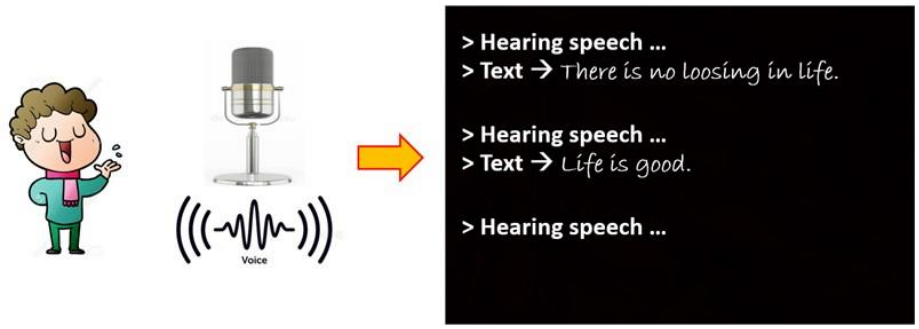
Architecture Description

Speech-to-Handwritten Text, (SThT)

Application that generates a Personalized Handwritten Text for a given Speech



November 2022

Revision History

Version	Date	Author	Revisions Made
0.8	11/10(Th), 9pm	이현웅	(Interim) Defining the Skeleton Architecture
0.9	11/20(Sun), 9pm	이현웅	(Pre-final) Designing Architecture with Viewpoints
1.0	12/05(Mon), 9pm	이현웅	**(Final) Architecture with All Activities Applied**

삼성전자 첨단기술연수소

TABLE OF CONTENTS

Architecture Description of Speech-to-Handwritten Text

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

Speech to Handwritten Text System (SThT System) is a system that converts various voice recordings such as conversations, lectures, and speeches into the user's own handwriting. The system learns the user's handwriting and provides various functions through it.

- Functionality for Users

  The system provides the functionality of registering user information and registering User's handwriting provided as a sample to learn handwriting their own handwriting samples. The system learns the user's handwriting from these samples and provides a font file or provides a handwritten text that transformed voice recording or audio file.

- Functionality for Staff

  The system provides the functionality of registering company staffs who manages the app and server system. The staff also manage that the handwriting learning function maintains high performance and can create reports as needed.

1.3. Definitions, Acronyms, and Abbreviations

- Pangram

  Pangram is a user's handwriting provided as a sample to learn handwriting. Users write Pangram sentences through a pen connected to the device.

- STT

  STT is an abbreviation of speech-to-text and is a service that converts speech into text. It is a machine learning-based service provided by companies such as Google, Amazon, and Microsoft. This system uses Azure STT provided by MS.

- Speech records / audio file

  This is the original speech data used in the STT service. Audio file is provided in the form of an mp3 file within 10 minutes. For smooth operation of STT during voice recording, the interval between voices should not exceed 15 seconds.

- ML Model

  The handwriting learning of the system is done through machine learning. The ML Model is a model that has learned the user's handwriting, and there may be differences in performance depending on the learned pangram. The system calculates a performance score and asks for pangrams until the performance exceeds a certain level.

1.4. References

[Kim 22a] Soo Dong Kim, Associate Architect Program, 2021-A5, CEP Specification of Speech-to-Handwritten Text System, Version 0.9, 삼성전자 첨단기술연수소, Nov. 2022

[Kim 22b] Soo Dong Kim, Associate Architect Program, 2021-A5, CEP Template for Speech-to-Handwritten Text System, 삼성전자 첨단기술연수소, Nov. 2022.

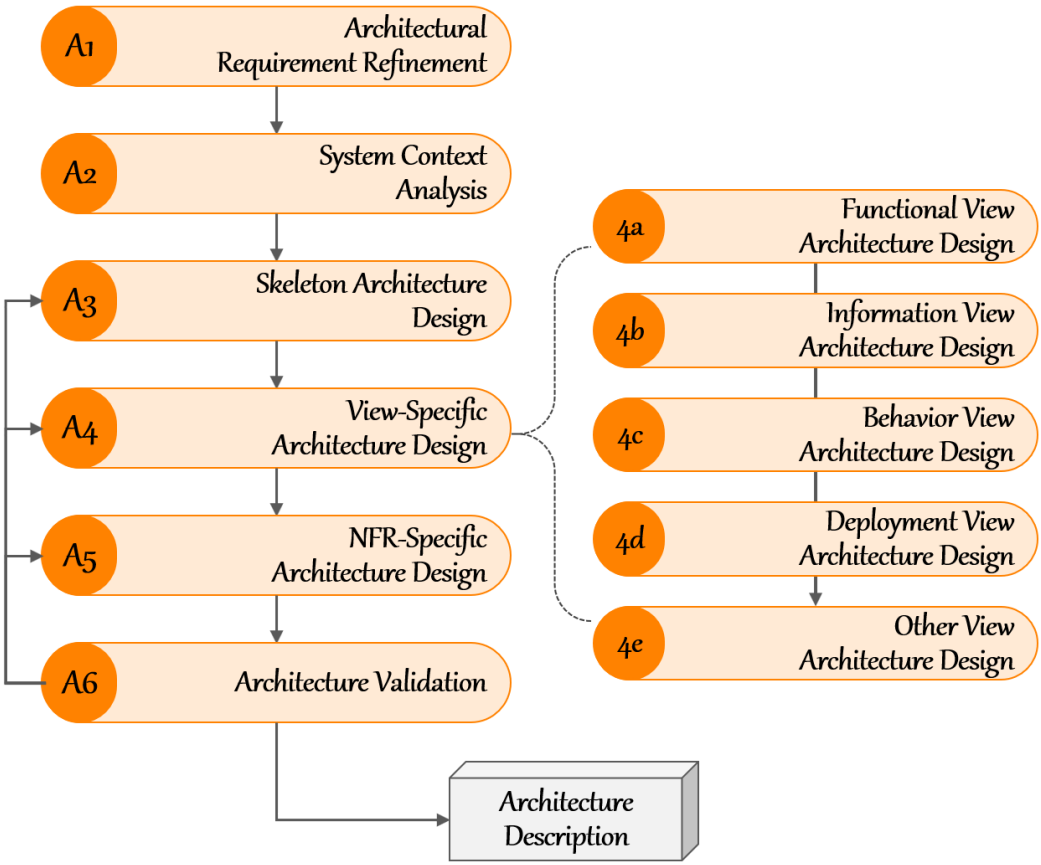
[Kim 22c] Soo Dong Kim, Architecture Design Process & Instructions, 2021-A5, Lecture Note #1, 삼성전자 첨단기술연수소, Nov. 2022

[Kim 22d] Soo Dong Kim, 2021-A5, CEP Specification of Speech-to-Handwritten Text System, Version 1.0, 삼성전자 첨단기술연수소, Nov. 2022.

[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.


A1
Architectural Requirement Refinement
A2
System Context Analysis
A3
Skeleton Architecture Design
A4
View-Specific Architecture Design
A5
NFR-Specific Architecture Design
A6
Architecture Validation
4a
Functional View Architecture Design
4b
Information View Architecture Design
4c
Behavior View Architecture Design
4d
Deployment View Architecture Design
4e
Other View Architecture Design
Architecture Description

- Activity 1. Architectural Requirement Refinement

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

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

- Activity 2. System Context Analysis

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

- Activity 3. Skeleton Architecture Design

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

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

- Activity 4. View-specific Architecture Design

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

- Activity 5. NFR-specific Architecture Design

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

- Activity 6. Architecture Validation

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

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

1.6. Template used for Architecture Description

The template used for writing this architect description is given in [Kim 22b].

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. Client

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

- ❑ Stakeholder 2. App User

  This represents the users who uploads their handwritings for generating their own handwriting font and to use SThT function.

- ❑ Stakeholder 3. Data Scientist

  This represents a company staff who maintains ML Model performance and generates datasets suitable for training.

- ❑ Stakeholder 4. System Manager

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

Describe the information of each stakeholder.

Stakeholder Group	Representative Name	Contact Information	Availability
Client	Lindsey Stirling	604-662-4700 Vancouver, BC	After 2pm, only on WF, Office Visit
Data Scientist	Hasan Piker	604-662-4720 Vancouver, BC	3-5pm, MWF, Phone, Office Visit
App User	John Joe	604-434-4231 Burnaby, BC	2-4pm, WF, Office Visits
System Manager	Harry Gray	604-662-4824 Vancouver, BC	10am-4pm, M-F, Office Visit

2.2. [Step 2] Refining Functional Requirements (Revise)

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

- ❑ Deficiency #1

Deficiency ID	FR.DEF.01	Deficiency Type	Ambiguity	Location	SRS 4.3
Original Context	In this case, the system asks the user to handwrite different pangram sentences repeatedly until the training is successfully completed.				
Questioning	How does the system set the criteria for successful completion and limit the number of iteration requests for a pangram sentence?				
Refined Context	The completion condition for training is a performance score of 80% or higher, calculated according to the metrics provided by the data scientist. Pangram sentence requests are requested up to 4 times, 5 sentences at a time, and after that, model initialization will be suggested to the user.				

❑ Deficiency #2

Deficiency ID	FR.DEF.01	Deficiency Type	Incompleteness	Location	SRS 4.4
Original Context	In Task 3, The system also computes the confidence value, i.e., the performance measure for the generated handwritten text.				
Questioning	After checking the confidence value of the generated handwritten text, what further action should we take?				
Refined Context	A data scientist analyzes the values and, if necessary, modifies the parameters of the ML model.				

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. (revise)

❑ Concern-1. Generated custom fonts should be distinguishable and like handwriting.

The system should be designed to provide a high accuracy in generating each user's unique handwriting. In particular, the change in the font according to the preceding character should be sufficiently reflected.

❑ Concern-2. System server should be reliable for every situation.

The machine learning functions included in this system can put a lot of loads on the server. For this function to be used by many users, high reliability is required for the server.

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

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

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

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

- Concern-1. Generated fonts should be distinguishable and like handwritten

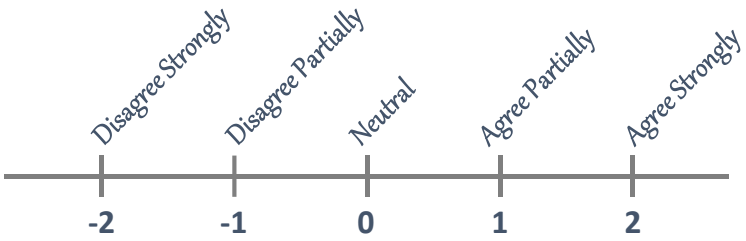
  → NFR-1. Accuracy of personalized Handwritten Texts.

- Concern-2. Reliable server for every moment → NFR-2. High Reliability of the Server
- Concern-3. Minimal Time loss → NFR-3 High Usability for Users

List all the NFR items in *NFR-to-Stakeholder Mapping* table.

NFR Items	Relevance to Stakeholders				Average Relevance	Standard Deviation	Selection (Y/N)
	Client	Data Scientist	App User	Server Manager			
NFR-1	2	2	2	1	1.75	0.43	Y
NFR-2	2	2	0	2	1.5	0.86	Y
NFR-3	2	-2	2	-2	0	2	N

We use the following scoring systems for filling in the table.



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**	Ambiguity	**Location**	SRS 4.3
Original Context	Note that the handwriting of each letter may vary depending on the surrounding characters of a letter.				
Questioning	Should We also be concerned about the number-related parts?				
Refined Context	It would be nice to be able to learn numbers according to the above situation, but because there is a limit to the pangram request, only handwriting learning is carried out and the part about preceding letters is excluded.				

❑ Deficiency #2

Deficiency ID	NFR.DEF.02	**Deficiency Type**	Ambiguity	**Location**	SRS 5.1
Original Context	The system should provide a high level of reliability in terms of its sub-quality attributes as defined.				
Questioning	To satisfy all reliability requirements, we'll need a secondary server or a spare server. How many servers can we purchase?				
Refined Context	It is difficult to invest too much money because of financial problems. It should be developed with minimal cost.				

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 20d].

3. Activity 2. System Context Analysis

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

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

Additional type of the context can be described.

3.1. [Step 1] System Boundary Context

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

3.1.1. Context Diagram

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

❑ Level 0 DFD for Context View

The SRS implies 2 tiers and accordingly the context diagram includes 2 processes, as shown in Figure 1.

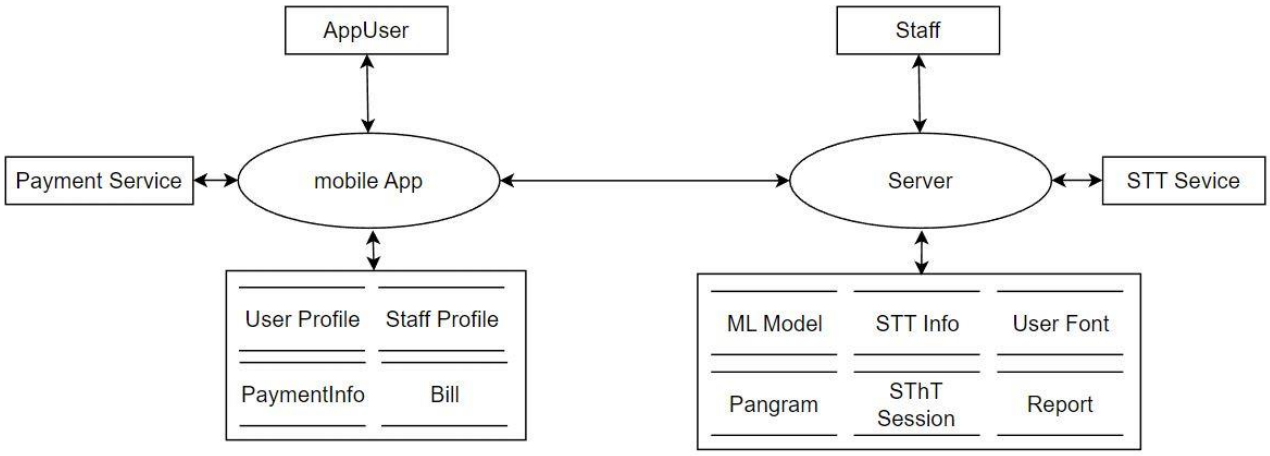


Figure 1. Context Diagram of SThT System

3.1.2. Description of Context Diagram

❑ Terminals Description

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

❍ App User
This represents the users who use the mobile app to use SThT System.

❍ Staff (Data Scientist and System Manager)
This represents the employee of the company who manage mobile App and Server.

❍ Payment Service
This represents an external service for processing payment with a registered card.

❍ STT Service
This represents an external system used to convert speech into text.

❍ Archival Manager

This represents a user who manages the archival-related operations.

- Processes, i.e. Tiers

  The system consists of 4 tiers and each tier is represented as a process of DFD.

  - Mobile App
  - Server

- Store

  Server maintains datasets for machine learning and STT Service and mobile app manages the payment system and profile data to reduce the load on the server. as shown in the figure.

- Data Flow

  Each arrow in the diagram indicates a flow of data, and the names of the data on arrows are omitted in the diagram.

3.2. [Step 2] Functional Context

3.2.1. Use Case Diagram for the System

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

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

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

- Functional Groups

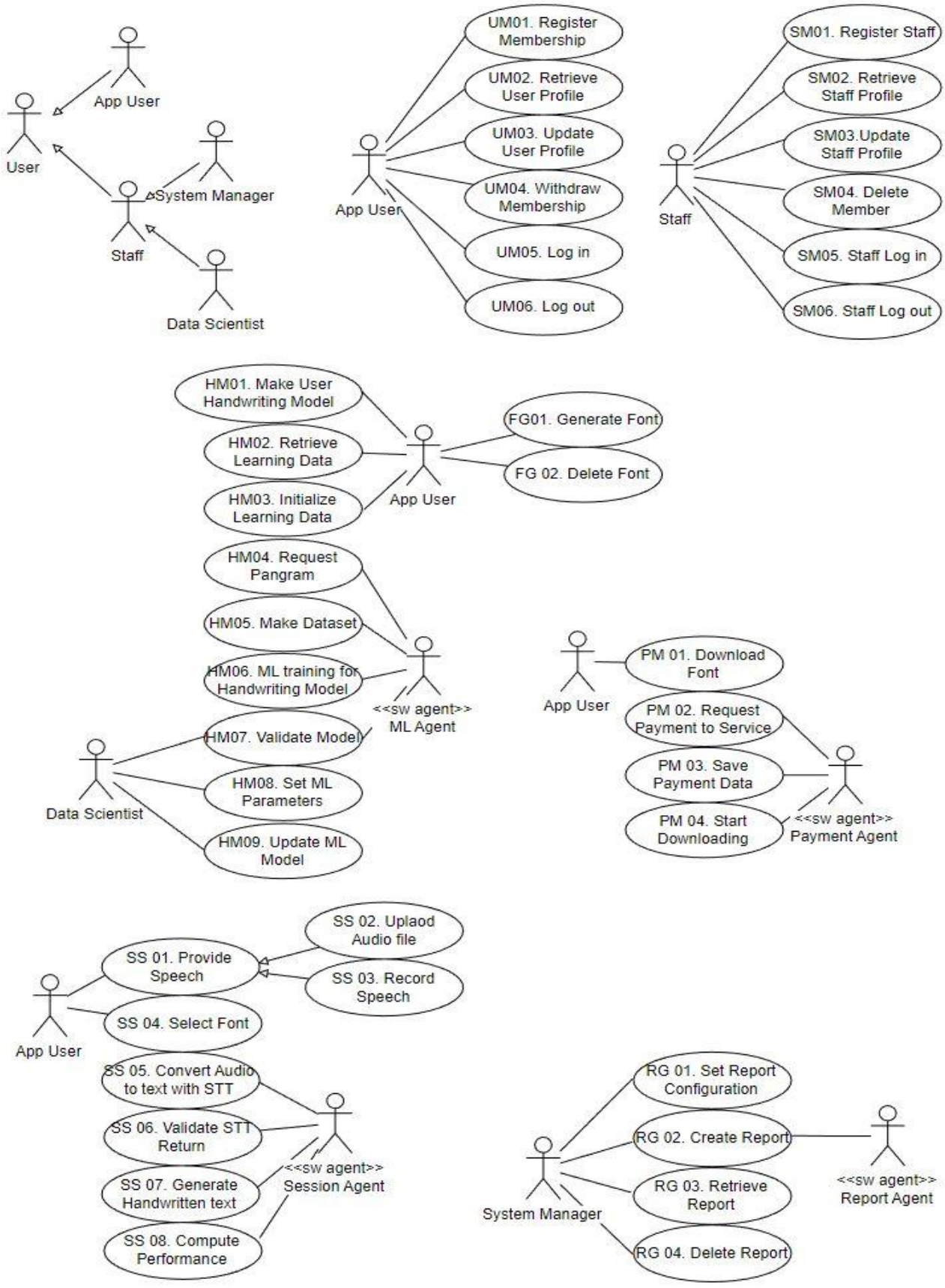
  We apply a scheme for numbering the use cases by considering functional groups. A functional group is a collection of *closely related* use cases. And we assign a two-character prefix to each functional group. A use case diagram with use case identification numbers becomes easier to comprehend and to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

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

  - User Profile Management → UM
  - Staff Profile Management → SM
  - Handwriting Training Management → HM
  - Personalized Font Generation → FG
  - Billing and Payment Management → PM
  - SThT Session Management → SS
  - Report Generation → RG

- Use Case Diagram

  The following use case diagram shows all the use cases of the target system.



