

**I-Campus Lecture Sub-System**

**Software Design Specification**

2021.05.16.

**Introduction to Software Engineering 41**

**TEAM 4**

Team Leader Jisun Lee

Team Member Chihyun Lee

Team Member Deokjae Kang

Team Member Dongjun Lee

Team Member Seongwoo Kim

Team Member Krogross Ryan

**CONTENTS**

[1. Preface 8](#_Toc71945060)

[1.1. Readership 8](#_Toc71945061)

[1.2. Scope 8](#_Toc71945062)

[1.3. Objective 8](#_Toc71945063)

[1.4. Document Structure 9](#_Toc71945064)

[2. Introduction 9](#_Toc71945065)

[2.1. Objectives 10](#_Toc71945066)

[2.2. Applied Diagrams 10](#_Toc71945067)

[2.2.1. UML 10](#_Toc71945068)

[2.2.2. Use case Diagram 10](#_Toc71945069)

[2.2.3. Sequence Diagram 10](#_Toc71945070)

[2.2.4. Class Diagram 11](#_Toc71945071)

[2.2.5. Context Diagram 11](#_Toc71945072)

[2.2.6. Entity Relationship Diagram 12](#_Toc71945073)

[2.3. Applied Tools 12](#_Toc71945074)

[2.3.1. Microsoft PowerPoint 12](#_Toc71945075)

[2.3.2. Draw.IO 12](#_Toc71945076)

[2.4. Project Scope 12](#_Toc71945077)

[2.5. References 13](#_Toc71945078)

[3. System Architecture – Overall 13](#_Toc71945079)

[3.1. Objectives 13](#_Toc71945080)

[3.2. System Organization 13](#_Toc71945081)

[3.2.1. Context Diagram 14](#_Toc71945082)

[3.2.2. Sequence Diagram 15](#_Toc71945083)

[3.2.3. Use Case Diagram 16](#_Toc71945084)

[4. System Architecture – Frontend 16](#_Toc71945085)

[4.1. Objectives 16](#_Toc71945086)

[4.2. Subcomponents 17](#_Toc71945087)

[4.2.1. User controller 17](#_Toc71945088)

[4.2.1.1. Attributes 17](#_Toc71945089)

[4.2.1.2. Methods 17](#_Toc71945090)

[4.2.1.3. Class Diagram 18](#_Toc71945091)

[4.2.1.4. Sequence Diagram 18](#_Toc71945092)

[4.2.2. Bookmark Controller 19](#_Toc71945093)

[4.2.2.1. Attributes 19](#_Toc71945094)

[4.2.2.2. Methods 19](#_Toc71945095)

[4.2.2.3. Class Diagram 20](#_Toc71945096)

[4.2.2.4. Sequence Diagram 21](#_Toc71945097)

[4.2.3. Lecture Controller 21](#_Toc71945098)

[4.2.3.1. Attributes 21](#_Toc71945099)

[4.2.3.2. Methods 21](#_Toc71945100)

[4.2.3.3. Class Diagram 22](#_Toc71945101)

[4.2.3.4. Sequence Diagram 23](#_Toc71945102)

[5. System Architecture – Backend 23](#_Toc71945103)

[5.1. Objectives 23](#_Toc71945104)

[5.2. Overall Architecture 24](#_Toc71945105)

[5.3. Subcomponents 25](#_Toc71945106)

[5.3.1. Canvas LMS Functions 25](#_Toc71945107)

[5.3.1.1. Canvas LMS Functions – Course System 25](#_Toc71945108)

[5.3.1.1.1. Endpoint Hander 25](#_Toc71945109)

[5.3.1.1.2. Course DB Handler 25](#_Toc71945110)

[5.3.1.1.3. Course System 25](#_Toc71945111)

[5.3.1.2. Canvas LMS Functions – User System 26](#_Toc71945112)

[5.3.1.2.1. User DB Hander 26](#_Toc71945113)

[5.3.1.2.2. User System 26](#_Toc71945114)

[5.3.2. xAPI 26](#_Toc71945115)

[5.3.3. SIS Data 27](#_Toc71945116)

[5.3.3.1. SIS imports 27](#_Toc71945117)

[5.3.3.2. SIS Integration 27](#_Toc71945118)

[5.3.4. Bookmark System 28](#_Toc71945119)

[5.3.4.1. Class Diagram 28](#_Toc71945120)

[5.3.4.2. Sequence Diagram 28](#_Toc71945121)

[6. Protocol Design 29](#_Toc71945122)

[6.1. Objectives 29](#_Toc71945123)

[6.2. OAuth2 29](#_Toc71945124)

[6.2.1. Log-in 29](#_Toc71945125)

[6.3. User 30](#_Toc71945126)

[6.3.1. Get User Profile 30](#_Toc71945127)

[6.3.2. Store User Data 30](#_Toc71945128)

[6.3.2.1. Description 30](#_Toc71945129)

[6.3.2.2. Request and Response 31](#_Toc71945130)

[6.4. File 31](#_Toc71945131)

[6.4.1. Upload Lecture 31](#_Toc71945132)

