Refine Persistent Object Model 41](#_Toc138110266)

[5.2.3. [Step 3] Derive Data Components 43](#_Toc138110267)

[5.2.4. [Step 4] Refine Data Components for Tiers 44](#_Toc138110268)

[5.2.5. [Step 5] Allocate Data Components 44](#_Toc138110269)

[5.2.6. [Step 6] Design Data Components 46](#_Toc138110270)

[5.2.7. [Step 7] Define Interfaces of Data Components 46](#_Toc138110271)

[5.3. Behavioral View 46](#_Toc138110272)

[5.3.1. [Step 1] Observe Behavioral Characteristics 46](#_Toc138110273)

[5.3.2. [Step 2] Refining Control Flow of the Platform 47](#_Toc138110274)

[5.3.3. [Step 3] Choosing Element for Detailed Control Flow 50](#_Toc138110275)

[5.3.4. [Step 4] Detailed Control Flow for ‘Create GPT Prompt to evaluate.’ 50](#_Toc138110276)

[5.4. Deployment View 53](#_Toc138110277)

[5.4.1. [Step 1] Observe Deployment Characteristics 53](#_Toc138110278)

[5.4.2. [Step 2] Define Nodes 53](#_Toc138110279)

[5.4.3. [Step 3] Define Network Connectivity 54](#_Toc138110280)

[5.4.4. [Step 4] Define Artifacts to Deploy 54](#_Toc138110281)

[5.4.5. [Step 5] Allocate Artifacts on Nodes 54](#_Toc138110282)

[6. Activity 5. NFR-specific Architecture Design 56](#_Toc138110283)

[6.1. Design for High Reliability of the System 56](#_Toc138110284)

[6.1.1. [Step 1] Identify Facts and Policies 56](#_Toc138110285)

[6.1.2. [Step 2] Define Criteria for Tactics 57](#_Toc138110286)

[6.1.3. [Step 3] Define Candidate Tactics 57](#_Toc138110287)

[6.1.4. [Step 4] Evaluate Candidate Tactics 64](#_Toc138110288)

[6.1.5. [Step 5] Analyze Impacts of Tactics 64](#_Toc138110289)

[6.1.6. [Step 6] Apply Tactics 65](#_Toc138110290)

[6.1.7. [Step 7] Validate Conformance 67](#_Toc138110291)

[6.2. Design for NFR-2. High Personalization of Instruction and Coding Exercises 67](#_Toc138110292)

[6.2.1. [Step 1] Identify Facts and Policies 68](#_Toc138110293)

[7. Activity 6. Architecture Validation 68](#_Toc138110294)

[7.1. [Step 1] Presenting ATAM 68](#_Toc138110295)

[7.2. [Step 2] Presenting Business Drivers 68](#_Toc138110296)

[7.3. [Step 3] Presenting the Architecture 69](#_Toc138110297)

[7.4. [Step 4] Identifying the Architectural Approaches 69](#_Toc138110298)

[7.5. [Step 5] Generating Quality Attribute Tree 69](#_Toc138110299)

[7.6. [Step 6] Analyzing Architectural Approaches 71](#_Toc138110300)

[7.7. [Step 7] Brainstorming and Prioritizing Scenarios 72](#_Toc138110301)

[7.8. [Step 8] Analyzing Architectural Approaches 73](#_Toc138110302)

[7.9. [Step 9] Presenting the Results 73](#_Toc138110303)

[8. Concluding Remarks 74](#_Toc138110304)

**List of Figures**

[Figure 1. Concept of the GPT-Assisted Coding Trainer System 7](#_Toc138110305)

[Figure 2. Process to design Software Architecture 10](#_Toc138110306)

[Figure 3. Degree of Relevance of NFRs to Stakeholders 14](#_Toc138110307)

[Figure 4. DFD Diagram for Boundary Context 16](#_Toc138110308)

[Figure 5. Use Case Diagram for the Functional Context 19](#_Toc138110309)

[Figure 6. Class Diagram for showing Information Context 20](#_Toc138110310)

[Figure 7. Activity Diagram for showing Behavior Context of System 23](#_Toc138110311)

[Figure 8. Activity Diagram for showing Behavior Context of SW Agent 24](#_Toc138110312)

[Figure 9. Applying MVC Architecture Style 29](#_Toc138110313)

[Figure 10. Applying Event-driven Architecture Style 30](#_Toc138110314)

[Figure 11. Applying Dispatcher Architecture Style 30](#_Toc138110315)

[Figure 12. Resulting Schematic Architecture 31](#_Toc138110316)

[Figure 13. Refined Use Case Diagram 34](#_Toc138110317)

[Figure 14. Deriving Functional Components 35](#_Toc138110318)

[Figure 15. Functionality Place Holders of the Target System 37](#_Toc138110319)

[Figure 16. Allocaiton of Components on Functionality Place Holders 39](#_Toc138110320)

[Figure 17. Refined Class Diagram 42](#_Toc138110321)

[Figure 18. List of Data Components 43](#_Toc138110322)

[Figure 19. Data Component Holders of the Target System 45](#_Toc138110323)

[Figure 20. Allocation of Data Components 45](#_Toc138110324)

[Figure 21. Refined Control Flow of Explicit Functional Group 48](#_Toc138110325)

[Figure 22. Refined Control Flow of SW Agent(Event-based, Timer) 49](#_Toc138110326)

[Figure 23. Target Element for Designing Detailed Control Flow 50](#_Toc138110327)

[Figure 24. Detailed Control Flow of Create Evaluation GPT Prompt 51](#_Toc138110328)

[Figure 25. Deployment Diagram for the Target System 55](#_Toc138110329)

**List of Tables**

Table 1. Profiles of Stakeholders 12

Table 2. Relevance of NFRs to Stakeholders 13

Table 3. Allocation of Functionality over Tiers 21

Table 4. Invocation Patterns defined for Functional Groups 22

Table 5. Functional Components and their Use Cases 36

Table 6. Functional Components of Dispatcher Architecture Style 36

Table 7. Partitioning Functional Components 38

Table 8. Design of Functional Components 40

Table 9. Data Components allocated on Tiers 44

Table 10. Invocation Patterns defined for Functional Groups 46

**Architecture Description of  
GPT-Assisted Coding Trainer**

# 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

The system of interest is Coding Trainer which is a software system, designed to facilitate the teaching of programming languages without the need for human instructors. GPT-Assisted Coding Trainer is an advanced coding trainer that harnesses the power of GPT-4 to deliver its core functionality. The system leverages the capabilities of the GPT-4 model through its API to perform essential tasks and provide a comprehensive learning experience.

The GPT-Assisted Coding Trainer takes advantage of the latest advancements in GPT language modeling and artificial intelligence to create an immersive and effective learning environment for programming languages.

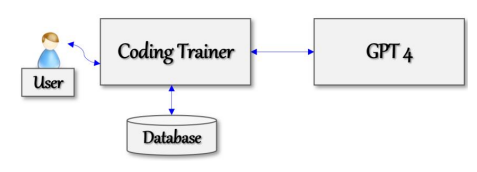


Figure . Concept of the GPT-Assisted Coding Trainer System

The system is deployed as a web server, which subscribes to GPT 4, as shown in the Figure 1. A user interacts only with the Coding Trainer, not directly with GPT 4. The system maintains a database for managing the essential datasets such as user profiles, programming language profile, sessions for using the system including teaching contents, exercise problems, users’ program codes, and their evaluations.

## Definitions, Acronyms, and Abbreviations

* GPT (Generative Pre-trained Transformer)

GPT refers to a type of artificial intelligence model developed by OpenAI. GPT models are based on the Transformer architecture, which is a deep learning model architecture known for its ability to handle sequential data efficiently.

GPT models are renowned for their ability to generate coherent and contextually relevant text. They can be used for various natural language processing tasks, including language translation, text summarization, chatbot interactions, and creative writing assistance.

* Language Profile

Language Profile serves as a meta-description of each programming language and is defined by the course director.

Language Profile includes the following attributes:

* Profile Name
* List of Units in the programming language
* List of Topics for each Unit

The content of a unit is further organized into topics, with each topic representing a specific language construct related to the unit's main theme.

* Unit

The content of a programming language is organized into units, with each unit representing a key topic or concept of the language.

* Topic

The content of a unit is further organized into topics, with each topic representing a specific language construct related to the unit's main theme.

* Quantitative Evaluation

GPT-Assisted Coding Trainer offers a quantitative evaluation of user submissions, utilizing a rating scale ranging from 1 to 10. This evaluation provides a measurable assessment of the quality or performance of the submitted code.

* Qualitative Evaluation

The system goes beyond quantitative evaluation and provides comprehensive feedback on user submissions. It offers a detailed analysis that highlights the strengths and weaknesses of the code. The feedback includes explanations for necessary corrections, suggestions for improvement, and alternative approaches to writing the code. This thorough feedback helps users understand their mistakes, learn from them, and enhance their coding skills effectively.

* Training Report

Training Report represents comprehensive reports on the learning progress of learners and certificates of completion.

* Progress Report

This progress report provides comprehensive details of the training sessions conducted. It encompasses the complete history of training sessions conducted for the specific programming language, highlighting the exercises undertaken and the evaluation results of code submissions.

* Certificate of Completion

This certificate confirms that the recipient has successfully completed all the required training units and fulfilled the necessary criteria for certification. It is a concise, one-page document that bears the official seal of the training institute.

## References

[Kim 23a] Soo Dong Kim, Associate Architect Program, 2023-A3, CEP Specification of GPT-Assisted Coding Trainer*,* Version 0.9, 삼성전자 첨단기술 아카데미, May 2023.

[Kim 23b] Soo Dong Kim, Associate Architect Program, 2023-A3, CEP Template for GPT-Assisted Coding Trainer, 삼성전자 첨단기술 아카데미, May 2023.

[Kim 23c] Soo Dong Kim, Architecture Design Process & Instructions, 2023-A3, Lecture Note #1, 삼성전자 첨단기술 아카데미, May 2023.

[Kim 23d] Soo Dong Kim, Associate Architect Program, 2023-A3, CEP Specification of GPT-Assisted Coding Trainer*,* Version 1.0, 삼성전자 첨단기술 아카데미, May 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 22c]. It consists of the following six activities as shown in Figure 2.

A picture containing text, screenshot, font, design

Description automatically generated

Figure . 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 of the target system is specified in the context model of the target system.

* Activity 3. Schematic Architecture Design

This activity is to design the initial and high-level architecture of the target system, called schematic architecture. This 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 schematic 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 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 is 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. System Manager

This represents the staff who will manage the overall operation of the Coding Trainer system.

* Stakeholder 2. Client

This represents the organization that will sponsor the development of the system and distribute the system to application users.

* Stakeholder 3. Learner

This represents a learner who uses the Coding Trainer to learn a program language through this system.

* Stakeholder 4. Course Director

This represents the course director who responsible for managing program Language course contents(Language Profile).

The profile of each stakeholder is summarized in Table 1.

Table . Profiles of Stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
| **Stakeholder Group** | **Representative Name** | **Contact Information** | **Availability** |
| System Manager | Linda Johnson | 251-546-9442 Gulf Shores, AL | After 2pm, only on F, Phone Only |
| Client | James Brown | 415-546-4478 San Francisco, CA | Before Noon, MWF, Phone Only |
| Learner | Susan Tayler | 949-569-4371 Santa Clara, CA | 10-Noon, T. Th, Office Visits |
| Course Director | 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 4.4 |
| **Original Context** | “Learners have the opportunity to engage with the system by typing questions or queries, and the system responds with appropriate and accurate answers.” | | | | |
| **Questioning** | Is Learner always able to ask questions in any session? | | | | |
| **Refined Context** | No. After each Topic's Foundation, exercise, and submitted code evaluation, a question box will be created to ask questions. | | | | |

* Deficiency #2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Deficiency ID** | FR.DEF.02 | **Deficiency Type** | Ambiguity | **Location** | SRS 4.6 |
| **Original Context** | “Upon completion of the exercise, learners can submit their program codes through the system's interface.” | | | | |
| **Questioning** | What are the methods of the system interface for submitting program code? | | | | |
| **Refined Context** | The system interface supports direct typing and file upload. File uploads support txt, jpg, and png files. | | | | |

## [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. Stable Operation of Coding Trainer

The target system provides services potentially to a large number of users, and hence the system should be designed to provide high-level reliability.

* Concern-2. Efficient Personalization for Leaners

The system should be designed to offer a high level of personalization.

* Concern-3. Accuracy of Language Profile

The system should be designed that directors provide an accurate language profile.

Merge newly derived NFR items and the NFR items of the SRS. We now have 2 NFR items.

* Concern-1. Stable Operation of Coding Trainer 🡪  
   (NFR #1) High Reliability of the System
* Concern-2. Efficient Personalization for Leaners 🡪  
   (NFR #2) High Personalization of Instruction and Coding Exercises
* Concern-3. Accuracy of Language Profile 🡪 NFR #3 ((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)** |
| Learner | System Manager | Client | Course Director |
| NFR-1 | 2 | 2 | 2 | 1 | 1.5 | 0.58 | Y |
| NFR-2 | 2 | 0 | 2 | 0 | 1 | 1.15 | Y |
| NFR-3 | 2 | -1 | -1 | 2 | 0.5 | 1.73 | N |

We apply a 5-level Relevance rating scheme as shown in Figure 3 to fill in the table.



Figure . 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** | “Reliability in ISO 9126 is defined with three sub-quality attributes, and they should be satisfied by the system.” | | | | |
| **Questioning** | In order to satisfy the reliability of ISO 9126, I think we need to run a redundant server. How many units can we operate? | | | | |
| **Refined Context** | It's good to run a lot of servers, but we need to minimize costs due to financial problems. | | | | |

* Deficiency #2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Deficiency ID** | NFR.DEF.02 | **Deficiency Type** | Incompleteness | **Location** | SRS 5.2 |
| **Original Context** | the system should be designed to offer a high level of personalization across three key aspects | | | | |
| **Questioning** | What is the third key aspect? | | | | |
| **Refined Context** | Personalizing the solution evaluation.  The evaluation of the solution code should be based on the individual's code and convey information that fits the individual's competency, such as advantages, disadvantages, and points to be improved. | | | | |

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 the followings.

* 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 users, hardware devices, external systems or other sources in the operational environment. *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.

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 the Boundary Context

The target system is a versatile software platform designed to support the development of various applications. The architectures of Coding Trainer consist of 1 tier and accordingly the context diagram includes 1 process.

We define the boundary context in a DFD, as shown in Figure 4.

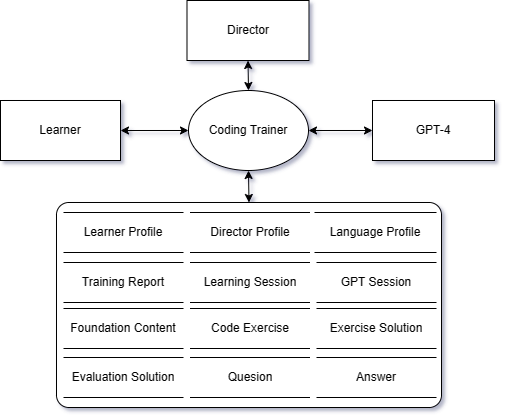


Figure . DFD Diagram for Boundary Context

### Description of the Context Diagram

* Process

The system consists of 1 tier and each tier is represented as a process of DFD.

* Coding Trainer
* Terminals

A terminal represents a source providing inputs to the system and/or a destination consuming output from the system.

* The terminals of human user types are *Learner, Director* and System *Manager*.
* Learner

This represents learners who are responsible for managing learner profile and training reports.

* Director

This represents course directors who are responsible for managing language profile.

* Administrator

This represents administrator who is responsible for activating director profile & language profile and deactivating director profile & language profile.

* The terminal of external system is *GPT-4.*
* GPT-4

The system leverages the capabilities of the GPT-4 model through its API to perform essential tasks and provide a comprehensive learning experience.

* Store
* The application maintains a number of distinct data stores. All the essential persistent information is captured in the data store set.
* Learner Profile

The Learner Profile includes the following attributes: name, official identification, address, phone, email, and affiliation. Learners also register their login ID and password.

* Director Profile

The Director Profile includes the following attributes: name, official identification, director identification, address, phone, email, department, specialty, and courses managed. Directors also register their login ID and password.

* Language Profile

The Language Profile includes the following attributes: Profile ID, Programming Language Name, Units, and Topics

* Training Report

The Training Report includes the following attributes: Certification

* Learning Session

The Learning Session includes the following attributes: Language Progress, Unit Progress, Topic Progress.

* GPT Session

The GPT Session include Generated GPT Prompt and GPT Completion.

* Foundation Content
* Code Exercise
* Exercise Solution
* Evaluation Solution
* Q&A List
* 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

### Representing the Functional Context

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.