3.2.2. Description of Use Case Diagram

- Actors as Human Users

  There are several actors in the diagram that represent human users who use the system;

  - App User, Staff (Data Scientist, System Manager)

- Actors as Software Agents

  There are several actors in the diagram that represent software components which run in background mode. They can be implemented as daemon processes/threads.

  - Session Agent, Payment Agent, ML Agent, Report Agent

- Use Cases

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

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

3.3. [Step 3] Information Context

3.3.1. Class Diagram for the System

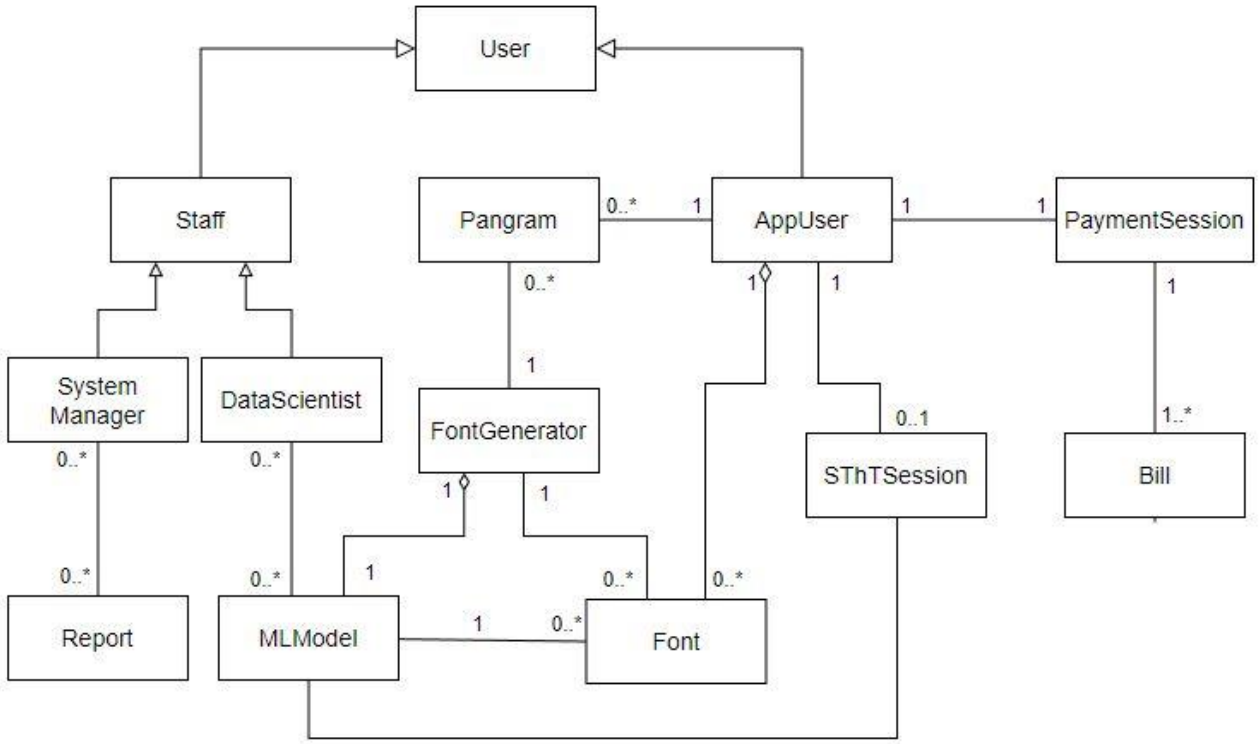
The information context of the system shows the datasets manipulated by the system. Class diagram can be effectively used to capture the information context.

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

- Class Diagram

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

  The context-level class diagram of the target system is shown in the following figure.



3.3.2. Description of Classes

- User Profile-related Classes

  User is the superclass which captures the common property of its subclasses.

  - *AppUser* captures the information of the user of mobile app.
  - *Staff* is the superclass which captures the common property of its subclasses.
    - *System Manager* captures the information of report setting change history.
    - *Data Scientist* captures the information of ML Model evaluation and history.

- Pangram

  Pangram captures the information from raw handwriting data provided by AppUser

- Payment Session

  This class captures the information payment transactions.

- Bill

  This class captures the content of accrued bill.

- SThT Session

  This class captures the information and content of SThT service.

- Font

  This class captures the information of generated font.

- ML Model

This class captures the information of trained handwriting ML model.

- ❑ FontGenerator

  This class captures the information of datasets of ML and font generating data.

- ❑ ML Model

  This class captures the information of generated and/or deployed machine learning models.

- ❑ Report

  This class captures the information of generated report

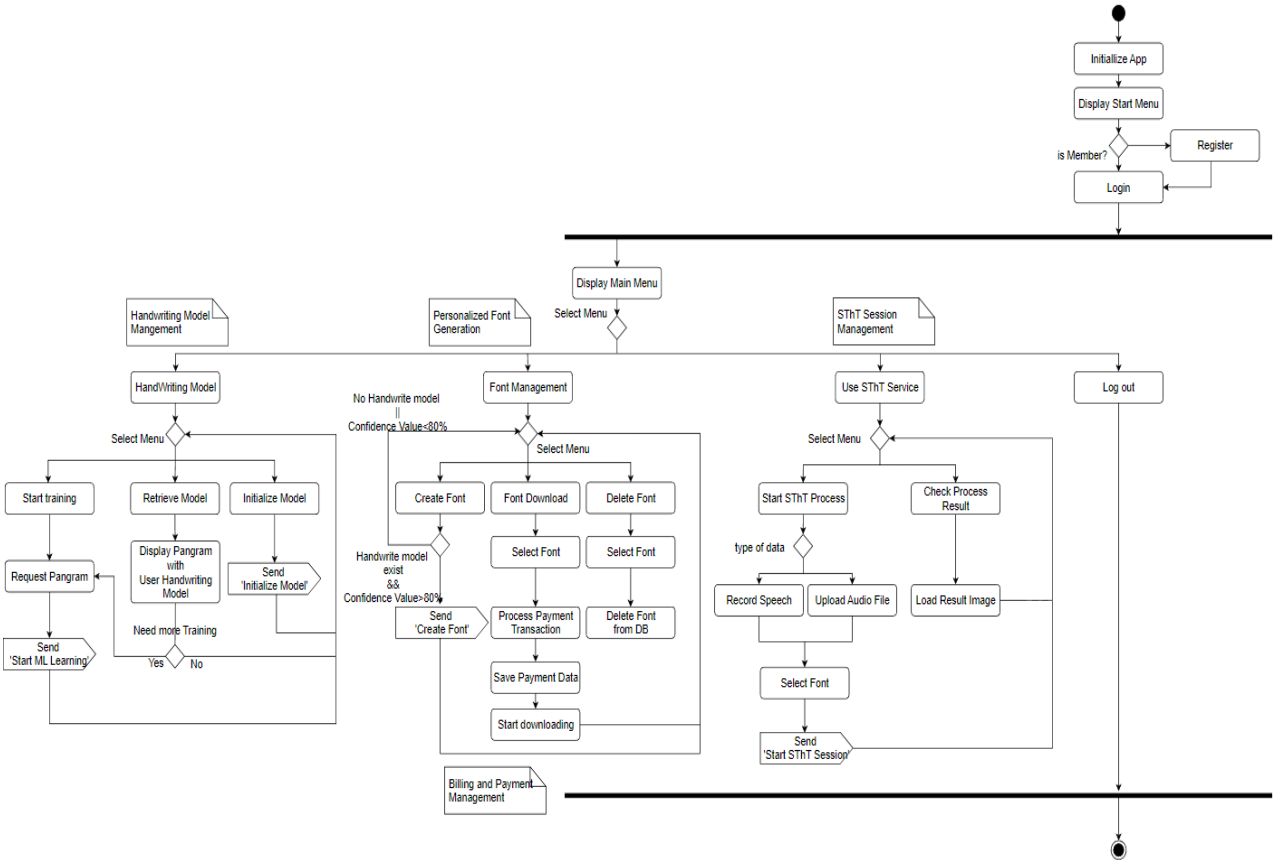
3.4. [Step 4] Behavioral Context

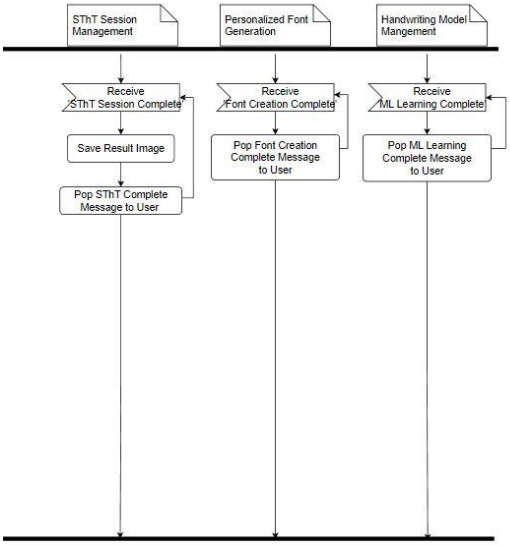
Behavioral context of the system shows the execution and control flow at runtime. Behavioral context may be more important for systems with complex workflows, parallel processing, and timing constraints.

3.4.1. Activity Diagram

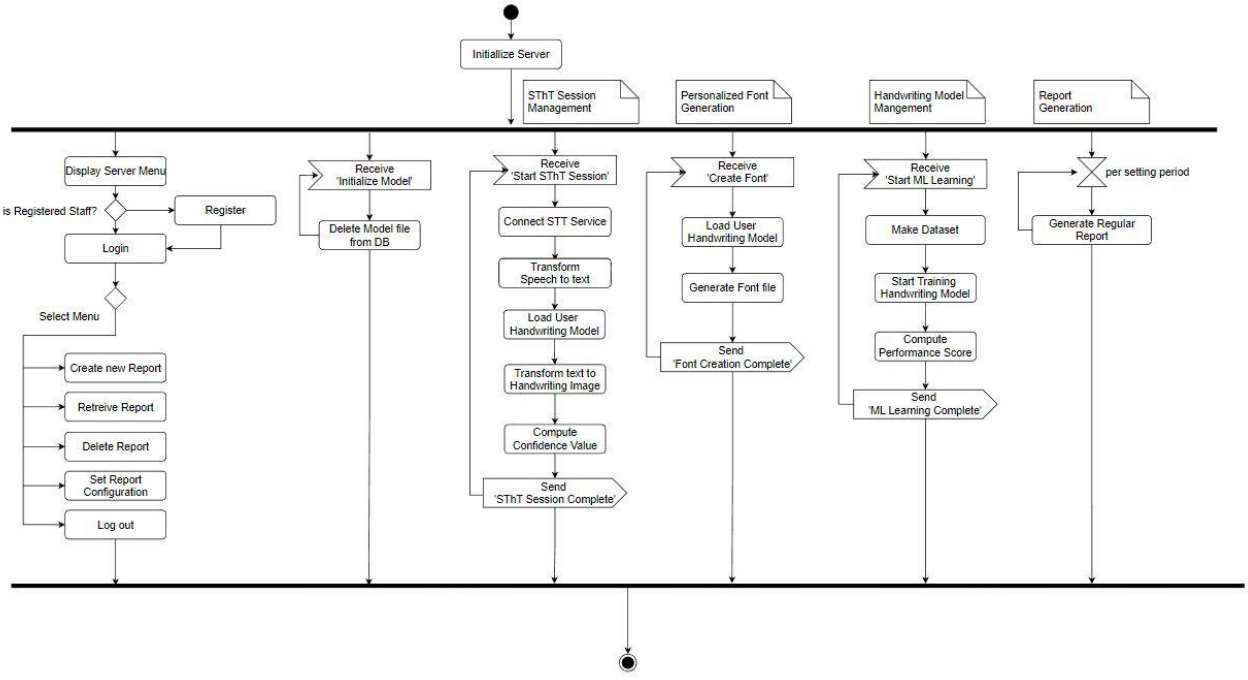
Activity Diagram can be used to define the overall control of the target system. There are 2 tiers in the target system, and we specify the context-level control flows of the 2 tiers.

- ❑ Control Flow of Mobile App





❑ Control Flows of Server



3.4.2. Description of the Diagram

❑ Mobile App

The control flow for Mobile App has two parts: menu-driven invocation-based behavior and a closed loop behavior(thread).

The invocation-based behavior includes the actions like managing handwrite model, font, STT Service.

In a closed loop, it is mainly configured to notify the user by receiving a message that the task requested to the server is completed.

- ❑ Server

  The control flow for Server has three parts. A menu-driven invocation-based behavior includes actions for staffs: evaluating ML Model and managing report. Closed-loop operations consist of operations with the ML model, since these operations are time consuming. Timed behavior is only for generating regular report.

3.5. [Step 5] Additional Contexts

Any additional contexts of the target system can be described.

- ❑ None

4. Activity 3. Skeleton Architecture Design

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

4.1. [Step 1] Observe Architectural Characteristics

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

- ❑ Consisting with Several Tiers → Client-Server Architecture Style

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

- ❑ Manipulating datasets in several phases → Batch-Sequential Architecture Style

  In SThT function, Raw data goes through various components and becomes handwritten text.

- ❑ Conventional Mobile Application → MVC Architecture Style

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

- ❑ High Reliability on Server → Dispatcher Architecture Style

  The SThT system needs a load balancing for a reliable server.

4.2. [Step 2] Candidate Architectural Styles

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

- ❑ Client-Server architecture style
- ❑ Batch-Sequential Architecture style
- ❑ MVC architecture style
- ❑ Dispatcher Architecture Style

4.3. [Step 3a] Applicability of 'Candidate 1. Client-Server Architecture Style'

The SThT System is typical mobile application. It is composed of two tiers and functions can be separated similarly to the functions of client and server.

4.3.1. Match on the Applicable Situation

Applicable Situations	Match	Demands on the System

Functions can be separated from client and server	❍	The SThT system can be divided into a function based on user interaction including GUI and a function that requires a lot of resources such as machine learning and font generation.
Multiple users share resources	❍	Since It's typical mobile application, server resources must be shared when many users use the app.
Shared data between client and server is not too much.	❍	The functions of this system are clearly separated into client and server, so there is not much shared data.

4.3.2. Applicable Benefits

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

4.3.3. Handling Drawbacks

Cons of the Style	Match	Handling the Drawbacks
Failure due to server overload	❍	For scalability, use a cloud server such as Paas or have a secondary server.
Communication overhead between client and server	Δ	There is not much communication between client and server in this system. It doesn't need to consider.

4.3.4. Summary of the Applicability

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

4.4. [Step 3b] Applicability of 'Candidate 2. Batch-Sequential Architecture Style'

The Batch-Sequential architecture Batch Sequential Architecture Style is used when independent components operate sequentially and manipulate data.

4.4.1. Match on the Applicable Situation

Applicable Situations	Match	Demands on the System
System consists of independent components manipulating data.	❍	During SThT task, recording or uploading raw data, text transform with STT Service, generating handwritten text will be proceed sequentially.
The components are connected sequentially ordered by data flow.	❍	

No interaction with user during data manipulation.	❍	There is no need to interact between system and user during the task.

4.4.2. Potential to benefit the *Pros*

Advantages of the Style	Match	Benefits Applicable to the System
High maintainability	❍	Each component is functionally separated. (SRP)

4.4.3. Potential to handle the *Cons*

Cons of the Style	Match	Handling on Cons on the System
After all the data is ready, the user can get the result.	Δ	Users do not need to check the data until the result is available in SThT process.

4.4.4. Summary of the Applicability

Batch-Sequential Architecture style is well applicable to the target system with the justification made above. There is no significant issue which prevents the application of this style.

4.5. [Step 3c] Applicability of 'Candidate 3. MVC Architecture Style'

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

4.5.1. Match on the Applicable Situation

Applicable Situations	Match	Demands on the System
Systems can be divided by 3 parts: user interaction, control, persistent data manipulation.	❍	Mobile app and server are divided by 3 parts. This division of the system is efficient in management.
The system is highly user-interactive	❍	Many parts of the system are menu driven behavior.
Processes need to be modeled independently from DB data	❍	The main functions of the system are related to ML training and the use of STT service, so it is often carried out separately from the DB.

4.5.2. Potential to benefit the *Pros*

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

4.5.3. Potential to handle the *Cons*

Cons of the Style	Match	Handling on Cons on the System
Design complexity increases.	❍	The advantages of maintenance and area expansion outweigh the difficulties in design.

4.5.4. Summary of the Applicability

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.

4.6. [Step 3c] Applicability of 'Candidate 4. Dispatcher Architecture Style'

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

4.6.1. Match on the Applicable Situation

Applicable Situations	Match	Demands on the System
Need replicate Servers for Reliability	❍	This system requires a reliable server and can install multiple servers.
The client program needs to interact directly with a server with a high QoS.	❍	The server handles main business logic, so achieving high QoS is very important.

4.6.2. Potential to benefit the *Pros*

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	❍	The performance should be good Because of many resource-intensive tasks of the System.

4.6.3. Potential to handle the *Cons*

Cons of the Style	Match	Handling on Cons on the System
Difficulty of changing dispatcher interface.	❍	The advantages of maintenance and area expansion outweigh the difficulties in design.

4.6.4. Summary of the Applicability

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.

4.7. [Step 4] List of Selected Architecture Styles

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

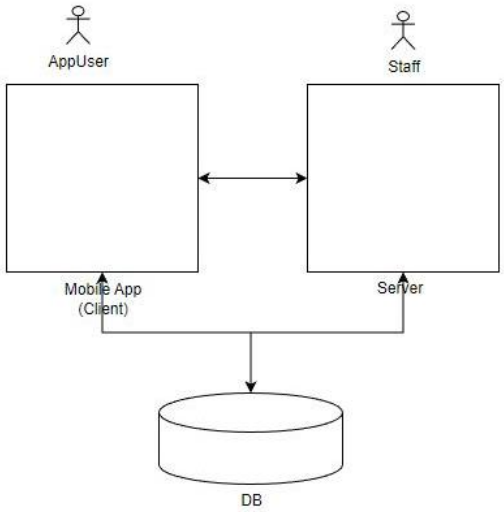
- ❑ Client-Server Architecture Style
- ❑ Batch-Sequential Architecture Style
- ❑ MVC Architecture Style
- ❑ Dispatcher Architecture Style

4.8. [Step 5] Integrating Architecture Styles

We apply the selected architecture styles incrementally.

4.8.1. Applying Client-Server Architecture Style

We define 2 tiers for the target system by considering the sub-functionalities The whole system consists of two tiers, and is divided into mobile app, which is a client, and a development server.



