Voice Campus

Software Design Specification Document

Team 6

Kang Hyunmuk

Kim Jihye

Park Jiye

Shin Wonchul

Oh Seungjun

Date : 2020.05.16.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Preface

This chapter contains the readership information, readership, scope, objective of this document and the document structure of this Software Design Document for Voice Campus of Team6.

1.1. Readership

This Software Design Document is divided into 10 sections with various subsections. The structure of the Software Design Document can be found as listed below, in the Document Structure subsection of this SDD. In this document, Team 6 is the main reader. Additionally, professors, TAs, and team members in the Introduction to Software Engineering class can be the main readers.

1.2. Scope

This document gives a detailed description of the software architecture of Voice Campus. It specifies the structure and design of some of the modules discussed in the SRS. It also displays some of the use cases that had transformed into sequential and activity diagrams. The class diagrams show how the programming team would implement the specific modules.

1.3. Objective

The purpose of this document is to present a detailed description of the design of the Voice Campus, created for the students with visual impairments. Firstly, this document is intended for Team6, to use the designs as guidelines to implement the project. Equally, this document is also for the team's professor, Mr. Lee, as it fulfills one of the requirements of the project. Lastly, this document could be used for designers who try to upgrade or modify the present design of the LearningX application.

1.4. Document Structure

- 1. Preface: this chapter describes readership, scope of this document, object of this system, and structure of this document.
- 2. Introduction: this chapter describes several tools used for this document, several diagrams used in this document and the references, and object of this project.

- 3. Overall System Architecture: this chapter describes overall architecture of the system using context diagram, sequence diagram, and use case diagram.
- 4. System Architecture(frontend): this chapter describes architecture of the frontend system using class diagram and sequence diagram.
- 5. System Architecture(backend): this chapter describes architecture of the backend system using class diagram and sequence diagram.
- 6. Protocol Design: this chapter describes design of several protocols which used for communication of client and server.
- 7. Database Design: this chapter describes database design
- 8. Testing Plan: this chapter describes testing plan for our system.
- 9. Development Plan: this chapter describes which tools to use to develop the system, constraints, assumption, and dependencies for developing this system.
- 10. Supporting Information: this chapter describes baseline of this document and history of this document.

2. Introduction

2.1. Objectives

In this chapter, we describe the various tools and diagrams which we have applied to this project in the design phase.

2.2. Applied Diagrams

2.2.1. UML

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps our project team communicate, explore potential designs, and validate the architectural design of the software.

2.2.2. Use case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

2.2.3. Sequence Diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

2.2.4. Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

2.2.5. Context Diagram

The Content diagram is an extension of UML notation. The purpose of the Content diagram is to generate or represent a project structure (diagrams) and relations between them. The Content table works as a table of contents for a project. The Content Shape creates a table of contents of all diagrams of the project. All Content diagrams have their own specifications. We can specify their name, documentation, and view the relationships in which they participate. We can also add stereotypes, tagged values, and constraints.

2.2.6. Entity Relationship Diagram

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities.

2.3. Applied Tools

2.3.1. Microsoft PowerPoint

It is an easy program to use and a powerful tool for drawing text and figures. It is convenient to draw diagrams using various shapes.

2.4. Project Scope

The Voice Campus system is a learning aid application for students with visual impairments. One of the biggest inconveniences visually impaired students may experience when studying is that they cannot read a textbook consisting of type. To help alleviate this inconvenience, we provide visually impaired students with text-to-speech(TTS) and test assistance. This system is based on the UI familiar to visually impaired people and TTS and STT functions using open source to make VA functions easier to use. Above all, we hope that this application will help students with visual impairments to solve a little bit of the difficulties they face while studying in college.

2.5. References

The user of this SDD may need the following documents for reference:

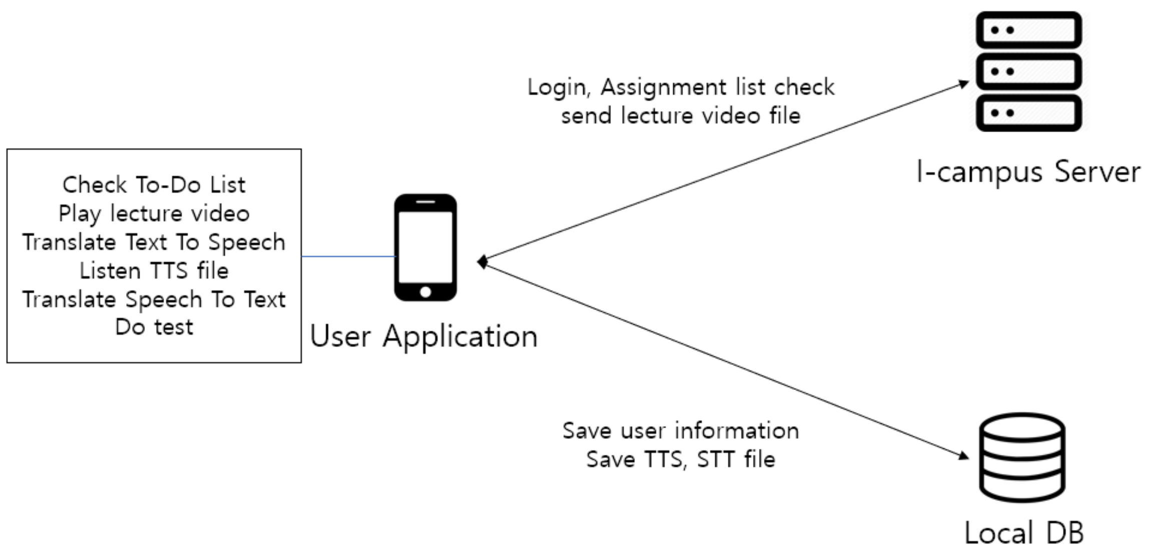
(1) Canvas LMS - REST API and Extensions Documentation. (n.d.). Retrieved April 23, 2021, https://canvas.instructure.com/doc/api

(2) React-Native-Voice. (n.d.). React-native-voice/voice. Retrieved May 08, 2021, from https://github.com/react-native-voice/voice

3. System Architecture – Overall

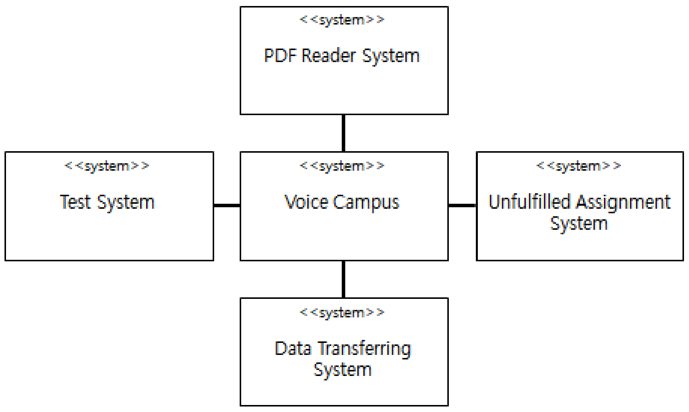
3.1. System Organization

The diagram below illustrates the brief configuration of this system. User application performs the function of logging in, checking to-do list, and receiving lecture video files in conjunction with I-campus server. For course material file reading function, TTS system uses ‘react-native-doc-viewer’ text extraction library and ‘react-native-tts’ library for translating extracted text from PDF, WORD file and save to database. For playing voice translated course material files, voice play class requests sound file to DB handler and get sound file from database. For test taking assistance, STT system uses ‘react-native-stt’ library to translate user’s voice into text file, save it to DB.



