Architecture Description

GPT-Assisted Coding Trainer

Software System for Training Coding with GPT Assistance

June 2023

Revision History

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

삼성전자 첨단기술아카데미

TABLE OF CONTENTS

9

LIST OF FIGURES

LIST OF TABLES

Architecture Description of GPT-Assisted Coding Trainer

1. Introduction

1.1. 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.

1.2. System of Interest

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

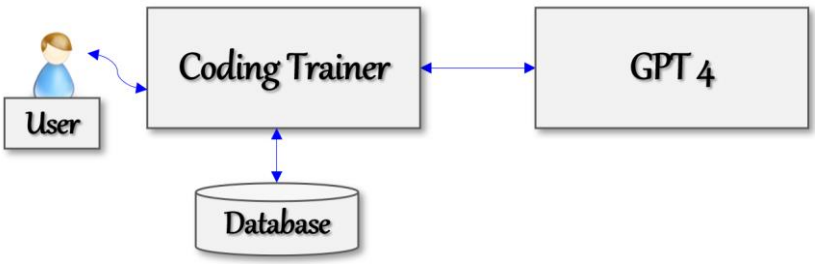
The GPT-Assisted Coding Trainer takes advantage of the latest advancements in GPT language modeling and artificial intelligence to create an immersive and effective learning environment for programming languages.

Once a target programming language is chosen, the training is conducted in the following order:

- Step 1: Learning the Foundation of current Unit
- Step 2: Solving Coding Exercises
- Step 3: Evaluating Program Codes
- Step 4: Repeating Steps 1 through 3

The overall goal is for learners to successfully complete all learning units of the chosen language by passing the coding exercises, demonstrating their understanding and proficiency in the programming language.

The target system is deployed as a web server, which subscribes to GPT as shown in the figure.



1.3. Definitions, Acronyms, and Abbreviations

- Coding Trainer

  Coding Trainer is a software system designed to facilitate the teaching of programming languages with the minimal support of human instructors.

- 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-Assisted Coding Trainer

  GPT-Assisted Coding Trainer is an advanced coding trainer that harnesses the power of GPT to deliver its core functionality. The system leverages the capabilities of a recent version of GPT model, i.e., GPT-4, through its API to perform essential tasks and provide a comprehensive learning experience.

- Language Profile

  A Language Profile is a meta-description of a programming language and is defined by the course director. A language profile includes the key attributes of the target language, including a sequence of units to learn and topics within each unit.

- Foundation of each Unit

  Each unit of a language profile is defined with its foundation to teach. It is the fundamental concepts of the topics as specified in a language profile, including the introduction to the topics, syntax, and example codes.

- Coding Exercise

  Coding exercises are the coding exercises for learners to solve. The system generates coding exercises that align with the learning objectives, ensuring the learner has opportunities to practice and apply the concepts covered in the current topic.

- Training Reports

  Training reports are in two types. Progress reports describe the progress of learners'

achievements. Certificate of Complement is to confirm that the learner has successfully completed all the required training units and fulfilled the necessary criteria for certification.

1.4. 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.

1.5. 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 1.