### Defining 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.

* Learner Profile Management (LE)
* Director Profile Management (DI)
* Administrator Profile Management (AD)
* Language Profile Management (LA)
* Foundation Teaching (FT)
* Exercise Generation (EG)
* Solution Evaluation (SE)
* Training Report Generation (RP)
* Question Answer (QA)
* Training Operation (TO)

### Defining Actors

We define actors that interact with the use cases. Each functional group is given its relevant actors as shown below.

|  |  |  |
| --- | --- | --- |
| **Actors**  **Functional Groups** | **Active Actors** | **Passive Actors** |
| Learner Profile Management (LE) | Learner |  |
| Director Profile Management (DI) | Director, Administrator |  |
| Administrator Profile Management (AD) | Administrator |  |
| Language Profile Management (LA) | Director, Administrator |  |
| Foundation Teaching (FT) | Foundation Agent | GPT-4 |
| Exercise Generation (EG) | Exercise Agent | GPT-4 |
| Solution Evaluation (SE) | Evaluation Agent | GPT-4 |
| Training Report Generation (RP) | Learner,  Foundation Agent,  Exercise Agent,  Evaluation Agent,  Question Agent | GPT-4 |
| Question Answer (QA) | Question Agent | GPT-4 |
| Training Operation (TO) | Learner | Foundation Agent,  Exercise Agent,  Evaluation Agent,  Question Agent |

### Defining Use Cases

The use cases in the diagram can be derived from the functional requirement of SRS. The name of each use case begins with a prefix which indicates the functional group it belongs to. The use cases in a functional group are placed together in the diagram for readability.

A use case typically has an active actor and optional passive actors.

### Context-level Use Case Diagram

The use case diagram for the functional context is shown in Figure 5.



Figure . Use Case Diagram for the Functional Context

[Step 3] Information Context

### Representing the Information Context

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

In this context-level class diagram, we only show the entity-type classes, their relationships, and the cardinalities. No need to specify attributes and methods at this stage.

### Identifying Persistent Object Classes

Persistent object classes of the target system can be the following types of classes.

* Classes for Physical Objects
* Classes for Logical Objects
* Classes for Session-related Objects

### Context-level 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 6.

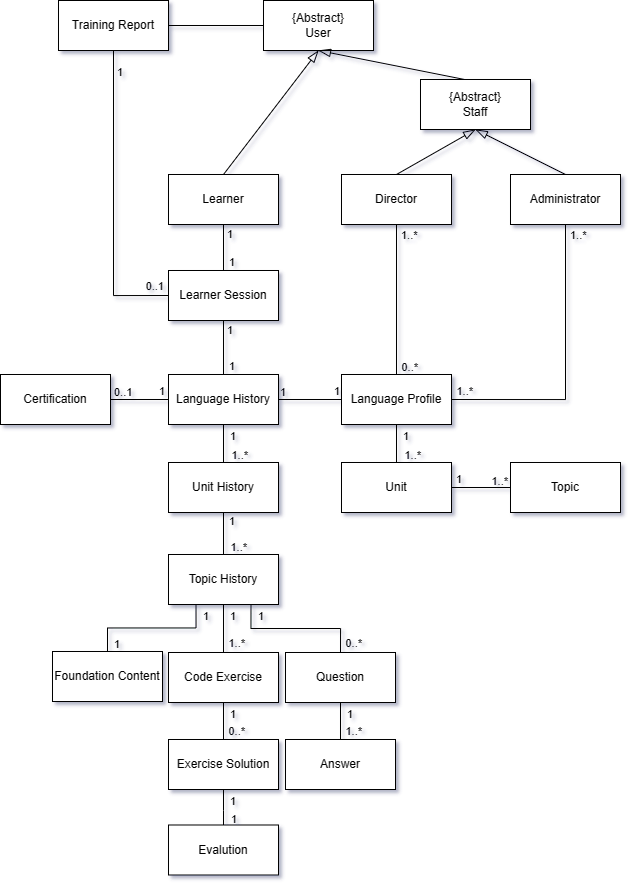


Figure . Class Diagram for showing Information Context

* Relationships

Various types of relationships are defined between classes.

* Cardinalities

Cardinalities are defined on every relationship.

## [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.

### Representing the Behavior Context

Activity Diagram of UML provides a rich set of constructs that can be used to model the runtime behavior of the system.

### Allocate Functionality over Tiers

This task is to allocate the system functionality over the tiers. Use the table of functionality allocation as in Table 3.

Since the target system runs as a single tier, there is NO need to allocate on tiers.

Table . Allocation of Functionality over Tiers

|  |  |
| --- | --- |
| Tiers  Functional Groups | **Coding Trainer** |
| Learner Profile Management (LE) | ✓ |
| Director Profile Management (DI) | ✓ |
| Administrator Profile Management (AD) | ✓ |
| Language Profile Management (LA) | ✓ |
| Foundation Teaching (FT) | ✓ |
| Exercise Generation (EG) | ✓ |
| Solution Evaluation (SE) | ✓ |
| Training Report Generation (RP) | ✓ |
| Question Answer (QA) | ✓ |
| Training Operation (TO) | ✓ |

### 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.
* This is not appliable to 1-tier system.

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

Table . Invocation Patterns defined for Functional Groups

|  |  |
| --- | --- |
|  | **Coding Trainer** |
| Learner Profile Management (LE) | Explicit |
| Director Profile Management (DI) | Explicit |
| Administrator Profile Management (AD) | Explicit |
| Language Profile Management (LA) | Explicit |
| Foundation Teaching (FT) | Event I, C-Loop |
| Exercise Generation (EG) | Event I, C-Loop |
| Solution Evaluation (SE) | Event I, C-Loop |
| Training Report Generation (RP) | Event I, C-Loop, Timer |
| Question Answer (QA) | Event I, C-Loop |
| Training Operation (TO) | Explicit |

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

### Context-level Activity Diagram

Based on the invocation patterns defined over the tiers, we draw an activity diagram for each tier in the system as shown in Figure 7.

* Control Flow of Coding Trainer

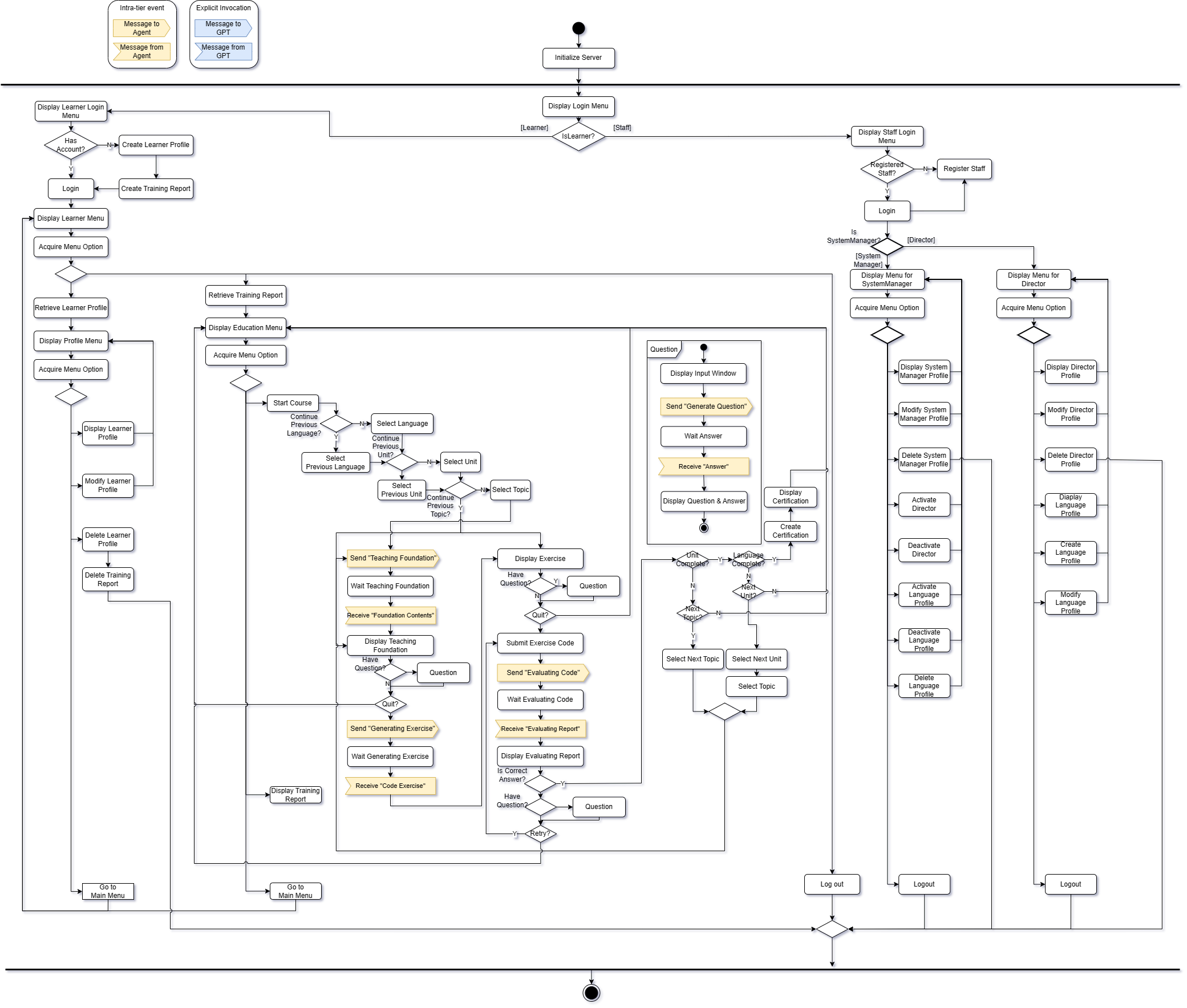


Figure . Activity Diagram for showing Behavior Context of System

The control flow begins with initialize process. Then, it runs 5 parallel threads.

* The thread #1 is to handle the functionality with explicit invocation pattern using a menu.
* The Thread #2~#5 are SW agent as shown in Figure 8.

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

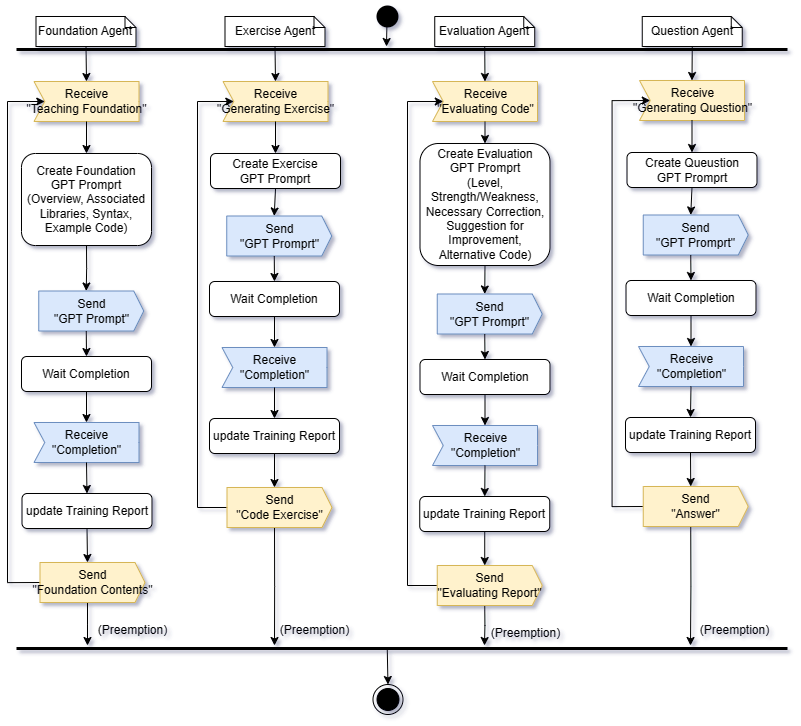


Figure . Activity Diagram for showing Behavior Context of SW Agent

* All Agents operate event-driven.
* Foundation agent and Exercise Agent build GPT-Prompt based on Topic in Language profile.
* Evaluating Agent builds GPT-Prompt based on submitted code.
* Question Agent builds GPT-Prompt based on Learner’s Question.

## [Step 5] Additional Contexts

Any additional contexts of the target system can be described.

* None

# Activity 3. Schematic Architecture Design

A schematic 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 and derive candidate architecture styles from the observation.

* System consisting with Distinct Roles 🡪 MVC Architecture Style

The Coding Trainer whole system consists of 3 distinct roles of presentation, business logic, and data persistency management.

* Considering Loosely-coupled Event Handling 🡪 Event-driven Architecture Style

The Coding Trainer server can be operated by several events. To handle the events, additional processes/threads in the background are essential.

* High Reliability on Server 🡪 Dispatcher Architecture Style

The Coding Trainer system needs a load balancing for a reliable server.

[Step 2] Candidate Architecture Styles

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

* MVC architecture style
* Event-driven architecture style
* Dispatch architecture style

## [Step 3a] Evaluating ‘Candidate 1. MVC Architecture Style’

The MVC architecture decomposed a system into View, Control and Model. Each role has different responsibilities in the system.

### Evaluate the Applicable Situations

|  |  |  |
| --- | --- | --- |
| **Applicable Situations** | **Match** | **Demands on the System** |
| The target system can be divided into 3 distinct roles of View, Control and Model. | 🔾 | Coding Trainer system consists of View for users, Control for its logic and Model data. |
| View and Control are frequently changed. | 🔾 | Coding Trainer system provides user program language contents for learning which may be changed frequently as userfriendly-format. |
| This system is used for presenting Model Data as number of different types of View. | 🔾 | Coding Trainer system has multiple type of User and each user interact with system with different view. |

### Evaluate the Benefits

|  |  |  |
| --- | --- | --- |
| **Advantages of the Style** | **Match** | **Benefits Applicable to the System** |
| The dependency of each layer is minimal; depending on its immediately lower layer. | 🔾 | Each layer of Coding Trainer system should depend only on its immediate lower layer, reducing the dependency on other parts. |
| Various View are implemented with one Model. | 🔾 | Coding Trainer system provides different View for each types of Users with one Model. |
| Guarantee Synchronization of all independent View and Control. | 🔾 | Model changes immediately propagated to all Views and Controls and represented. |

### Evaluate the Drawbacks

|  |  |  |
| --- | --- | --- |
| **Cons of the Style** | **Match** | **Handling the Drawbacks** |
| Increase design complexity. | 🔾 | MVC introduces some extra classes/code due to the separation of model, view and controller. However, each functionality is distinct in the target system. |
| Difficulty of allocation functionality to appropriate component. | 🔾 | Coding Trainer system, each components’ role is clear. This problem is not expected. |
| Each View can be notified unexpected Model changes. Not all View has concern with Model changes. | 🔾 | This is not critical. The majority of Model changes are Learner related, so most of concern is about View, respectively, and it is essential. |

### Result of the Evaluation

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 2b] Evaluating ‘Candidate 2. Event-driven Architecture Style’

The target system can be well configured with the Event-driven architecture style. This is applied Generating Coding Trainer’s Learning Contents.

### Evaluate the Applicable Situations

|  |  |  |
| --- | --- | --- |
| **Applicable Situations** | **Match** | **Demands on the System** |
| The system consists of event emitters and event sinks. Occurrence of events invoke specific functionality of events. | 🔾 | Coding Trainer system use various event signal to service specific functionality. |
| The communication between event emitter and event sinks is asynchronous, providing benefits of parallelism, modularity, and extendibility. | 🔾 | Coding Trainer system uses asynchronous event handling to have parallelism, modularity and extendibility. The communication between event emitter and event sinks is asynchronous. |
| Upon arrival, the event listener locates an appropriate event handler for the received event. | 🔾 | Components of Coding Trainer server processes the event in its way according to a service type. |

### Evaluate the Benefits

|  |  |  |
| --- | --- | --- |
| **Advantages of the Style** | **Match** | **Benefits Applicable to the System** |
| This style achieves high modularity through its asynchronous capabilities. | 🔾 | Separation between a part which invokes event and a part handles that event makes possible to implement independently and is helpful in analysis. |
| It is possible to process the event handling in parallel. | 🔾 | The target system handles the event listener in parallel by using an asynchronous thread |

### Evaluate the Drawbacks

|  |  |  |
| --- | --- | --- |
| **Cons of the Style** | **Match** | **Handling the Drawbacks** |
| Event-driven architecture pattern is a relatively complex pattern to implement, primarily due to its asynchronous distributed nature. | 🔾 | The number of event sources and sink sources are limited since the number of functionalities is small. |
| It is difficult to figure out cause of errors immediately and hard to understand whole system data control flow. | 🔾 | The Coding Trainer System does not complex event algorithm. The number of events is understandable. |

### Result of the Evaluation

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 2c] Evaluating ‘Candidate 3. Dispatcher Architecture Style’

The target system can be well configured with the Dispatcher architecture style. This is applied between Web Browser and Server for high QoS.