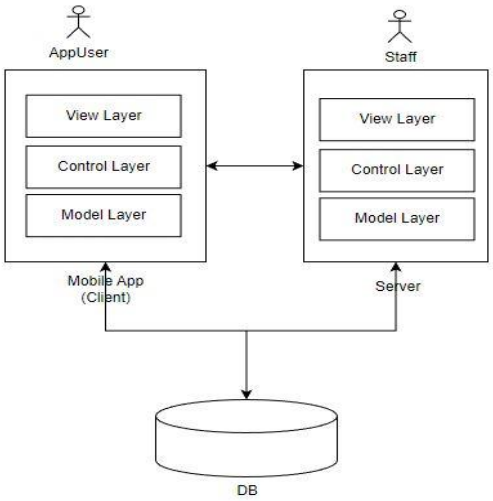
- ❑ Mobile App Tier

  This tier is to run a mobile app for users to access the server and DB.

- ❑ Server Tier

  This tier is to manage core functions such as ML training and SThT Service.

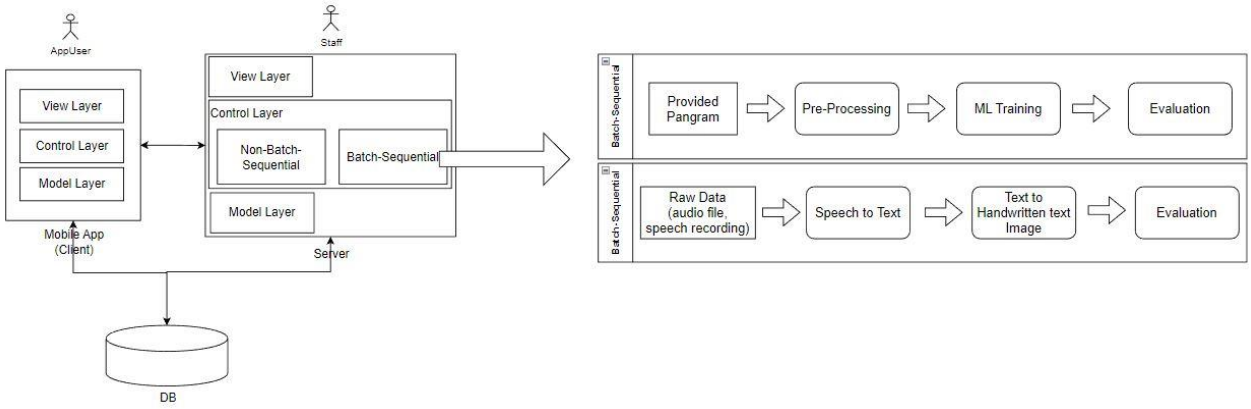
4.8.2. Applying MVC Architecture Style



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

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

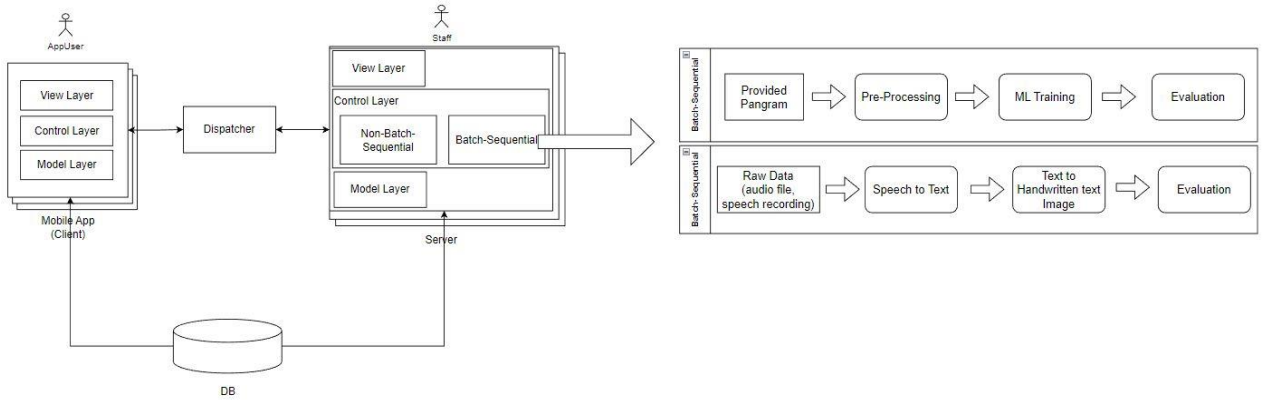
4.8.3. Applying Batch-Sequential Architecture Style



Batch-Sequential Architecture is mainly applied to ML model processing, and it unify a series of sequential tasks.

- ❑ Handwritten model training process.
- ❑ SThT Process.

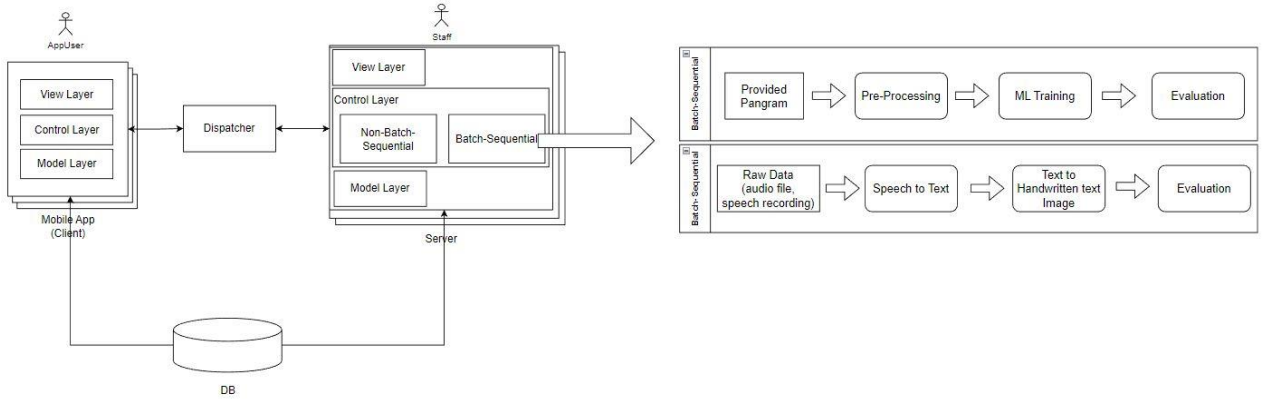
4.8.4. Applying Dispatcher Architecture Style



The system applied Dispatcher Architecture for a reliable server. To apply this architecture, you need to add an extra server and develop a Dispatcher. If the cloud server (Paas) can be used, the corresponding architectural style can be excluded from the design.

4.8.5. Resulting Skeleton Architecture

The resulting skeleton architecture of applying all the selected architecture styles is shown in the following figure.



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

4.8.6. Interactions of the Skeleton Architecture

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

❑ Interaction Paths derived from the Styles

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

❑ Additional Interaction Paths

None

4.9. [Step 6] Strength and Limitations of the Skeleton Architecture

4.9.1. Strengths

Specify the advantages of the proposed skeleton architecture.

❑ Separation of Concern

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

❑ Complexity of the System Design and Implementation

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

❑ High Maintainability

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

4.9.2. Drawbacks

Specify the drawbacks and risks of the proposed skeleton architecture.

❑ Not Anticipated.

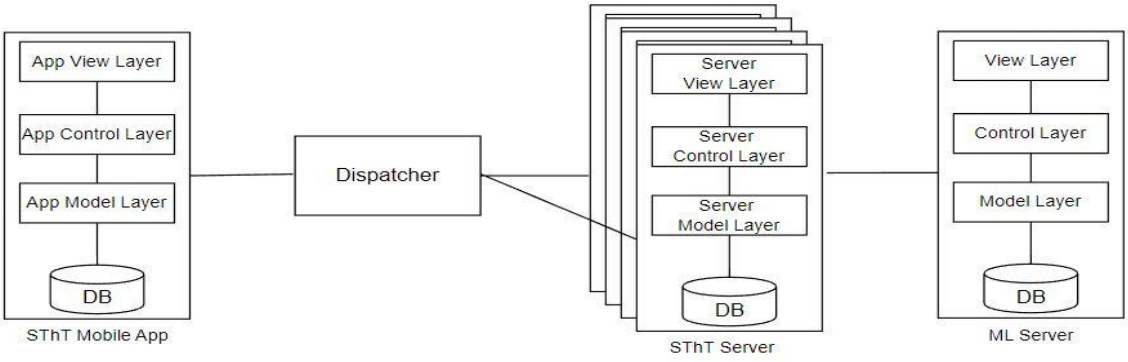
4.9.3. [REVISED] Skeleton Architecture

For operating Machine Learning functions in this system stably, A separate server is needed. And batch-sequential architecture is excluded because the data flow of the machine learning function is not complicated.

❑ N-Tier Architecture Style

❑ MVC Architecture Style

❑ Dispatcher Architecture Style



5. Activity 4. View-specific Architecture Design

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

5.1. Functional View

5.1.1. [Step 1] Observe Functional Characteristics

- Functionality of Managing Various Profiles

  The target system has two kinds of profiles to manage; users, staffs.

- Functionality of Managing Handwriting Training.

  The target system makes handwriting model training from pangram of users.

- Functionality of Transforming speech to handwrite text.

  The target system has function for transforming speech to handwrite text.

- Functionality of Personalized Font Generation.

  The target system generates personalized font from trained handwriting model

- Functionality of Billing and Payment Management.

  The target system charges a certain amount of money for font creation and proceeds with the payment.

- Functionality of Generating Reports.

  The target system creates a periodic report.

5.1.2. [Step 2] Refine Use Case Diagram

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

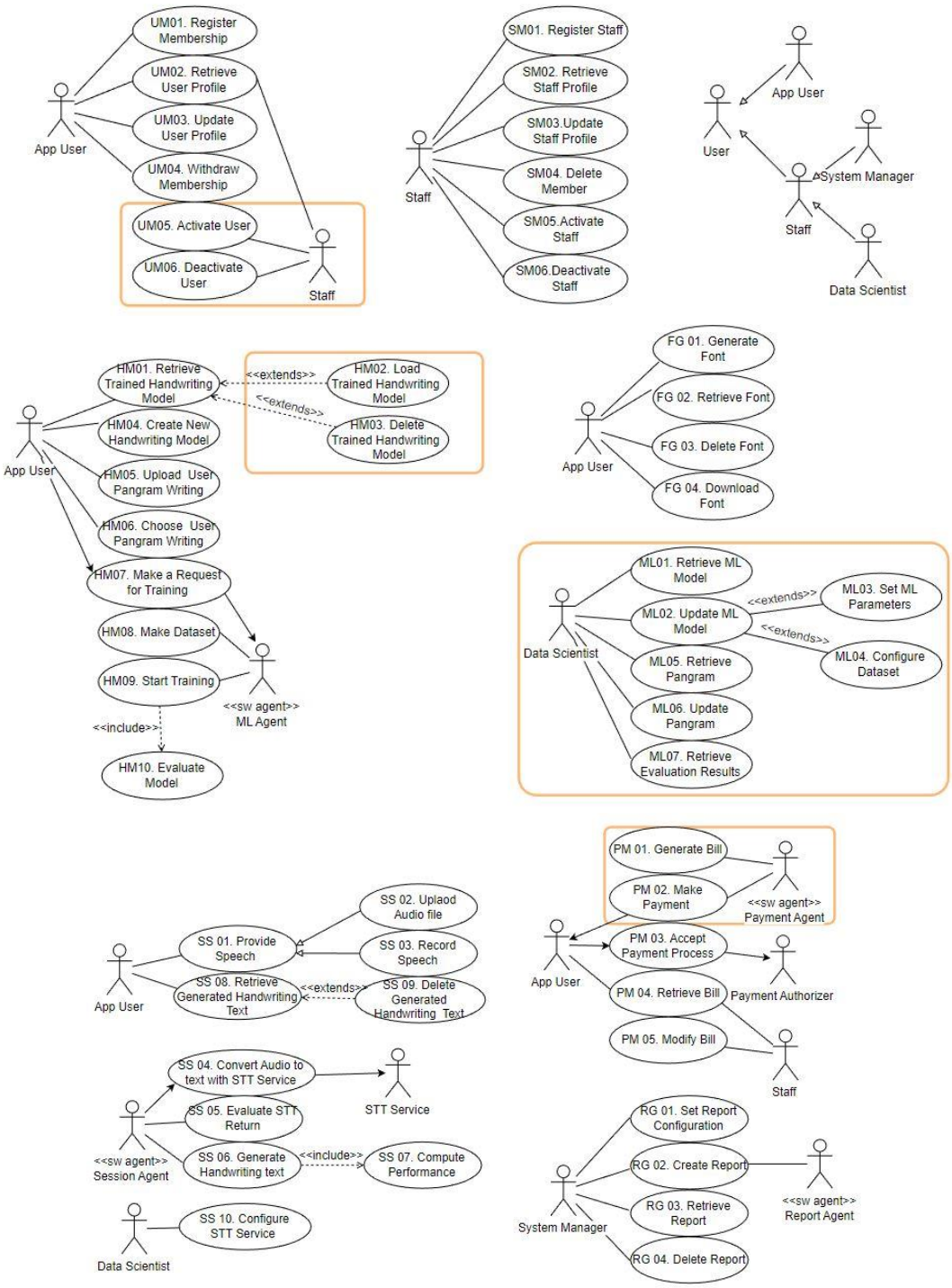
- Functional Groups

  We use the same set of functional groups defined earlier.

  - User Profile Management → UM
  - Staff Profile Management → SM
  - Handwriting Training Management → HM
  - Personalized Font Generation → FG
  - Billing and Payment Management → PM
  - SThT Session Management → SS
  - Report Generation → RG
  - ML Model Management → ML

❑ 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.



5.1.3. [Step 3]. Derive Functional Components

There are three categories of functional components to consider; SRS-derived components, skeleton architecture-derived components, and interface-centric functional components.

- Category 1. Functional Components derived from the SRS

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

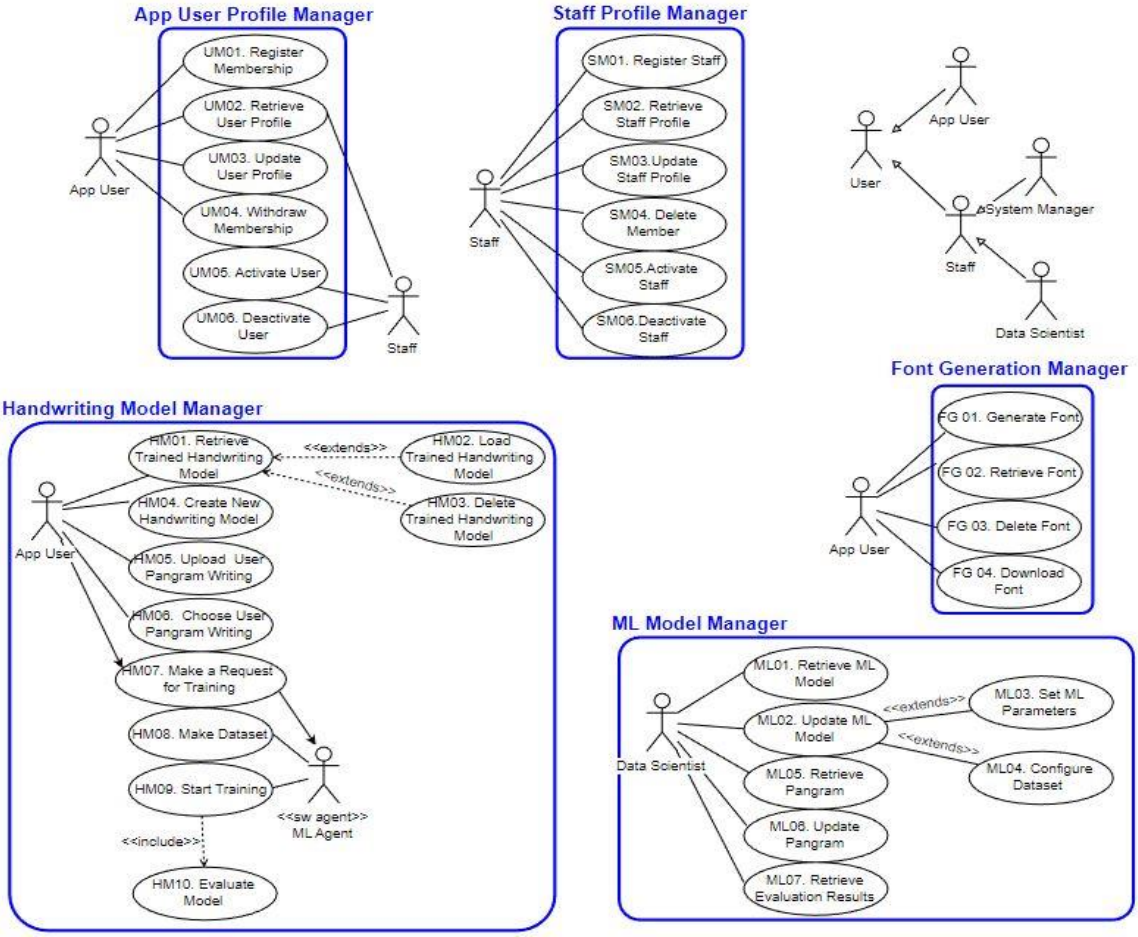
- Category 2. Functional Components derived from Skeleton Architecture

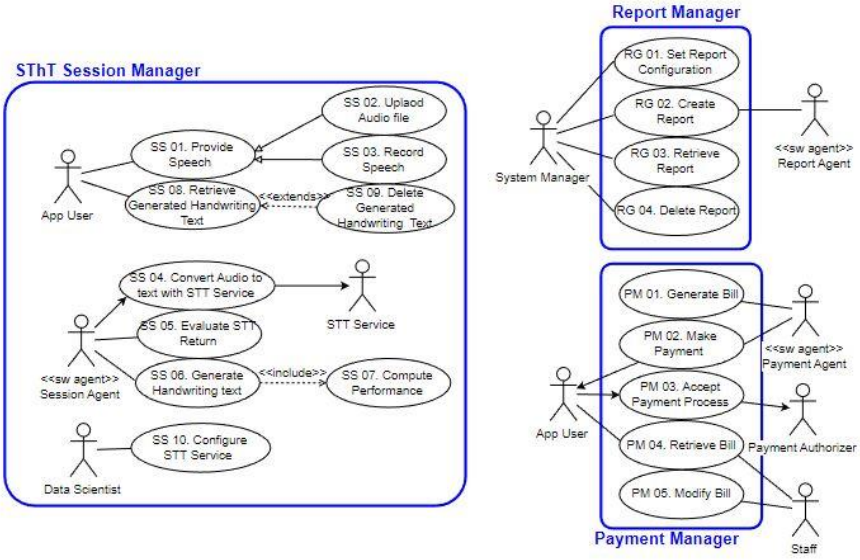
  A skeleton architecture is typically designed by applying architecture styles. An architecture style consists of components and connectors. Some of the components and connects may need to be modeled as functional components. That is, functional components could be derived from architecture styles in the skeleton architecture. Many of these functional components are pre-defined with a specific role in the architecture.