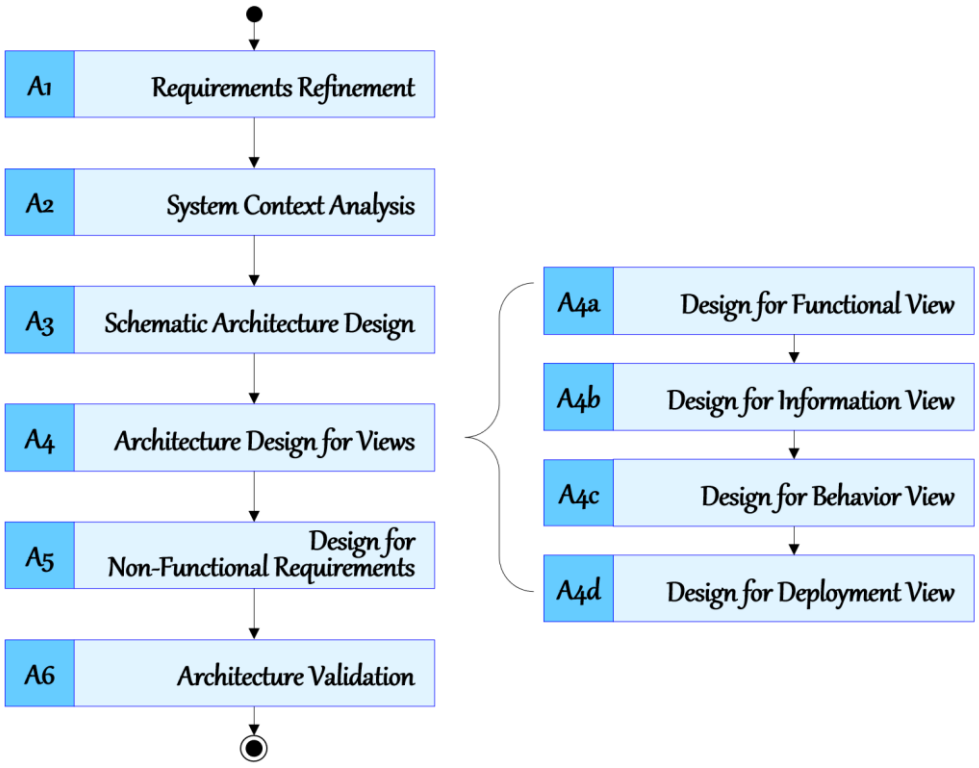


Figure 1. Process to design Software Architecture

- 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.

1.6. Template used for Architecture Description

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

2. Activity 1. Architectural Requirement Refinement

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

2.1. [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. Learner

  This represents the user group using the target system to learn programming.

- Stakeholder 2. (Course) Director

  This represents the user group writing language profiles and monitoring the learning processes.

- Stakeholder 3. Marketing Staff

  This represents the use group specialized in marketing the target system.

- Stakeholder 4. IP Lawyer

  This represents the use group specialized in legal issues of Intellectual property (IP). IP is a category of property that includes intangible creations of the human intellect. The contents from GPT can be used for GPT users and paid subscribers. However, IP issues could be raised in utilizing GPT contexts to provide value-added services like Coding Trainer. Hence, IP Lawyer is a valuable stakeholder.

The profile of each stakeholder is summarized in Table 1.

Table 1. Profiles of Stakeholders

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

2.2. [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 4.5
Original Context	"Using the GPT model, the system analyzes the current topic and learner's profile to generate exercise problems that are relevant, engaging, and appropriately challenging. The GPT model's natural language processing capabilities allow the system to understand the learner's level of proficiency, preferred learning style, and specific areas that require focus."				
Questioning	How does the coding trainer system understand the learner's proficiency? The system relies on GPT heavily and GPT does not store the sessions of long-term learning transactions.				
Refined Context	GPT is limited to handling a current topic, not remembering the previous topics and dialogues. Hence, the coding trainer has to store the learners' proficiencies and generating context-rich prompts to guide the output of GPT.				

❑ Deficiency #2

Deficiency ID	FR.DEF.02	**Deficiency Type**	Incompleteness	**Location**	SRS 4.7
Original Context	"The evaluation is based on the criteria of correctness, efficiency, adherence to coding standards, and other relevant criteria. The system assesses the submitted code by consulting the GPT."				
Questioning	The consistency on evaluating submitted exercise codes among learners is an important criterion as a training system. How to ensure the consistency of evaluation?				
Refined Context	Coding Trainer does not perform the evaluation by itself, and it relies on the capability of GPT to maintain the consistency. Stakeholders performed limited experiments on the consistency issue and have positive reliability on GPT. However, their investigation was performed with limited test cases, and hence, this issue of consistency has to be further investigated by the development team.				

2.3. [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. Availability and Stability of the System

  The system runs as a web server, which is assessed by potentially many users. Hence, the system should be designed to provide high availability and stability.

- Concern-2. Instructions tailored to Learners

  The instructions to learners should be tailored for each learner to yield a high effectiveness of training. Avoid using instructions of one-size-fits-all type.