[6.4.2. Upload Assignment 32](#_Toc71945133)

[6.5. Bookmark 33](#_Toc71945134)

[6.5.1. Open User Bookmark 33](#_Toc71945135)

[6.5.2. Open Instructor Bookmark 33](#_Toc71945136)

[6.5.3. Create User Bookmark 34](#_Toc71945137)

[6.5.4. Create Instructor's Bookmark 35](#_Toc71945138)

[6.5.5. Edit Bookmark 35](#_Toc71945139)

[6.5.6. Delete Bookmark 36](#_Toc71945140)

[7. Database Design 37](#_Toc71945141)

[7.1. Objectives 37](#_Toc71945142)

[7.2. ER Diagram 37](#_Toc71945143)

[7.2.1. Entities 38](#_Toc71945144)

[7.2.1.1. User 38](#_Toc71945145)

[7.2.1.2. User Group 39](#_Toc71945146)

[7.2.1.3. Course 39](#_Toc71945147)

[7.2.1.4. File 40](#_Toc71945148)

[7.2.1.5. Media 40](#_Toc71945149)

[7.2.1.6. Bookmark 41](#_Toc71945150)

[7.3. Relational Schema 42](#_Toc71945151)

[7.4. SQL DDL 42](#_Toc71945152)

[7.4.1. User 42](#_Toc71945153)

[7.4.2. User\_group 43](#_Toc71945154)

[7.4.3. Course 43](#_Toc71945155)

[7.4.4. File 44](#_Toc71945156)

[7.4.5. Media 44](#_Toc71945157)

[7.4.6. Bookmark 45](#_Toc71945158)

[8. Development Plan 45](#_Toc71945159)

[8.1. Objectives 45](#_Toc71945160)

[8.2. Frontend Environment 45](#_Toc71945161)

[8.2.1. Adobe Photoshop 45](#_Toc71945162)

[8.2.2. Adobe Xd 46](#_Toc71945163)

[8.2.3. Android Studio 46](#_Toc71945164)

[8.3. Backend Environment 47](#_Toc71945165)

[8.3.1. Github 47](#_Toc71945166)

[8.3.2. Firebase 47](#_Toc71945167)

[8.3.3. Android Studio 48](#_Toc71945168)

[8.4. Constraints 48](#_Toc71945169)

[8.5. Assumptions and Dependencies 49](#_Toc71945170)

[9. Supporting Information 49](#_Toc71945171)

[9.1. Software Design Specification 49](#_Toc71945172)

[9.2. Document History 49](#_Toc71945173)

**LIST OF FIGURES**

[[Figure 1] Overall system architecture 14](#_Toc71945289)

[[Figure 2] Overall context diagram 15](#_Toc71945290)

[[Figure 3] Overall sequence diagram 15](#_Toc71945291)

[[Figure 4] Use case diagram 16](#_Toc71945292)

[[Figure 5] Class diagram – User Controller 18](#_Toc71945293)

[[Figure 6] Sequence diagram – User Controller 18](#_Toc71945294)

[[Figure 7] Class diagram – Bookmark Controller 20](#_Toc71945295)

[[Figure 8] Sequence diagram – Bookmark Controller 21](#_Toc71945296)

[[Figure 9] Class diagram – Lecture Controller 22](#_Toc71945297)

[[Figure 10] Sequence diagram – Lecture Controller 23](#_Toc71945298)

[[Figure 11] Overall architecture 24](#_Toc71945299)

[[Figure 12] Class diagram – Course system 25](#_Toc71945300)

[[Figure 13] Class diagram – User system 26](#_Toc71945301)

[[Figure 14] Class diagram – SIS System 27](#_Toc71945302)

[[Figure 15] Class diagram – Bookmark System 28](#_Toc71945303)

[[Figure 16] Sequence diagram – Bookmark System 28](#_Toc71945304)

[[Figure 17] ER-diagram 38](#_Toc71945305)

[[Figure 18] ER diagram, Entity, User 38](#_Toc71945306)

[[Figure 19] ER diagram, Entity, Search History 39](#_Toc71945307)

[[Figure 20] ER diagram, Entity, Cart 39](#_Toc71945308)

[[Figure 21] ER diagram, Entity, Laptop 40](#_Toc71945309)

[[Figure 22] ER diagram, Entity, Laptop 40](#_Toc71945310)

[[Figure 23] ER diagram, Entity, Laptop 41](#_Toc71945311)

[[Figure 24] Relational Schema 42](#_Toc71945312)

[[Figure 34] Adobe Photoshop logo 45](#_Toc71945313)

[[Figure 35] Adobe Xd logo 46](#_Toc71945314)

[[Figure 36] Android Studio logo 46](#_Toc71945315)

[[Figure 37] Github logo 47](#_Toc71945316)

[[Figure 38] Firebase logo 47](#_Toc71945317)

[[Figure 39] Android Studio logo 48](#_Toc71945318)

LIST OF TABLES

[[Table 1] Table of Log-in request 29](#_Toc71945319)

[[Table 2] Table of Log-in response 29](#_Toc71945320)