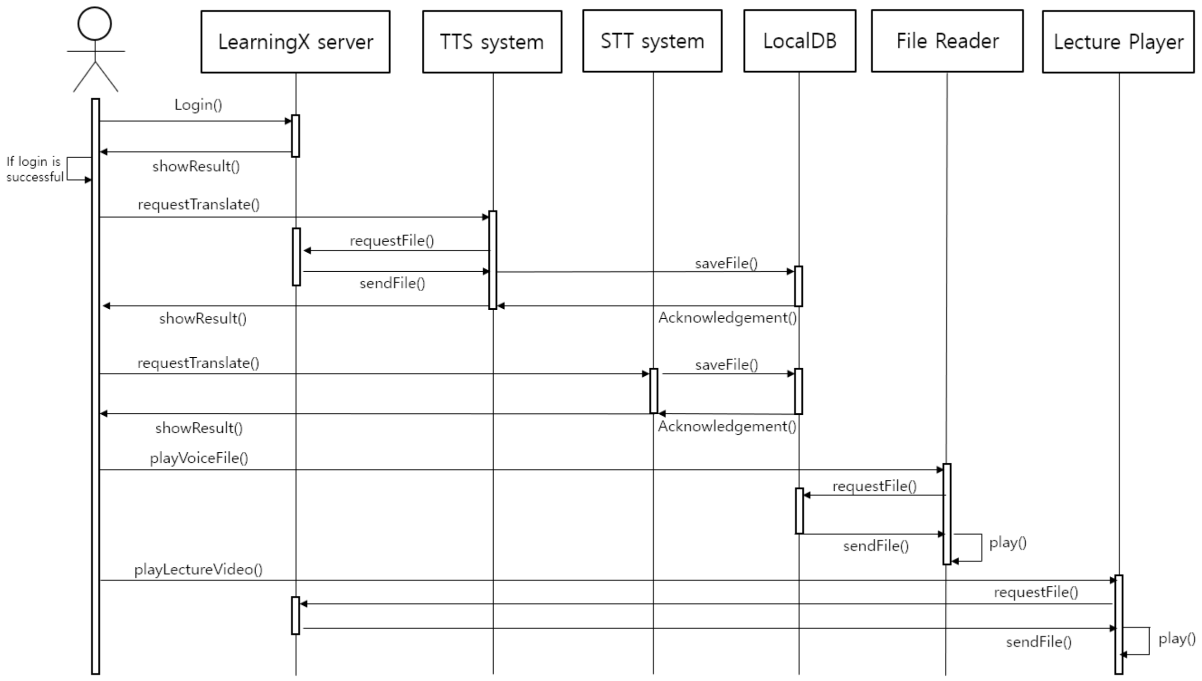
[Figure 1] Overall System Architecture

3.1.1. Context Diagram



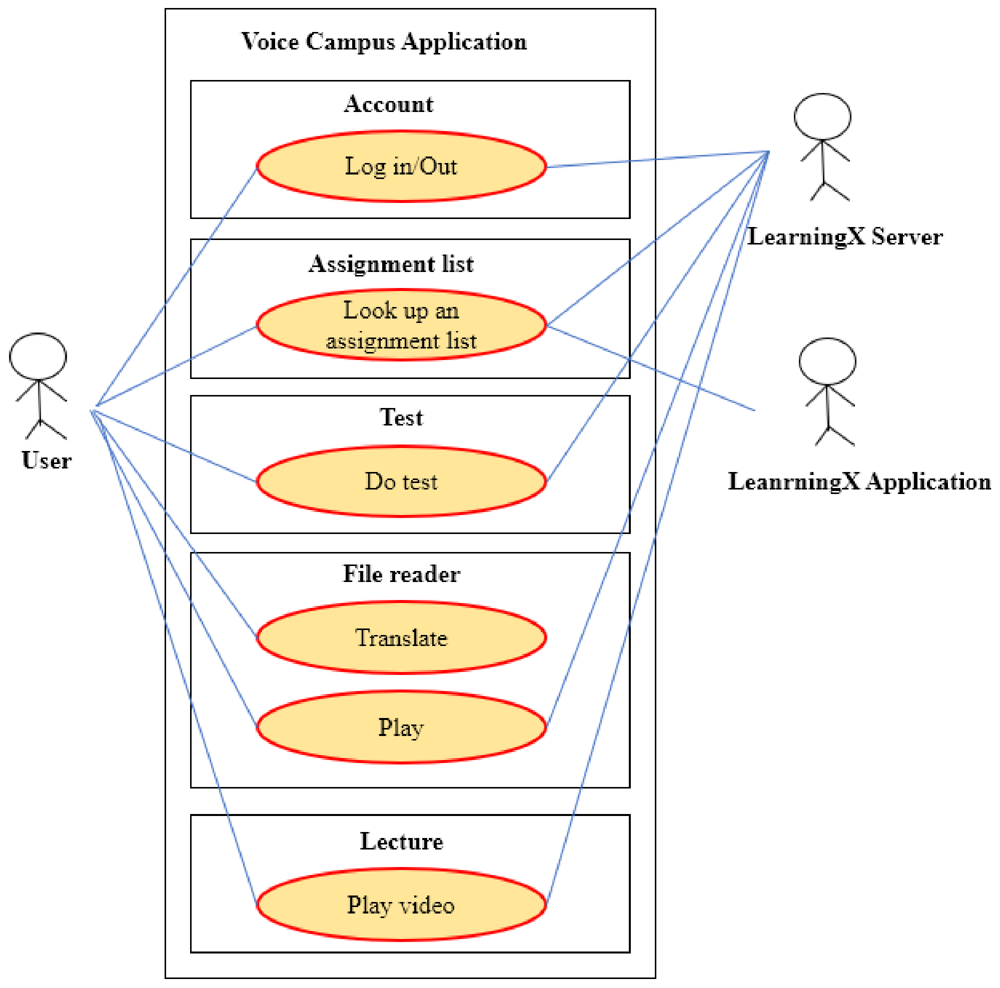
[Figure 2] Context diagram

3.1.2. Sequence Diagram



[Figure 3] Sequence diagram

3.1.3. Use Case Diagram



[Figure 4] Use Case diagram

4. System Architecture – Frontend

4.1. Objectives

4.2. Subcomponents

4.2.1. Voice Play

The voice play class plays voice files according to the user setting.

4.2.1.1. **Attribute**

These are the attributes that voice play object has.

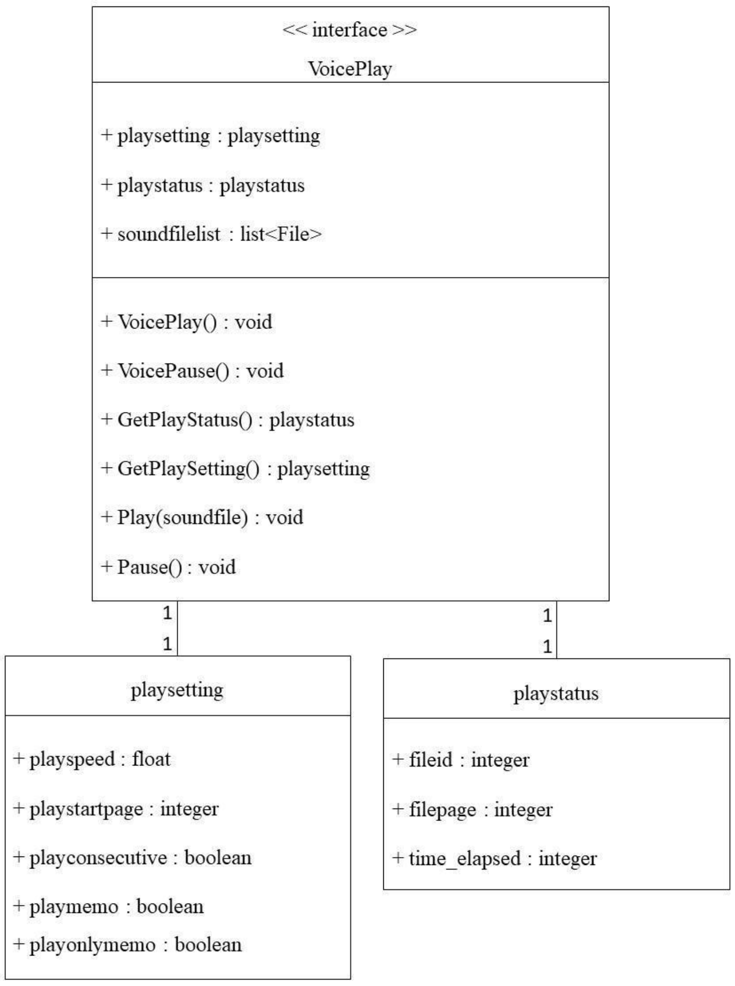
- playsetting : play setting of current voice play
- playstatus : play status of current voice play (fileid, filepage, time_elapsed)
- soundfilelist : list of voice file to play

4.2.1.2. **Methods**

These are the methods that play voice class has.

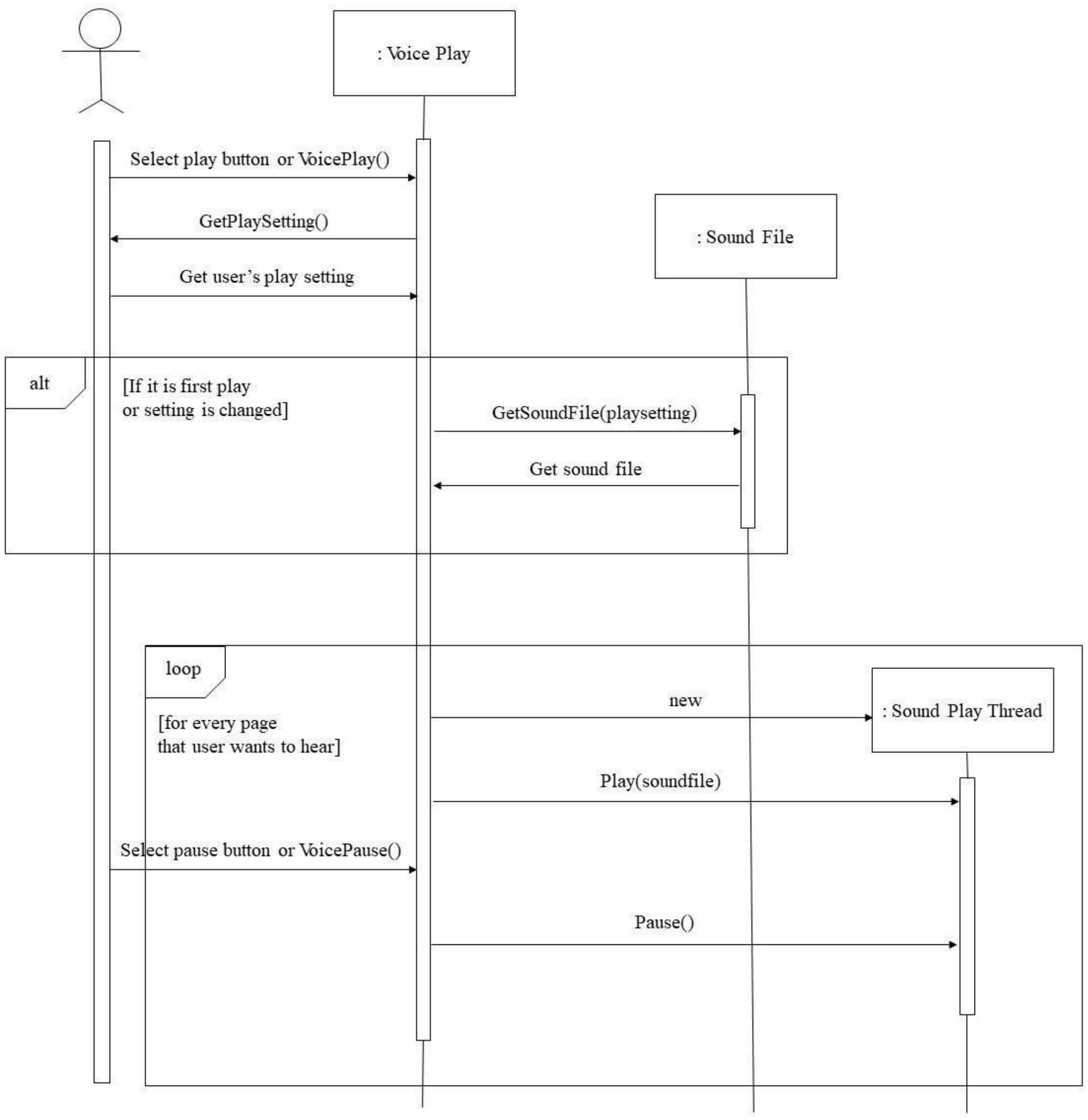
- VoicePlay()
- VoicePause()
- GetPlayStatus()
- GetPlaySetting()
- Play(soundfile)
- Pause()