- Concern-3, Quality Assessment by Professional Instructors

  The quality of education provided by Coding Trainer should be evaluated by professional instructors, and their evaluations and feedbacks should be reflected in enhancing the quality of education.

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

- Concern-1. Availability and Stability → (NFR #1) High Reliability
- Concern-2. Instructions tailored to Learners → (NFR #2) High Personalization
- Concern-3. Quality Assessment by Professional Instructors → (NFR #3) Newly Added !

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

Table 2. Relevance of NFRs to Stakeholders

NFR Items	Relevance to Stakeholders				Average Relevance	Standard Deviation	Selection (Y/N)
	User	Staff	Client	Scientist			
NFR-1	2	1	2	1	1.5	0.58	Y
NFR-2	2	0	0	2	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 2 to fill in the table.

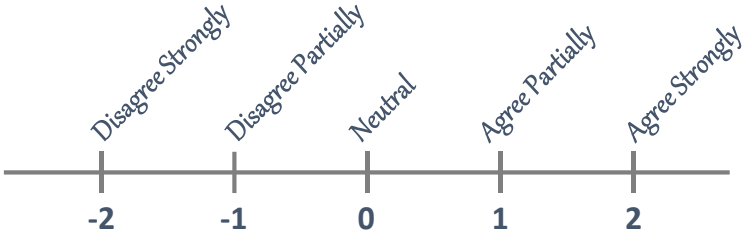


Figure 2. Degree of Relevance of NFRs to Stakeholders

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.

2.4. [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**	Incompleteness	**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: Maturity, Fault Tolerant, and Recoverability.				
Questioning	What is the degree of Recoverability expected by stakeholders, in terms of recoverability time efficiency and the elements to be recovered.				
Refined Context	Even in case of failures, the downtime of Coding Trainer should be less than an hour. All the learning transaction-related datasets must be recovered.				

❑ Deficiency #2

Deficiency ID	NFR.DEF.01	**Deficiency Type**	Incompleteness	**Location**	SRS 5.2
Original Context	"By considering the proficiency level of learners, the system can generate exercise problems that align with their current abilities and knowledge."				
Questioning	Accordingly, different learners will be given different sets of exercise problems. Nonetheless, there must be a required level of exercise difficulty and scope of exercise problems that will be applied to all learners. What are the required level and scope?				
Refined Context	Stakeholders were consulted by an architect, and they agree on the need to set the required level and scope to meet. However, they did not provide any further details on these and expected that the development team could propose them.				

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

2.5. [Step 5] Write Refined Software Requirement Specification

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

3. 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

An additional type of the context can be described.

3.1. [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 architects 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.

3.1.1. Level 0 DFD for the Boundary Context

The boundary context is shown in Figure 3.