### Evaluate the Applicable Situations

|  |  |  |
| --- | --- | --- |
| **Applicable Situations** | **Match** | **Demands on the System** |
| The system adopts Dispatcher middle layer between client and server. | △ | The Coding Trainer System has performance impact. High QoS needs to meet for Stakeholder’s concern. Coding Trainer system has high demand of Load-balancing. |
| For high scalability and availability in Distributed environment, this system maintains multiple servers those are considered as One Server at Client. | 🔾 | The Coding Trainer system can have multiple Server with Dispatcher. |
| Need replicate Servers for Reliability | 🔾 | Need to have multiple servers for the NFR of availability. |

### Evaluate the Benefits

|  |  |  |
| --- | --- | --- |
| **Advantages of the Style** | **Match** | **Benefits Applicable to the System** |
| High Availability and Reliability | 🔾 | It is very important to minimize failure and data loss between operations. |
| High Performance | 🔾 | There will be a lot of service request from learner in the target system. The performance and QoS should be high. |

### Evaluate the Drawbacks

|  |  |  |
| --- | --- | --- |
| **Cons of the Style** | **Match** | **Handling the Drawbacks** |
| Difficulty of changing dispatcher interface | △ | The interface of Coding Trainer system would be fixed before deployment. When the change is happened, we can apply design pattern like adapter to fix interface mismatch. |
| Inefficient connection manner between client-dispatcher-server. It provides an explicit and bypass connection. | △ | This is not critical for the Coding Trainer system. Availability is more important. |

### Result of the Evaluation

Dispatcher 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 2e] List of Selected Architecture Styles

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

* MVC Architecture Style
* Event-driven Architecture Style
* Dispatcher Architecture Style

## [Step 3] Applying Architecture Styles

We apply the selected architecture styles incrementally.

### Applying MVC Architecture Style

Coding Trainer Server are composed of MVC as shown in Figure 9.

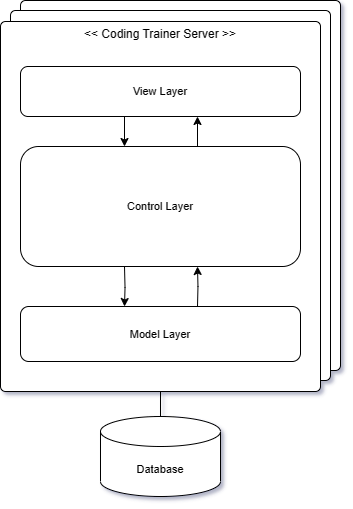


Figure . Applying MVC Architecture Style

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

### Applying Event-driven Architecture Style

For Coding Trainer Server, Event-driven Architecture Style is applied as shown in Figure 10.

Each user can request direct function execution through the displayed screen, which is included in the explicit invocation control layer.

According to Learner's direct function performance, the SW Agent performed in the background is designed to operate on an event basis.

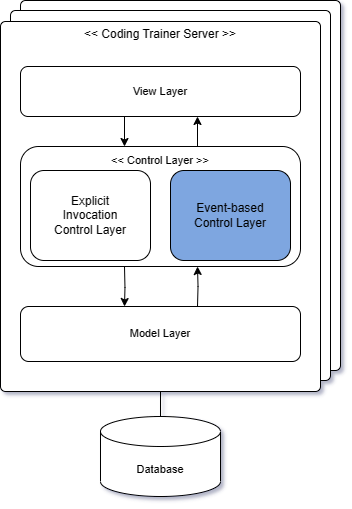


Figure . Applying Event-driven Architecture Style

### Applying Dispatcher Architecture Style

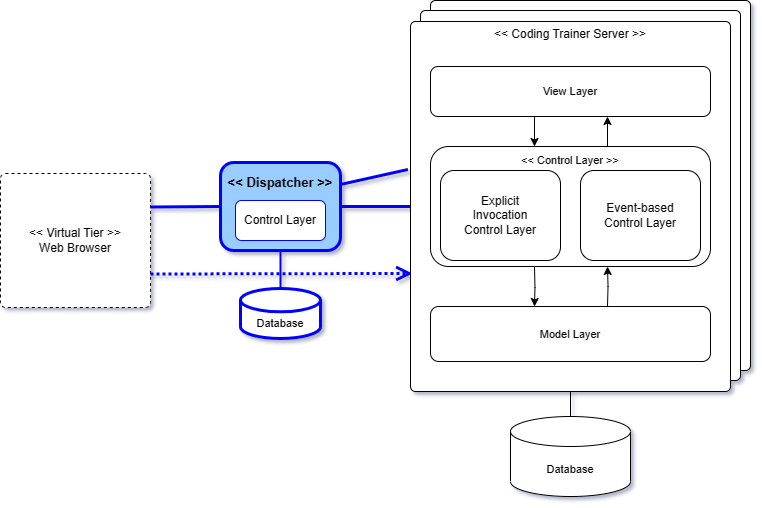


Figure . Applying Dispatcher Architecture Style

The system applied Dispatcher architecture style. Dispatcher located between Web Browser and Coding Trainer Server.

The coding trainer system itself is deployed as a web server, and there is no Client Tier with Control, but it has a Virtual Tier, a web browser, which connects users to the server in consideration of Load Balancing and QoS when accessing the server from the web browser. It is expected to behave like a router on the network.

### Resulting Schematic Architecture

The resulting Schematic architecture of applying all the selected architecture styles is shown in Figure 12.

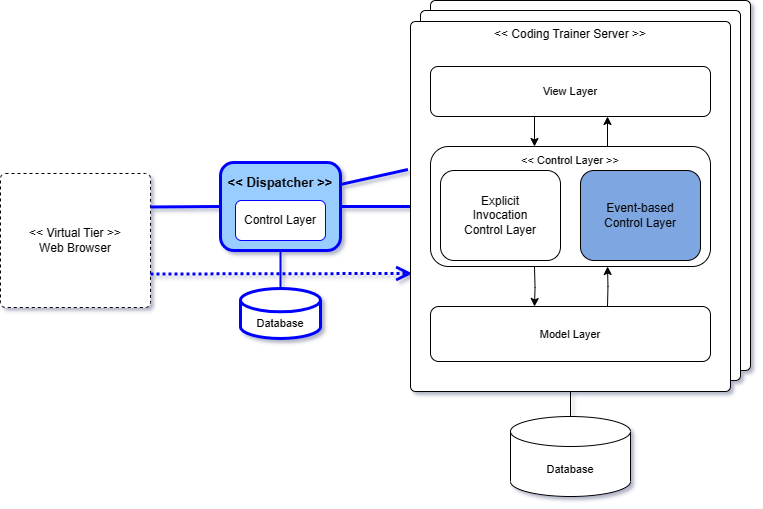


Figure . Resulting Schematic Architecture

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.

## [Step 4] Refining Interaction Paths

Define interaction paths among places in the Schematic 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 architecture style are adopted and remain unchanged in the Schematic architecture.

* Interaction Paths from MVC Architecture Style
* Interaction Paths from Event-driven Architecture Style
* Interaction Paths from Dispatcher Architecture Style

### Refinements on the Default Interaction Paths

* None

The target platform does not require refinements on the interaction paths. Therefore, the default interaction paths remain the same.

## [Step 4] Elaborating the Schematic Architecture

### Strengths

Specify the advantages of the proposed Schematic architecture.

* Separation of Concern

Each component or layer represents 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 Schematic architecture, the impact of modification would be minimal.

### Drawbacks

Specify the drawbacks and risks of the proposed Schematic architecture.

* Not Anticipated.

# Activity 4. View-specific Architecture Design

This chapter describes the results of applying essential architecture viewpoints. The Schematic 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 User Profiles

The target system has three kinds of user to manage; Learners, Director and Administrator.

* Functionality of Managing Language Profile
* Functionality of Foundation Teaching
* Functionality of Exercise Generation
* Functionality of Solution Submission
* Functionality of Solution Evaluation
* Functionality of Training Report Generation
* Functionality of Question Answer
* Functionality of Learner Session

### [Step 2] Refine Use Case Diagram

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

* Functional Groups (from the SRS)
* Learner Profile Management (LE)
* Director Profile Management (DI)
* Administrator Profile Management (AD)
* Language Profile Management (LA)
* Foundation Teaching (FT)
* Exercise Generation (EG)
* Solution Submission (SS) (added)
* Solution Evaluation (SE)
* Training Report Generation (RP)
* Question Answer (QA)
* Learner Session (LS) (changed)
* 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 13.

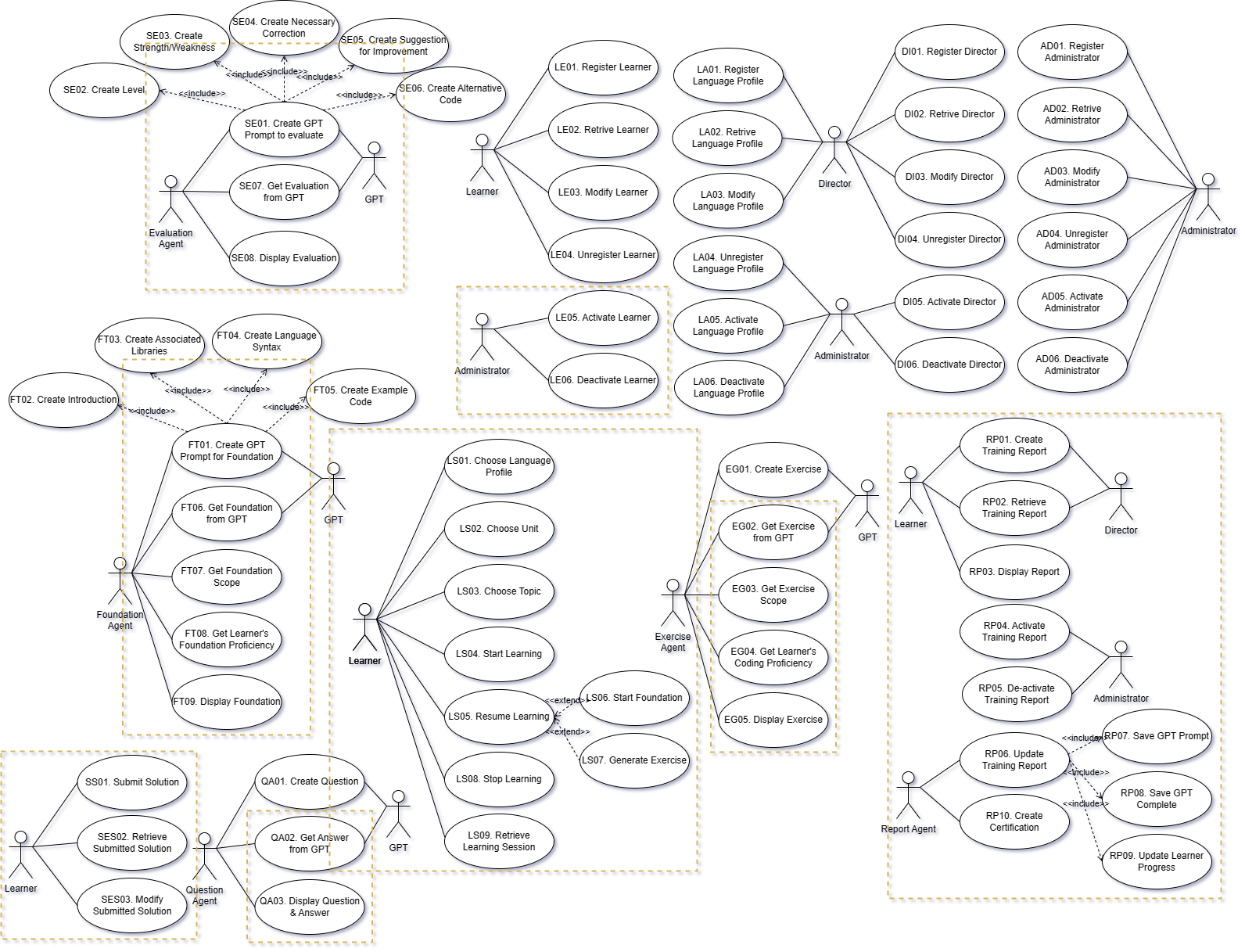


Figure . Refined Use Case Diagram

* Refinements made

The refinements made on the initial use case diagram are shown with rectangle markers.

* The Training Operation Group has been changed to Learner Session.
* Added Scope and Proficiency to be reflected in the Exercise Generation and Foundation Teaching group.
* The existing create use cases of Foundation Teaching and Solution Evaluation Group were changed to the include relationship of the GPT Prompt generation function.
* The set of 3 use cases for Solution Submission are now defined.
* The set of 10 use cases for Training Report Generation are now added.
* We refine Training Report Generation Group such as Saving GPT sessions, Updating Learner Progress.

### [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 Schematic Architecture

Schematic 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 are shown in Figure 14.

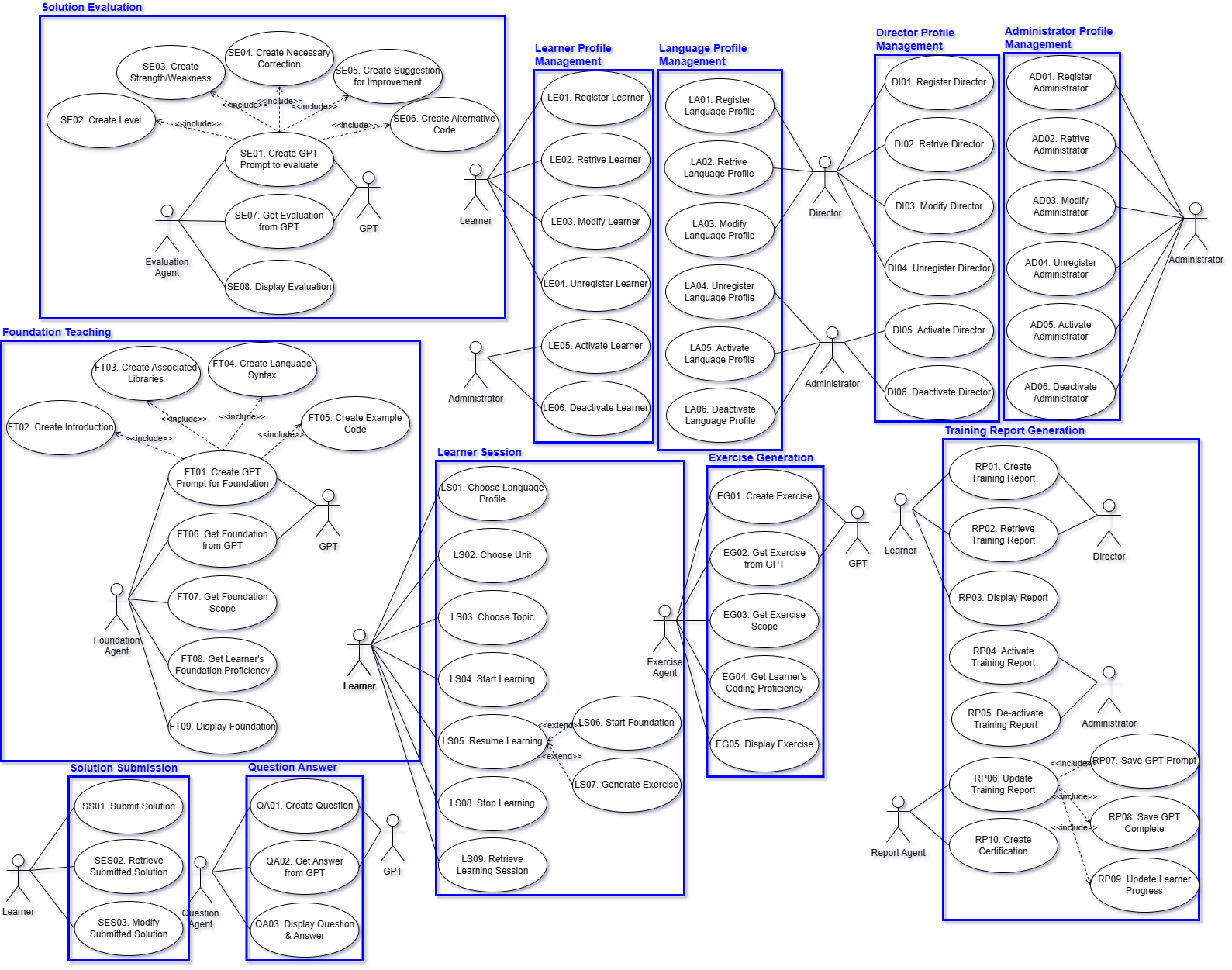


Figure . Deriving Functional Components

The summary of the functional components and their relevant use cases is shown in Table 5.