4.2.1.3. **Class Diagram**



[Figure 5] Class diagram – Voice Play

4.2.1.4. **Sequence Diagram**



[Figure 6] Sequence diagram – Voice Play

4.2.2. Voice Memo

The voice memo class lets user record user's own voice memo.

4.2.2.1. **Attributes**

These are the attributes that voice memo object has.

- **memofile** : voice memo file

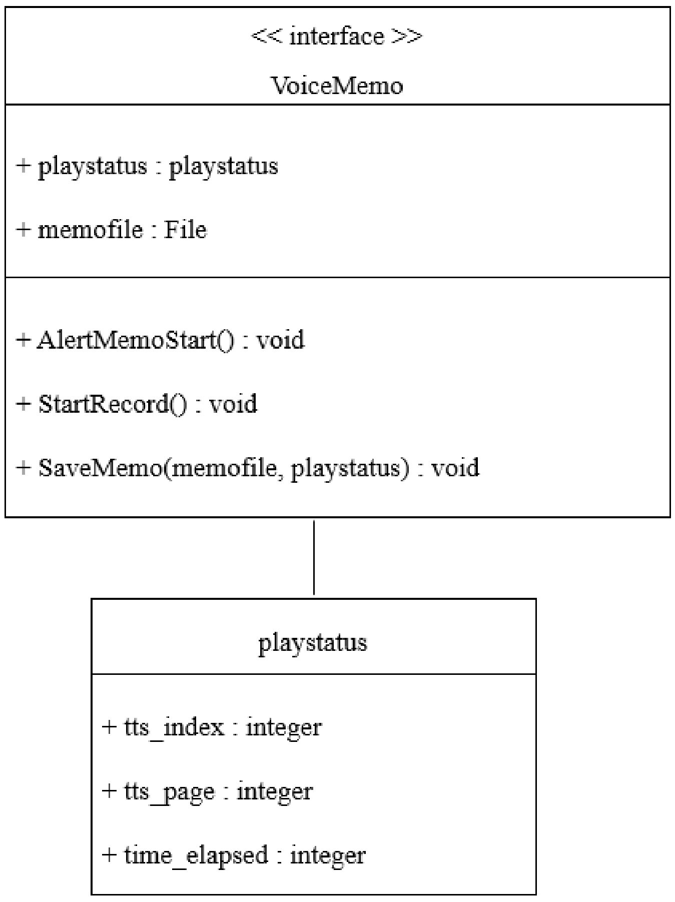
- **playstatus** : play status of current voice play (fileid, filepage, time_elapsed)

4.2.2.2. **Methods**

These are the methods that voice memo class has.

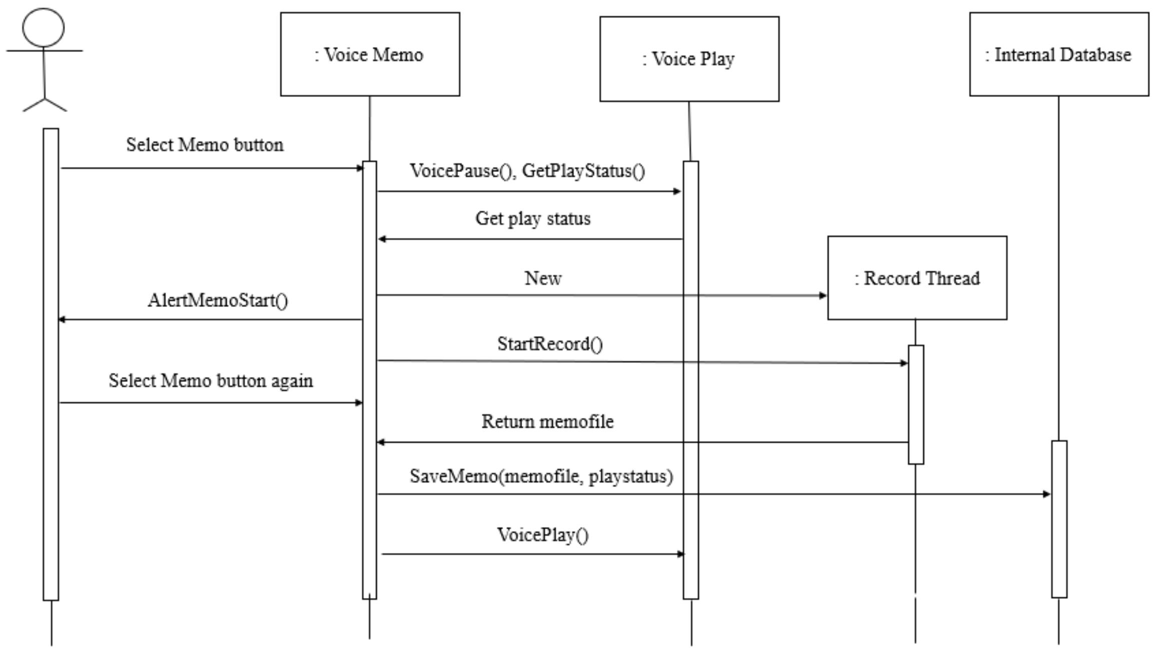
- AlertMemoStart()
- StartRecord()
- SaveMemo(memofile, playstatus)

4.2.2.3. **Class Diagram**



[Figure 7] Class diagram – Voice Memo

4.2.2.4. **Sequence Diagram**



[Figure 8] Sequence diagram – Voice Memo

4.2.3. Test

The test class gets the test problems from LearningX server and let the user solve the problems. The answers of the user should be send to the server.

4.2.3.1. **Attributes**

These are the attributes that test object has.

- **test id**: ID of the test
- **test title**: title of the test
- **test description**: description of the test
- **test type**: title of the test
- **start at:** title of the test
- **end at**: title of the test

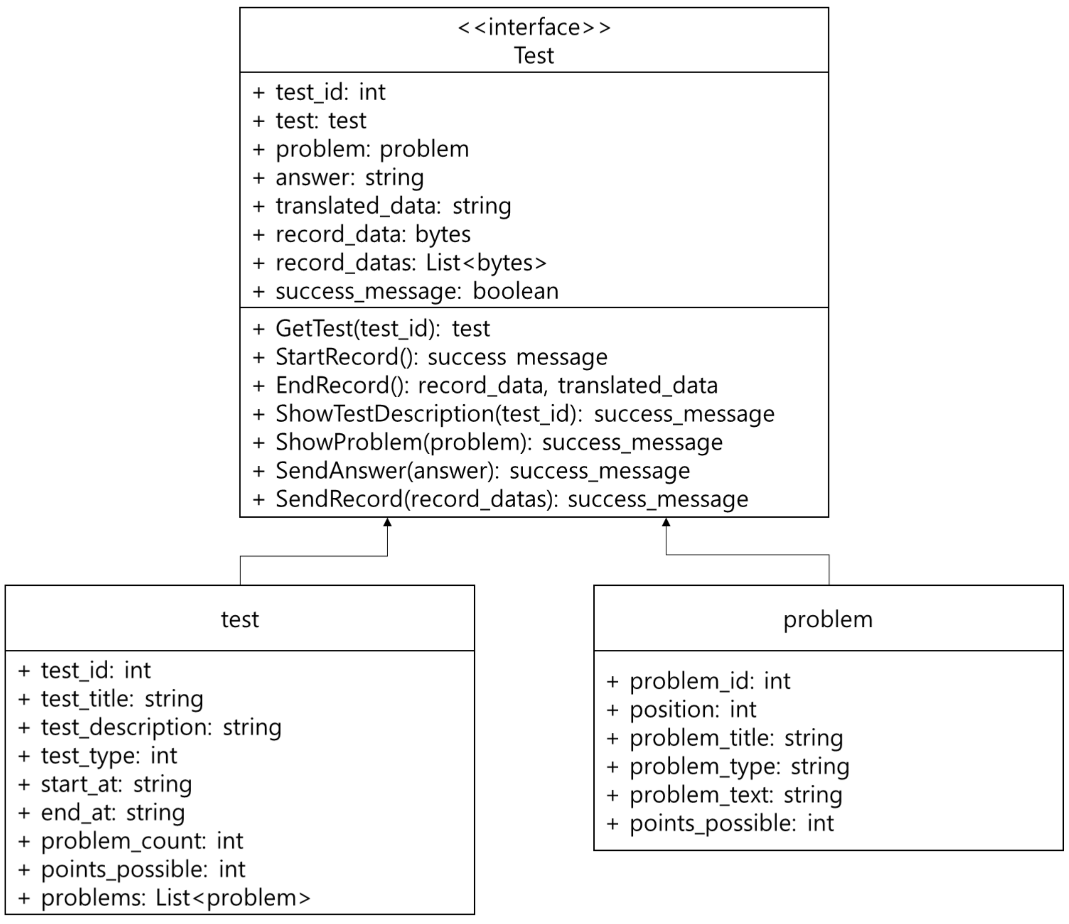
- **problem count**: title of the test
- **points possible**: title of the test
- **problems**: problems of the test