- Category 3. Interface-centric Functional Components

  An interface-centric functional component specifies a stable and public interface, which will be realized/implemented. For example, *Hardware Abstraction Layer (HAL)* is often adopted to designing architecture of embedded systems which interact with hardware devices such as sensors and actuators. Here, the *HAL* layer only specifies the interfaces of hardware devices in the system, and the layer itself does not include any implementation. Rather, its lower layer, *Hardware Adapter Layer*, implements the HAL for the given specific hardware devices.

- Functional Components derived from SRS





- Table of Functional Components derived from SRS

  This table shows the functional components of the system and their relevant use cases. It basically is the tabular representation of the use case clustering.

Functional Components	Use Cases
App User Profile Manager	UM 01. Create User Profile UM 02. Retrieve User Profile UM 03. Update User Profile UM 04. Delete User Profile UM 05. Activate User UM 06. Deactivate User
Staff Profile Manager	SM 01. Create Staff Profile SM 02. Retrieve Staff Profile SM 03. Update Staff Profile SM 04. Delete Staff Profile SM 05. Activate Staff SM 06. Deactivate Staff
Handwriting Model Manager	HM 01. Retrieve Trained Handwriting Model HM 02. Load Trained Handwriting Model HM 03. Delete Trained Handwriting Model HM 04. Create New Handwriting Model HM 05. Upload User Pangram Writing HM 06. Choose User Pangram Writing HM 07. Make a Request for Training HM 08. Make Dataset HM 09. Start Training HM 10. Evaluate Model

Font Generation Manager	FG 01. Generate Font FG 02. Retrieve Font FG 03. Delete Font FG 04. Download Font
Payment Manager	PM 01. Generate Bill PM 02. Make Payment PM 03. Accept Payment Process PM 04. Retrieve Bill PM 05. Modify Bill
SThT Session Manager	SS 01. Provide Speech SS 02. Upload Audio file SS 03. Record Speech SS 04. Convert Audio to text with STT Service SS 05. Evaluate STT Return SS 06. Generate Handwritten Text SS 07. Compute Performance SS 08. Retrieve Generated Handwriting text SS 09. Compute Performance SS 10. Configure STT Service
Report Manager	RG 01. Set Report Configuration RG 02. Create Report RG 03. Retrieve Report RG 04. Delete Report
ML Model Manager	ML 01. Retrieve ML Model ML 02. Update ML Model ML 03. Configure Dataset ML 04. Set ML Parameters ML 05. Retrieve Pangram ML 06. Update Pangram ML 07. Retrieve Evaluation Results

- Functional Components derived from Skeleton Architecture

  Skeleton Architecture includes Dispatcher for QoS. Dispatcher consists of QoS Evaluator and Server Allocator.

  - QoS Evaluator evaluates each SThT Server's QoS
  - Server Allocator allocates optimal QoS SThT Server to SThT App.

5.1.4. [Step 4] Refine Functional Components for Tiers

The skeleton architecture of the target system has the following tiers, and hence we refine functional components for the tiers.

Functional Components	SThT Mobile App	SThT Server	ML Server
App User Profile Manager	cApp User Profile Manager	sApp User Profile Manager	
Staff Profile Manager		sStaff Profile Manager	mStaff Profile Manager
Handwriting Model Manager	cHandwriting Model Manager	sHandwriting Model Manager	mHandwriting Model Manager
Font Generation Manager	cFont Generation Manager	sFont Generation Manager	mFont Generation Manager
Payment Manager	cPayment Manager	sPayment Manager	
SThT Session Manager	cSThT Session Manager	sSThT Session Manager	mSThT Session Manager
Report Manager		Report Manager	
ML Model Manager			ML Model Manager

- Description of Functional Components (for 3 Tiers)

  Each functional component is described for its key functionality. In CEP, we provide a brief textual description for each component.

  Some components are allocated to both tiers, and their functionalities would partially be different.

  - cApp User Profile Manager

    This component provides the functionality of managing the App user profiles for App users.

  - sApp User Profile Manager

    This component provides the functionality of managing the App user profiles for System manager.

  - mApp User Profile Manager

    This component provides the functionality of managing the App user profiles for Data Scientist.

  - Staff Profile Manager

    This component provides the functionality of managing the profiles of staff who works for system maintenance and data analysis.

  - cHandwriting Model Manager

    This component provides handwriting model generation functions for app users.

  - sHandwriting Model Manager

    This component provides a connection for ML training between Mobile app and ML server.

  - mHandwriting Model Manager

    This component executes functions for training Handwriting ML model with handwriting dataset.

  - cFont Generation Manager

    This component provides the functionality of managing font generation for App users.

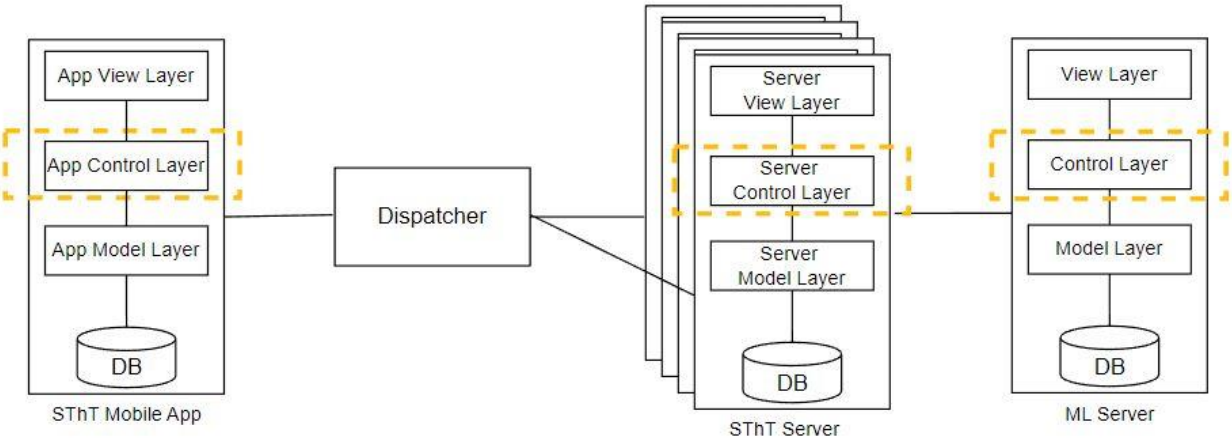
- sFont Generation Manager
  This component generates a personalized font files.
- mFont Generation Manager
  This component generates a personalized font from handwriting model.
- cPayment Manager
  This component provides the functionality of managing payment for App users.
- sPayment Manager
  This component provides transaction to credit card payment authorization service.
- cSThT Session Manager
  This component provides the functionality of SThT service for app user.
- sSThT Session Manager
  This component transforms speech to text and communicate with ML server.
- mSThT Session Manager
  This component generates a handwriting text image from transformed text.
- Report Manager
  This component provides the functionality of managing report generation for staff.
- ML Model Manager
  This component provides the functionality of managing ML model configuration.

- Interface-centric Functional Components
  - None

5.1.5. [Step 5] Allocate Functional Components

Allocate functional components onto 'functionality place holders' of the skeleton architecture. A functional place holder is a layer, a partition, or any place which is defined to host some functionality.

❑ Place Holders for Functional Components

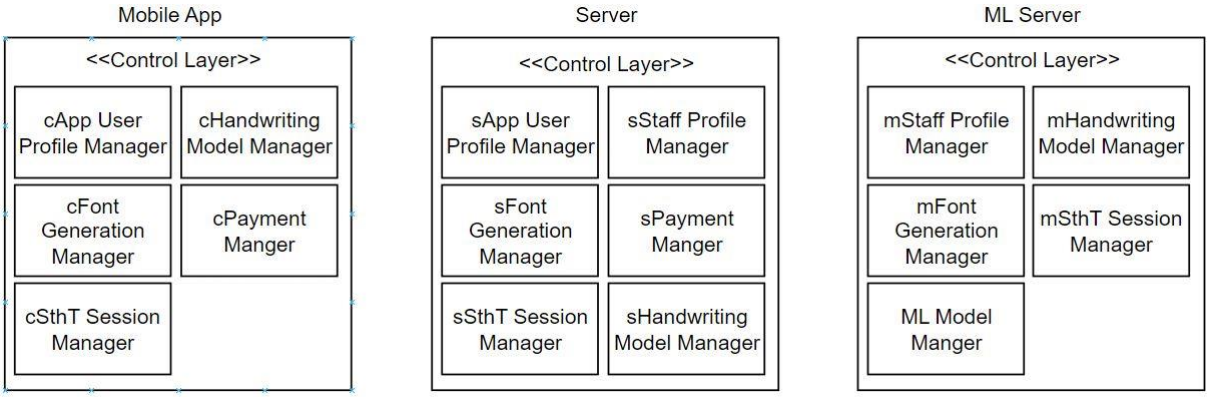


The control layer of each tier becomes the functionality place holders, as shown in the figure.

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

❑ Functional Components allocated

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



Each place holder is assigned with several functional components.

The skeleton architecture is now more complete with functional components.

5.1.6. [Step 6] Design Functional Components

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

- ❑ Criterion 1. Visibility of Functional Component

  Whitebox Component or Blackbox Component

- ❑ Criterion 2. Type of Interface (for Blackbox Components)

  Façade-type or Mediator-type

- ❑ Criterion 3. Variability of Functional Component

  Closed Component or Open Component

- ❑ Criterion 4. Variation Points (for Open Component)

  Variation point where the variability occurs.

We use the table of Functional Component Design Decisions as the following.

Refined Functional Components	Visibility	Interface Type	Open/Closed	Variant Points
cApp User Profile Manager	Blackbox	Façade	Closed	N/A
cHandwriting Model Manager	Blackbox	Façade	Closed	N/A
cFont Generation Manager	Blackbox	Façade	Closed	N/A
cPayment Manager	Blackbox	Façade	Closed	N/A
cSThT Session Manager	Blackbox	Façade	Closed	N/A
sApp User Profile Manager	Blackbox	Façade	Closed	N/A
sStaff Profile Manager	Blackbox	Façade	Closed	N/A
sHandwriting Model Manager	Blackbox	Façade	Closed	N/A
sFont Generation Manager	Blackbox	Façade	Closed	N/A
sPayment Manager	Blackbox	Façade	Open	Authorization Service
sSThT Session Manager	Blackbox	Façade	Open	STT Service
Report Manager	Blackbox	Façade	Closed	N/A
mStaff Profile Manager	Blackbox	Façade	Closed	N/A
mHandwriting Model Manager	Blackbox	Façade	Closed	N/A
mFont Generation Manager	Blackbox	Façade	Closed	N/A
mSThT Session Manager	Blackbox	Façade	Closed	N/A
ML Model Manager	Blackbox	Façade	Open	Algorithm

- ❑ Whitebox Components

- None

❑ Designing Component with Openness

Some functional components are designated as open designs. Because external services and algorithms can be used in a variety of ways for performance.

5.1.7. [Step 7] Define Interfaces of Functional Components

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

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

Functional Components	Center System	ML Server	Archival Server
App User Profile Manager	icApp User Profile Manager	isApp User Profile Manager	
Staff Profile Manager		isStaff Profile Manager	imStaff Profile Manager
Handwriting Model Manager	icHandwriting Model Manager	isHandwriting Model Manager	imHandwriting Model Manager
Font Generation Manager	icFont Generation Manager	isFont Generation Manager	imFont Generation Manager
Payment Manager	icPayment Manager	isPayment Manager	
SThT Session Manager	icSThT Session Manager	isSThT Session Manager	imSThT Session Manager
Report Manager		iReport Manager	
ML Model Manager			iML Model Manager

❑ Interface of Components

In this CEP, we define the interface of one component, *'SThT Session Manager'*. The use cases included in the functional component are here;

- SS 01. Provide Speech
- SS 02. Upload Audio file
- SS 03. Record Speech
- SS 04. Convert Audio to text with STT Service
- SS 05. Evaluate STT Return
- SS 06. Generate Handwritten Text
- SS 07. Compute Performance
- SS 08. Retrieve Generated Handwriting text
- SS 09. Compute Performance
- SS 10. Configure STT Service

- Provided Interface of SThT Session Manager

  There are 3 interfaces of SThT Session Manager: icSThT Session Manager, isSThT Session Manager, imSThT Session Manager

  - icSThT Session Manager (Mobile App Tier)

    This interface is to get a speech material such as audio file and speech recording.

    - selectHandwritingModel(): void;

      This method is to request choice of handwriting model for handwriting text generation.

    - getSpeechMaterial(audio file): MP3 file; , getSpeechMaterial(recording):MP3 file

      Both methods are used to request speech material to App User. The return value is the mp3 file format available in the STT Service.

    - startSThTSession(): void;

      This method sends a message to the server to begin the session for whole SThT transforming sequence.

  - isSThT Session Manager (Server Tier)

    This interface communicates with STT Service for transforming speech to text.

    - transformSpeechtoText(): text;

      This method sets the configuration of the STT service, such as the subscription key, the audio input type, and the audio file name, and proceeds with the conversion to return text.

    - startTransformHandwritingText(): void;

      This method sends a message to the ML server to begin converting a text image into a handwritten text image.

  - imSThT Session Manager (ML Server Tier)

    This interface transforms return text of STT Service to a handwriting text with trained ML handwriting model.

    - transformHandwritingText(text): jpeg file

      This method generates handwriting text image with trained handwriting model.

    - evaluateGeneratedImage(): score;

      This method computes evaluation score of generated text image.

5.2. 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.

5.2.1. [Step 1] Observe Informational Characteristics

We first observe the informational characteristics of the target system.

❑ Profile-related Classes

There are several classes for capturing various profiles in SThT system.

❑ Data from SThT process

The SThT process requires a lot of data, and a lot of data comes out such as pangram image, generated text, handwriting image. From the user's point of view, all data should be maintained.

❑ Meta-information of Machine Learning Models

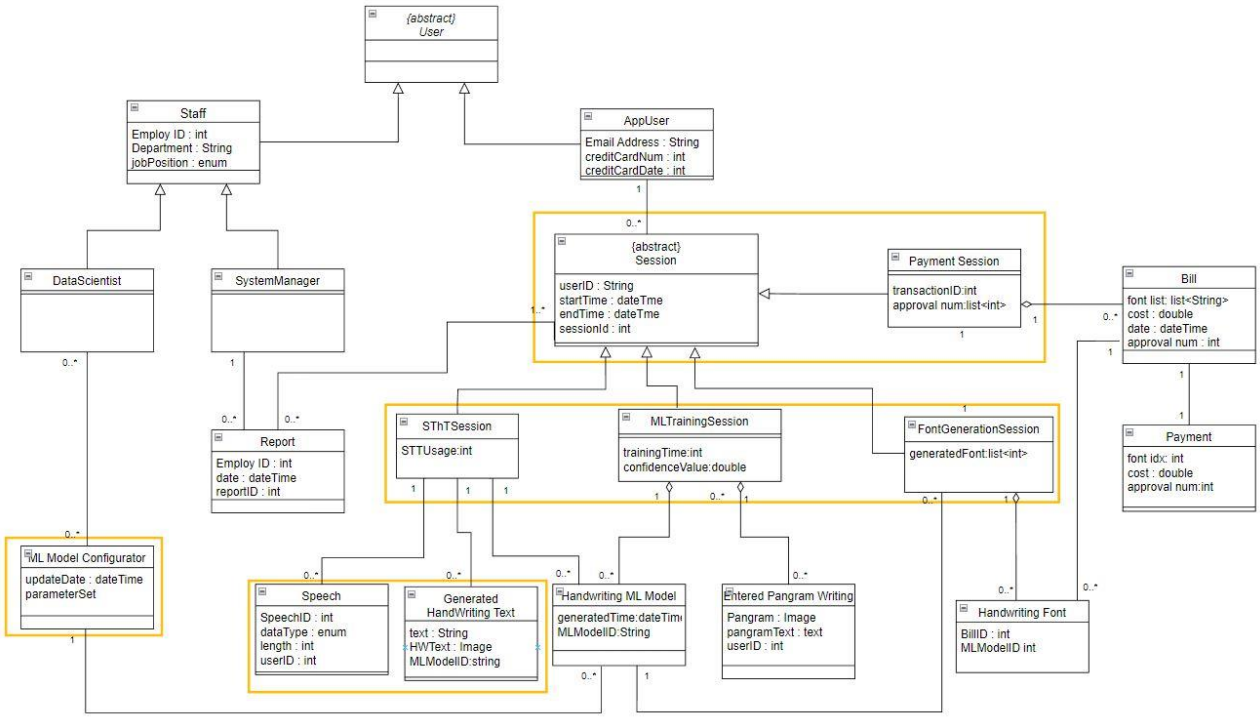
Since SThT System provides handwriting font generation using Convolutional Neutral Network model, the meta-information of the models should be captured in persistent classes.

5.2.2. [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 following diagram shows additional classes and refined relationships among classes.



❑ Refinements made on the Class Diagram

The following refinements are made;

- ❍ Adding Session of each main classes as subclasses of Session : Payment Session, SThT Session, ML Training Session, FontGeneration Session
- ❍ Adding *classes* of SThT Service Data: Speech, Generated Handwriting Text

- Adding key Attributes and Methods to Classes

  Each class is refined with key attributes and methods.

  - Omitted in Sample Solution

Each class is specified with a textual description.

- User Profile-related Classes

  User is the superclass which captures common property of its subclasses.

  - *App User* captures information of the user who wants to use SThT Service.
  - *Staff* captures information of the staff who manages SThT System. There are 2 subclasses divided depending on their job: Data Scientist, System Manager.

- Session classes

  Session is the superclass of other session classes which captures session information of main functions.