Table . Functional Components and their Use Cases

|  |  |  |
| --- | --- | --- |
| **Functional Components** | **Use Cases** | **# of Use Cases** |
| Learner Profile Management (LE) | LE01~LE06 | 6 |
| Director Profile Management (DI) | DI01~DI06 | 6 |
| Administrator Profile Management (AD) | AD01~AD06 | 6 |
| Language Profile Management (LA) | LA01~LA06 | 6 |
| Foundation Teaching (FT) | FT01~FT09 | 9 |
| Exercise Generation (EG) | EG01~EG05 | 5 |
| Solution Submission (SS) | SS01~SS03 | 3 |
| Solution Evaluation (SE) | SE01~SE08 | 8 |
| Training Report Generation (RP) | RP01~RP10 | 10 |
| Question Answer (QA) | QA01~QA03 | 3 |
| Learner Session (LS) | LS01~LS09 | 9 |
| Total # of Use Case | | 71 |

* Components derived from Architecture Styles

Table . Functional Components of Dispatcher Architecture Style

|  |  |
| --- | --- |
| **Functional Components** | **Use Cases** |
| Server Allocation Manager | QM01. Register Server  QM02. Evaluate QoS  QM03. Allocate Server |

The Coding Trainer System supports communication with the server through the web browser. At this time, the dispatcher structure is intended to be used for stable service support. The Web Browser communicates with the dispatcher first and receives the server assignment to proceed with the server.

* Interface Components

The target system does not have an interface-centric Functional Components.

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

This step is to refine functional components on the tiers of Schematic architecture. Since the Schematic architecture of the target system has only one tier, this step is not applicable.

* Not Applicable

### [Step 5] Allocate Functional Components

This step is to find the Functionality Place Holders and allocate functional components onto functionality place holders in the Schematic architecture.

* Functionality Place Holders

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 functionality place holders for the target system are shown in Figure 15.

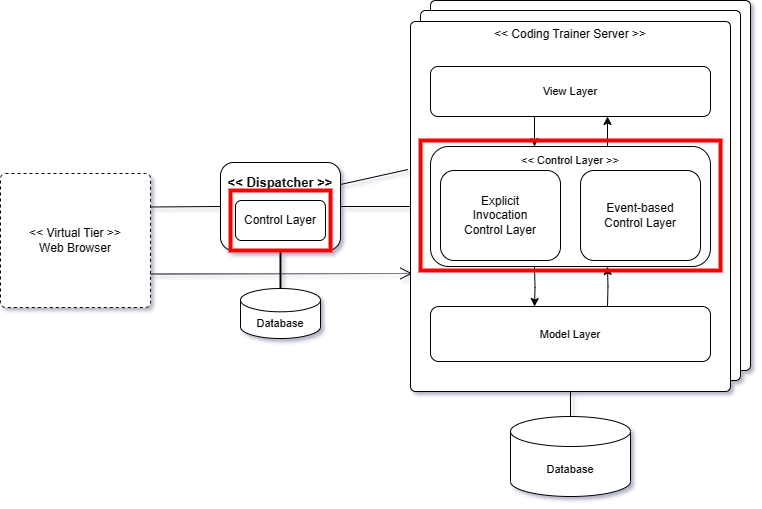


Figure . Functionality Place Holders of the Target System

* Allocating Functional Components

Since the Schematic architecture is defined with 1-Tier Architecture style, we assigned all functional components to server. However, Server Allocation Manager, a Functional Component by Architecture, was specifically assigned to Dispatch.

The classification of the functional components is shown in Table 7

Table . Partitioning Functional Components

|  |  |  |
| --- | --- | --- |
| **Place Holders**  **Functional Components** | **Cording Trainer Server** | **Dispatcher** |
| Learner Profile Manager | Learner Prof. Manager |  |
| Director Profile Manager | Director Prof. Manager |  |
| Admin Profile Manager | Admin Prof. Manager |  |
| Language Profile Manager | Language Prof. Manager |  |
| Foundation Teaching Generator | Foundation Teaching Generator |  |
| Exercise Generator | Exercise Generator |  |
| Solution Submitter | Solution Submitter |  |
| Solution Evaluator | Solution Evaluator |  |
| Training Report Generator | Training Report Generator |  |
| Question Answer Generator | Question Answer Generator |  |
| Learner Session Manager | Learner Session Manager |  |
| Server Allocation Manager |  | Server Allocation Manager |

As shown in the table, most functional components were assigned to servers, but Server Allocation Manager for distributed processing and stable server operation was assigned to Dispatcher to connect the appropriate server from Server Allocation Manager when connected through a web browser.

* Server Allocation Manager

Server Allocation Manager is like a router and is responsible for connecting connections to the optimal server. Due to the characteristics of the user accessing through the web browser, it is difficult to maintain a separate control and database, and the dispatcher integrates and manages the function. It will be possible to maximize the stability and efficiency of the service by playing the role of a Load Balancer even in the use of many users.

We now allocate the refined functional components on the functionality holders according to the table of refined functional components, as shown in Figure 16.

Each place holder is defined with its functional components. The Schematic architecture is now more complete with functional components.

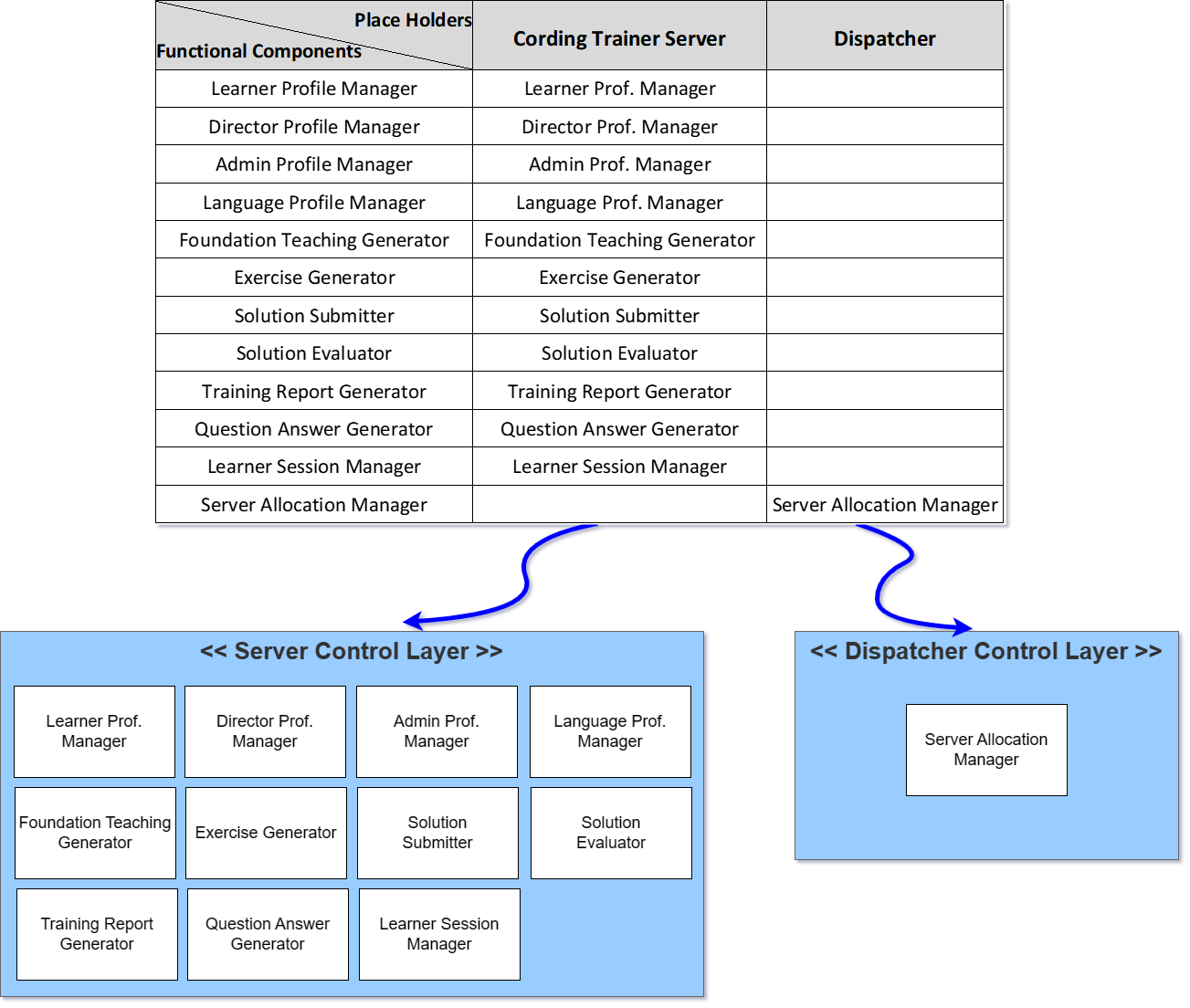


Figure 16. Allocaiton of Components on Functionality Place Holders

### [Step 6] Design Functional Components

Each functional component must be designed in greater detail. 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 points where the variability occurs.

We use the table of Functional Component Design Decisions as shown in Table 8.

Table . Design of Functional Components

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Refined Functional Components** | **Visibility** | **Interface Type** | **Open/Closed** | **Variant Points** |
| Learner Profile Manager | Blackbox | Façade | Closed | N/A |
| Director Profile Manager | Blackbox | Façade | Closed | N/A |
| Admin Profile Manager | Blackbox | Façade | Closed | N/A |
| Language Profile Manager | Blackbox | Façade | Closed | N/A |
| Foundation Teaching Generator | Blackbox | Façade | Closed | N/A |
| Exercise Generator | Blackbox | Façade | Closed | N/A |
| Solution Submitter | Blackbox | Façade | Closed | N/A |
| Solution Evaluator | Blackbox | Façade | Closed | N/A |
| Training Report Generator | Blackbox | Façade | Closed | N/A |
| Question Answer Generator | Blackbox | Façade | Closed | N/A |
| Learner Session Manager | Blackbox | Façade | Closed | N/A |
| Server Allocation Manager | Blackbox | Façade | Closed | N/A |

* Blackbox Components

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

* Whitebox Components

None

* Components with Openness

The components with openness shows 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.

However, we do not describe the contents because there is no use of open-type components in our target system.

## 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.

* Adding Session-related Classes
* Refining Relationships for various Conceptual Object Classes

### [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 17. Note that attributes are omitted in this sample solution.

The refined class diagram shows refinements in colors.

* The class used for the previous training report has been changed to Learner Session. Also, I changed the term from Progress to Learning History.
* At the same time as Learner's registration, a Training Report is generated to maintain a one-on-one relationship. This ensures the Learner's behavior to the Training Report.
* The Learner Session includes all Language History, Unit History, and Topic History so that you can include all of the history of language learning.
* Language History contains ordered content based on the Language Profile.
* Administrators have more accessible components than ever before.
* GPT-4 Session added. To overcome the limitations of short-term session of GPT, all information of Foundation Content, Excellence, Evaluation, and Q&A generated through GPT can be recorded.
* Refinements on Relationships
* Relationships between Training Report and *User type*.
* Relationships between Administrator and Director.
* Relationships between Administrator and Learner.
* Relationships between Learner Session and Language Learning History.
* Relationships between Language Learning History and Unit Language History.
* Relationships between Topic Learning History and Each Learning Contents.
* Each class is specified with a textual description.

Omitted in this sample solution.

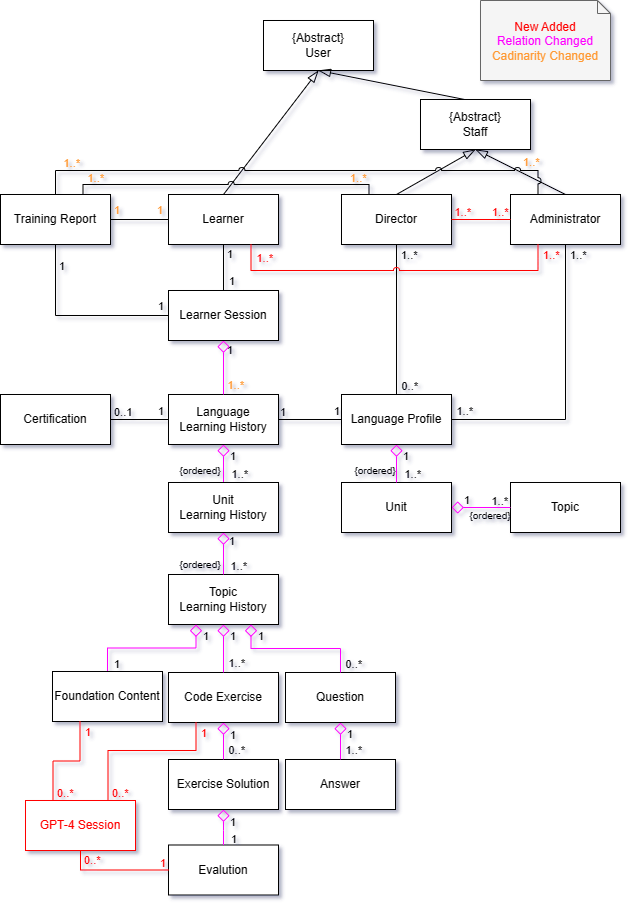


Figure . Refined Class Diagram

### [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 18.

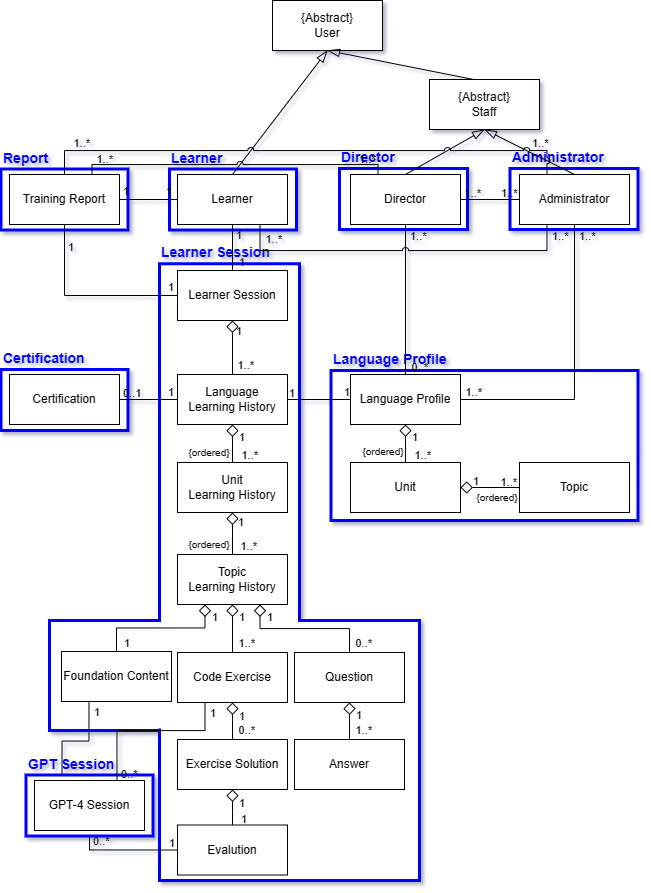


Figure . List of Data Components

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

* Data Components derived from Architecture Style

None

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

The Schematic 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.

* In fact, the Coding Trainer System has only one tier.

Table . Data Components allocated on Tiers

|  |  |
| --- | --- |
| **Tier**  **Data Components** | **Coding Trainer Server** |
| Learner | Learner |
| Director | Director |
| Administrator | Administrator |
| Language Profile | Language Profile |
| Learner Session | Learner Session |
| Report | Report |
| Certification | Certification |
| GPT Session | GPT Session |

* 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 19.

* Place Holders for Data Components

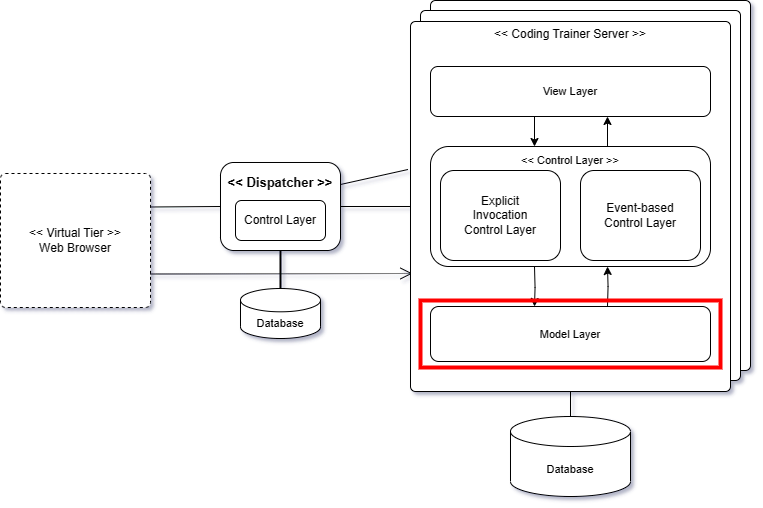


Figure . Data Component Holders of the Target System

There is only one data component holder in the system: *Model Layer*.