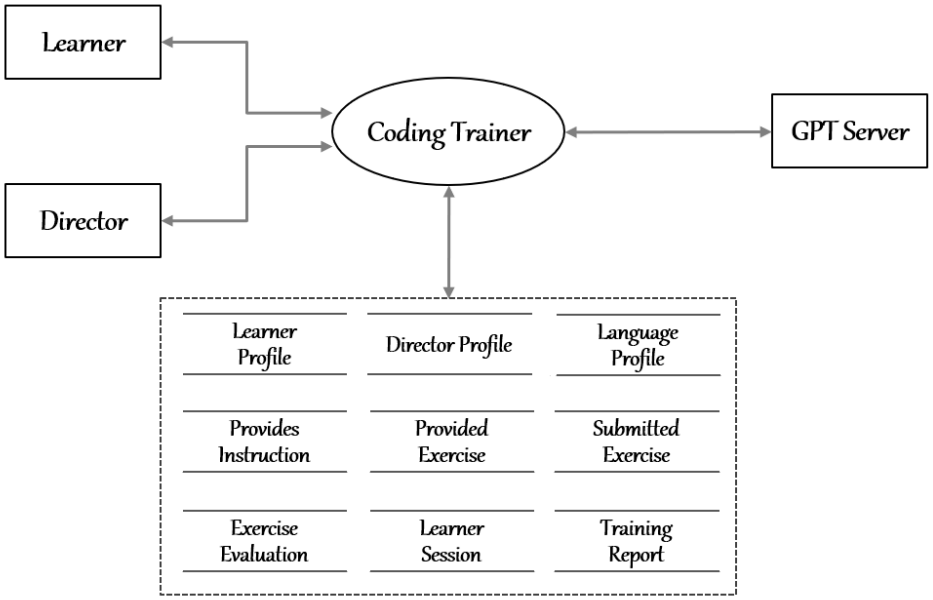


Figure 3. DFD Diagram for Boundary Context

3.1.2. Description of the Context Diagram

- Process
  - The SRS specifies that the target system is deployed as a single web server, i.e., a single tier system. Accordingly, only one process is defined: *Coding Trainer*.
- Terminals
  - The SRS specifies two types of users: Learner and Director.
  - The SRS specifies an external system to interface: Chat GPT.
- Store
  - The SRS lists a number of functional groups, from which the persistent datasets are derived.
- Data Flow

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

3.2. [Step 2] Functional Context

3.2.1. 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.

3.2.2. 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.

From the SRS of the target system, the following functional groups are derived, and their prefixes are specified as a two-character label.