  - *SThT Session* captures information of SThT process. The other classes included in the STT process are Speech and Generated Handwriting text.
    - *Speech* captures original speech data which App user provides.
    - *Generated Handwriting Text* captures the result data from the SThT process
  - *MLTraining Session* captures information of ML Model and training data. The other classes included in the ML training Session are Handwriting ML Model and Entered Pangram Writing.
    - *Handwriting ML Model* captures information of trained ML model.
    - *Entered Pangram Writing* captures pangram images provided by App user.
  - *FontGeneration Session* captures information of generated handwriting fonts and its generation.
    - *Handwriting font* captures information of font generation.
  - *Payment Session* captures information of transactions with credit card authorization. The other classes included in the Payment Session are Bill and Payment.
    - *Bill* captures information of bill for payment generated by system.
    - *Payment captures information about payment details.*

- ML Model Configurator

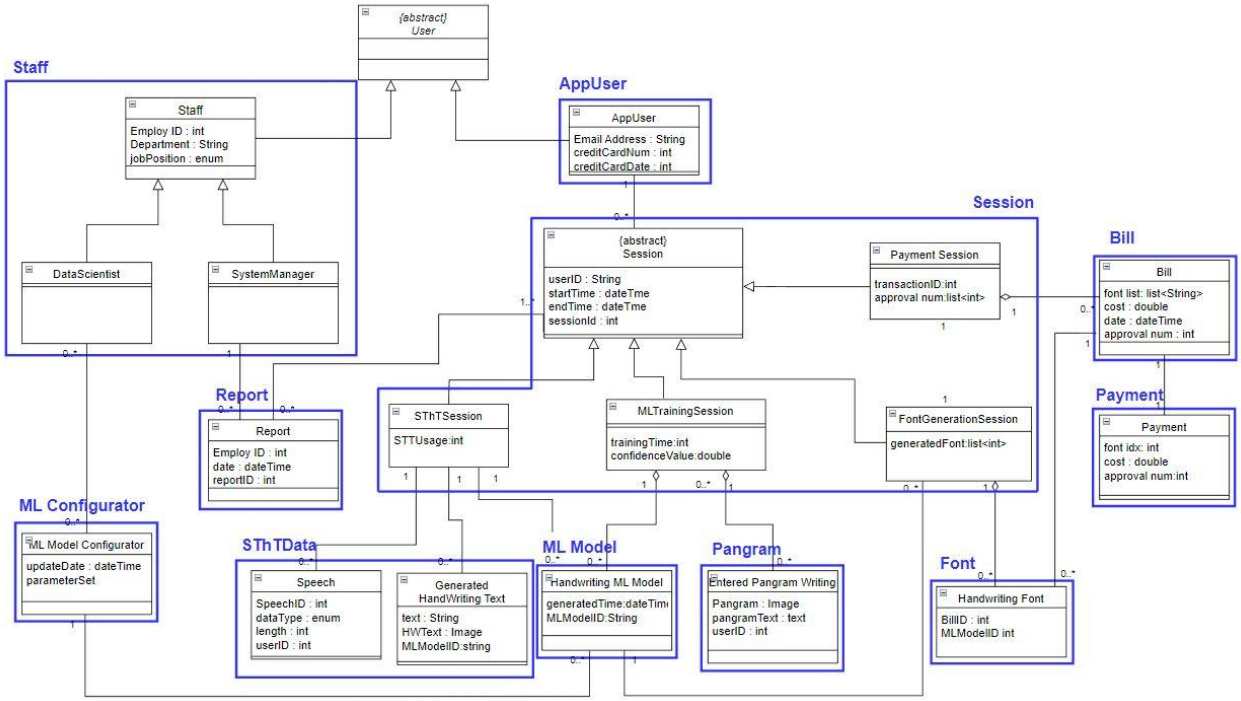
  This class captures the information of modify history about ML Model specification.

- Report Class

  This class captures the information of report.

5.2.3. [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 the figure, classes with strong dependency such as inheritance and compositions are grouped into a same data component.

- ❑ Data Components derived from Class Diagram/SRS

  There are several data components that are derived by considering the strengths of relationships among classes.

- ❑ Data Components derived from Skeleton Architecture

  None

5.2.4. [Step 4] Refine Data Components for Tiers

The skeleton architecture of the target system has multiple tiers, and hence we need to refine them for the tiers.

Data Components	SThT Mobile App	SThT Server	ML Server
AppUser	cAppUser	sAppUser	
Staff		sStaff	mStaff
Session	cSession	sSession	mSession
SThTData	cSThTData	sSThTData	mSThTData
MLModel		sMLModel	mMLModel
Pangram	cPangram	sPangram	mPangram
Font	cFont	sFont	
Bill	cBill	sBill	

Payment	cPayment	sPayment	
Report		Report	
ML Configurator			ML Configurator

- ❑ Data Components on *SThT Server*

  Most of the data components are allocated to *SThT Server*.

- ❑ Data Components allocated on multiple tiers

  When a data component is allocated on both tiers, the contents of the data components on the tiers are not same.

  - ❍ The data components of the *SThT server* store the necessary information from the *Server* or *ML Server* among the data generated by the Mobile App.
  - ❍ Data components of the *Mobile App* store data input from the user.
  - ❍ The data component of the *ML Server* stores data required for ML training or the resulting data.

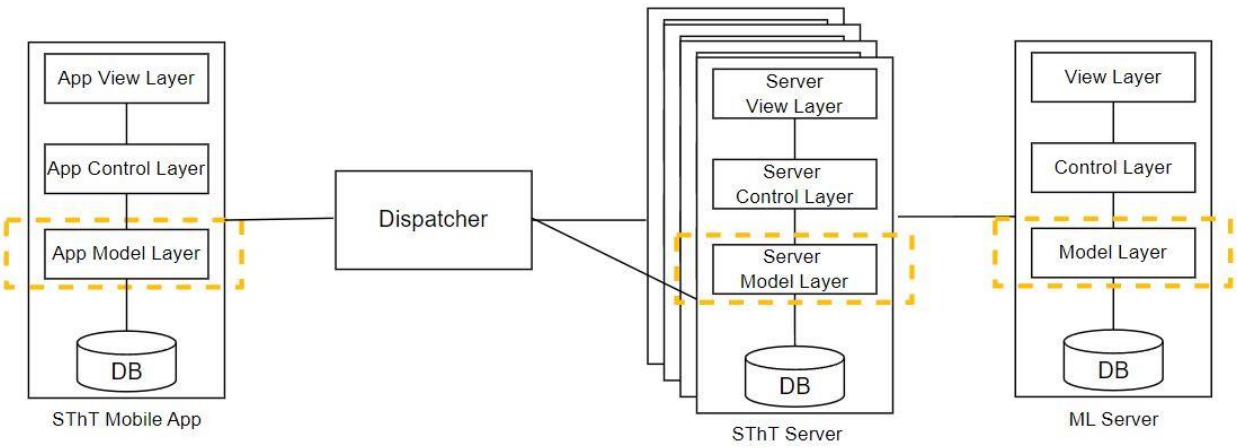
- ❑ Considering the Consistency with Allocation of Functional Components

  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.

5.2.5. [Step 5] Allocate Data Components

Allocate the data components to the appropriate place holders of the architecture

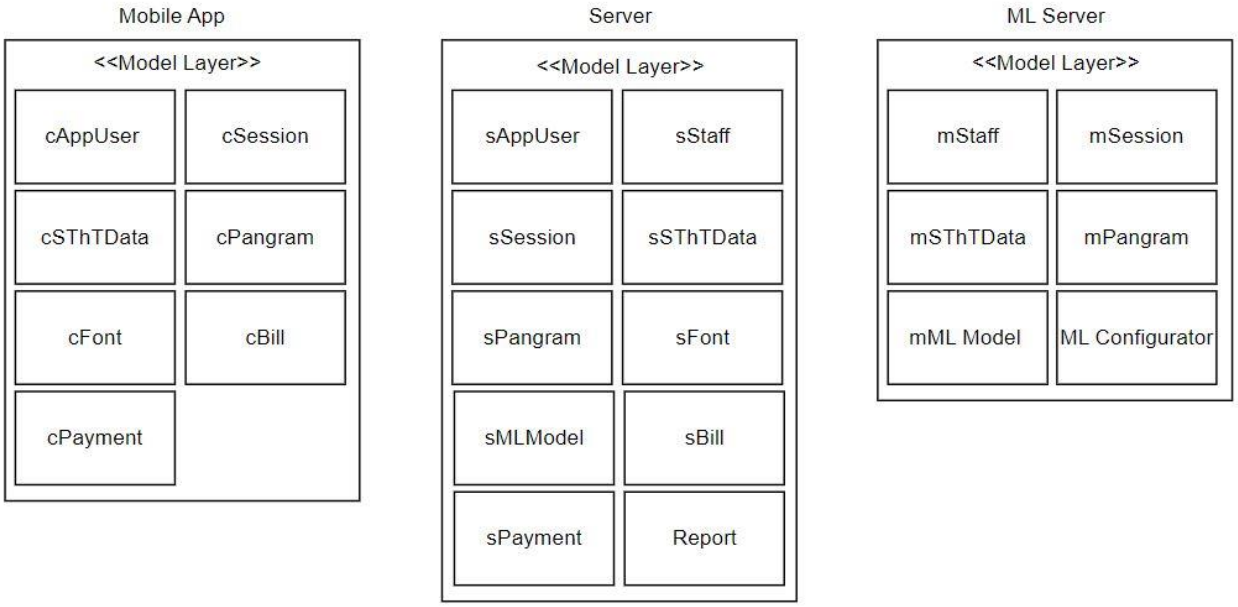
- ❑ Place Holders for Data Components



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

❑ Data Components allocated

Based on the table of refined data components for tiers, we allocate the data components on the client tier.



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.

5.2.6. [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

5.2.7. [Step 7] Define Interfaces of Data Components

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

❑ Omitted in Sample Solution

5.3. Behavioral View

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

5.3.1. [Step 1] Observe Behavioral Characteristics

We observe the following behavior of the target system.

- ❑ Need for Explicit Invocation-based Control Flow
- ❑ Event-driven Control Flow
- ❑ Parallel Processing to support Realtime Processing
- ❑ Closed-loop Control Flow for Treads
- ❑ Timer-based Invocation for Optimizing Machine Learning model (ML Server)

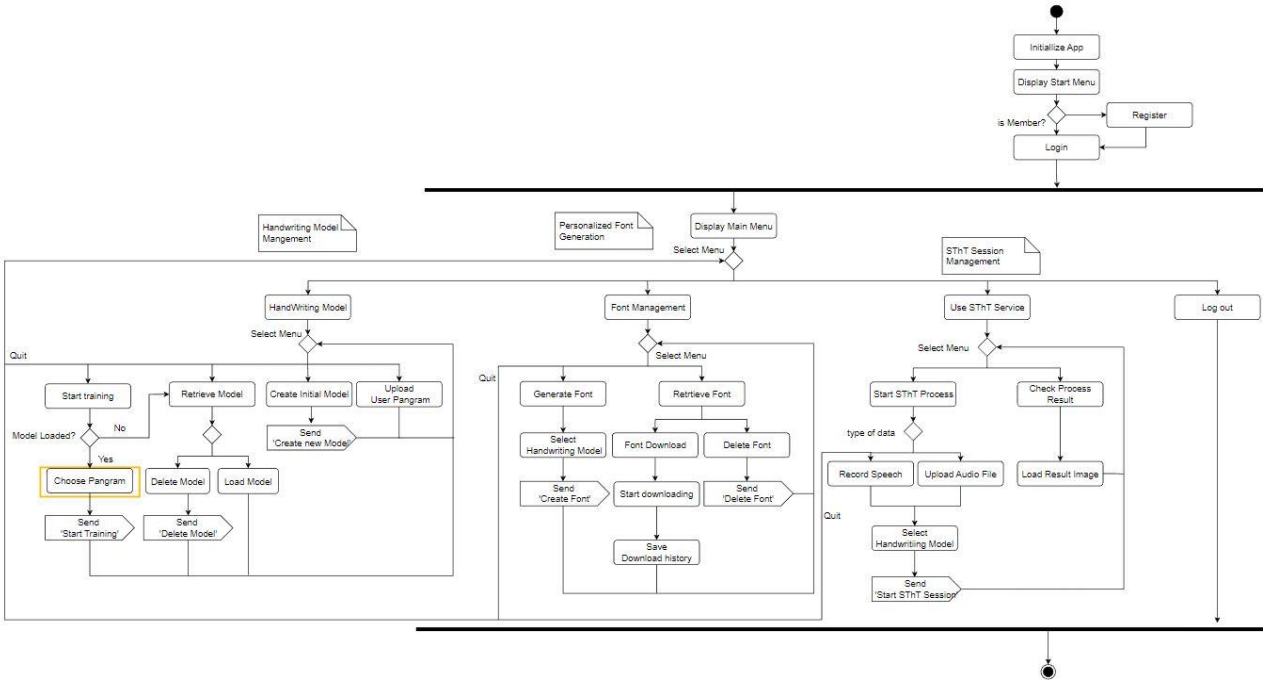
5.3.2. [Step 2] Refining Control Flow for whole System

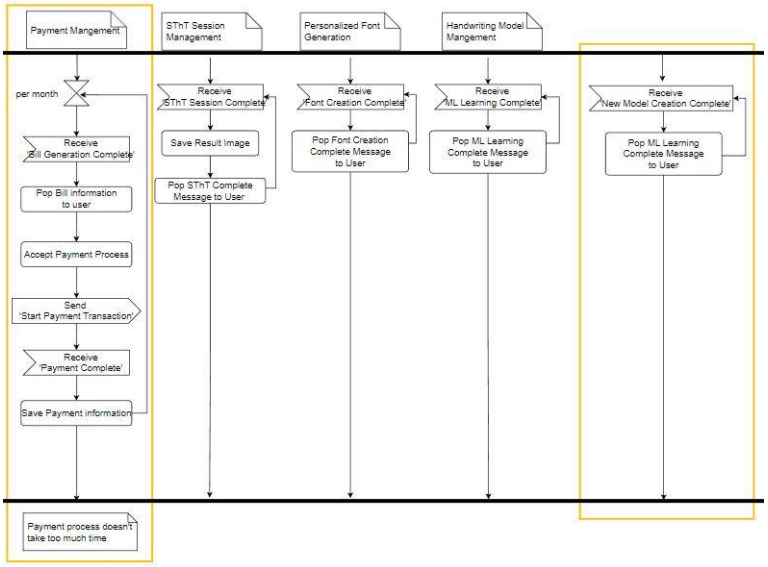
We design the overall control flow of the target system. This is done by refining the context-level activity diagram. If the target system has multiple tiers, each tier has its own control flow and the interaction between the tiers should also be designed.

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.

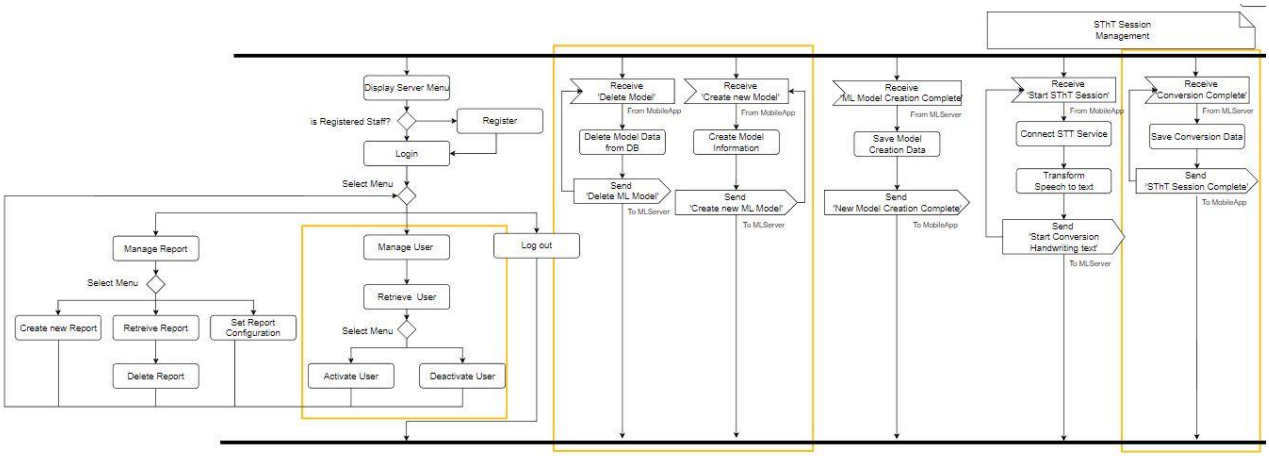
We specify the control flows of *Mobile App,Server and ML Server* in this CEP solution.

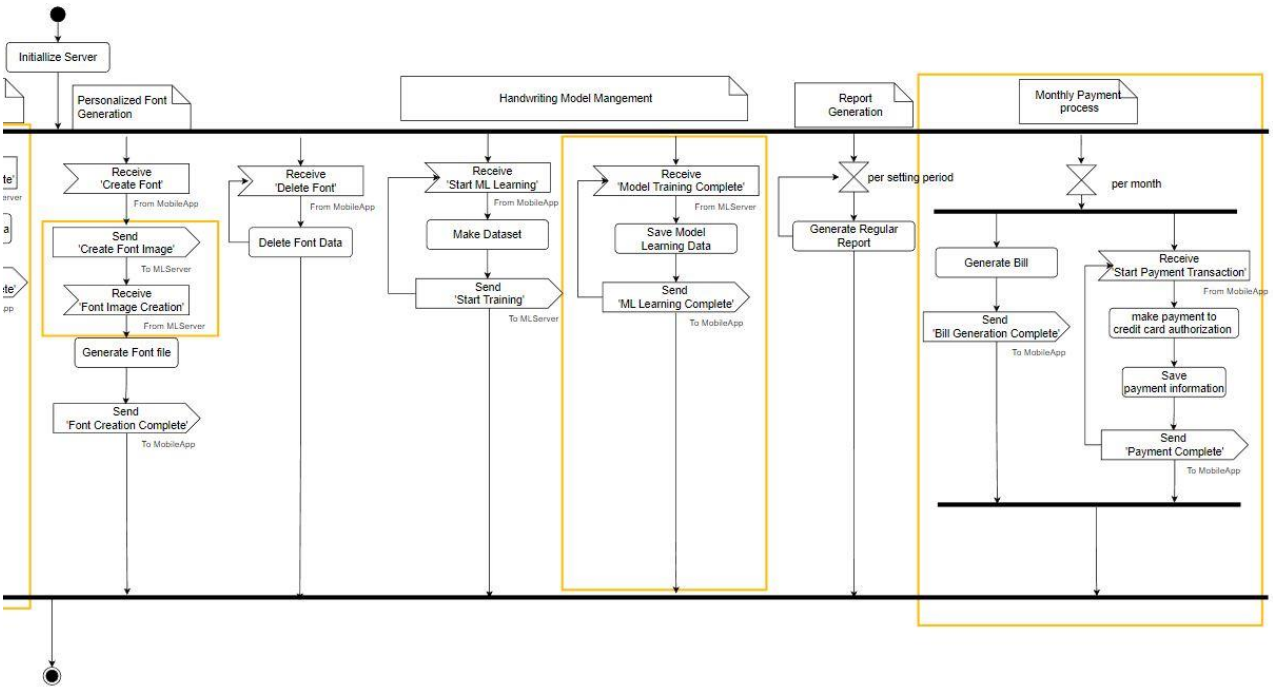
- ❑ Refined Control Flows of Mobile App



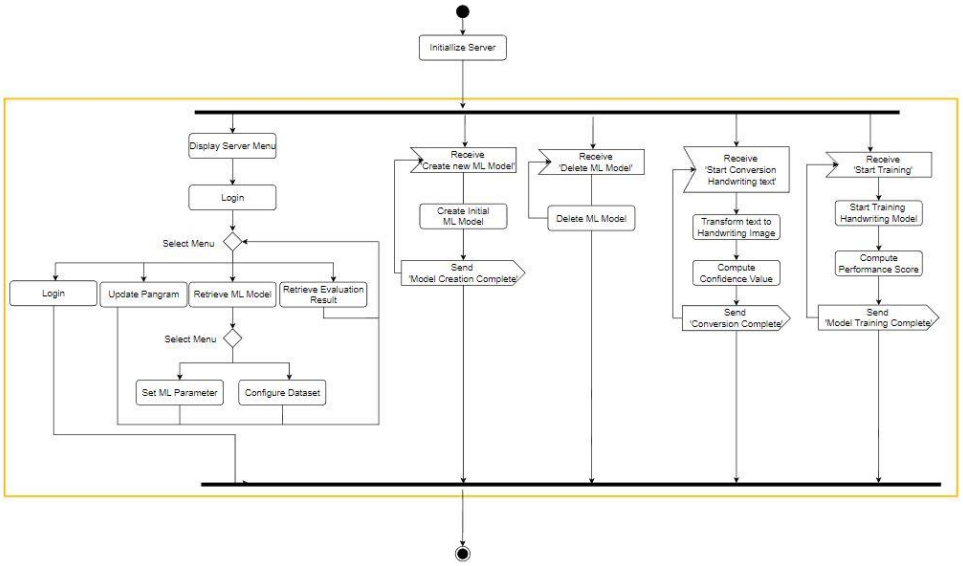


- Refined Control Flows of Server





❑ Refined Control Flows of MLServer

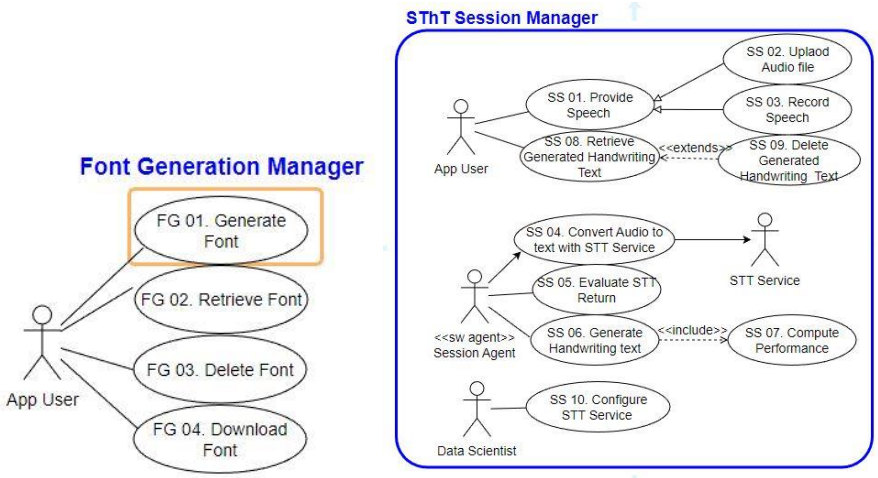


5.3.3. [Step 3] Choosing Elements for Detailed Control Flow

From the overall control flow, we choose some parts of the overall control for designing the detailed control flow. For CEP, we choose only one element for modeling the detailed control flow.