We now allocate the data components on the data component placeholders. We use the table of refining data components over tiers as shown in Figure 20.

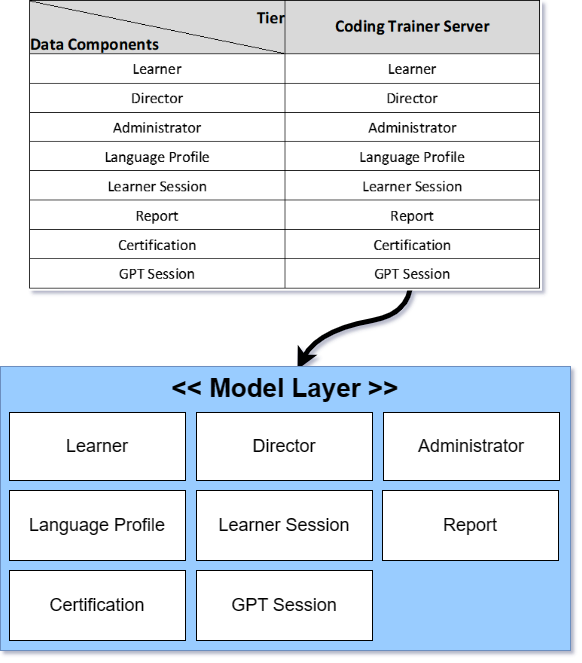


Figure . Allocation of Data Components

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.

The invocation patterns for functional groups are refined as shown in Table 10.

Table . Invocation Patterns defined for Functional Groups

|  |  |
| --- | --- |
|  | **Coding Trainer System** |
| Learner Profile Management | Explicit |
| Director Profile Management | Explicit |
| Administrator Profile Management | Explicit |
| Language Profile Management | Explicit |
| Foundation Teaching | Event-driven, Closed Loop |
| Exercise Generation | Event-driven, Closed Loop |
| Solution Submission | Explicit |
| Solution Evaluation | Event-driven, Closed Loop, Timer |
| Training Report Generation | Event-driven, Closed Loop |
| Question Answer | Event-driven, Closed Loop |
| Learner Session | Explicit |

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

### [Step 2] Refining Control Flow of the Platform

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 platform is shown in Figure 21, Figure 22.

* Refinements made on the Control Flow
* The system starts and selects whether it is Staff or Learner to proceed with Login. A different View will be shown for each User Type.
* In the case of Learner, a training report was generated with the register step so that it could have all of Learner's Language Study History.
* When you enter the Education Menu, you decide whether to proceed with the previous session or choose Language, Unit, and Topic, and proceed with the learning. If you select the previous session, you start with the previous breakpoint, and if you select Topic, you continue learning if you have a previous learning record, or you start with Foundation Teaching.
* A procedure has been added to take into account the scope for the Learner's proficiency and learning efficiency when creating personalized Foundation Content and Exercise.
* Solution submission was separated from the learning process and changed to be executed through the menu. After the solution is submitted, Evaluation Agent is designed to perform all the contents. Evaluation Agent is designed to determine whether to use GPT, notify the user of the completion of the evaluation after processing immediately if it can be processed immediately, and conduct the evaluation in the background for the submitted history of the solution that has not been processed.
* Each agent uses GPT using the GPT API, and is designed to store GPT Session and Learner's learning history by delivering an event to the Report Agent every cycle. When all learning of Language is completed, the report agent judges and creates a certification, and the user can check the training report in real time in the Display Training Report.

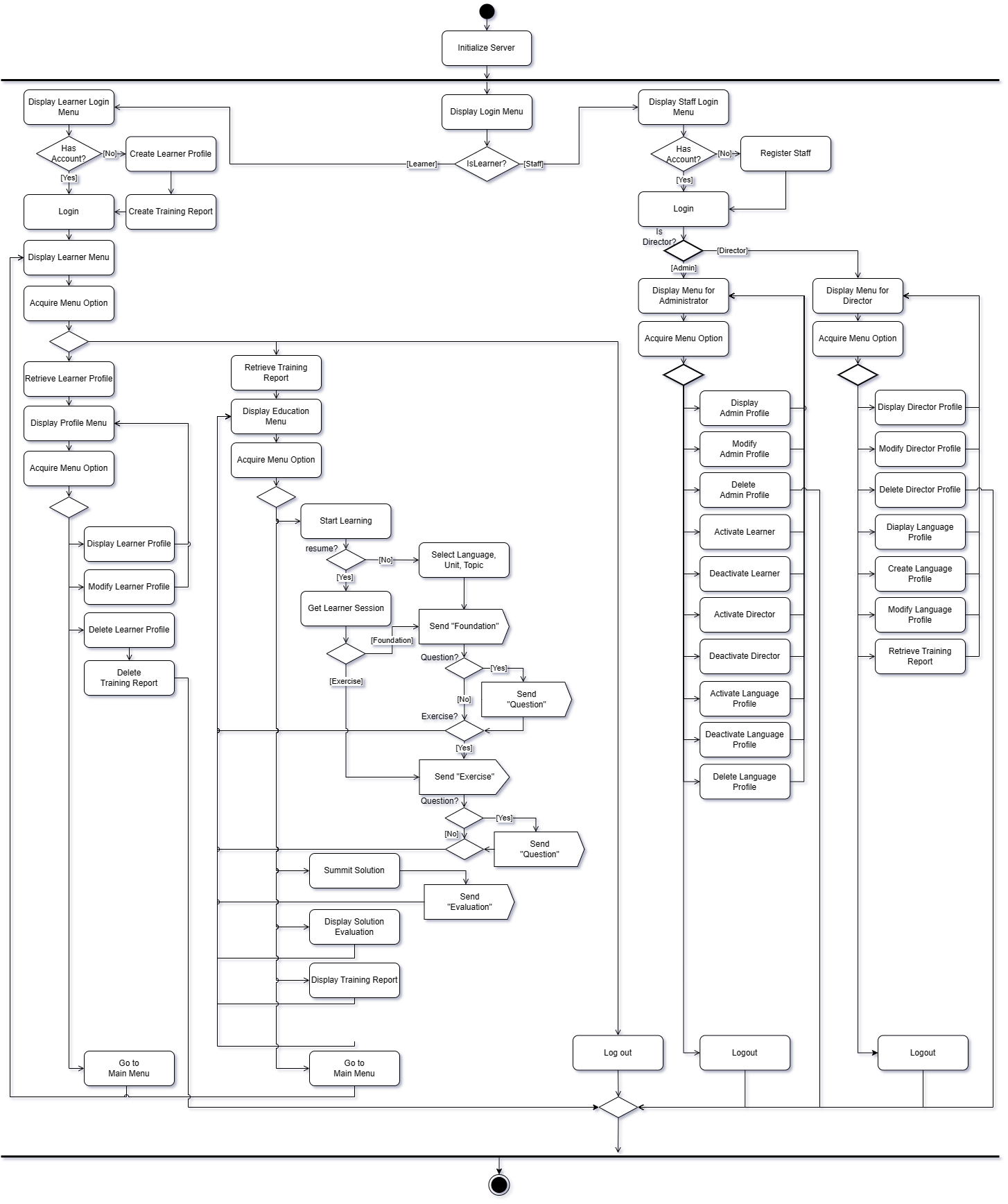


Figure . Refined Control Flow of Explicit Functional Group

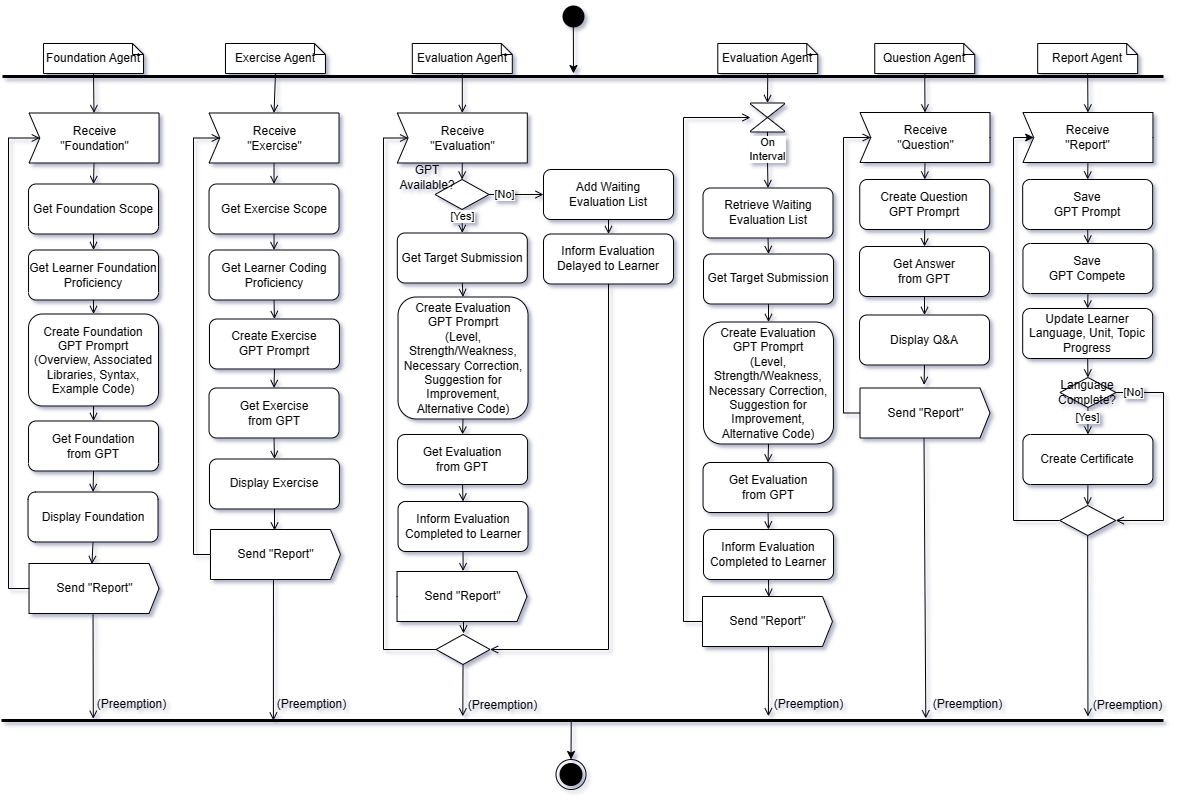


Figure . Refined Control Flow of SW Agent(Event-based, Timer)

### [Step 3] Choosing Element 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 23.

* Use case, SE01. Create GPT Prompt to evaluate.

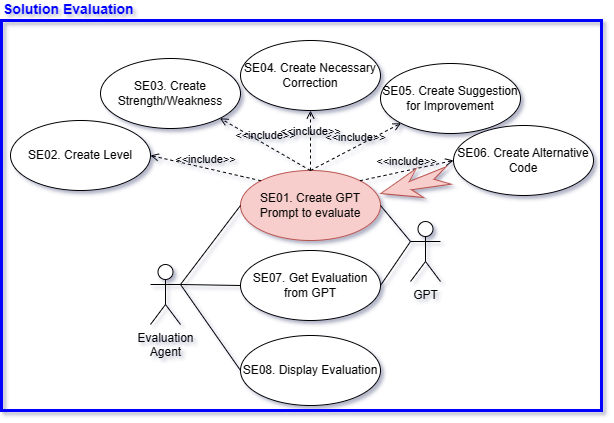


Figure . Target Element for Designing Detailed Control Flow

### [Step 4] Detailed Control Flow for ‘Create GPT Prompt to evaluate.’

* Functionality

This functionality is to make GPT Prompt to evaluate the program code submitted by learners and provide comprehensive feedback on the content submitted. Expected GPT Completion, evaluation should present accuracy, efficiency, compliance with coding standards, and other relevant criteria.

In order to deliver accurate evaluation to learners, GPT Prompt must be generated in detail and the service must be requested through the GPT API to receive the intended evaluation.

* Activity Diagram

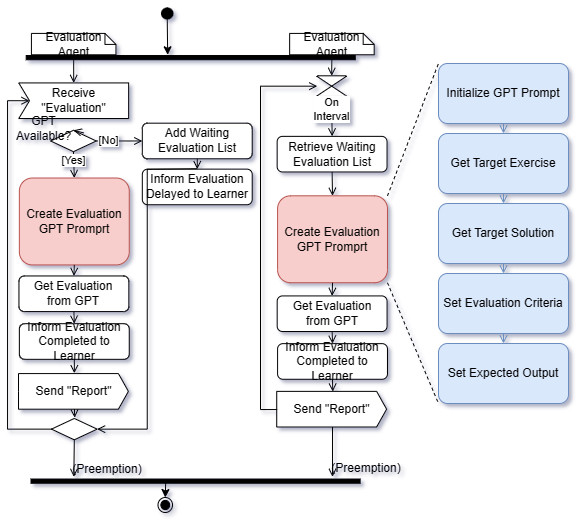


Figure . Detailed Control Flow of Create Evaluation GPT Prompt

* Input
* Target Exercise’s Conditions and Constraints

ex) time or space complexity limit

* Learner’s Target Solution
* Evaluation Criteria

ex) Accuracy, Efficiency, Readability, Exception Handling

* Expected Output

ex) Level, Strength/Weakness, Necessary Correction, Suggestion for Improvement, Alternative Code

* Output
* GPT Prompt

An accurate and detailed GPT Prompt should be created to evaluate the learner's solution.

* Precondition
* GPT is Available.
* Target Exercise is existed.
* Target Solution is existed.
* Begin

//Step 1. Initialize GPT Prompt.

Initialize GPT Prompt.(“Evaluate the learner's solution for exercise based on the following criteria.”)

//Step 2. Set Target Exercise’s Conditions & Constraints

GPT Prompt Add(Get Target Exercise)

GPT Prompt Add(Conditions & Constraints)

//Step 3. Set Target Solution

GPT Prompt Add(Get Target Solution)

//Step 4. Set Evaluation Criteria

GPT Prompt Add(“Please evaluate the level according to the Criteria below. The range is from 1 to 10, and the higher the level, the closer it is to 10.”)

GPT Prompt Add(“Accuracy, Efficiency, Readability”)

//Step 5. Set Expected Output

GPT Prompt Add(“Please make a Complete including the contents below.”)

GPT Prompt Add(“Strength/Weakness, Necessary Correction, Suggestion for Improvement, Alternative Solution)

End

* Postcondition

Use the service through GPT API with the generated GPT Prompt

* Example

Evaluate the learner's solution for exercise based on the following criteria.

[Exercise]

Print out Hello world.

[Condition]

Use standard input and output

[Constraints]

N/A

[Solution]

Print(“Hello World”)

[Criteria]

Evaluate the Accuracy Level. Range is 0 to 10 and the Higher it is, the closer it is to 10.

Evaluate the Efficiency Level. Range is 0 to 10 and the Higher it is, the closer it is to 10.

Evaluate the Readability Level. Range is 0 to 10 and the Higher it is, the closer it is to 10.

[Expected Output]

Please make a Complete including the contents below.

Strength/Weakness, Necessary Correction, Suggestion for Improvement, Alternative Solution.

## 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

* Programming Languages to support

Coding Trainer System will initially be available for Python implementations.

* Python
* Execution Environment

The execution environment will depend on the programming language.

* For Python implementation
* Operating System
* Ubuntu
* The Coding Trainer Server will be realized with a number of replicated medium-sized servers. The number of server instances can initially be determined with an estimation of peak-time load. If the number of users increases a lot, it will be expanded.
* The Dispatcher tier will be realized with a single high-end server since its functionality is not complex.

### [Step 2] Define Nodes

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

* Node 1. Coding Trainer Server
* Hardware Specification
* High-end CPU with a minimum of 10 cores
* Execution Environment
* Operating System: Ubuntu 22.04
* Development Language: Python
* Node 2. Dispatcher
* Hardware Specification
* CPU with Moderate Computing
* Execution Environment
* Operating System: Ubuntu 22.04

### [Step 3] Define Network Connectivity

* Between Web Browser and Coding Trainer Server: HTTP-based network
* Between Server and Archival Server: HTTP-based network

### [Step 4] Define Artifacts to Deploy

* Artifacts for Coding Trainer Server
* Functional Components specified in Functional View-design
* Data Components specified in Information View-design
* Artifacts for Dispatcher
* Functional Components specified in Functional View-design
* Data Components specified in Information View-design

### [Step 5] Allocate Artifacts on Nodes

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

* Deployment for Python version

We use a Deployment Diagram to represent the nodes and artifacts as shown in Figure 25.