4.2.3.2. **Methods**

These are the methods that test class has.

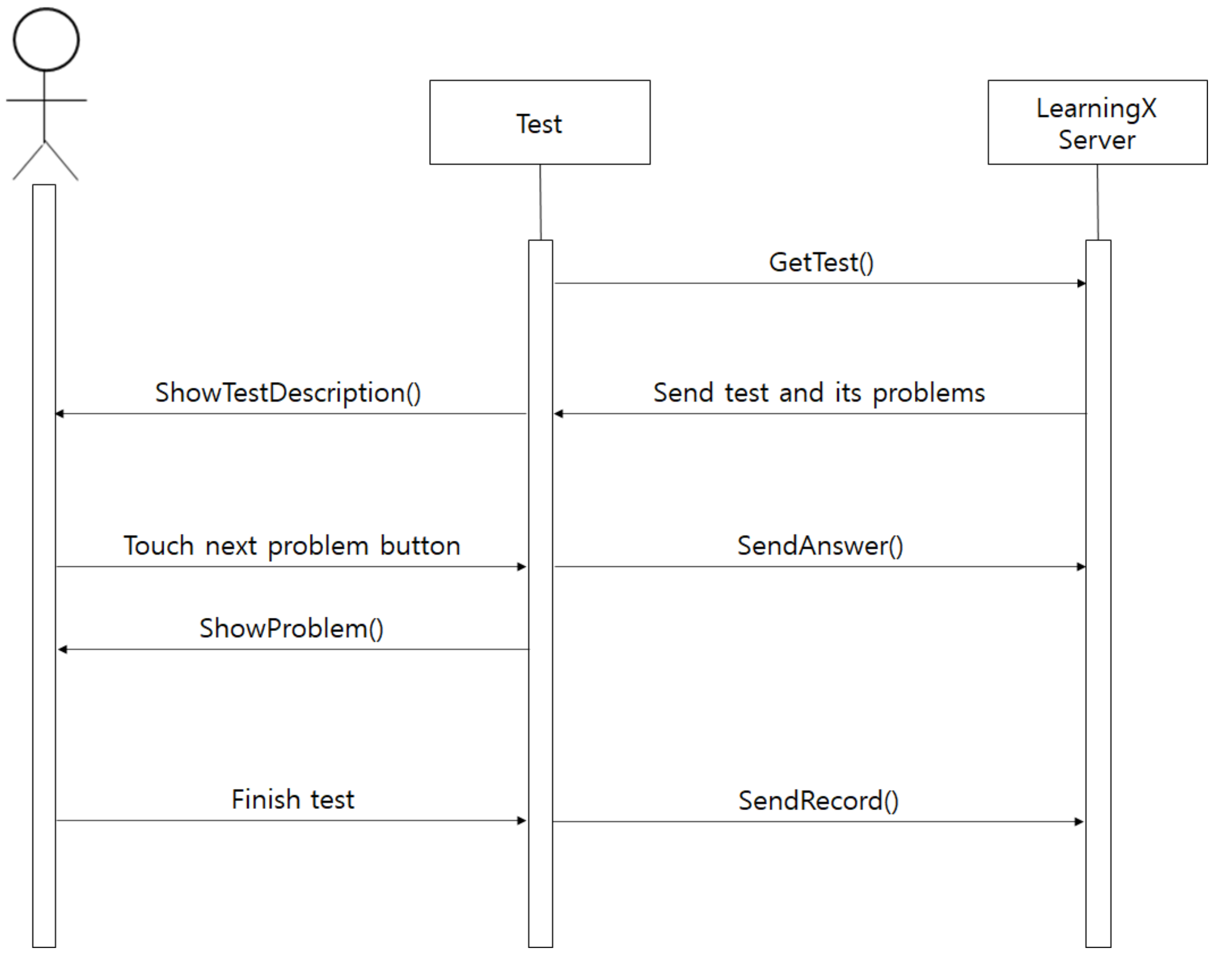
- GetTest()
- ShowTestDescription()
- ShowProblem()
- SendAnswer()
- SendRecord()

4.2.3.3. **Class Diagram**



[Figure 9] Class diagram – Test

4.2.3.4. **Sequence Diagram**



[Figure 10] Sequence diagram – Test

4.2.4. To-do List

To-do List class lets user to access class lecture, material, and test.

4.2.4.1. **Attributes**

These are the attributes that voice memo object has.

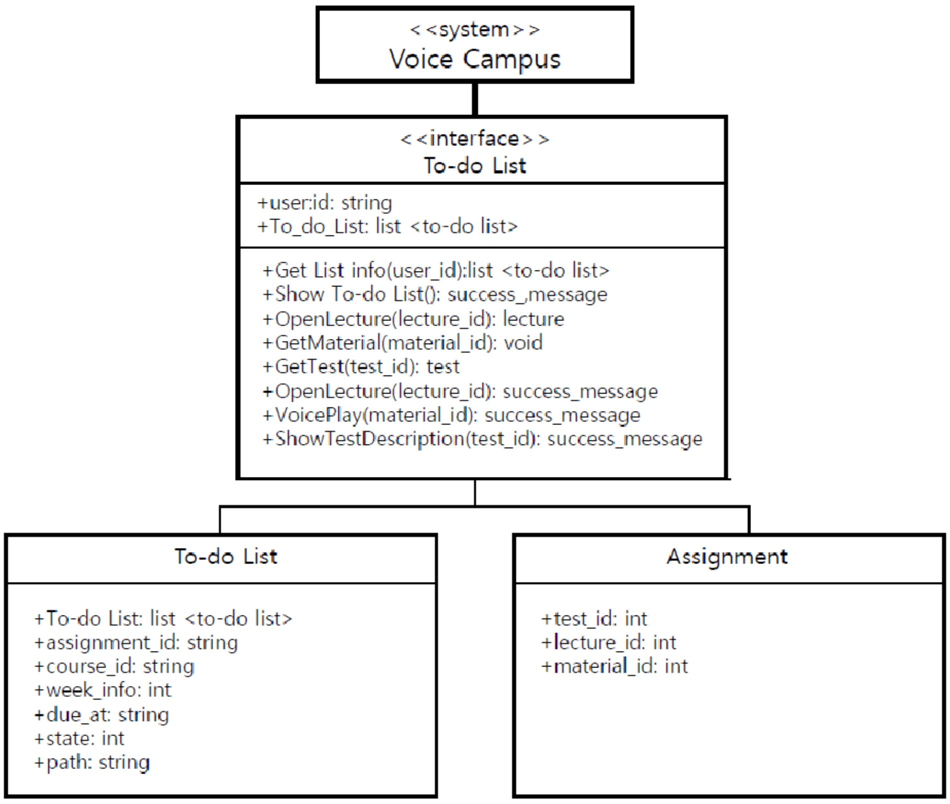
- **Assignment id** : id of the assignment
- **To-do List** : List of assignments and their information

4.2.4.2. **Methods**

These are the methods that voice memo class has.