❑ "FG01. Generate Font" use case of Font Generation Manager

❑ "SThT Session Manager" Functional Component for Converting Speech to Handwriting Text.



5.3.4. [Step 4a] Defining Detailed Control Flow for Generate Font

We should design the detailed control flows for all the selected elements.

❑ Functionality of "FG01. Generate Font"

This use case generates personalized font from trained handwriting model. There are two main considerations for this use case.

❑ Detail Design Description of Generate Font

- Handwriting model

  This use case starts with trained handwriting model, so we don't need to consider about how to train the model. However, we should specify the details of the model.

  - Using GAN

    There will be numerous strategies for the accuracy of handwriting font generation, but we will use machine learning algorithms that are considered the most competitive among them. I think DCGAN is the best fit for this function. GAN have generative models that want to generate as real data as possible and classification models that want to distinguish between real and fake, respectively, learning against each other. DCGAN has the same concept as GAN, and it is a representative model that improves performance by changing GAN's fully connected layer to Convolution layer.

  - Generation character image with Generator

    The image file of each character is generated through the trained generator model.

  - Discriminating with Discriminator

    It measures the quality of the image generated by the discriminator. The quality can be confirmed through the loss value, and if it is 0.8 or more, it can be determined that there is no abnormality in the quality of the generated image.

- Font file Generation

  The output of this function is font file such as FNT, TTF, TTC. For generating Font file, we should fill the font template with each character images which is generated by handwriting model. In Python, fonts can be created through a library called fontmake. Fontmake is a library that changes UFO(Unified Font Object) to font files.

  - UFO

    It is an object that contains information necessary for generating Font. Information mapped to each letter and image file (.png) is put in a .glif file and managed as a list. It contains other details necessary for generating the font.

- Main Control Flow of Generate Font
  - Input: Generator Model, Discriminator Model, UFO Format
  - Output: Generated Font file

```
Begin
    Load Trained Generator and Discriminator Model
    For Number of Character
        While
            Generate character's Image with Generator
            Get the discriminating loss of image with Discriminator
            If(discriminating loss > 0.8)      // 1 is real, 0 is fake
                Save the Image as a fixed name of each character
                Break
```

End If

End While

End For

//Mapping is already done with template

Generate font file with fontmake library

End

5.3.5. [Step 4b] Defining Detailed Control Flow for 'SThT Session Manager'

We should design the detailed control flows for all the selected elements.

- Functionality of SThT Session Manager

  This functional component converts the input speech to text and converts the converted text to handwritten text.

- Detail Description of SThT Session Manager
  - Speech to text

    The speech input from the user is converted into text through the STT Service. The STT Service uses Microsoft's Azure STT service and backs up Google and Amazon's services. This is because MS azure provides the most free resources.

    - Input Speech file

      Speech is provided by users through audio files or recordings. For compatibility with STT Service, the format is limited to .mp3 or .wav.

    - STT Parameter Setting

      To use the STT Service, parameters such as authenticate key, region, language, audio input method, etc. must be entered.

  - Converted text to Handwriting text

    As mentioned above, we will use a handwriting model using Gan. In this step, the conversion process will be defined with a fully learned model already.

    - Handwriting text

      An image of text is generated through Gan's generation model. If text is generated using Font, it cannot be used on other platforms without Font. Therefore, it provides a handwriting font image.

    - Interpolation

      The system supplements according to the connectivity between letters in order to create cursive handwriting with a high consistency rate.

- Main Control Flow of 'Start Monitoring'

  We define the main control flow for the use case using the primitive methods.

  - Input : Speech audio file
  - Output : Handwriting text image
  - Begin

```
Get an audio file from user (.mp3 or .wav file)
Set STT Parameter
Convert audio file to text with STT Service

Load Trained Generator and Discriminator Model
For Words in text
    While
        For character in Words
            Generate Characters Image with Generator
            Combine with previous character
            Interpolation between characters
        End For
        Get a loss value with Discriminator
        If (loss value >0.8) break
    End While
End For
Return combined handwriting text image
End
```

5.4. 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.

5.4.1. [Step 1] Observe Deployment Characteristics

- ❑ The SThT Mobile tier will be realized with a large number of App users.
- ❑ The SThT Server tier will be realized with a small number of medium-sized servers.

  The number of servers is determined by the number of applicants, but the purpose of multiple servers is to ensure reliability, so it does not need to be large. If the number of users increases a lot during operation, it will be expanded.
- ❑ The ML server tier will be realized with a medium-sized servers, but it requires higher computing capability to process highly-complex machine learning algorithms.

5.4.2. [Step 2] Define Nodes

The skeleton architecture of the target system consists of three tiers. For deep learning development compatibility, all systems will be developed in Python languages.

- ❑ Node 1. SThT Mobile App

This node denotes a mobile app running on Android or iOS. Hence, the computation power and resources are already specified by the mobile OS and smart phone vendors.

- Node 2. SThT Server

  The server manages all major data and manages many transactions as a connection point between Mobile App-ML server. There is no heavy workload other than multi-threaded tasks for messaging, but many files such as audio files and image files need to be stored. Therefore, if sufficient storage space and CPU core are secured, other specifications may be configured as a mid-level server PC.

- Node 3. ML Server

  Since the system performs all tasks using machine learning on the ML server, high-end GPUs and CPUs are required. In particular, the training work is expected to take a lot of time, so GPUs are prepared in plenty of more than 4 units.

- Node 4. Dispatcher Server

  This node is deployed with a python-based web server.

5.4.3. [Step 3] Define Network Connectivity

- Between Server and Mobile App: HTTP protocol-based network configuration.
- Between Server and ML Server: HTTP protocol-based network configuration.

5.4.4. [Step 4] Define Artifacts to Deploy

- Artifacts for SThT Mobile App
  - Functional Components specified in Functional View-design
  - Data Components specified in Information View-design
  - Implementation of Behavior View-design
- Artifacts for SThT Server
  - Functional Components specified in Functional View-design
  - Data Components specified in Information View-design
  - Implementation of Behavior View-design
- Artifacts for ML Server
  - Functional Components specified in Functional View-design
  - Data Components specified in Information View-design
  - Implementation of Behavior View-design

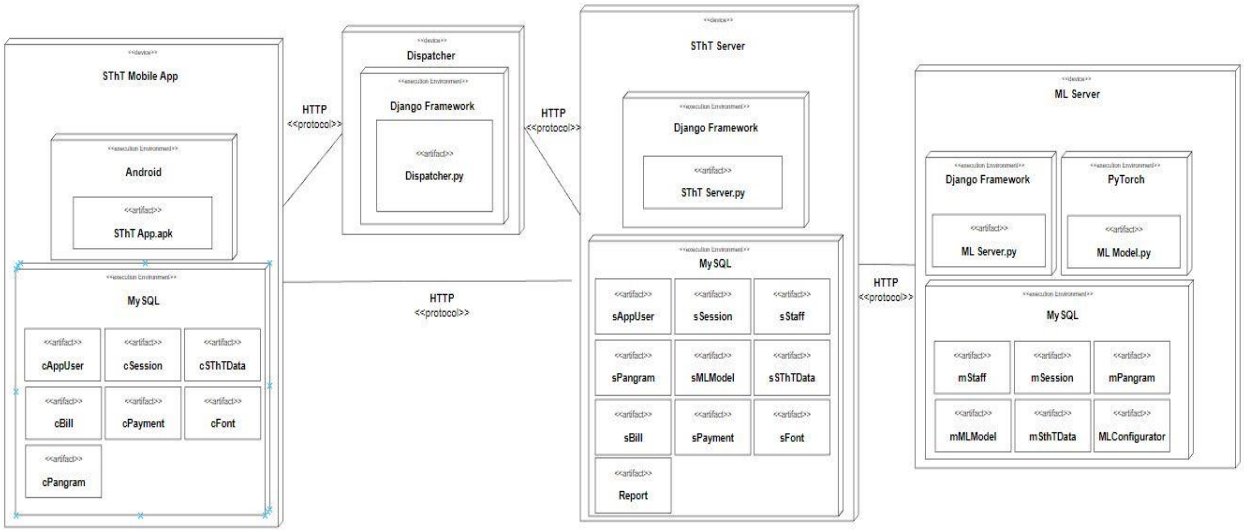
5.4.5. [Step 5] Allocate Artifacts on Nodes

Allocate all the deployable artifacts and show the network connections

- Deployment Diagram

  Allocate components and sub-systems to the appropriate place holders of the diagram as

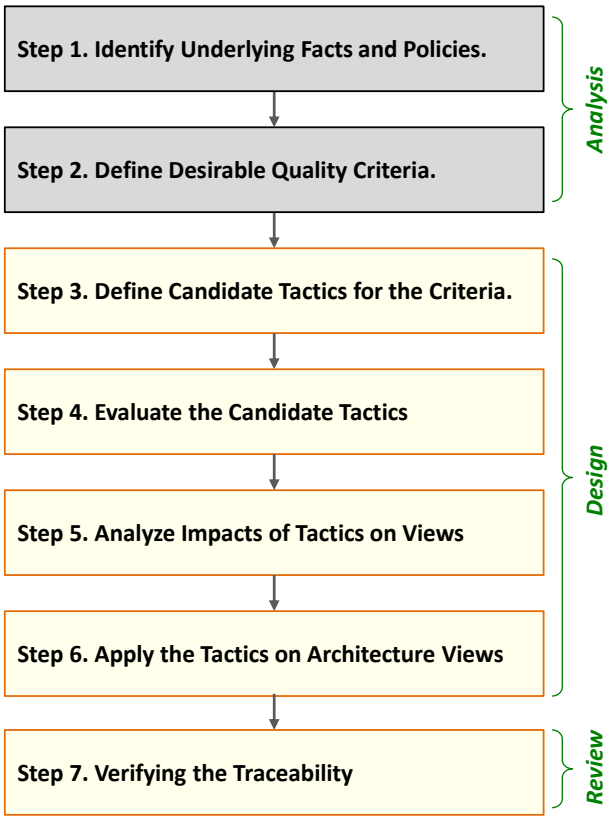
shown in the following diagram.



We define network connection between the tiers.

6. 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.



6.1. Design for NFR-1. Accuracy of Personalized Handwritten Texts

The system should be designed to provide a high accuracy of personalized handwritten texts. That is, the system has to learn the user's specific handwriting styles for each letter of the alphabet with a high accuracy.

Note that the handwriting of each letter may vary depending on the surrounding characters of a letter. That is, the way to writing a letter depends on the preceding letter and the subsequent letter. For example, The way to write a letter 'a' could vary depending on what the sequence of letters including 'a'.

- apple
- access
- axis
- diary
- jazz
- Kenya
- Corona

The system should capture all different writing styles of each letter from the users' handwritten texts.

In order to evaluate the accuracy, i.e., the performance of generated handwriting style, appropriate performance measures must be defined by the development team and the system should compute the accuracy by using the defined accuracy metrics.

6.1.1. [Step 1] Underlying Facts and Policies

We define the following facts and policies regarding the NFR.

- (F1) Learning the Personalized Handwriting with High Accuracy.

  There are various methods for learning User handwriting, such as machine learning and image processing. For high accuracy, machine learning has recently been widely used. Among them, DCGAN-based algorithms show high accuracy.

- (F2) High Quality and Variety of Dataset

  Successful training requires many high-quality datasets. This system needs many datasets that can represent changes in handwriting in various locations, as well as uppercase and lowercase letters of all alphabets. And Datasets with clearly exposed characteristic of handwriting are good for training.

- (F3) Evaluation of Generated Text Image

  It could be possible that a low-quality text image is generated through the system. If such a product is simply provided, the user's reliability will be reduced. If the system can evaluate the generated text image and manage it with scores, the reliability of the product can be maintained.

- (F4) Manage Generated Text Quality

  In general, the quality of text generated by ML algorithm is low. It is necessary to improve image quality by removing unnecessary lines or making the lines clear.

- (F5) Maintain ML Algorithms and Model

  ML Model can lose performance if people change over time. If the system automatically monitors the ML model and informs the staff of the performance degradation, it will be convenient to maintain.

6.1.2. [Step 2] Criteria for Satisfying NFR

- (C1) Using DCGAN-based algorithms for High Accuracy (Relevant to F1)

  The system should be designed to use DCGAN-based algorithms for High Accuracy. This is because many papers on handwriting learning have produced good results through DCGAN-based models.

- (C2) Collecting Variety of Dataset (Relevant to F2)

  Datasets should be collected to reflect all possible handwriting. The types of datasets to

be collected are as follows.

- ❍ Lower case of each letter
- ❍ Upper case of each letter
- ❍ Numbers

- ❑ (C3) Preprocessing Dataset for Effective Training (Relevant to F2)

  The system should preprocess datasets to secure high-quality datasets. Since the quality of the dataset during training also affects, it is necessary to create a clean image through filtering.

- ❑ (C4) Evaluation of Generated Text Image (Relevant to F3)

  The system should be able to evaluate the generated text image. Evaluation should be provided in Score form through Metric and supported by a convincing theory

- ❑ (C5) Postprocessing Generated Text Image (Relevant to F4)

  Text images generated by the system should be post-processed. The system should improve quality by eliminating unnecessary lines through filtering or by making the lines neat.

- ❑ (C6) Performance Monitor (Relevant to F5)

  The system should notify the staff if the score of the generated text image is continuously low. If the performance of the model itself is poor and does not produce good quality, Data Scientist should check this part and re-optimize it.

6.1.3. [Step 3] Candidate Tactics for the Criteria

The following tactics are proposed for the identified criteria.

- ❑ (T1) Applying ScrabbleGAN (Relevant to C1)

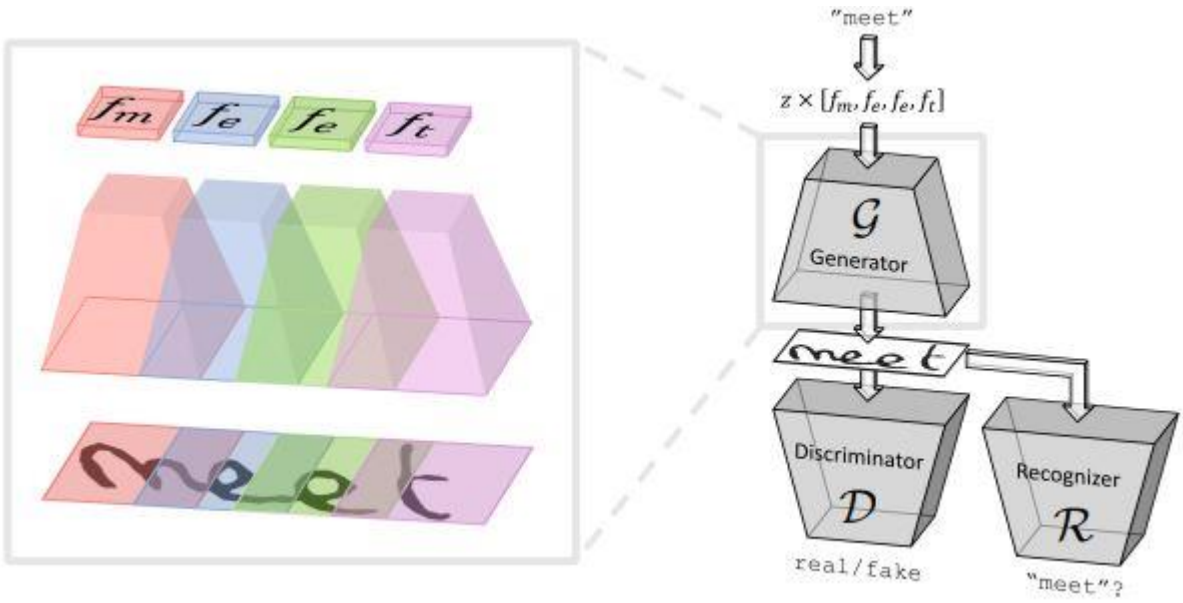
  The function of learning handwriting and making text is a high-level technique. Currently, ML Algorithm based on DCGAN, the latest technology related to text generation, should be used. Three possible options were identified by searching the paper.

  - ❍ Few Shot Learning (Component combination-based Algorithm)
  - ❍ SkelGAN, SE GAN (Skeleton analysis-based Algorithm)
  - ❍ **Scrabble GAN Algorithm (Area Capturing based) (Selected)**

  Component combination-based algorithm is a technique commonly known as new-shot learning. It is a method of generating text by selecting and recombining necessary components among the components of the pangram provided. Although this technique can be learned in a short time through a small number of pangrams, it has a problem of overfitting, and it is difficult to determine the association of consecutive letters.

  Skeleton analysis-based algorithm is a method of learning by constructing the framework(skeleton) of the characters provided. It is one of the best ways to provide personalized font, but it is difficult to reflect changes caused by surrounding letters.