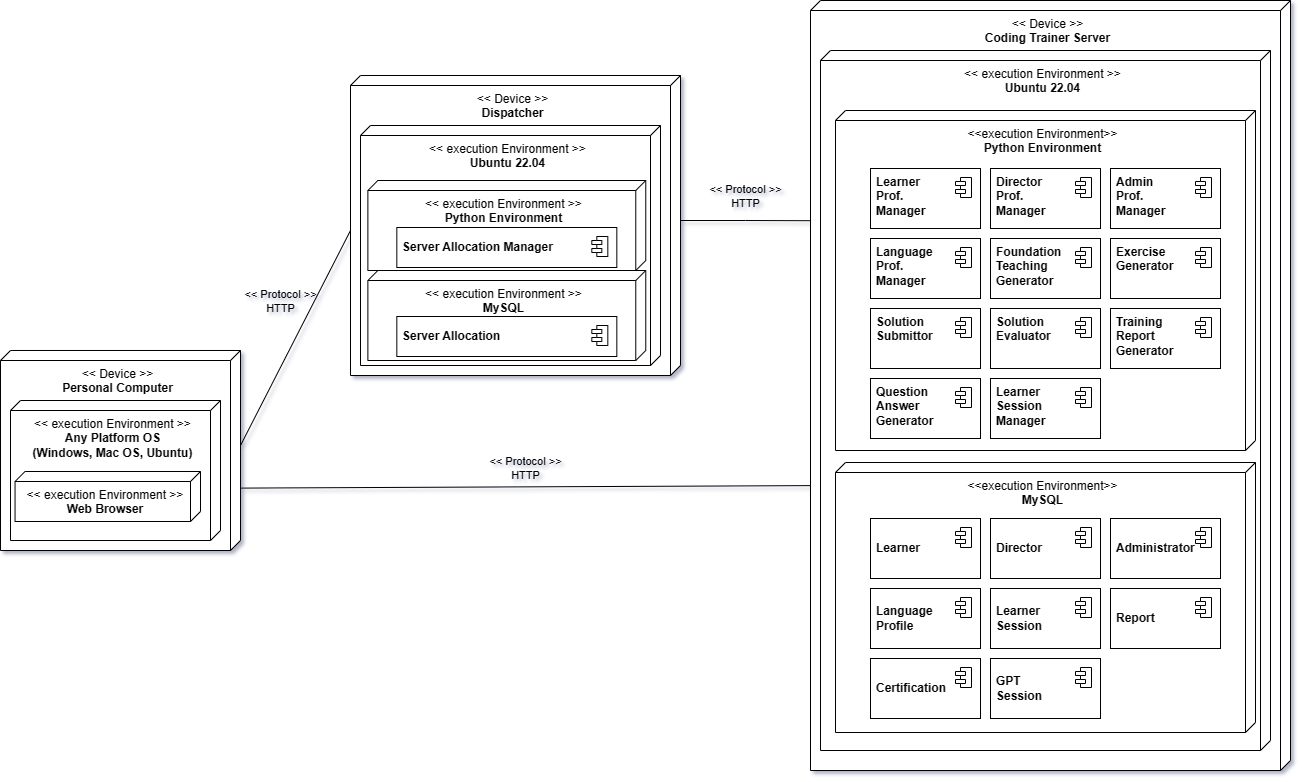


Figure . Deployment Diagram for the Target System

* Deployment for Java version

Omitted in this sample solution.

# 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 High Reliability of the System

The target system provides services potentially to a large number of users, and hence the system should be designed to provide high-level reliability.

By emphasizing reliability in the design process, the system can effectively mitigate potential failures, minimize downtime, handles various faults, recovering efficiently from failures, and enhance the overall user experience.

Reliability in ISO 9126 is defined with three sub-quality attributes, and they should be satisfied by the system.

* Maturity
* Fault Tolerant
* Recoverability

### [Step 1] Identify Facts and Policies

Given the NFR of Platform Reliability, we can reasonably establish the following facts.

* (F1) High Reliability on Coding Trainer Server

Coding Trainer Server can be impossible to use by failure. Failures can be caused by fault on software or physical damage. In the event of a problem with the server, the user experiences great inconvenience. If these cases are repeated, the user loses confidence in the system and is not used.

* (F2) High Reliability on Dispatcher

If a problem occurs in the dispatcher, Users may not be able to access the server side. The Dispatcher does not perform many functions and each function may be processed in a short time.

* (F3) High Reliability on GPT-4 Service

A problem may also occur in the server of the GPT-4 service. They may have tried to secure reliability in their own way, but errors may occur.

* (F4) Occur Software Fault

Coding Trainer system may cause a fault in internal software. This may lead to poor performance or service failure.

* (F5) Server Recovering upon a failure

If the software and dataset are not recovered after server failure, the service may not be provided.

### [Step 2] Define Criteria for Tactics

* (C1) Fault Tolerance of Coding Trainer Server (Relevant to F1)

The server should be fault tolerant. For fault tolerance, three elements should be satisfied: fault detection, diagnosis, and recovery. The system should be operated so that users cannot feel it even in the event of a fault.

* (C2) Distributed processing of service requests (Relevant to F1)

The server should be able to manage so that requests are not concentrated on a specific server when there are multiple physical servers that provide services. It should be possible to maintain the minimum performance by lowering the service latency.

* (C3) Recoverable Dispatcher (Relevant to F2)

Dispatcher should recoverable. Dispatcher fault will not occur much, but if it occurs, the web browser will not be properly assigned to the server. Therefore, it is necessary to recover as soon as possible.

* (C4) Minimize the impact of Failures on GPT-4 Service (Relevant to F3)

If a fault occurs in the GPT-4 Service, recovery of the fault must be made. However, this service is not a service that we can restore like the Coding Trainer Server. In the case of GPT-4 service, it is very important to learner because this is a target system's main business point.

* (C5) Fault Detection and Notification (Relevant to F4)

The server should be able to detect fault and inform to the administrator for recovery.

* (C6) Handling Software Agent Fault (Relevant to F4)

The server should detect and handle software agent fault to restart them to normal operation.

* (C7) System Recovery (Relevant to F5)

The target system should detect and handle corruptions on delivered data and current state to offer normal service.

### [Step 3] Define Candidate Tactics

The following tactics are proposed for the identified criteria.

* (T1) Prepare Passive Redundancy of Server (Relevant to C1)

The coding trainer server uses the passive redundancy tactic to ensure the reliability of the service in the state that the server can no longer operate due to a software failure or physical damage.

Coding Trainer System uses Dispatcher Architecture style for QoS, so two or more servers are being built. By creating an active-passive state between servers, it synchronizes the state and data in the server failure situation so that the ongoing service can be continued. If an error occurs as shown in the Figure 26, the server's status is synchronized and the ongoing service is continued.

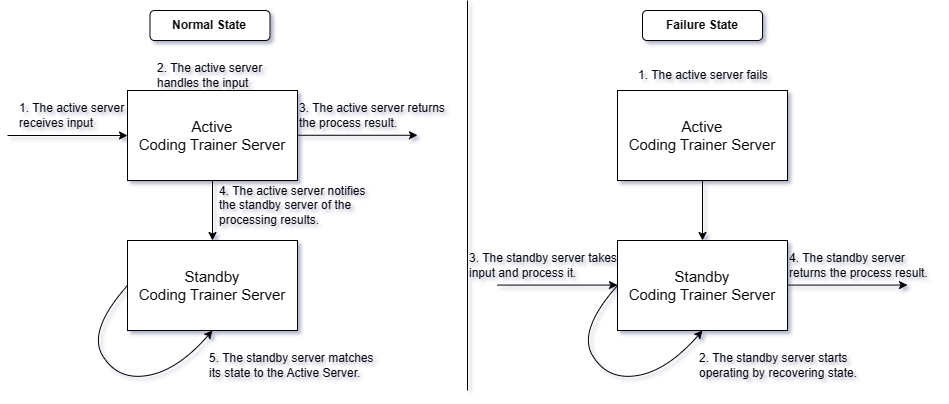


Figure . Passive Redundancy of Server

* (T2) Dispatcher for distributing service request (Relevant to C1, C2)
* Already Applied in View-based Design

This tactic of applying reliability design has also been used during the architecture design for functional, information, and behavior views.

* (T3) Prepare Passive Redundancy of Dispatcher (Relevant to C3)

Since dispatcher does not perform many functions and focus on network transactions, the probability of problems occurring is not high. However, in the event of a problem, if an abnormality occurs in a new user or server, the logic for reallocation does not proceed, causing inconvenience to the user. Therefore, if a problem occurs, replace dispatcher with passive redundancy. In the case of dispatcher, it is not necessary to maintain active redundancy because state re-synchronization does not take long if only the function operates normally.

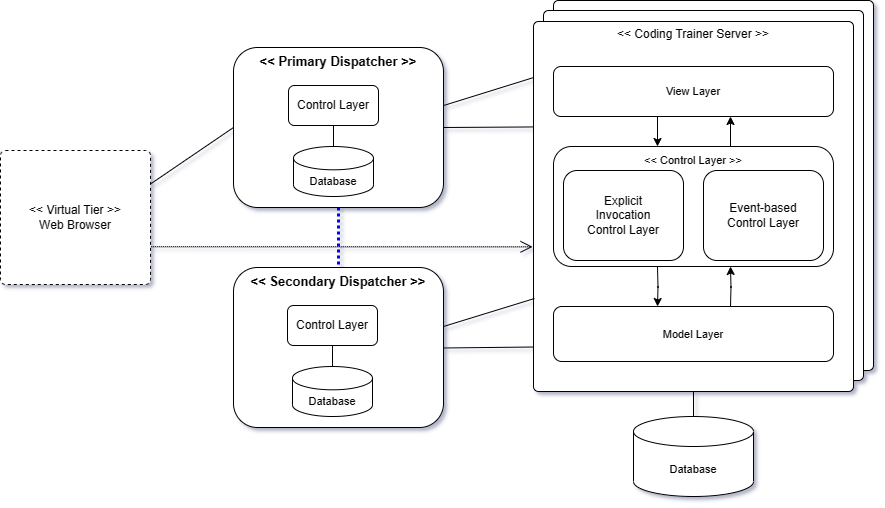


Figure . Passive Redundancy of Dispatcher

* (T4) Prepare Other Service for GPT-4 (Relevant to C4)

When GPT-4 provided by OpenAI is used as the main service and the service becomes unavailable, the system is configured to be linked to other services so that users do not know if GPT-4 is unavailable.

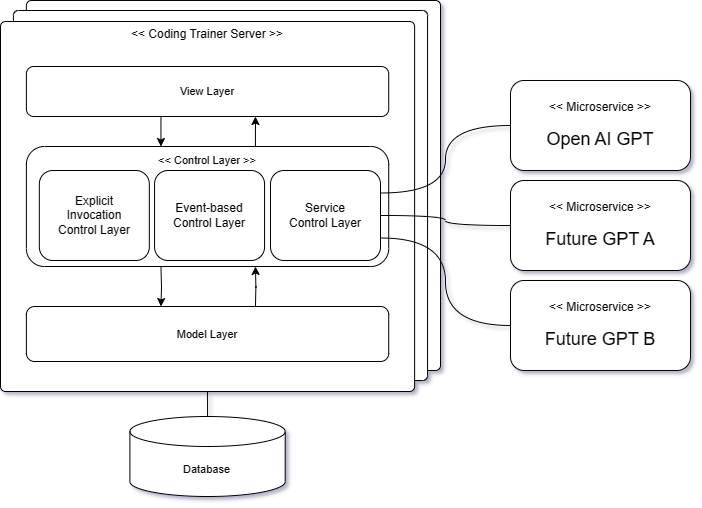


Figure . Apply Microservice Architecture Style

* (T5) Server Monitoring via Dispatcher (Relevant to C5)

For high reliability, it is important to detect a failure in order to overcome the failure that occurs in the coding trainer server. The system can use dispatcher to detect server failures. By applying heartbeat tactic to the dispatcher of the coding trainer system, periodic signals are delivered to the server, and when there is no response to the dispatcher's signal, it is recognized as a server failure, and the server's failure status is notified to the administrator through the screen.

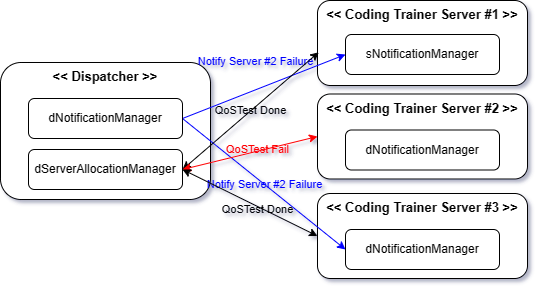


Figure . Server Monitoring via Dispatcher

* (T6) Monitor and Restart S/W Agent via watchdog (Relevant to C6)

Watchdog can monitor if threads and processes are in normal operations. It sends ping to threads and processes periodically and waits their responses. If a thread or process doesn’t respond, watchdog restarts the thread or process. Watchdog itself can also be failed. Therefore, divide watchdog into two threads. The first watchdog thread monitors other threads and processes including the second watchdog thread. The second watchdog thread only checks if the first watchdog thread alive. If the second watchdog thread cannot receive ping for a certain period, it restarts the first thread.

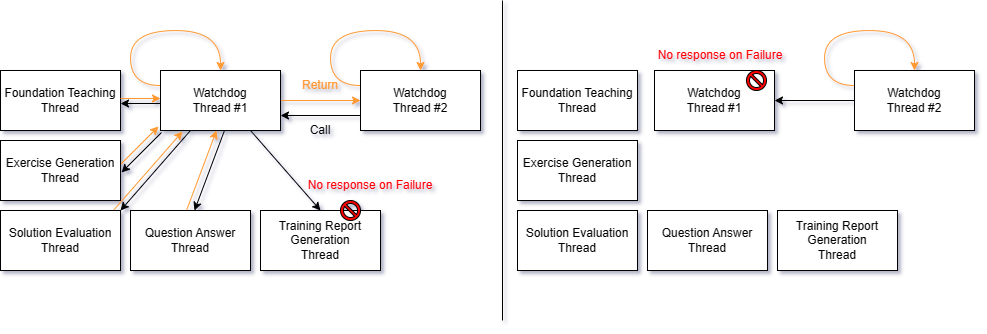


Figure . Monitor and Restart S/W Agent via watchdog

* (T7) Data Duplication and Shared Storage (Relevant to C7)

The system is configured to duplication database server for high reliability and share storage to minimize data integrity issues. Since there are two database servers that access the storage, if there is a problem with the active database, the service can be continued through the standby database.

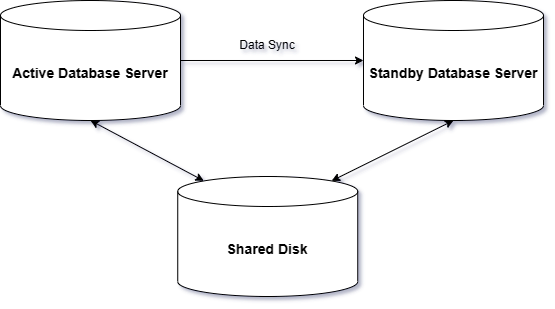


Figure . Data Server Duplication & Shared Repository

### [Step 4] Evaluate Candidate Tactics

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Tactics** | **Y/N** | **Justification** |
| T1 | Prepare Passive Redundancy of Server | Y | Benefit) Even if the server fails, the service can continue to be provided through Redundancy Server.  Cost) Redundancy server construction and operation cost increases.  Decision) This is a necessary strategy for high reliability. |
| T2 | Dispatcher for distributing service request | Y | Benefit) When operating multiple servers, efficient resource management is possible and availability can be increased.  Cost) Runtime and operating cost increase  Decision) The cost-benefit ratio is large. |
| T3 | Prepare Passive Redundancy of Dispatcher | Y | Benefit) Sufficient reliability of the dispatcher can be secured through this tactic.  Cost) There is a cost of constructing Passive Redundancy.  Decision) This tactic can be implemented with a minimal cost, but it provides sufficient reliability improvement. |
| T4 | Prepare Other Service for GPT-4 | Y | Benefit) Even if there is a problem with the GPT service, the service can be provided without a big problem.  Cost) Development costs increase as developers need to interface GPT components for interworking with other services.  Decision) Although development costs increase somewhat, redundancy of core services in terms of reliability can significantly reduce downtime from a system operation perspective. |
| T5 | Server Monitoring via Dispatcher | Y | Benefit) Servers in Failure state can be quickly detected and recovered.  Cost) Because we utilize dispatcher, the cost is minimal.  Decision) The tactic can be implemented with a minimal cost, but the benefit is effective. |
| T6 | Monitor and Restart S/W Agent via watchdog | Y | Benefit) It can detect and recover software agent failure, even watchdog itself is failed.  Cost) It needs extra cost to send pings and respond, but it is minimal with appropriate ping period.  Decision) The tactic can be implemented with a minimal cost, but the benefit is effective. |
| T7 | Data Duplication and Shared Storage | Y | Benefit) This tactic can solve data integrity problems that occur when data exists on each server and data loss problems due to server failure.  Cost) It takes additional core to build and manage the data server.  Decision) It is a necessary strategy in terms of recoverability among the quality attributes of reliability. |

### [Step 5] Analyze Impacts of Tactics

We analyze the impacts of each selected tactic.