- Get List info(user_id):list <to-do list>
- Show To-do List(): success_,message
- OpenLecture(lecture_id): lecture
- GetMaterial(material_id): void
- GetTest(test_id): test
- OpenLecture(lecture_id): success_message
- VoicePlay(material_id): success_message
- ShowTestDescription(test_id): success_message

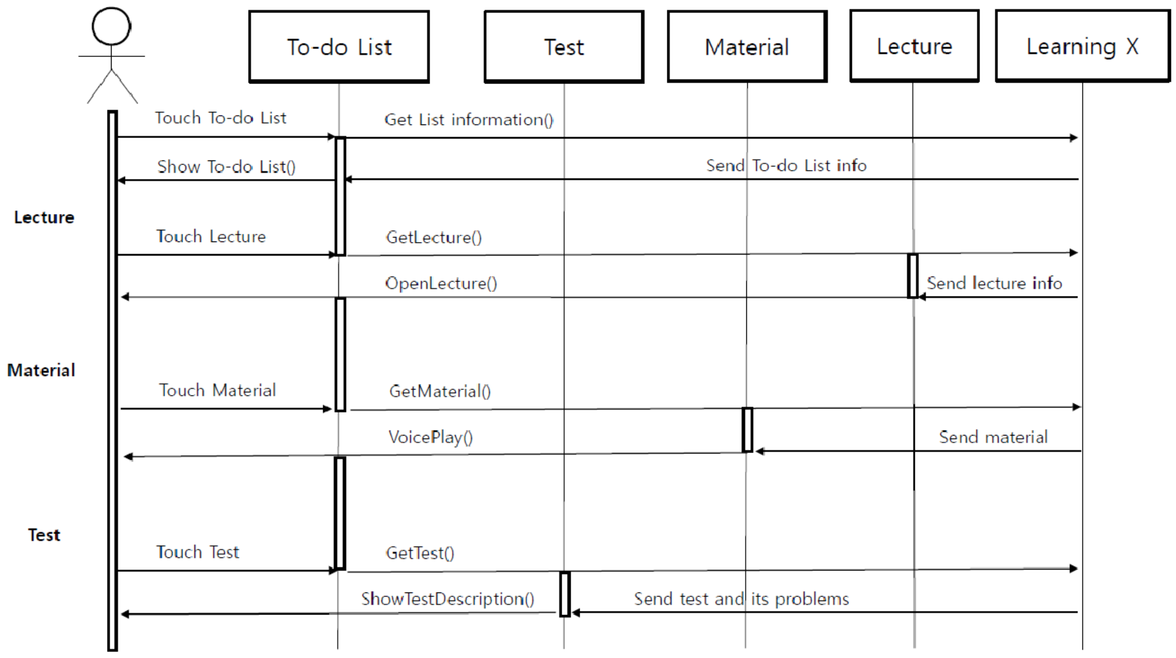
4.2.4.3. **Class Diagram**



<<system>>
Voice Campus
<<interface>>
To-do List
+user:id: string
+To_do_List: list <to-do list>
+Get List info(user_id):list <to-do list>
+Show To-do List(): success_,message
+OpenLecture(lecture_id): lecture
+GetMaterial(material_id): void
+GetTest(test_id): test
+OpenLecture(lecture_id): success_message
+VoicePlay(material_id): success_message
+ShowTestDescription(test_id): success_message
To-do List
+To-do List: list <to-do list>
+assignment_id: string
+course_id: string
+week_info: int
+due_at: string
+state: int
+path: string
Assignment
+test_id: int
+lecture_id: int
+material_id: int

[Figure 11] Class diagram – To-do List

4.2.4.4. **Sequence Diagram**



[Figure 12] Sequence diagram – To-do List

4.2.5. Login

The login class deals with admin user's account. After filling required fields with ID and password, users can login to the system using the login button. After login to the system, the user can use an approach to the functions of VOICE CAMPUS.

There is no explicit logout function, and it is automatically logged out when the application session expires.

4.2.5.1. **Attributes**

These are the attributes that login object has.

- **User id**: id of the user (Learning X id)
- **User password**: password of the user (Learning X password)

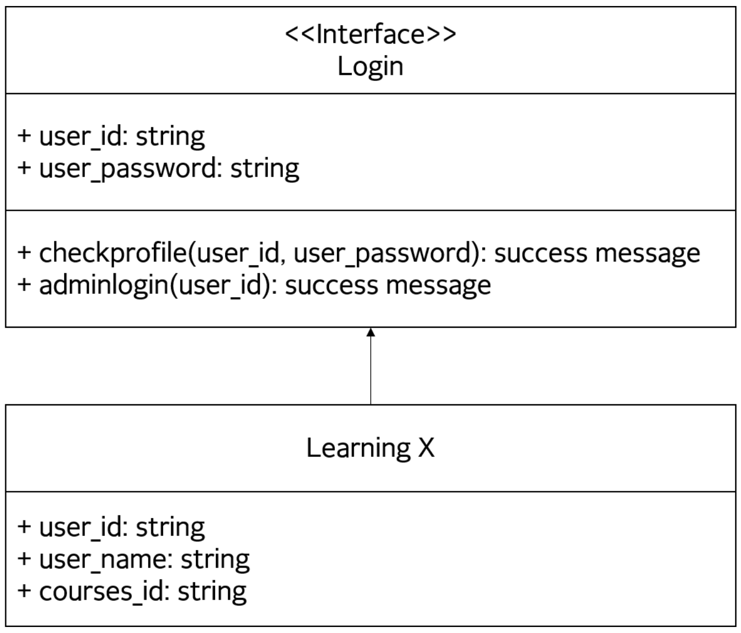
4.2.5.2. **Methods**

These are the methods that login class has.

- checkProfile()

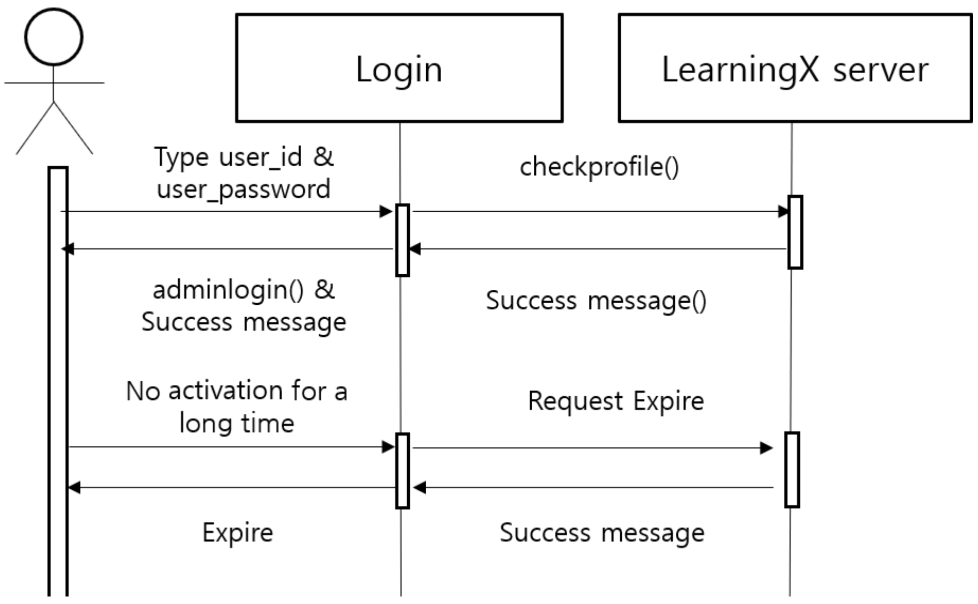
- adminlogin()

4.2.5.3. **Class Diagram**


<<Interface>>
Login
+ user_id: string
+ user_password: string
+ checkprofile(user_id, user_password): success message
+ adminlogin(user_id): success message
Learning X
+ user_id: string
+ user_name: string
+ courses_id: string

[Figure 13] Class diagram – Login

4.2.5.4. **Sequence Diagram**


Login
LearningX server
Type user_id & user_password
checkprofile()
adminlogin() & Success message
Success message()
No activation for a long time
Request Expire
Expire
Success message

[Figure 14] Sequence diagram – Login

4.2.6. Video Play

The video player class deals with playing a Learning X lecture video, this class get the lecture video from the Learning X server and shows the video to the user's screen.

4.2.6.1. **Attributes**

These are the attributes that video play object has.

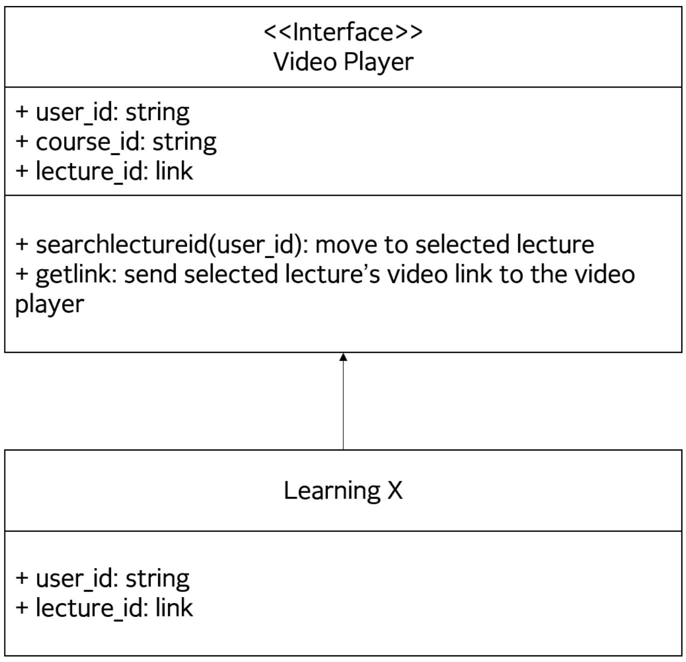
- **Lecture id**: id of the lecture in the course that the user is taking
- **link**: link which can move to the lecture that user choose

4.2.6.2. **Methods**

These are the methods that video play class has.