ScrabbleGAN Algorithm is an algorithm captures and reflects some of the preceding and trailing characters of the letter. The algorithm comes from the inclusion of information on the entire word through LSTM, focusing on the fact that the factors affecting the actual letters are the preceding and trailing letters. The characters generated through this have different shapes depending on the surrounding characters, and this characteristic is consistent with the requirements of the system we define. So, We finally adopt this algorithm.



This algorithm slightly different from the structure of a typical GAN. First Difference is the second input of the generator. Each input character is filtered, including some regions of leading and trailing characters, classified via CNN, and then concatenated. This is the second input that comes with noise. Through this algorithm, output creates characters modified by surrounding letters.

Another point is the existence of a *Recognizer*. The adversarial learning between the generator and the discriminator is the same with typical GAN, but it introduces a Recognizer to help the generator learn generation. The generator conducts training so that the recognizer can recognize the generated text. See below for detailed structure and implementation.

- Paper : https://arxiv.org/abs/2003.10557 (ScrabbleGAN)
- Implementation : https://github.com/amzn/convolutional-handwriting-gan
- Structure of Generator and Discriminator: https://arxiv.org/pdf/1809.11096.pdf (BigGAN)
- Structure of Recognizer: GitHub - GitYCC/crnn-pytorch: Convolutional Recurrent Neural Network (CRNN) for image-based sequence recognition using Pytorch
  (Exclude bidirectional LSTM layer on this structure)

❑ (T2) Collecting Pangrams with Three types (Relevant to C2)

Acquiring various Pangram sentences is effective in training. The types of Pangrams that are widely used are as follows.

- The quick brown fox jumps over a(the) lazy dog.
- Pack my box with five dozen liquor jugs.
- Jackdaws love my big sphinx of quartz.
- The five boxing wizards jump quickly.
- Adjusting quiver and bow, Zompyc killed the fox
- Bright vixens jump; dozy fowl quack.
- Quick wafting zephyrs vex bold Jim.
- My faxed joke won a pager in the cable TV quiz show.
- Grumpy wizards make toxic brew for the evil Queen and Jack.
- Sphinx of black quartz, judge my vow.

As described in C3, a pangram sentence reflecting the following is needed for system learning.

- Lower case Text
- Upper case Text
- Normal case Text

When requesting a pangram once, the system requests three types of sentences: Lower case, Upper case, and Normal case (15 types in total). It doesn't matter if the pangram is duplicated, but if possible, it would be less objectionable for the user to request other pangram other than the previously entered pangram.

- (T3) Preprocessing Dataset (Relevant to C2)

The input Pangram must be preprocessed in a form that can be used for training.

- Word Tokenization

  Changes in letter style due to other letters are noticeable within words. When crossing the space between words, the position of the pen can be reliably initialized, but within the word, the previous character can affect the next character. Therefore, it is better to divide it into words.

- Image data Transform

  It recognizes image data and transforms it for training. The necessary functions are basic functions such as brightness, contrast, and size adjustment provided in libraries such as torchvision. Since image size needs to be filtered by letter, it is better to regularize it, and according to the GAN paper we will use, 16x32 pixels is the most suitable.

- Filtering (additional)

  If the system receives a pangram input through a photo or scan, adds filter that will eliminate unnecessary parts other than letters.

- (T4) Generated Text Image Evaluation with indicator (Relevant to C4)

The generated text image will be provided to the user. Therefore, quality must be guaranteed. The model presented in T1 can already determine whether it is the same as

an actual image and whether it is recognition as text through Discriminator and Recognizer. Therefore, only the loss values of the two modules can sufficiently evaluate the text image. At this time, the loss of the Discriminator and the Recognizer is closer to 1.

However, the system manages the Character Error Rate (CER) to check the accuracy of the text itself. CER is a number that is usually used in speech recognition, but it is a good index to manage the accuracy of the generated text. It can be easily obtained using the Levenshtein Distance algorithm. This algorithm is an algorithm that calculates how many times two strings need to be inserted, deleted, and replaced in order to be the same. It is suitable for quantifying similarity. The closer this result is to zero, the better, and if possible, it is better to manage it targeting zero.

When it is judged that the quality is low through evaluation, it is possible to recreate an image with a high-quality value and provide it to the user and increase the user's reliability. If low-quality text images are continuously produced, it should notify to the staff.

- (T5) Postprocessing Generated Text Image (Relevant to C5)

  The generated text image may be difficult to recognize or not neat due to unnecessary lines that are not organized. Post processing is carried out to deal with this part. Post processing is carried out before evaluation, along with simple brightness, contrast, and size adjustment, and additional filtering if necessary.

  In this part, it can also be changed to a desired image using Pix2Pix, a type of GAN. However, if there are many complex functions, it affects performance, so try to achieve quality through simple filtering if possible.

- (T6) Model Evaluation and Monitoring (Relevant to C6)

  ML Model needs continuous management. Evaluation indicators are necessary for model management, and a plan to monitor and maintain them should be prepared. Since the system can be maintained through data scientists, it is necessary to define an evaluation index.

  - Evaluation indicator

    There are two widely used indicators for evaluating the GAN model.

    - IS (Inception Score)

      It uses a pre-trained neural network to capture desirable properties of generated samples such as highly classifiable and diverse with respect to class labels. It measures the mean KL difference between the conditional label distribution p(y | x) of the sample and the marginal distribution p(y) obtained from all samples. IS shows a reasonable correlation with the quality and diversity of generated images. However, IS cannot detect overfitting. One model trapped in bad mode is not detected. Since IS uses an Inception model trained on ImageNet with many object classes, it may rather prefer a good object model that produces realistic images.

    - FID (Frechet Inception Distance)

      This metric is one of the metrics used to measure the feature distance between the actual image and the generated image. Frechet Distance is a method of

measuring the similarity between curves considering the position and order of points along the curve. It is also used to measure the distance between two distributions. FID is a method of calculating the distance between the set of products and the distribution of class data to be generated, so if the number of samples is insufficient, the reliability is low.

FID is an indicator created to complement IS, but it is reliable only when the number of samples generated exceeds 10000. Therefore, both indicators should be managed and interpreted according to the situation.

❍ Monitoring

The system manages IS and FID Score each time text is generated and informs the Data Scientist when this indicator does not improve below a certain number so that the ML Model can be re-optimized.

6.1.4. [Step 4] Evaluation of the Candidate Tactics

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

ID	Tactics	Y/N	Justification
T1	Applying ScrabbleGAN	Y	Benefit) It is a suitable idea to achieve the quality requirements of the system. Cost) Implementation is difficult and requires a lot of resources during operation. Decision) The cost is high, but it is the most important tactic.
T2	Collecting Pangrams with Three types	Y	Benefit) Various and important datasets can be obtained. Cost) The request is made to the user, but the cost is very small. Decision) It is an important tactic to achieve the quality of the system.
T3	Preprocessing Dataset	Y	Benefit) This tactic can improve the quality of the dataset and increase the learning degree of the ML Model. Cost) Implementation difficulty is not high and can be obtained through opensource. Decision) The cost is minimal, but benefits are high.
T4	Generated Text Image Evaluation with indicator	Y	Benefit) By managing the generated text as an indicator, quality management of the ML Model can be conveniently performed, and furthermore, user reliability can be improved. Cost) When low-quality text images are produced, there is a cost for repair, but there is no high cost for evaluation and indicator management. Decision) It is essential as a good tactic for managing potential risks.
T5	Postprocessing Generated Text Image	Y	Benefit) It can produce high-quality images that the user can satisfy, and that the system can permit. Cost) The cost is not high if you use basic functions, but the cost is high if you use complex functions.

			Decision) It is a tactic necessary for the user's satisfaction.
T6	Model Evaluation and Monitoring	Y	Benefit) It is possible to monitor for preparing and maintaining indicators that can evaluate the function of the model itself in the long term. Cost) Evaluation indicators are not complicated and easy to manage Decision) It is a necessary tactic for the management of the system.

6.1.5. [Step 5] Impact Analysis of Tactics on Views

We analyze the impacts of each selected tactic.

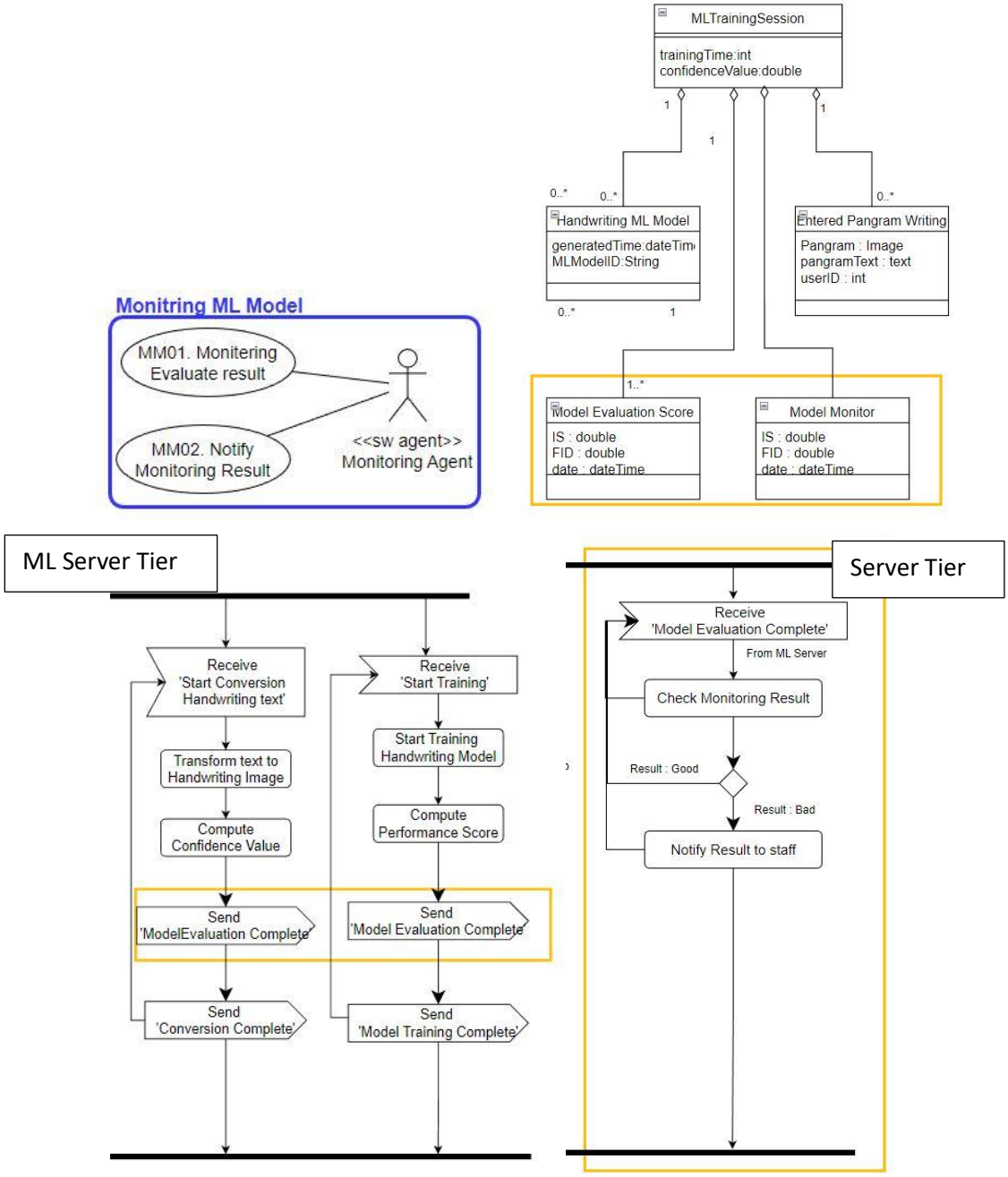
ID	Tactics	Functional View	Information View	Behavior View	Deployment View
T1	Applying ScrabbleGAN			Define Control flow of ML Training with Scrabble GAN	
T2	Collecting Pangrams with Three types	Segmentation of already reflected pangram input requests			
T3	Preprocessing Dataset	Define Preprocessing Functions		Add Preprocessing Control Flow between pangram provision and start ML training	
T4	Generated Text Image Evaluation with indicator		Add Text Evaluation Indicator	Add an Evaluation action after Post processing	
T5	Postprocessing Generated Text Image	Define Postprocessing Functions		Add Postprocessing Control Flow after ML training	
T6	Model Evaluation and Monitoring	Define 'Monitoring Evaluation results' function	Add ML Model Evaluation Indicator	Add Monitoring Thread.	

6.1.6. [Step 6] Architecture with Tactics Applied

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

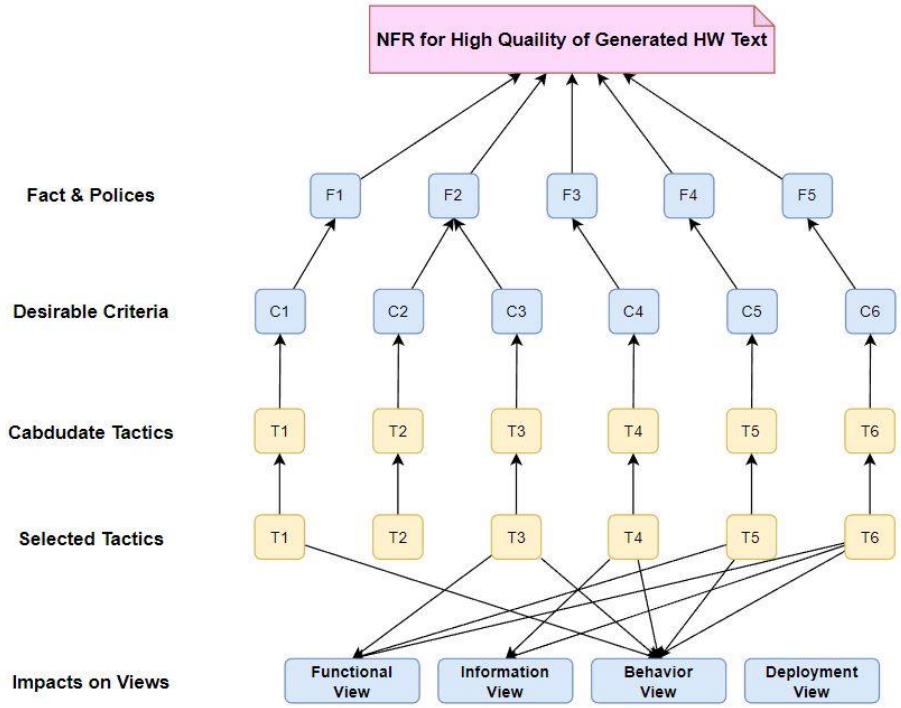
❑ T6. Model Evaluation with Indicator and Monitoring

This tactic needs to be changed to Functional View, Information View, and Behavior View.



6.1.7. [Step 7] Verifying the Traceability

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



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

6.2. NFR-2. High Reliability of the Server

The server side of the system should be developed and operated with high reliability since all the mobile clients access the SThT functionality through the server. Therefore, the system should provide a high level of reliability in terms of its sub-quality attributes as defined.

- High Maturity of the Server
  The system should avoid the failure in case of fault occurrences.
- High Fault Tolerance of the Server
  The system should detect various types of faults and tolerate the faults to continue provide the system functionality with the specialized level of performance.
- High Recoverability of the Server
  The system should be designed to be recovered for its software and the transaction dataset in effective and efficient manner.

6.2.1. [Step 1] Underlying Facts and Policies

High reliability of the server should be considered equally for other parts of the server's operation. Therefore, the scope of this requirement is as follows.

- SThT Server
- ML Server
- Dispatcher

In addition, external services used should also be considered for reliability.

- STT Service
- Payment Service (Card Payment Authorization System)

- (F1) High Reliability on SThT Server

  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) Resource Management ML Server

  Machine learning training takes a lot of time and requires a lot of resources. If there is more work, problems may occur in the ML server. ML Server does not have much transaction, so if the system manages resources well, there will be no problem with the server.

- (F3) Problem occurs on Dispatcher

  If a problem occurs in the dispatcher, the client 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.

- (F4) QoS of STT Service

Problems arising from the STT Service cannot be handled by the system. The same is true for low QoS. However, STT Service is part of the main function in the system, so if a problem occurs, the service will be disrupted.

- (F5) Error occurs on Payment Service

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

6.2.2. [Step 2] Criteria for Satisfying NFR

- (C1) Fault Tolerance of SThT Server (Relevant to F1)

The SThT 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) Resource Management ML Server (Relevant to F2)

The resource management of the ML server should be thorough. If a lot of training is performed at the same time, the load on the server may increase and become a problem. If this situation is avoided, there will be no problem with server operation.

- (C3) Recoverable Dispatcher (Relevant to F3)

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

- (C4) Prepare STT Service Alternatives (Relevant to F4)

Alternatives should be prepared for the reliability of the STT service. Reliability of each service cannot be secured in the system. For this reason, alternatives should be prepared so that other services can be used in the event of problems in mainly used services.

- (C5) Minimize the impact of Fault on Payment Service

If a fault occurs in the Payment Service, recovery of the fault must be made. However, this service is not a service that we can restore like the STT service. In the case of payment service, it is a very important service to user because a lot of sensitive information is required, and financial damage can occur.

6.2.3. Step 3] Candidate Tactics for the Criteria

The following tactics are proposed for the identified criteria.

- (T1) State resynchronization without redundancy (Relevant to C1)

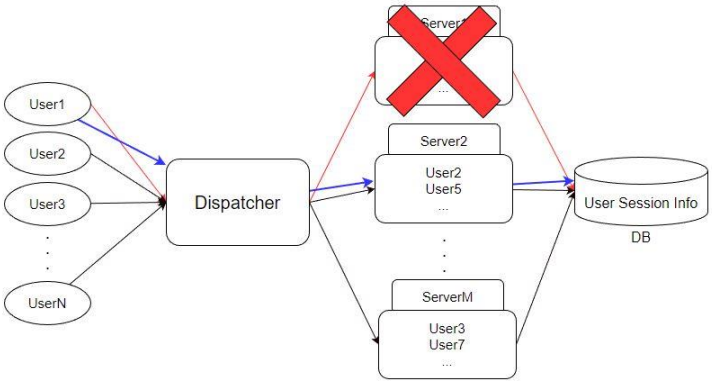
The most convenient way to ensure server reliability is to create redundancy. If there is redundancy, if there is a problem with the server, simply change it to a redundant server and operate it. This is a very simple but expensive and resourceful way. We suggest different way to combine several tactics.

- Retry

  First, when a failure occurs, a logic that checks whether there is an error in the server through the dispatcher, and if there is no abnormality, retry logic should precede it. In this case, the number of retries should be limited. If there is a problem with the server, or the retry limit is exceeded, proceed with the next tactic.

- Non-stop forwarding with Dispatcher

  Non-stop forwarding through shared session storage with dispatchers is implemented. First, a session storage for storing information on a session is configured so that information on the entire user session can be stored and accessed by servers. And, through dispatcher, users are relocated to healthy servers except for servers with abnormalities, and session information is read to continue using the service. Since the dispatcher periodically checks the health of the server, it can immediately check the abnormality of the server.