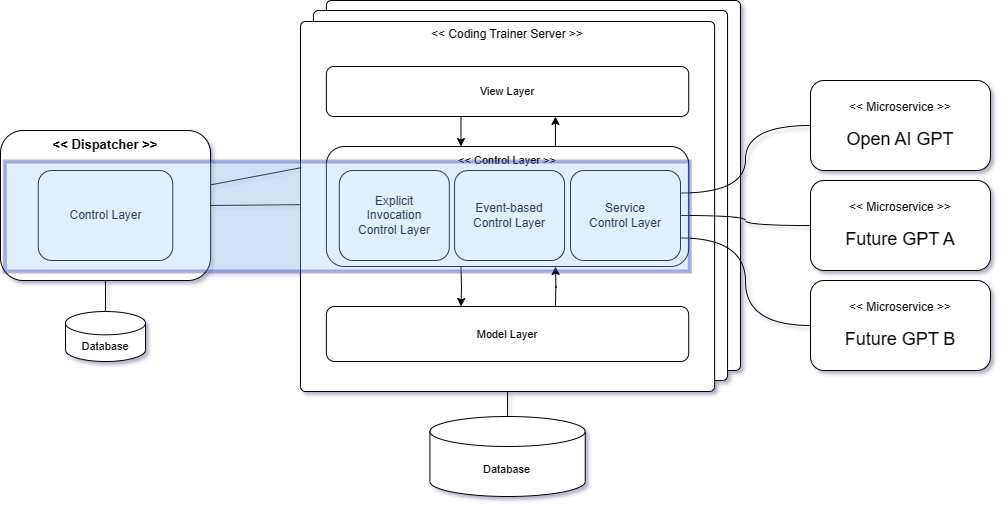
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Tactics** | **Functional View** | **Information View** | **Behavior View** | **Deployment View** |
| T1 | Prepare Passive Redundancy of Server | Add functionality for passive node operation. |  | Add control flow for passive node operation. |  |
| T2 | Dispatcher for distributing service request | There is nothing to reflect in Views. | | | |
| T3 | Prepare Passive Redundancy of Dispatcher | Add a functionality to heart heat between dispatcher. |  | Add control flow to manage multiple dispatcher | Add secondary Dispatcher |
| T4 | Prepare Other Service for GPT-4 | Change the assignment of feature components to placeholders |  |  |  |
| T5 | Server Monitoring via Dispatcher | Add a functional component to detect fault |  | Add fault detect control flow. |  |
| T6 | Monitor and Restart S/W Agent via watchdog | Add a functional component to monitor s/w agent |  | Add monitoring control flow. |  |
| T7 | Data Duplication and Shared Storage | Add functional components for Data Server Operation |  | Add control flow component for Data Server Operation. | Update Data Server |

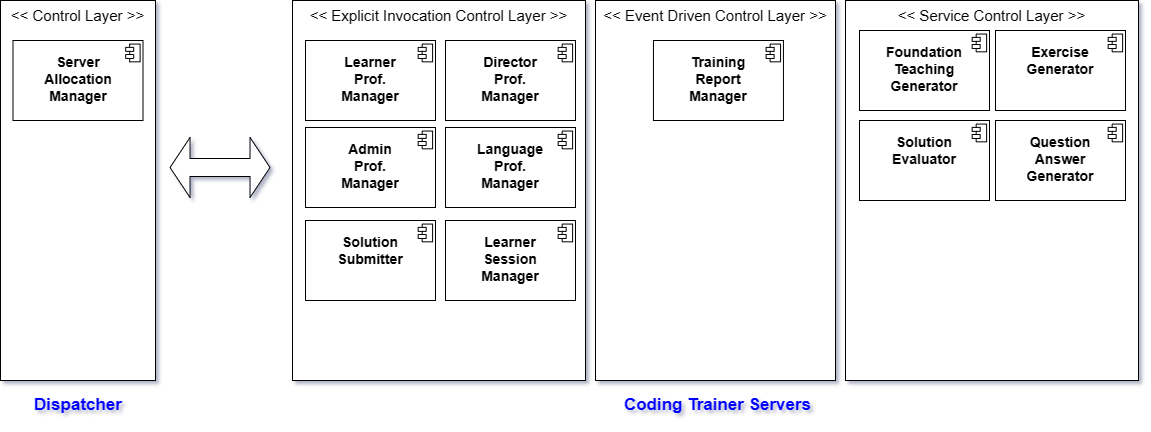
### [Step 6] Apply Tactics

We design the detailed control flow for selected tactics. In CEP, we choose only one tactic.

* Applying T4) Prepare Other Service for GPT-4
* Allocate Functional Components to Placeholders

There is no addition of the existing functional component, but as the service control layer is added, the placement change occurs for components using GPT-4 Service as shown in the figure below.





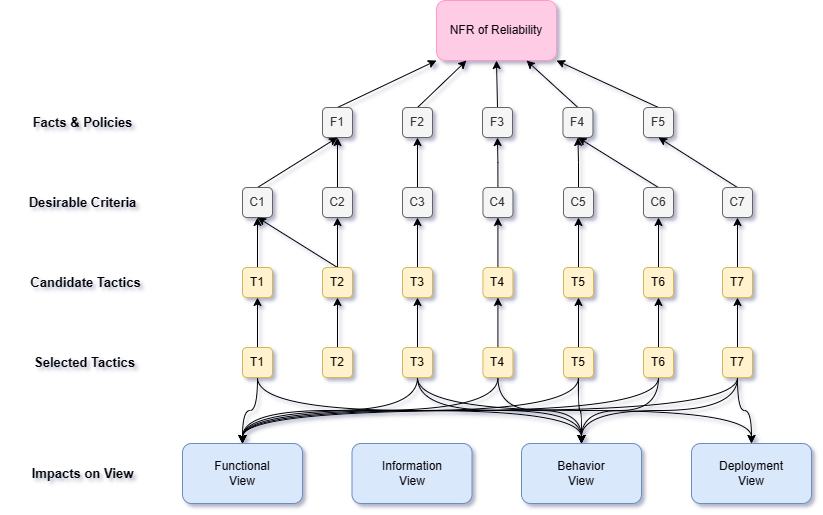
* Functional Component interfaces for Microservice

We choose the following components since their interfaces should be well-defined to effectively communicate microservices such as GPT.

* Foundation Teaching Generator
* Exercise Generator
* Solution Evaluator
* Question Answer Generator

### [Step 7] Validate Conformance

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.

## Design for NFR-2. High Personalization of Instruction and Coding Exercises

Learning can be maximally effective when training is delivered in a personalized manner. To ensure this, the system should be designed to offer a high level of personalization across three key aspects:

* Personalizing the Instruction

The instructions for explaining the language constructs should be customized for the level of details, the level of compactness, and types of examples for each learner.

* Personalizing the Coding Exercise

The coding exercise problems should be designed with consideration for the proficiency level of each learner, moving away from a 'one-size-fits-all' approach. Instead, the problems should be generalized to accommodate learners at different skill levels.

By considering the proficiency level of learners, the system can generate exercise problems that align with their current abilities and knowledge. This ensures that learners are appropriately challenged without feeling overwhelmed or bored.

### [Step 1] Identify Facts and Policies

Given the NFR of Platform Functionality, we can reasonably establish the following facts.

* (F1) Personalized Learning

Learners are provided with the appropriate content necessary for learning and proceed with language learning efficiently.

* (F2) Diversity of Learners

Learners have various differences such as job, age, academic background, major, and language learning experience.

* (F3) Learning Progress on Language

The learner has a learning history of the language to learn.

* (F4) Learner’s Level for Language

Learners have a level of language through their learning history.

* (F5) Personalization in Foundation Teaching

In the Topic of the language, basic explanation content, grammar, Syntax, etc. may be difficult to personalize. However, examples in language learning are one of the important explanatory materials, which is suitable for personalization.

* Degree of Overview
* Number of Examples
* Domain of Example
* Degree of Annotation Detail
* (F6) Personalization in Exercise Generation

For exercise to evaluate the learned content, various exercises can be asked for each learner.

* Number of Exercise
* Difficulty of Exercise
* Domain of Exercise
* (F7) Questions Entered by Learners

Learners ask questions about necessary contents such as contents that they do not understand and difficult contents during learning. This can be seen as the learner entering personalized content.

### [Step 2] Define Criteria for Tactics

* (C1) Learner’s Evaluation (Relevant to F1)

The system should be able to be evaluated for providing appropriate learning to learners. The system should be able to adjust the level of learning content according to the learner's evaluation.

* (C2) Account for Learner Background (Relevant to F2)

The system should be able to acquire and utilize learners' personal information. For example, depending on the job, domain knowledge should be used for learning, and learning methods should be changed according to academic background. Using each domain knowledge may improve understanding, and the learning methods of major college students and elementary school students cannot be the same. In the case of elementary school students, it should be easier to inform, and in the case of major college students, they should be able to explain professional content in detail without being too easy.

* (C3) Determining the Level of Learners (relevant to F3, F4)

The system should be able to determine the level of the learner through the learner's learning history. It should be possible to evaluate in stages 1 to 10 with quantitative evaluation, not qualitative evaluation.

* (C4) Degree of Overview’s Detail (relevant to F5)

The system should be able to adjust the level of detail of basic learning materials according to learners. For example, majors will need in-depth basic learning materials, and beginners with low academic background should use figurative expressions to make them easily accessible so that they can be interested in learning.

* (C5) Number of examples by Learner (Relevant to F5)

The system should be able to adjust the number of examples according to the level of learners. For majors or learners with previous language learning experience, only a simple example of the Topic is needed, but for beginners, various examples of the Topic should be provided to make it easier to understand.

* (C6) Degree of example’s annotation (Relevant to F5)

The system should be adjusted to show the annotations of the examples in detail or simply according to the learner's level.

* (C7) Number of exercises by Learner (Relevant to F6)

The system should be able to adjust the number of exercises according to the learner. If Learners are a senior, the system can evaluate one or two exercises of appropriate difficulty, and in the case of beginners, the system should be able to provide several questions of various difficulty.

* (C8) Degree of exercise’s difficulty (Relevant to F6)

The system should be able to adjust the degree of difficulty of the exercise according to the learner. In other words, the system should be provided to provide a more challenging solution by providing various difficulties for beginners and evaluating it as a problem of advanced difficulty.

* (C9) Difference of Domain Knowledge (Relevant to F5, F6)

The system should use the learner's domain knowledge to provide exercise, which should provide personalized exercise.

* (C10) Personalized Questions and Answers (Relevant to F7)

The system should provide instruction and exercise by reflecting learners' questions and answers.

### [Step 3] Define Candidate Tactics

The following tactics are proposed for the identified criteria.

* (T1) Learner's System Assessment Process (Relevant to C1)

By adding a procedure for evaluating the content provided by the system, the degree of personalization that the user thinks may be obtained from the system. The system will be able to reflect the evaluation and provide learning content according to the degree of personalization felt by the user.

The Learner Feedback procedure was added by extracting part of the Activity diagram as shown in Figure 32 to proceed with the System Feedback after each activity item.

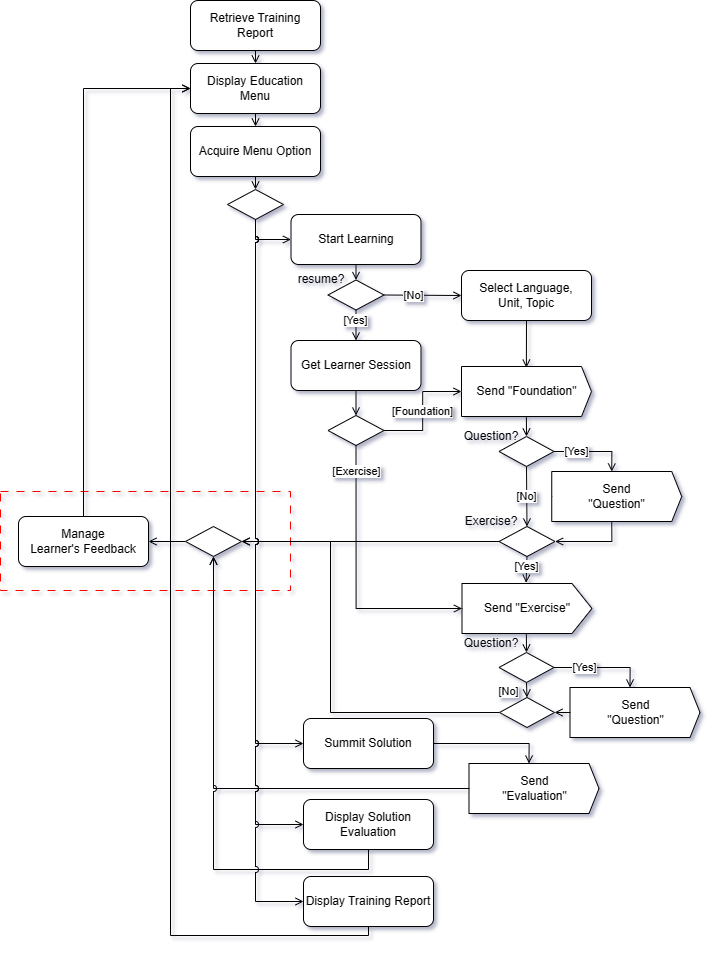
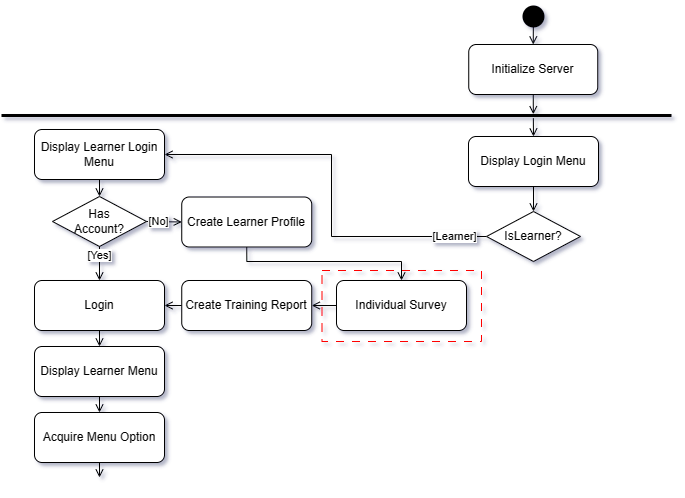


Figure . Add Managing Learner’s Feedback in Activity Diagram

* (T2) Individual Survey at Registration (Relevant to C2, C9)

It is determined that it is difficult to secure background information necessary for the learner's personalization only with simple subscription information, so we will conduct a survey on the background. Information necessary for personalization, such as academic background, major, job, programming language that was learned, and programming language learning period, is secured through a survey.



* (T3) Pre-test before programming language learning (Relevant to C2, C3)

Prior to starting the language chosen by the learner, a pre-test must be conducted to evaluate the learner's level so that the details and summary of the learning content can be determined. In addition, even if there is no prior evaluation, the learner's information accumulates as the unit progresses, and the level is evaluated, allowing personalization, but the preliminary test was selected as a tactic to proceed with personalization from the first unit.

* (T4) Make Learner’s Language Progress Table (Relevant to C3)

*ProgressTable* is a two-dimensional table representing progresses of all the units. This table is given the value of ‘F’ initially, meaning the learning activities are not performed yet. Each entry in the table is given a Boolean Value, where ‘T’ indicates the activity in the given unit has been completed and ‘F’ indicates ‘Not Yet Completed’.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Activity  Units | Foundation | Exercises | Submissions | Evaluation | Score |
| Unit1 | T | T | T |  |  |
| Unit2 | T | T |  |  |  |
| Unit3 | T | T | T | T | 90% |
| Unit4 | T | T |  |  |  |
| ... |  |  |  |  |  |

The system can determine the level of Learner based on the table above.

* (T5) Creating learning contents that reflect learners' personal information (Relevant to All)

By integrating the tactics mentioned above, it will be possible to acquire learners' personal information and provide personalized learning content based on this.

If the learner's domain knowledge, differences in content details according to academic background and major, and the learner's level is low, the number of examples is provided to help understand, and if the level is high, the learner can select and solve various problems.