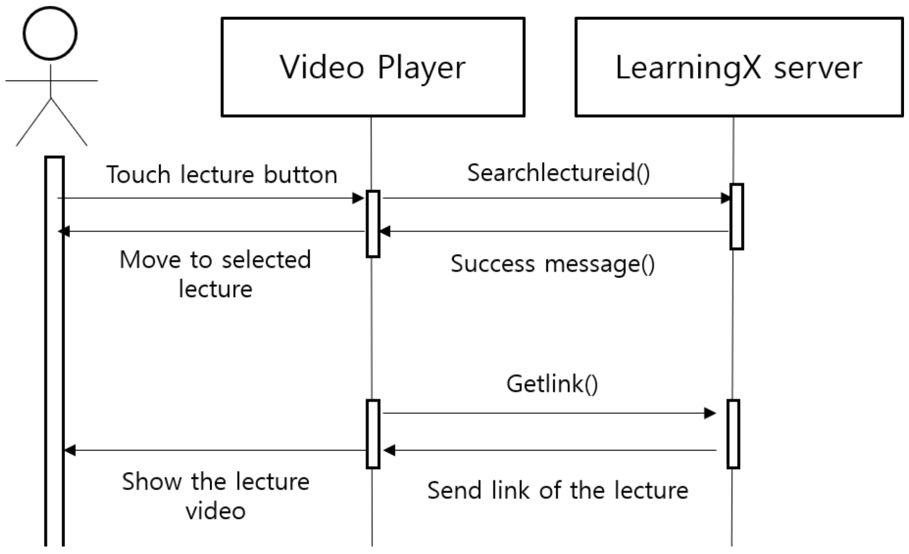
- Searchlectureid()
- Getlink()

4.2.6.3. **Class Diagram**



[Figure 15] Class diagram – Video Player

4.2.6.4. **Sequence Diagram**



[Figure 16] Sequence diagram – Video Player

5. System Architecture – Backend

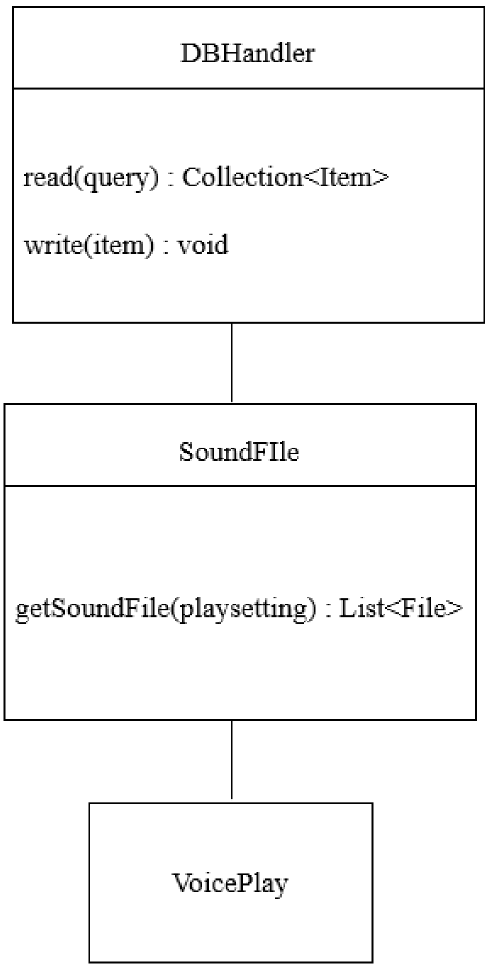
5.1. Objectives

5.2. Overall Architecture

5.3. Subcomponents

5.3.1. Sound File Generating System

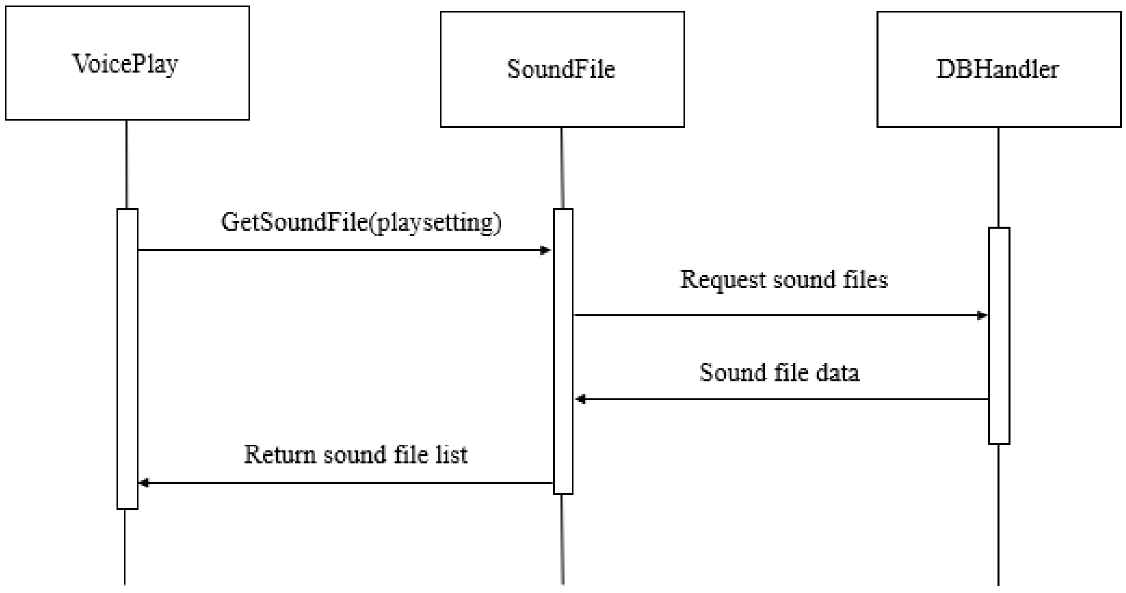
5.3.1.1. **Class Diagram**


DBHandler
read(query) : Collection<Item>
write(item) : void
SoundFIle
getSoundFile(playsetting) : List<File>
VoicePlay

[Figure 17] Class diagram - SoundFile

This class creates a combination of files that must be listened to according to the settings and hand over the data to VoicePlay class.

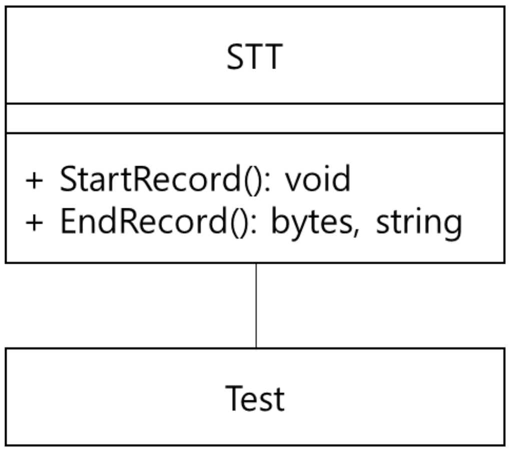
5.3.1.2. **Sequence Diagram**



[Figure 18] Sequence diagram - sound file generating system

5.3.2. STT System

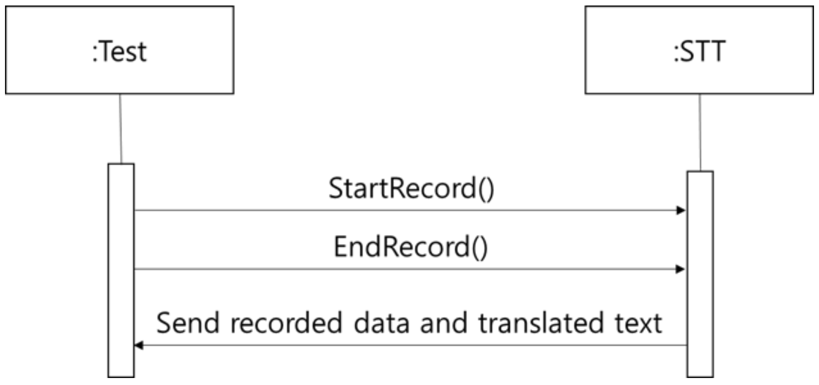
5.3.2.1. **Class Diagram**



[Figure 19] Class diagram - STT

STT is a class for recording and translating STT for Test. It uses the open source library 'react-native-voice' for Speech To Text translation.