- Learner Profile Management (LE)
- Director Profile Management (DI)
- Language Profile Management (LA)
- Foundation Teaching (FT)
- Exercise Generation (EG)
- Solution Submission (SS)
- Solution Evaluation (SE)

- Learner Session Management (SM)
- Training Report Generation (RP)

3.2.3. Defining Actors

From the functional groups of the system, their relevant actors are derived.

- Actors of User
  - Learner
  - Director
- Actors of External System
  - GPT Server
- Actors of Software Agent

  The software agents are not specified in the SRS. They are derived through observations on how each function in the system should be invoked. Four actors of software agent type are defined.
  - Teaching Agent
  - Exercise Agent
  - Evaluation Agent
  - Report Agent
- Table of Defining Actors for Functional Groups

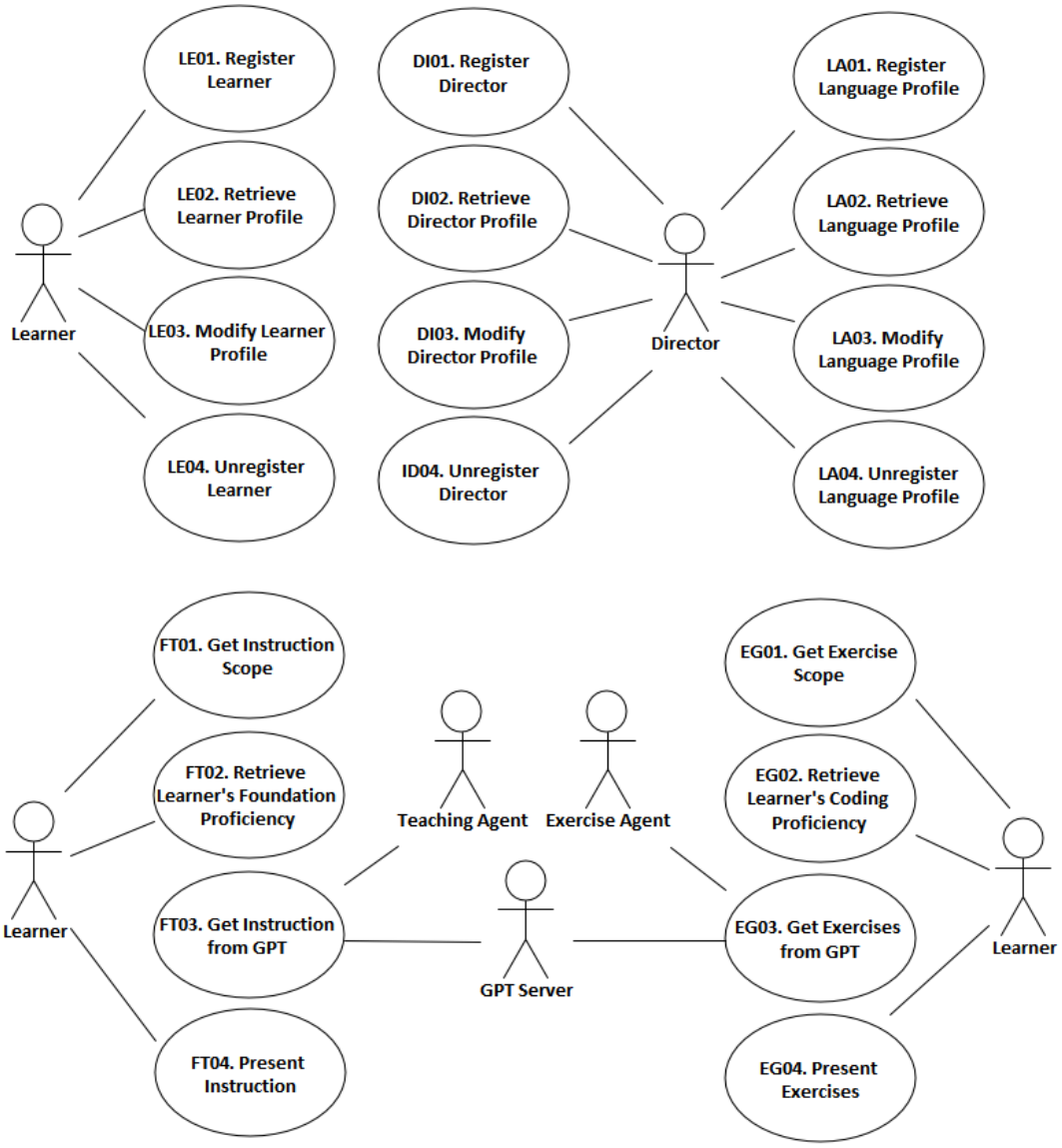
  The following table specifies actors for each functional group.