When Server1 fails, all User1's connections that were connected to Server1 disappear (red line) and connect to other servers that are available through Dispatch (blue line) All of the session's information is integrated and managed by DB, so it can be used immediately by the newly assigned server.

In the case of a process that should go through several steps, the state must be divided so that session information can be stored at each step for this configuration. These processes include Speech to Text and ML Model Training. Refer to the activity diagram specified in 5.3.2. and divide the corresponding steps and store session information from time to time.

- (T2) Using Docker, Kubernetes and Kubeflow (Relevant to C2)

The system must independently execute multiple ML training and pay attention to resource distribution and utilization. Such an environment is generally difficult to implement, but it is possible to build an environment through *Docker* with *Kubernetes* and create ML pipeline through *Kubeflow*.

Docker can simply be described as 'VM without OS'. Through container libraries, it is possible to create a highly distributed system by creating an independent virtual space in terms of resources and services.

Kubernetes can manage those *Docker* containers automatically. In addition, load

balancing is automatically performed through container orchestration. *Kubernetes* sets the scope of resources to be allocated through setting the Request and Limit values, which minimizes surplus resources and enables efficient resource management. Through these functions, resource management for ML training tasks can be performed simply through queuing.

It is recommended to use *Kubeflow* to conduct ML training in a *Kubernetes* environment. *Kubeflow* is an open-source platform that enables ML Workflow to be automated in a *Kubernetes* environment. Since Job definition is possible through Pytorch, there is no problem in applying it to this system. (The ML work of this system will be done with pytorch.) Among the main functions of *Kubeflow*, the functions available in the system are as follows.

- Pipeline

  Pipeline can make the process from Dataset preparation to parameter and default definition, training definition, model generation and deployment into a single pipeline, and provides monitoring capabilities via UI.

- Katib

  Katib consists of hyperparameter optimization and neural architecture navigation. Among them, Hyperparameter optimization will help data scientists optimize the SThT model.

Applying all three technologies listed above can be expensive to develop, but it will be of great help to smooth resource management and ML operation.

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

  Since dispatchers do 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 dispatchers, it is not necessary to maintain active redundancy because state resynchronization does not take long if only the function operates normally.

- (T4) STT Service health check (Relevant to C4)

  The STT Service to be used in the system is a service operated by three large IT companies: MS, Google, and AWS. That is why the availability of services is high enough. But we need to be prepared in case of trouble.

  Each STT Service includes a function to check Health in the API. Through this, the current QoS of each service may be checked. In the case of MS Azure STT, the status of the service can be checked through HealthStatus_Get, and the return value can be checked in three stages: Healthy, Degraded, and Unhealthy. If the service status is not Healthy, check the status of Google and AWS to implement other services.

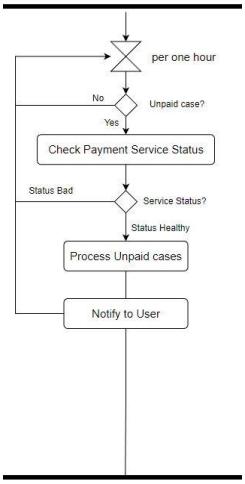
- (T5) Payment Service health check (Relevant to C5)

Like the STT service, the payment service also determines the service to proceed with payment through health check. There are many reliable services in the market. Since speed and accuracy are the most important things for payment, a system to replace in case of a problem should be secured.

- (T6) Monthly/Delayed payment (Relevant to C5)

  The system proceeds with payment for the entire use once a month. In this way, the dependency on the payment service decreases, so even if there is a problem with all the payment services in use, it is not fatal. And since the system has payment information, it is okay for the payment to be delayed if the user checks it in advance. Of course, this part should be notified to the user at the time of payment confirmation according to the status of the Payment service.

  If there is a problem with the payment service at the time of payment, the payment must be made after the service is restored. This part is managed through a separate thread on the server side, and if more than one unpaid case occurs, the status of the payment service is checked, and the payment completion notification is sent to the user after processing the unpaid cases.



6.2.4. [Step 4] Evaluation of the Candidate Tactics

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

ID	Tactics	Y/N	Justification
T1	State resynchronization without redundancy	Y	Benefit) Server reliability can be improved without redundancy and costs can be reduced. Cost) Overhead for Session information management occurs, and additional DB Storage is required. Decision) It is the most economical tactic to improve server reliability.
T2	Using Kubernetes and Kubeflow	Y	Benefit) Resources for ML can be effectively managed, and ML Workflow will be easier to manage in the future. Cost) It takes a lot of time and effort to understand and develop Kubernetes and Kubeflow. Decision) Although it is expensive to develop, it is the

			best way to stably implement ML servers.
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	STT Service health check	Y	Benefit) The reliability of the STT Service can be obtained through simple tactics. Cost) Only small development costs are needed. Decision) It is a tactic that can achieve great results at a low cost.
T5	Payment Service health check	Y	Benefit) Even if there is a problem with the payment service, the service can be provided without a big problem. Cost) Low development costs are required, and user consent is required. Decision) It is a tactic that provides convenience in system operation at a low cost.
T6	Monthly/Delayed payment	N	Benefit) Even if there is a problem with the payment service, the system can be operated without any major disruption. Cost) Low development costs are required, and user consent is required. Decision) It is not adopted because the probability of an error occurring in one or more services is very low.

6.2.5. [Step 5] Impact Analysis of Tactics on Views

We analyze the impacts of each selected tactic.

ID	Tactics	Functional View	Information View	Behavior View	Deployment View
T1	State resynchronization without redundancy	Define usecase on server ‘resynchronization user session’ and add ‘save session info’ to each process.		Design control flow of server about resynchronization user session.	
T2	Using Kubernetes and Kubeflow			Add control flow of configure Kubernetes environment.	Kubernetes and Kubeflow are reflected in the ML Server Tier.
T3	Prepare Passive Redundancy of Dispatcher	There is nothing to reflect in Views.			
T4	STT Service health check	Define usecase about STT health check and Change		Add an action ‘Verifying QoS’ and ‘Change STT	

		STT Service		Service' within SThT Process.	
T5	Payment Service health check	Define usecase about Payment service health check and Change Payment service		Add an action 'Health check of payment service' and 'Change payment Service' within payment Process.	

6.2.6. [Step 6] Architecture with Tactics Applied

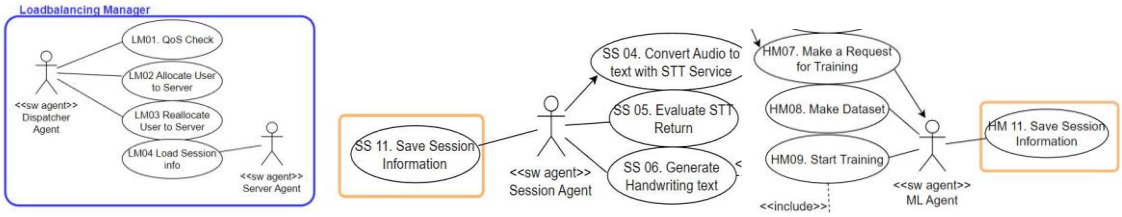
We design the detailed control flow for selected tactics. In CEP, we choose only one tactic.

❑ T1. State resynchronization without redundancy

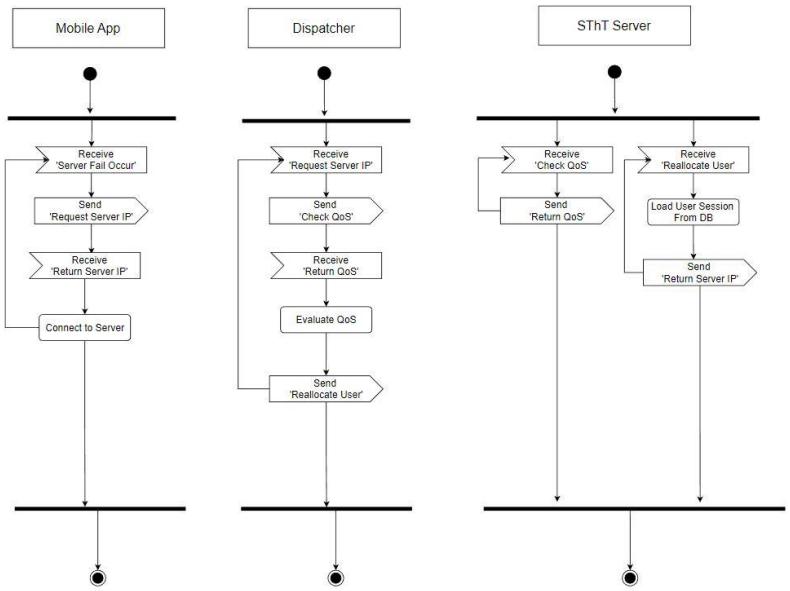
This tactic is about when a problem occurs in a server, and the definition of a use case and control flow is required for a situation in which a user assigned to the server is reallocated to another server.

First, Dispatch's reassignment logic should be the same as the allocation logic, but it should be stated that it is a reassignment user so that the server can load user sessions.

Thereafter, the server loads the reassigned user's session information from the session DB to provide a service. And Processes that are difficult to manage sessions allow the sw agent to store session information between tasks.

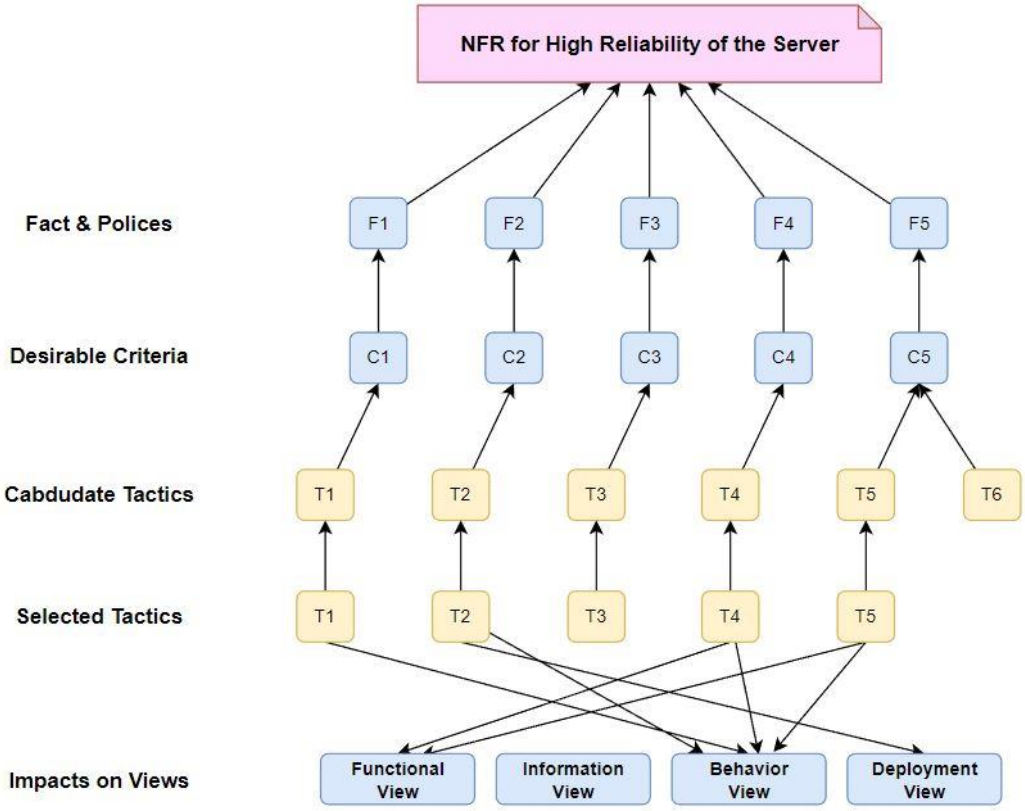


And then defines the control flow for the reassignment of Mobile App, Dispatcher and server.



6.2.7. [Step 7] Verifying the Traceability

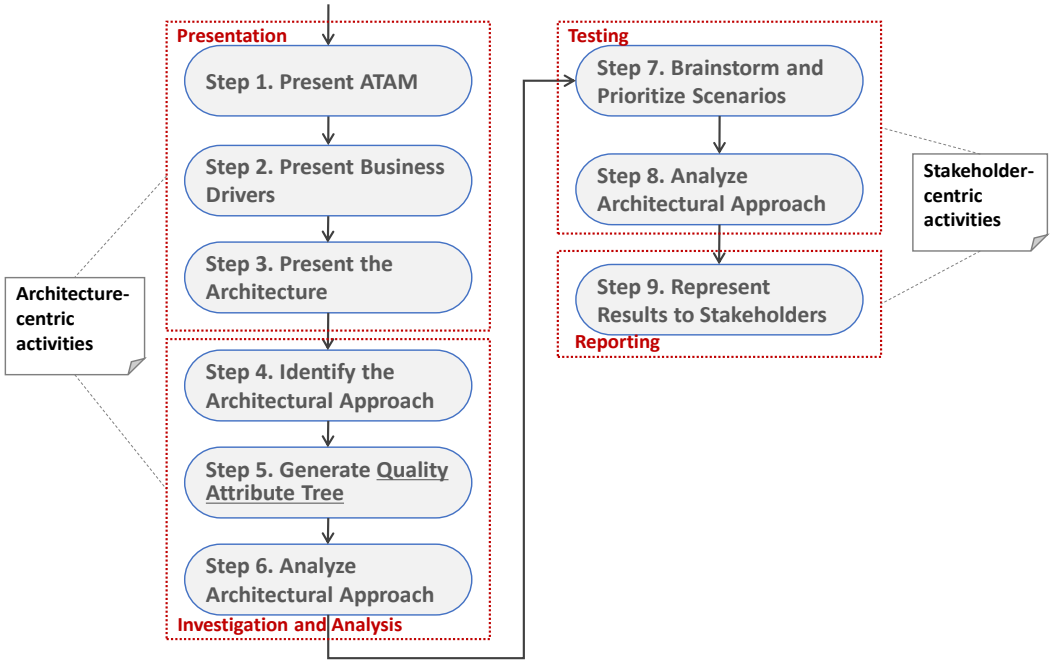
It is important to enforce the traceability among the facts/policies, criteria, tactics and impacts on views. The following figure shows the trace links among facts, criteria, tactics and their impacts on views.



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

7. Activity 6. Architecture Validation

❑ Steps of ATAM



7.1. [Step 1] Presenting ATAM

❑ Omitted

7.2. [Step 2] Presenting Business Drivers

❑ Omitted

7.3. [Step 3] Presenting the Architecture

❑ Omitted

7.4. [Step 4] Identifying the Architectural Approaches

The architectural approaches applied to *SThT System* are described in chapters 5 and 6. The architectural approaches are reflected in the following view;

❑ Functional View

❑ Information View

❑ Behavior View

❑ Deployment View

7.5. [Step 5] Generating Quality Attribute Tree

The following table describes the quality attribute utility tree used for evaluating the architecture of *SThT System*. We define 4 scenarios for evaluating functionality (i.e. Accuracy of Personalized Handwritten Texts) and for 4 scenarios for evaluating Reliability (i.e. High

Reliability of the Server) of the *SThT System*. Each scenario is described with a different degree of importance and difficulty.

Quality	Refinements	Scenarios	Importance	Difficulty
Functionality	NFR-1. Accuracy of Personalized Handwritten Texts	NFR-1-1. Evaluate CER between the output text and input text is close to zero.	High	High
		NFR-1-2. Evaluate Losses of Discriminator and Recognizer are close to 1 (at least more than 0.8)	High	High
		NFR-1-3. Evaluate IS and FID are close to target score.	High	High
		NFR-1-4. Evaluate how much portion of user satisfied with this system.	High	Low
Reliability	NFR-2. High Reliability of the Server	NFR 2-1. Evaluate server reliability with down time.(target is 4 nines)	High	High
		NFR 2-2. Evaluate ML server reliability with average training time.	High	High
		NFR 2-3. Evaluate STT Services reliability with average return time of text	High	High
		NFR 2-4. Evaluate Payment Services reliability with average return time payment result.	High	High

7.6. [Step 6] Analyzing Architectural Approaches

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

Analysis of Architectural Approach				
Scenario #	NFR-1-1. Evaluate CER between the output text and input text is close to zero.			
Attribute	Functionality			
Environment	Normal Operation.			
Stimulus	App User requests SThT Service and provides a speech or file. The converted text was returned through the STT Service.			
Architectural Decision	**Sensitivity**	**Trade-Off**	**Risk**	**Nonrisk**
D1-1. Applying ScrabbleGAN	S1-1-1	T1-1-1		
D1-2. Colleting various type of pangram	S1-1-2	T1-1-2		
D1-3. Preprocessing Dataset				N1-1-1
D1-4. Postprocessing Dataset				N1-1-2

Reasoning	The decisions are made for meeting this scenario since the chosen decisions are commonly used for improving accuracy of the results of running machine learning algorithms.
Architectural Diagram	Refer to refined component diagram in 6.1.

- Sensitivity Points
  - S1-1-1. It takes a lot of time to optimize the model.
  - S1-1-2. There is a lot of pressure on the user, and the increasing number of pangrams should be managed.
- Trade-off
  - T1-1-1. Accuracy (+) vs. Resource (-): ML training requires a lot of time and resources, so increasing accuracy requires a lot of resource consumption.
  - T1-1-2. Accuracy (+) vs. Useability (-): Obtaining a large variety of datasets increases the accuracy of the model, but there is a burden on the user to provide more parameters.
- Risk
  - R1-1-1. The risk is caused since detecting regular activities for multiple number of children is way more difficult than detecting ones for just one child since those children do different activities at the same time.
  - R1-1-2. This risk is caused since typically customers are not willing to give feedbacks.
- Nonrisk
  - S1-1-1 Dataset preprocessing is a very important factor in ML training.
  - S1-1-2 Through post processing, it is possible to increase the quality of output image and increase the user's satisfaction and image evaluation score.

7.7. [Step 7] Brainstorming and Prioritizing Scenarios

- List of scenarios collected by all stakeholders (i.e. clients, managers, users, data scientists)
  - About 3 scenarios are additionally acquired for evaluating functionality of SThT system. Most scenarios are gathered from users, managers, and data scientists.
    - Need a method that can effectively secure a pangram dataset.
    - Additional resources that can be used for ML training as much as possible.
    - Find a way to reduce the resources required for ML training.
  - About 3 scenarios are additionally acquired for evaluating reliability for SThT System. Most scenarios are gathered from clients and managers.
    - Review to use cloud service for server, ML server.
    - Find ways to make it easy to set up Kubernetes environment and create ML workflows.

7.8. [Step 8] Analyzing Architectural Approaches

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

7.9. [Step 9] Presenting the Results

- ❑ All evaluation team concludes the following results;
  - ❍ Concerns on the functionality, specifically on the generated handwriting text, need to be re-considered.
    - ➢ For ML training of SThT System, a lot of pangrams must be provided. The system cannot request hundreds of pangrams from all users.
  - ❍ Concerns on the Reliability, specifically on the server features, need to be re-considered.
    - ➢ In terms of industry trends, servers and ML servers in the on-premise environment have limitations in scalability and management. Cloud server environments should be considered.

8. 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 ≺