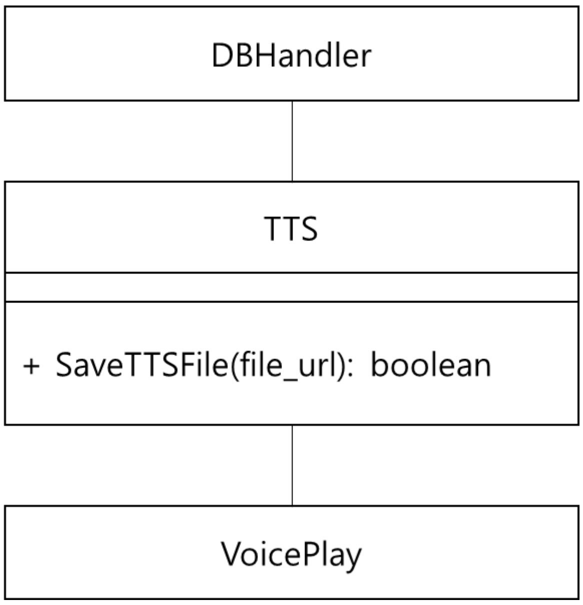
5.3.2.2. **Sequence Diagram**



[Figure 20] Sequence diagram - STT System

5.3.3. TTS System

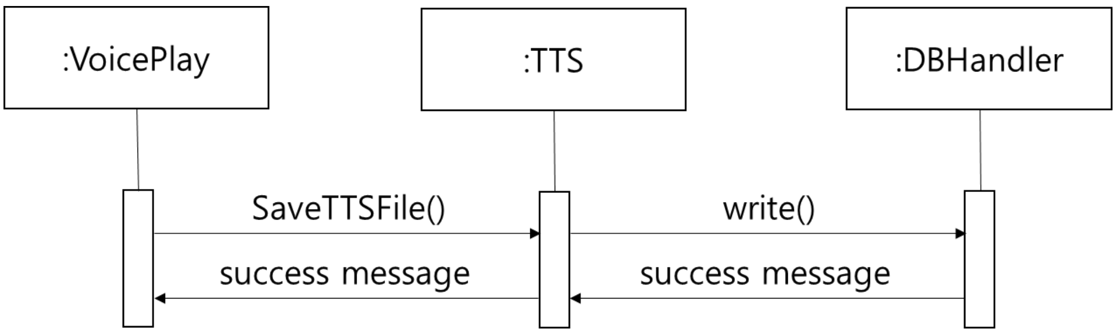
5.3.3.1. **Class Diagram**



[Figure 21] Class diagram - TTS

TTS is a class for extracting text from document files and synthesising speech audio file from text for VoicePlay. It uses the open source library 'react-native-doc-viewer' for text extraction from document and 'react-native-tts' for TTS.

5.3.3.2. **Sequence Diagram**



[Figure 22] Sequence diagram - TTS System

6. Protocol Design

6.1. Objectives

This chapter describes the protocol between Voice Campus application and other systems. We used existing systems, so documentation for each systems are presented.

6.2. LearningX Server

LearningX is a platform developed by Xinics, which extends the Canvas LMS. The protocol of interaction between Voice Campus and LearningX server follows the following document of Canvas LMS. [1]

6.3. STT Service

STT Service is provided by Google, but interaction between Voice Campus and Google is handled by react-native-voice library, which is MIT licensed open source library. You can check the documentation of library in the following document. [2]

7. Database Design

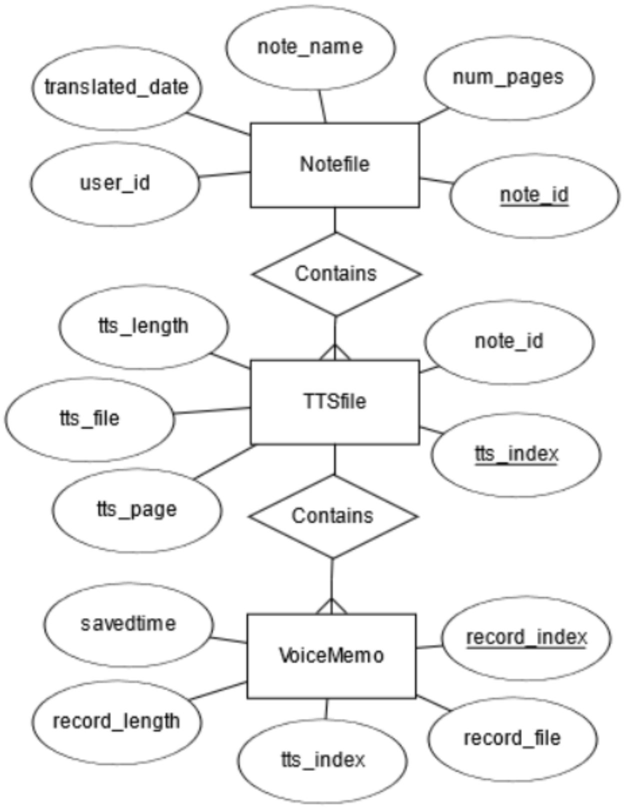
7.1. Objectives

This section is about database design. In this section, document describes entity relationship diagram of total database. And it gives relational schema and SQL data description language for the ER diagram of the project.

7.2. ER Diagram

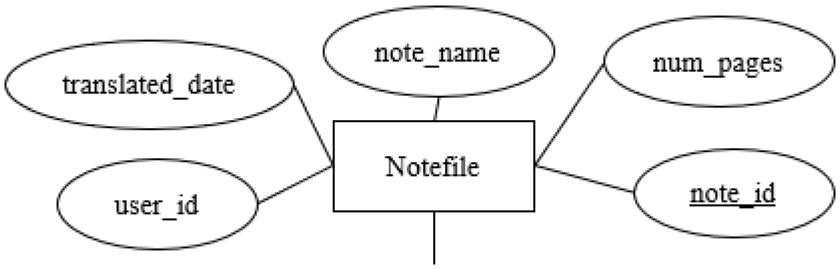
The system has 3 entities, Notefile, TTSfile, and VoiceMemo entities. Each entity is drawn by rectangular. And their attributes are drawn by circle. Especially, primary key is underlined. Each diamond shapes are representing relationship between entities. For 1:N relation, triangular shape is used for the end of the line.

7.2.1. Entities



[Figure 23] Total ER diagram

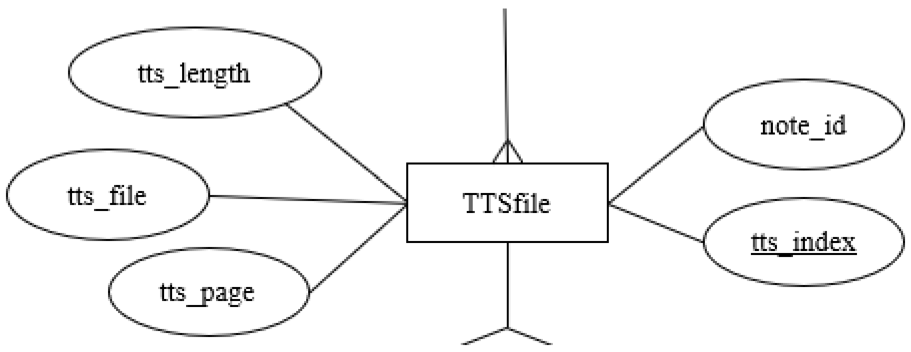
7.2.1.1. **Notefile**



[Figure 24] Notefile ER diagram

Notefile saves each file's information. It consists of note_id(PK, unique index), num_pages(the number of pages of lecture note file), note_name(the name of lecture note file), user_id(id of user), and translated_date(It stores when the file was converted to a voice file).

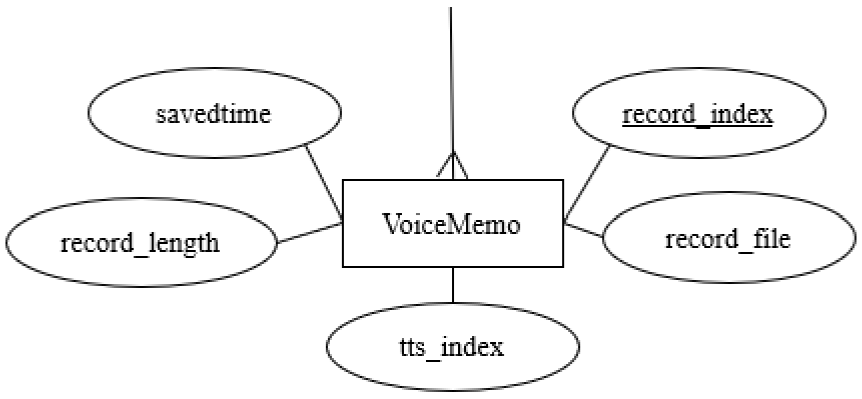
7.2.1.2. **TTSfile**



[Figure 25] TTSfile ER diagram

TTSfile saves each TTS file and its information. Each TTS file represents a TTS file of a page of each lecture note file. It consists of note_id(unique index of note file to which the TTS file belongs), tts_index(PK, unique index), tts_page(information about which page of the notefile is translated), tts_file(a path of tts file), and tts_length(playback length of tts_file).

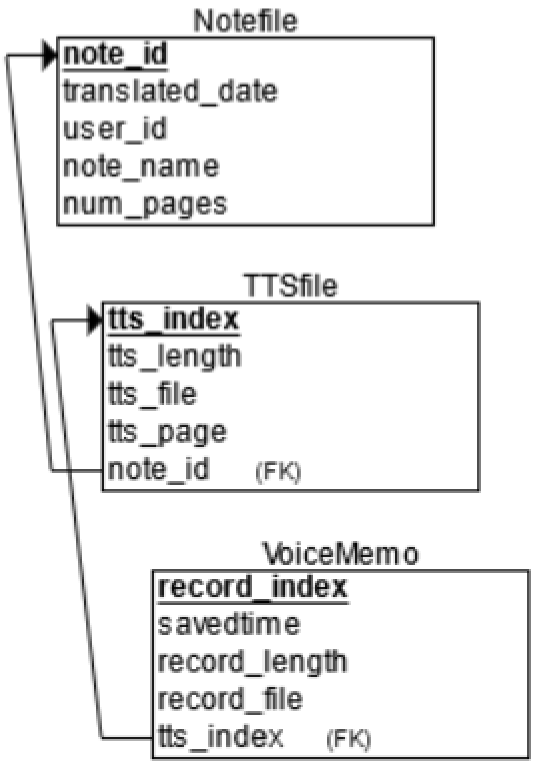
7.2.1.3. **VoiceMemo**



[Figure 26] VoiceMemo ER diagram

VoiceMemo saves each voice memo file and its information. It consists of savedtime (Saved time of memo file. This time is based on the playback time of the tts file on the page to which the voice memo belongs), record_length(playback length of record_file), tts_index(unique index of note file to which the voice memo belongs), record_file(a path of voice memo record file), and record_index(PK, unique index).