### [Step 4] Evaluate Candidate Tactics

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Tactics** | **Y/N** | **Justification** |
| T1 | Learner's System Assessment Process | Y | Benefit) Through the learner's system evaluation, it is possible to check whether appropriate contents have been provided to the system.  Cost) Additional costs can be calculated by the fact that interaction may enter learners in the middle and that developers need to add User Interface, but the impact is small.  Decision) This is a necessary strategy for high functionality. |
| T2 | Individual Survey at Registration | Y | Benefit) It is possible to obtain the learner's personal information, so that personalized learning information can be delivered to the learner.  Cost) Security-related costs for personal information may increase.  Decision) The use of personal information is essential to provide a high level of personalized learning services. |
| T3 | Pre-test before programming language learning | Y | Benefit) Since the first learning is performed, the level of the learner is confirmed, and thus more suitable learning may be provided.  Cost) It is necessary to develop additional functions because it is necessary to provide preliminary tests for each language.  Decision) If the template of the pre-test for each language can only be secured through GPT and displayed, it is considered a necessary item to provide more personalized learning to learners from the beginning. |
| T4 | Make Learner’s Language Progress Table | Y | Benefit) It is expected that the learner's level can be evaluated more accurately by managing the learner's learning history as a table.  Cost) It is analyzed that no additional costs are incurred.  Decision) It is a tactic that provides convenience in system operation. |
| T5 | Creating learning contents that reflect learners' personal information | Y | Benefit) It is possible to provide optimized learning services for each individual by making the most of personal information, and users can learn efficiently by experiencing one-on-one learning services.  Cost) There is a privacy issue, so the cost of protection is high.  Decision) Although the cost of information protection is largely incurred, it is a necessary tactic as a core service part. |

### [Step 5] Analyze Impacts of Tactics

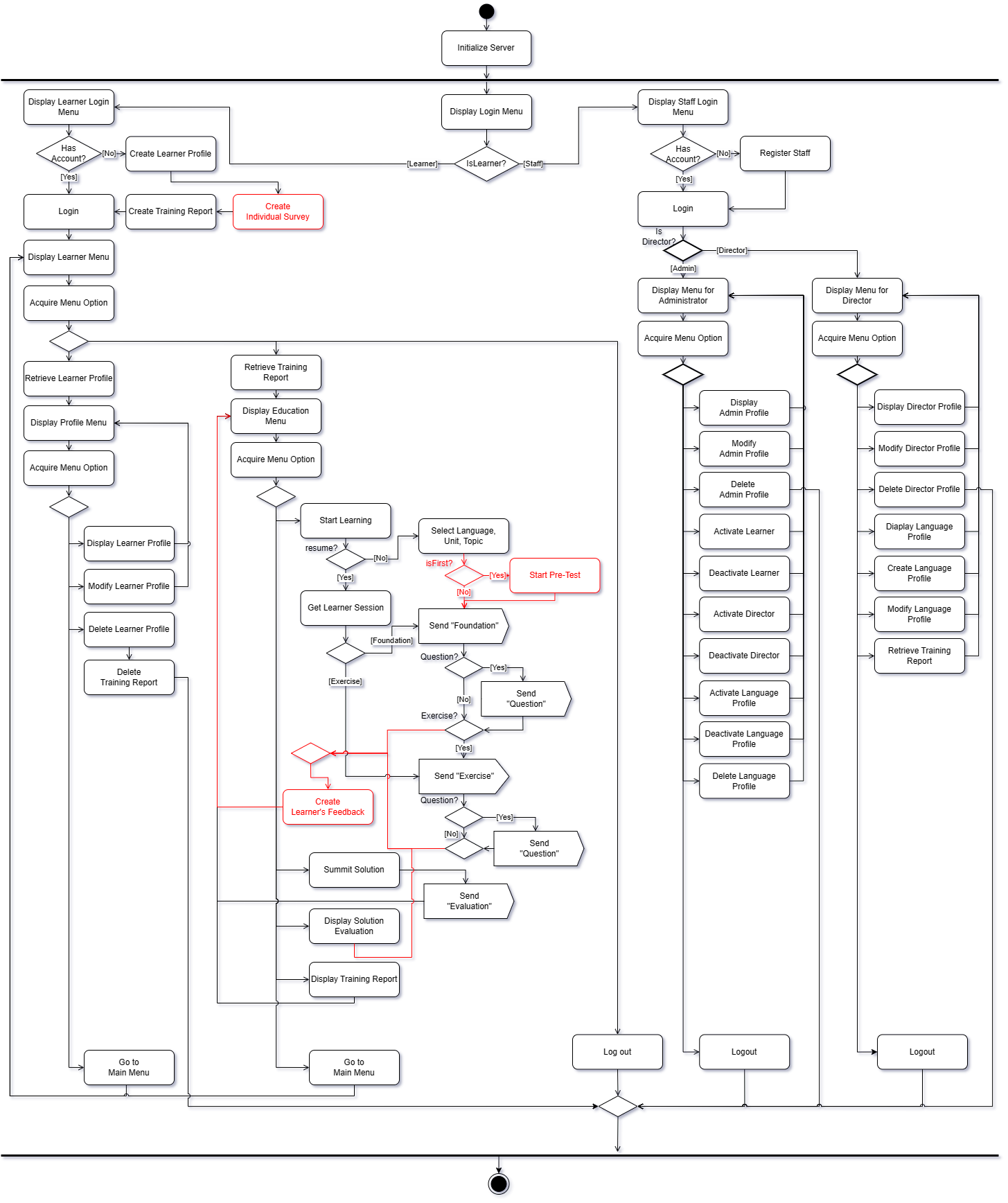
We analyze the impacts of each selected tactic.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Tactics** | **Functional View** | **Information View** | **Behavior View** | **Deployment View** |
| T1 | Learner's System Assessment Process | Add functionality for System Assessment. | Add Learner Feedback Component. | Add control flow for System Assessment Process. |  |
| T2 | Individual Survey at Registration | Add functionality for Individual Survey. | Add Learner Survey Component. | Add control flow for Individual Survey. |  |
| T3 | Pre-test before programming language learning | Add a functionality to Manage Pre-Test | Add Language Pre-Test Component | Add control flow to manage Pre-Test. |  |
| T4 | Make Learner’s Language Progress Table | The logic of updating Progress was already applied. | | | |
| T5 | Creating learning contents that reflect learners' personal information | Add a functionality to get Learner’s Profile. |  | Add detailed control flow to generate foundation and exercise. |  |

### [Step 6] Apply Tactics

We design the detailed control flow for selected tactics. In CEP, we choose only one tactic.

* Applying T5) Creating learning contents that reflect learners' personal information
* Refine control flow of the system



* Algorithm of ‘Foundation Teaching Generator’

This algorithm is invoked with an arrival of the event, ‘Foundation’.

* INPUT

Learner Profile (include survey)

Training Report (include learner session, pre-test result)

ProgressTable

Learner Feedback

* OUTPUT

Foundation Material for the given scope

Response from GPT

Refined Response

* BEGIN

*// Step 1. Acquire learner’s private information for generating the prompt.*

LANG ← Programming Language to learn

If (isFirst)

PROFICIENCY ← Scores in Pre-Test

else

PROFICIENCY ← Average of Scores in ProgressTable

*// Step 2. Reflect learner information*

LEARNER ← “Please process the following request by reflecting the learner information below.

1. Learner Feedback
2. Learner Level (PROFICIENCY)
3. Academic Background
4. Major
5. Domain Knowledge
6. Previous Learning Experience”

*// Step 3. Generate the prompt.*

PROMPT ← “A learner<LEARNER> wants to learn about <topics>of the programming language<LANG>.”

PROMPT += “Generate an instructional material for the given topics in the following order:

1. Instruction to the given topics,
2. Syntax of the relevant language constructs,
3. Example of using the syntax, and
4. Additional guidelines for utilizing the language constructs.”

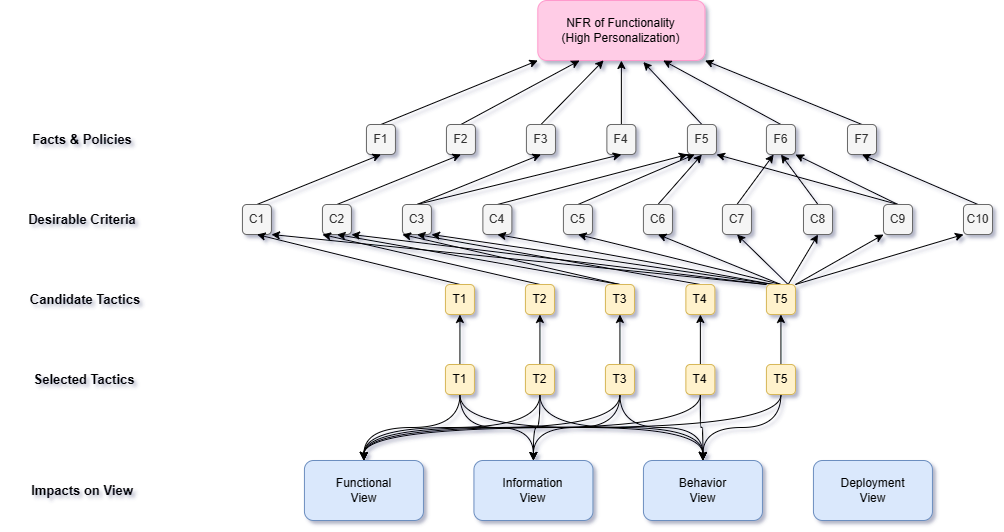
*// Step 4. Add Personalization Policy*

PROMPT += “A learner<LEARNER> wants to personalize the items below:

1. Instruction depth : The more beginners, the shallower the depth,
2. Number of Examples : If beginner, more than 5 examples,
3. Degree of Annotation detail : If beginner, expression of annotations in detail”

### [Step 7] Validate Conformance

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.



# Activity 6. Architecture Validation

* Steps of ATAM



## [Step 1] Presenting ATAM

This step is to present the ATAM process to the assembled stakeholders, typically customer representatives, the architect or architecture team, user representatives, maintainers, administrators, managers, testers, integrators, etc.

* Already Presented

## [Step 2] Presenting Business Drivers

This step is for the project manager to present what business goals are motivating the development effort and hence what will be the primary architectural drivers.

* Already Specified in the CEP SRS.
* Already Specified in this AD

The architecture drivers are further specified in chapter 2 of ‘Activity 1. Architectural Requirement Refinement’

## [Step 3] Presenting the Architecture

This step is for the architect to present the designed architecture, focusing on how the architecture addresses the business drivers.

* Already Presented in this AD.
* Chapter 4 of Schematic Architecture
* Chapter 5 for View-specific Architecture Design
* Chapter 6 for NFR-specific Architecture Design

## [Step 4] Identifying the Architectural Approaches

This step is for the architect to specific architectural approaches in the AD.

* Already Presented in this AD.
* Specific Design Decisions in Chapter 5 for View-specific Architecture Design
* Specific Design Decisions in Chapter 6 for NFR-specific Architecture Design

## [Step 5] Generating Quality Attribute Tree

This step is for the architect to define the core business and technical requirements of the system and map them to an appropriate architectural property. This is done by elicitating the quality factors that comprise system “utility” (performance, availability, security, modifiability, etc.), specify down to the level of scenarios, annotated with stimuli and responses, and prioritized.

A scenario is a specific situation or use case that demonstrates how the architecture will perform under different conditions. Each scenario should be specific and detailed, describing the actions of the user or system and the expected outcome.

* Use Case Scenarios
* Growth Scenarios
* Exploratory Scenarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Quality** | **Refinements** | **Scenarios** | **Importance** | **Difficulty** |
| Reliability | NFR-1. High Reliability of the System | NFR-1-1. Evaluate the availability of server operations should be at least 99.9% (in the event of a failure, the average recovery time is limited to less than an hour) | High | Medium |
| NFR-1-2. Evaluate load balancing, How well the system distributes its work by dispatcher. | High | Medium |
| NFR-1-3. Evaluate GPT Services reliability with average return time GPT Prompt Complete. | High | Medium |
| NFR-1-4. Evaluate how smoothly it changes to the Session state before the error occurs | High | Medium |
| NFR-1-5. Evaluate how quickly a single thread detects, restarts, or recovers when it fails | High | Medium |
| NFR-1-6. Evaluate whether consistency with previous data is maintained accurately in the event of an error | High | Medium |
| Functionality | NFR-2. High Personalization of Instruction and Coding Exercises | NFR-2-1. Evaluate how personalized learning is delivered | High | Medium |
| NFR-2-2. Evaluation of learners' satisfaction with learning content | High | Low |
| NFR-2-3. Evaluate whether learning content is provided considering the individual's background | High | Medium |
| NFR-2-4. Evaluate whether there was a better learning effect than the existing learning method | High | High |

Based on this analysis, prioritize the factors.

## [Step 6] Analyzing Architectural Approaches

This step is to analyze the architectural approaches that address those factors are elicited and analyzed, based upon the high-priority factors identified in Step 5. Identify architectural risks, sensitivity points, and tradeoff points.

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 the availability of server operations should be at least 99.9% (in the event of a failure, the average recovery time is limited to less than an hour) | | | |
| **Attribute** | Reliability | | | |
| **Environment** | Normal Operation. | | | |
| **Stimulus** | GPT-4 service has become unavailable. | | | |
| **Architectural Decision** | **Sensitivity** | **Trade-Off** | **Risk** | **Nonrisk** |
| D1. Microservice | S1 | T1 | R1 |  |
| D2. Monitoring SW Agent and Restarting | S2 | T2 |  |  |
| D3. Monitoring Server | S3 | T3 |  |  |
| D4. Dispatcher | S4 |  |  | NR1 |
| D5. Data Duplication & Shared Storage | S5 | T4 | R2 |  |
| **Reasoning** | The decisions are made for meeting this scenario since the chosen decisions are commonly used for improving reliability. | | | |
| **Architectural Diagram** | Refer to refined component diagram in 6.1. | | | |

* Sensitivity Points
* S1. Impact of target systems due to failure of other services
* S2. Importance of the type of item to monitor and restart.
* S3. Efficient handling of delay in response to monitor due to other service loads.
* S4. Possibility of failure on dispatcher and number of computing servers.
* S5. Effectiveness of data synchronization between data servers.
* Trade-off
* T1. Reliability (+) vs. Accuracy (-): The system reliability can be increased by using the default service and using the backup service when the default service is unavailable, but it is necessary to determine whether the backup service delivers the same GPT Complete as the default service.
* T2. Reliability (+) vs. Performance (-): The process of monitoring and restarting the thread may affect the performance of service.
* T3. Reliability (+) vs. Performance (-): Additional load to monitor the servers and notify on server failure information.
* T4. Reliability (+) vs. Performance (-): The cost of duplicating and managing data servers can affect performance.
* Risk
* R1. There are no replaceable GPT services available yet. We need to find out about the GPT service that will be developed in the future.
* R2. Data server redundancy may cause data loss and risk of integrity in the process of data processing and synchronization between servers.
* Non-risk
* NR1. Dispatcher has been sufficiently validated for load balancing operation. Load balancing is not a problem as long as there are enough available resources deployed.

## [Step 7] Brainstorming and Prioritizing Scenarios

This step is to elicit a set of scenarios from the entire group of stakeholders and prioritize the scenarios via a voting process.

* List of scenarios collected by all stakeholders (i.e. system managers, production managers, workers, and clients)
* About 6 scenarios are additionally acquired for evaluating high reliability of the *Coding Trainer System*. Most scenarios are gathered from system managers and production managers.
* Need to elaborate scenarios for NFR-1-2 to NFR-1-6. The current scenarios are identified at a coarse-grained level. Finer-grained details are needed.
* Measures how effectively a system can recover and reliably operate in the event of a failure. It also measures how complex it becomes to apply that technology.
* About 4 scenarios are additionally acquired for evaluating high personalization of the *Coding Trainer System*. Most scenarios are gathered from workers and production managers.
* Need to elaborate scenarios for NFR-2-1, NFR-2-4. The current scenarios are identified at a coarse-grained level. Finer-grained details are needed.
* Measures whether the system provides personalized content and enables efficient learning compared to traditional learning methods.

## [Step 8] Analyzing Architectural Approaches

This step reiterates step 6, but here the highly ranked scenarios from Step 7 are considered to be test cases for the analysis of the architectural approaches determined thus far. These test case scenarios may uncover additional architectural approaches, risks, sensitivity points, and tradeoff points which are then documented.

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

This step I sto for the ATAM team to present the findings to the assembled stakeholders and potentially write a report detailing this information along with any proposed mitigation strategies.

* The evaluation team concludes the following results;
* Concerns on the functionality, specifically on high reliability of the *Coding Trainer System*, need to be re-considered.
* The main point of the coding trainer system is to use GPT. There are not many partners who provide GPT services yet, so more partners will be needed to operate the system reliably.
* Concerns on the functionality, specifically on high personalization of the *Coding Trainer System*, need to be re-considered.
* Personal information must be utilized to provide personalized learning of the coding trainer system. Additional technical measures and legal reviews for privacy protection should be carried out.

# 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 ⮘**