Actors / Functional Groups	Active Actors	Passive Actors
Learner Profile Management (LE)	Learner	
Director Profile Management (DI)	Director	
Language Profile Management (LA)	Learner (Read-only)	Director
Foundation Teaching (FT)	Learner, Teaching Agent	GPT Server
Exercise Generation (EG)	Learner, Exercise Agent	GPT Server
Solution Submission (SS)	Learner	
Solution Evaluation (SE)	Learner, Evaluation Agent	GPT Server
Learner Session Management (SM)	Learner	
Training Report Generation (RP)	Learner	Report Agent

3.2.4. Use Case Diagram for Functional Context

The use cases in the diagram can systematically be derived from the functional groups.

- Use Case Diagram as Functional Context (Figure 4)



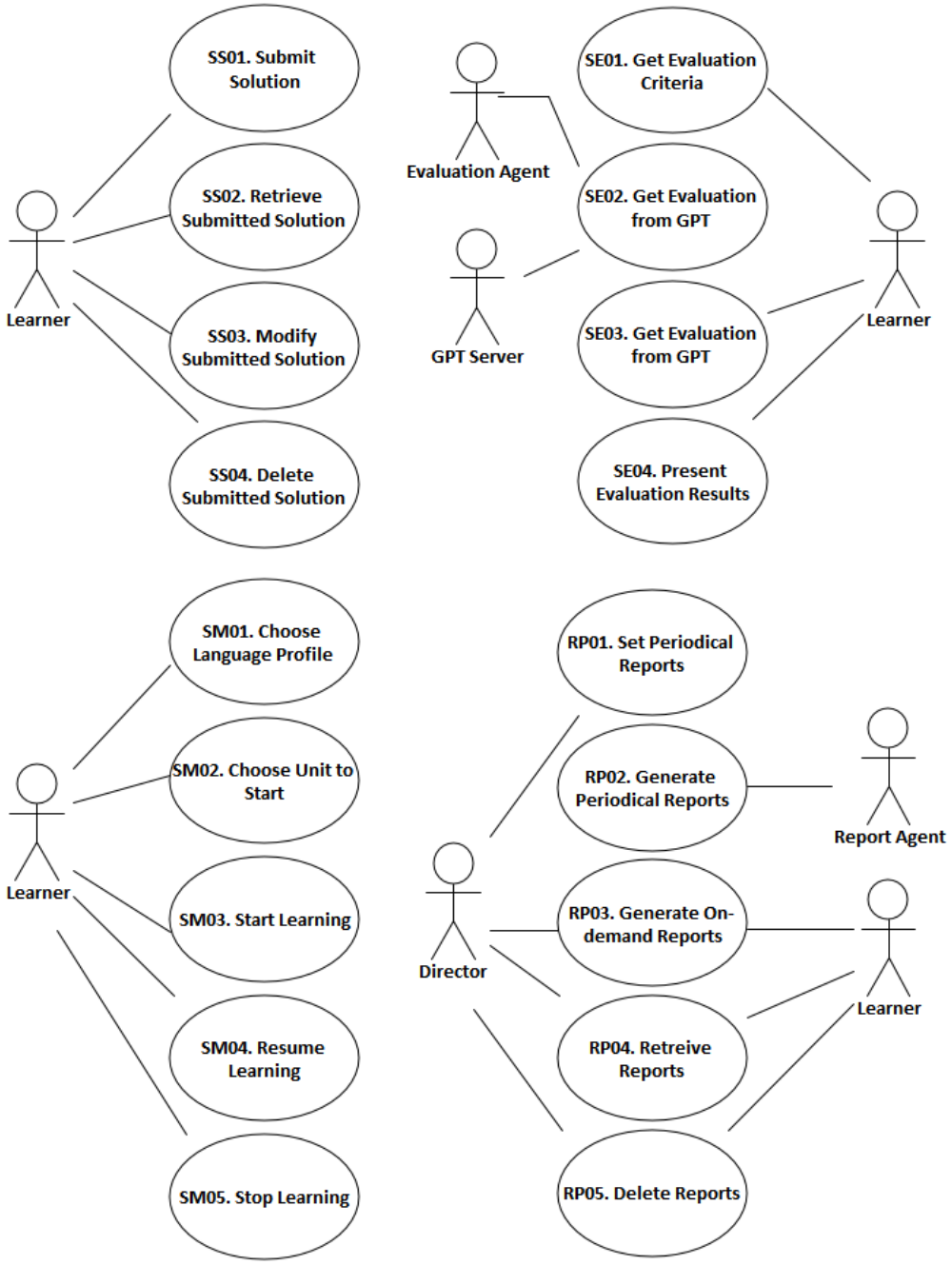


Figure 4. Use Case Diagram for the Functional Context

3.3. [Step 3] Information Context

3.3.1. 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.

3.3.2. 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

3.3.3. Context-level Class Diagram

The class diagram representing the information context may include only classes and their relationships. The context-level class diagram of the target system is shown in Figure 5.