7.3. Relational Schema



[Figure 27] Relational Schema

7.4. SQL DDL

7.4.1. Notefile

CREATE TABLE Notefile

(

```
  translated_date DATE NOT NULL,
  user_id VARCHAR(50) NOT NULL,
  note_name VARCHAR(200) NOT NULL,
  num_pages INT NOT NULL,
  note_id INT NOT NULL,
  PRIMARY KEY (note_id)
);
```

7.4.2. TTSfile

```
CREATE TABLE TTSfile
(
  tts_length INT NOT NULL,
  tts_file VARCHAR(200) NOT NULL,
  tts_page INT NOT NULL,
  tts_index INT NOT NULL,
  note_id INT NOT NULL,
  PRIMARY KEY (tts_index),
  FOREIGN KEY (note_id) REFERENCES Notefile(note_id)
);
```

7.4.3. VoiceMemo

```
CREATE TABLE VoiceMemo
(
 savedtime INT NOT NULL,
 record_length INT NOT NULL,
 record_index INT NOT NULL,
 record_file VARCHAR(200) NOT NULL,
 tts_index INT NOT NULL,
 PRIMARY KEY (record_index),
 FOREIGN KEY (tts_index) REFERENCES TTSfile(tts_index)
);
```

8. Testing Plan

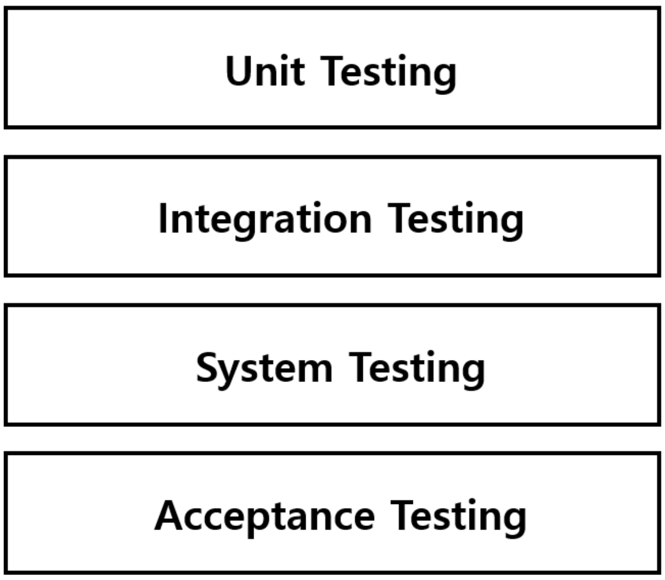
8.1. Objectives

Tests should be performed to ensure whether the system is performed as intended and check whether the system has internal flaws. This chapter describes testing policy which consists of development testing, release testing, and user testing. Development testing verifies the functionality of each component and subsystem. Release testing verifies whether the requirements in the requirements statement were reflected in the system. In User testing, testing takes place in the actual user environment.

8.2. Testing Policy

8.2.1. Development Testing

Development testing is performed in the construction stage in order to find possible errors and potential risks. Development testing should be conducted thoroughly because it gets much more expensive to change defects when passed this stage. Development testing consists of 4 processes: unit testing, integration testing, system testing and acceptance testing.



.

[Figure 28] Process of Development Testing

8.2.1.1. **Unit Testing**

Unit testing independently checks each component's functionality and defects. When unit testing is performed inefficiently and the issue is not defected, detecting and fixing cost increases dramatically. Therefore, proper unit testing should be done before integration. Developers will isolate each component and check whether each component such as PDF reader or Testing is producing desired output from test cases.

8.2.1.2. **Integration Testing**

In the integration testing, components tested in unit testing will be combined and tested. Integration will be performed incrementally to easily detect which interaction is causing the problem. Moreover, when performing the unit testing, new requirements might arise, and existing requirements can change. Implementation of the changed requirements should be tested in this part.

8.2.1.3. **System Testing**

Complete system is tested in the system testing. Often, system testing is final test of checking whether the system has met its specification. System testing assesses both functional and nonfunctional requirements for the system. System's non-functional requirements are firstly tested in this part as they can be only seen after the system is integrated.

8.2.1.4. **Acceptance Testing**

In the acceptance testing, whether requirements are meeting specification or contract is checked. Unlike tests performed before, acceptance tests are mainly conducted by customers and end users. Other stakeholders can also participate in the process as well. Acceptance test consists of two parts: user acceptance test and operational acceptance test. User acceptance test validates business flow and operational acceptance test validates system flow from installation to each use case and environment.

8.2.2. Release Testing

Main purpose of all the systems is to be delivered to the market and customer. In release testing, each release of the system is tested by a test group who are not in the development team. Entire functionality of the system is checked to check if the system is prepared for the users to use. Release testing not only tests functionality but also emergent properties such as reliability and performance.

8.2.3. User Testing

User testing can be understood as an assessment of usability. We should set up realistic situations like bringing user information and taking tests. However, we should also consider all user actions because unexpected user action might lead to problems.

9. Development Plan

9.1. Objectives

This chapter describes the tools used in the development process and the development environment.

9.2. Frontend Environment

9.2.1. Adobe Xd

Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc, available for macOS and Windows. It provides prototype features that can be previewed on supported mobile devices and real-time collaboration for fast feedback.

[Figure 29] Adobe Xd

9.2.2. React Native

React Native is an open-source mobile application framework created by Facebook, Inc. It is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. This will reduce the development time when both IOS and Android Apps have to be made. Also, it is easy to manage change in codes with React Native. Changing in one code base and deploying to both IOS and Android will make the iterative development efficient and cost-effective.

[Figure 30] React Native

9.3. Backend Environment

9.3.1. React Native TTS & STT

React Native TTS is a text-to-speech library for React Native on iOS and Android. It is an open-source library that works for the ability to convert text from class material files (PDF, WORD) to speech.

React Native STT is a speech-to-text library. It functions as a speech-to-text translation for the test-taking function of this system.

9.3.2. SQLite

Most applications need a way to store and use internal data. Designed for this purpose, SQLite is a built-in, open-source database written in C, and can be queried in typical SQL. SQLite has several features that can be viewed in advanced databases, including full-text indexing and JSON data support. Typically, application data stored in semi-structured formats, such as YAML or XML, can be stored as SQLite tables, making it easier to access and process data faster.

[Figure 31] SQLite

9.3.3. Github

Git is a distributed version management system developed to effectively manage source code. Github is a web hosting service that supports projects using Git that is a platform for developers' version control and collaboration. We use Github for developing and controlling versions of our system.

[Figure 32] Github

9.4. Constraints

The system is designed and developed with some of the limitations described below in mind.

Because we use a reuse-oriented approach, high attention needs to be paid to the copyright of the software used.

- The system is intended for visually impaired users, so all features should be developed to make it easier for them to use.
- Don't violate open source licenses when using it.
- Comply with open source license notice requirements.
- Optimization of code is always needed to improve system performance and prevent resource waste.
- The code should be written in a way that makes it easier for team members to understand each other's work outcomes.
- Functions that require thoroughly following the time limit, such as the test-taking function, must produce results within the time limit.
- Design the system scalable enough to add other features for visually impaired users to the system.
- Develop with the MacOS environment.
- Must support Android version 6 (API Level 23) to version 11 (API Level 30)
- Must support iOS version 11 to version 14

9.5. Assumptions and Dependencies

The overall content of the system described in the above document is written assuming that our system is developed using open source software and uses a cross platform framework to support both iOS and Android versions. Therefore, we need to develop with the MacOS environment because Xcode, the simulation and build tool of the iOS application, is only supported in MacOS.

10. Supporting Information

10.1. Software Design Specification

This software design specification was written in accordance with the IEEE Recommendation

(IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).

10.2. Document History

[Table 1] Document History

Date	Version	Description	Writer
4/24	1.0	3.1	Kang Hyunmuk
4/25	1.1	File Formatting	Kim Jihye
4/26	1.2	Login & Video play	Park Jiye
4/26	1.3	4.2.4.1, 4.2.4.2	Shin Wonchul
4/28	1.4	4.2.3	Oh Seungjun
4/30	1.5	9.1, 9.2, 9.3	Kang Hyunmuk
5/3	1.6	4.2.1, 4.2.2	Kim Jihye
5/3	1.7	4.2.5, 4.2.6	Park Jiye
5/4	1.8	4.2.4.3, 4.2.4.4	Shin Wonchul
5/6	1.9	5.3.1, 5.3.2, 5.3.3	Oh Seungjun
5/6	2.0	9.4, 9.5	Kang Hyunmuk
5/8	2.1	5.3.1, 7.1-7.4	Kim Jihye
5/8	2.2	1-2	Park Jiye
5/10	2.3	8	Shin Wonchul
5/10	2.4	6.1, 6.2, 6.3	Oh Seungjun