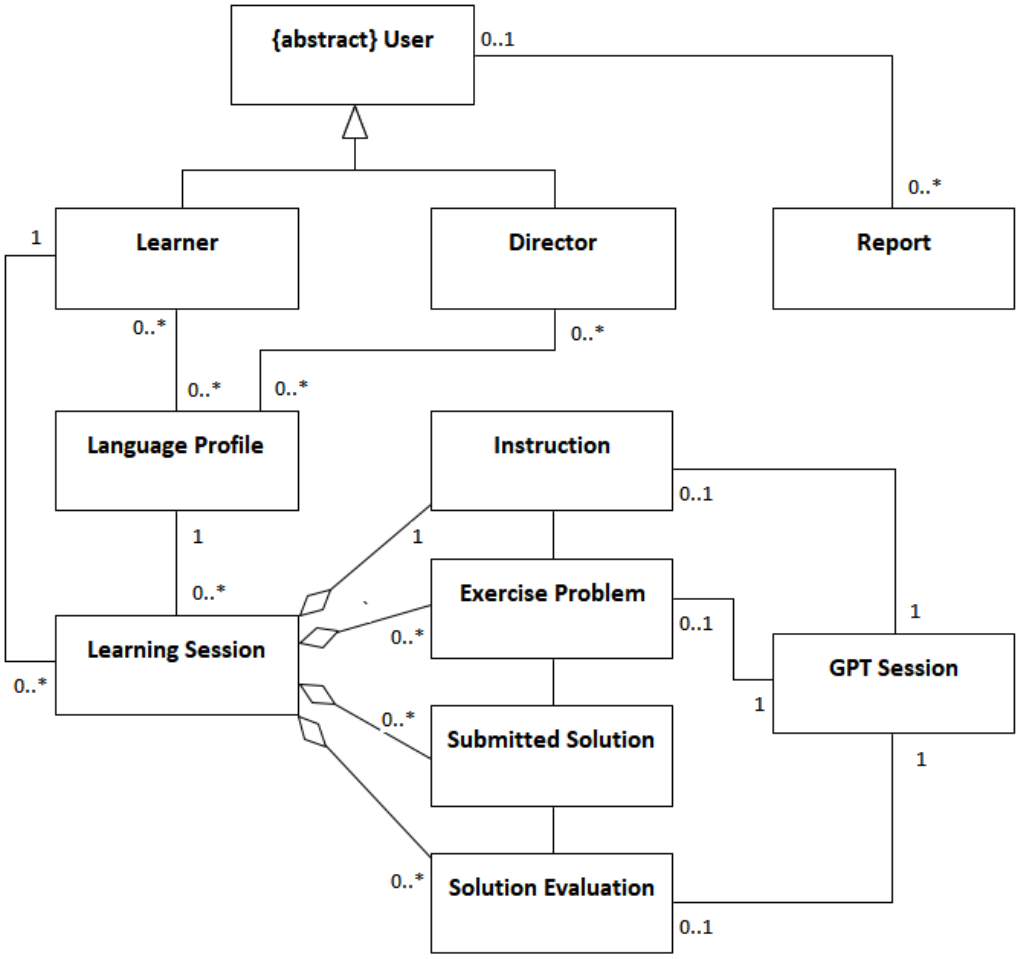


Figure 5. Class Diagram for showing Information Context

The class diagram shows entity-type classes with their relationships, and it provides the context of persistent datasets that should be managed in a secondary storage.

3.4. [Step 4] Behavioral Context

The 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.

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

3.4.1. Allocate Functionality on 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 3. Allocation of Functionality over Tiers

Tiers / Functional Groups	Coding Trainer Server
Learner Profile Management (LE)	✓
Director Profile Management (DI)	✓
Language Profile Management (LA)	✓
Foundation Teaching (FT)	✓
Exercise Generation (EG)	✓
Solution Submission (SS)	✓
Solution Evaluation (SE)	✓
Learner Session Management (SM)	✓
Training Report Generation (RP)	✓

3.4.2. Define Invocation Patterns

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

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

- **Event I** is for handling events within a tier, i.e., intra-tier event-driven invocation.
- **Event II** is for handling events among multiple tiers, i.e., inter-tier event-driven invocation.
  - The invocation with Event II type is not appliable to 1-tier system.

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

Table 4. Invocation Patterns for Functional Groups

Tiers / Functional Groups	Coding Trainer Server
Learner Profile Management (LE)	Explicit
Director Profile Management (DI)	Explicit
Language Profile Management (LA)	Explicit
Foundation Teaching (FT)	Explicit, Event, Closed Loop
Exercise Generation (EG)	Explicit, Event, Closed Loop
Solution Submission (SS)	Explicit
Solution Evaluation (SE)	Explicit, Timer, Closed Loop
Learner Session Management (SM)	Explicit, Event, Closed Loop
Training Report Generation (RP)	Explicit, Timer, Event, Closed Loop

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

3.4.3. 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 6.

- Control Flow of Coding Trainer

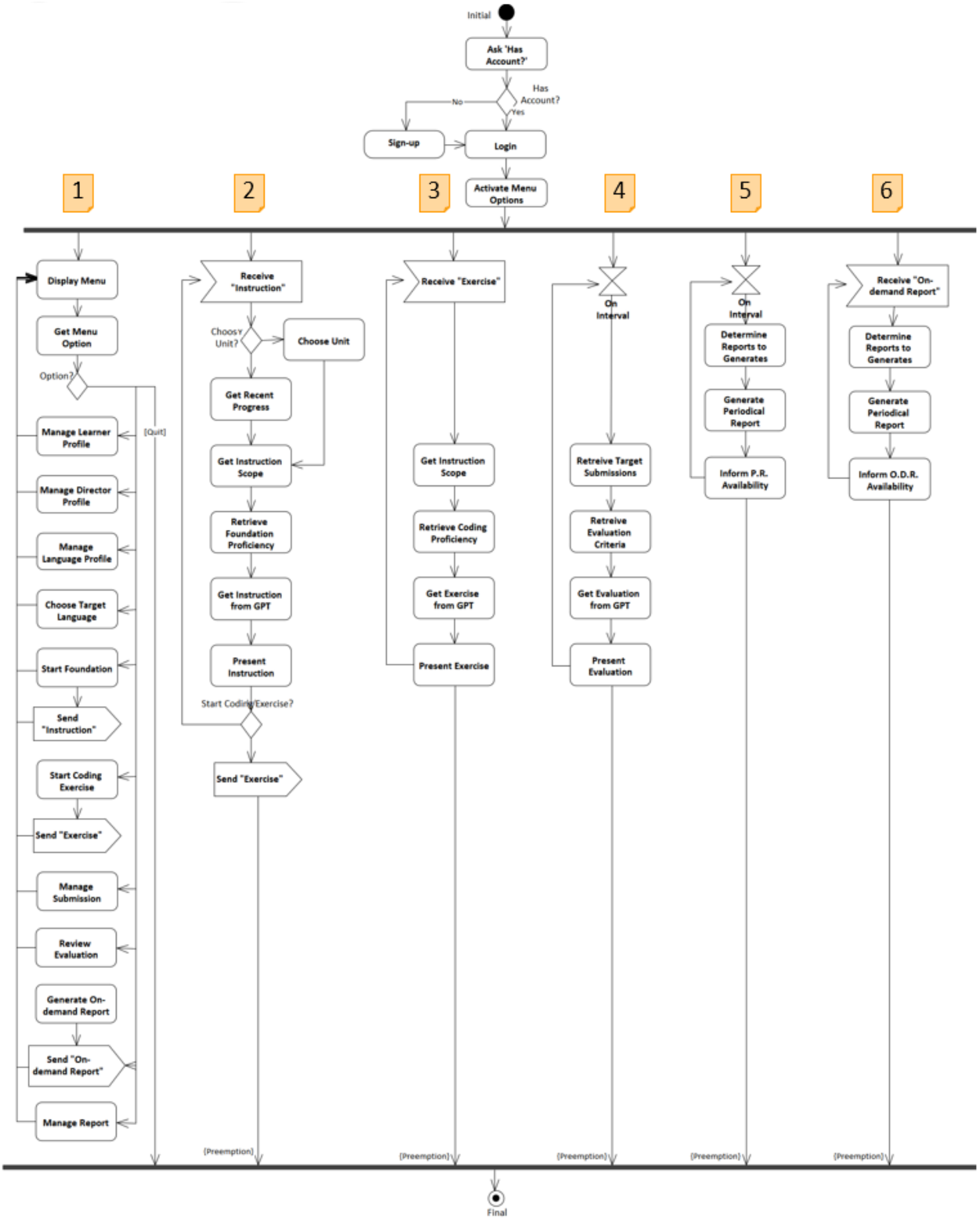


Figure 6. Activity Diagram for showing Behavior Context

The control flow of the system is defined with 6 parallel threads.

- The thread #1 is to handle the functionality with explicit invocation pattern using a menu.
- The thread #2 is to acquire the teaching instruction from GPT and display it.
- The thread #3 is to acquire the coding exercises from GPT and display it.
- The thread #4 is to initiate the evaluation of submitted solution, using a timer.
- The thread #5 is to generate periodical reports at a given time interval.
- The thread #6 is to generate on-demand reports using event-based invocation.

The behavior context shows the overall runtime behavior of the system functionality. All the use cases of the target system are reflected in this activity diagram.

3.5. [Step 5] Additional Contexts

Any additional contexts of the target system can be described.

- None