[[Table 3] Table of get user profile request 30](#_Toc71945321)

[[Table 4] Table of get user profile response 30](#_Toc71945322)

[[Table 5] Table of store user data request 31](#_Toc71945323)

[[Table 6] Table of store user data response 31](#_Toc71945324)

[[Table 7] Table of upload lecture request 31](#_Toc71945325)

[[Table 8] Table of upload lecture response 32](#_Toc71945326)

[[Table 9] Table of upload lecture request 32](#_Toc71945327)

[[Table 10] Table of upload lecture response 32](#_Toc71945328)

[[Table 11] Table of Open User Bookmark request 33](#_Toc71945329)

[[Table 12] Table of Open User Bookmark response 33](#_Toc71945330)

[[Table 13] Table of Open Instructor Bookmark request 33](#_Toc71945331)

[[Table 14] Table of Open Instructor Bookmark response 34](#_Toc71945332)

[[Table 15] Table of Create User Bookmark request 34](#_Toc71945333)

[[Table 16] Table of Create User Bookmark response 34](#_Toc71945334)

[[Table 17] Table of Create Instructor's Bookmark request 35](#_Toc71945335)

[[Table 18] Table of Create Instructor's Bookmark response 35](#_Toc71945336)

[[Table 19] Table of Edit Bookmark request 36](#_Toc71945337)

[[Table 20] Table of Edit Bookmark response 36](#_Toc71945338)

[[Table 21] Table of Delete Bookmark request 36](#_Toc71945339)

[[Table 22] Table of Delete Bookmark response 36](#_Toc71945340)

[[Table 25] Document History 49](#_Toc71945341)

1. Preface

This document describes the procedures and specifications that are involved with designing and implementing a bookmarking and notetaking feature to the i-Campus online learning platform. The following chapter discusses the intended audience for this document and gives brief descriptions on each topic discussed in this document.

* 1. Readership

This document is targeted to the developers and any stakeholders that are involved in the addition of this feature to i-Campus. Developers should use this document in order to view the software structure and procedures that are necessary for implementation. Stakeholders can use this document in order to make sure that their requirements are being met by the developers.

* 1. Scope

This Design Specification is to be used by Software Engineers and Software Quality Engineers as a guide to implement the i-Campus sub-system for video bookmark system.

* 1. Objective

The primary purpose of this Software Design Document is to provide a description of the technical design aspects for i-Campus sub system. This document describes the software architecture and software design decisions for the implementation of our project. It also provides an architectural overview of the system to depict different aspects of the system. Furthermore, it specifies the structure and design of some of the modules discussed in the SRS document and displays some of the use cases that have been transformed into sequential and activity diagrams. Class diagrams show how the programming team would implement the specific module. The intended audience of this document is, but is not limited to, the stakeholders, developers, designers, and software testers.

* 1. 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 using several ER diagrams and SQL DDL.
* **8. Development Plan**: this chapter describes which tools to use to develop the system, constraints, assumption, and dependencies for developing this system.
* **9. Supporting Information**: this chapter describes baseline of this document and history of this document.

1. Introduction

This goal project is to develop and design an application that can be used to learn lectures uploaded to i-Campus. This system will allow users to create bookmarks at any point in a lecture video and be able to move to the bookmarks as needed. This design document presents the designs that will be used in implementing the feature. The designs described fulfill the requirements specified in the Software Requirements Specifications document presented earlier.

* 1. Objectives

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

* 1. Applied Diagrams
     1. UML

UML is an acronym for Unified Modeling Language. UML is a widely used tool which helps to model and document software. UML is a language commonly used by business analysts, software architects, and developers to describe, specify, design, and document existing or new business processes, structure, and behavior of software system components. UML can be applied to diverse application domains (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component software development methods and for various implementation platforms (e.g., J2EE, .NET). It is based on diagrammatic representations of software components. Visual representations of software systems make finding and correcting errors easier.

* + 1. Use case Diagram

A central part of the system is the functional requirements that the system fulfills. Use Case diagrams are used to analyze the system’s high-level requirements. These requirements are expressed through different use cases. We notice three main components of this UML diagram: Functional requirements – represented as use cases; a verb describing an action. Actors – they interact with the system; an actor can be a human being, an organization or an internal or external application. Relationships between actors and use cases – represented using straight arrows.

* + 1. Sequence Diagram

Sequence diagrams are one of the most important UML diagrams used by the computer science community and business application developers. Lately, these diagrams have become popular in depicting business processes due to their visually self-explanatory nature. As the name suggests, sequence diagrams describe the sequence of interactions that happen between actors and objects. Actors or objects are only active when needed or when another object wants to communicate with them. All communication is represented in a chronological manner.

* + 1. Class Diagram

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. Just as classes are the building blocks of objects, class diagrams are the building blocks of UML. The various components in a class diagram represent the classes that will be programmed, the main objects, or the interactions between classes and objects. The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.

* + 1. Context Diagram

The system context diagram (also known as a level 0 DFD) is the highest level in a data flow diagram and contains only one process, representing the entire system, which establishes the context and boundaries of the system to be modeled. It identifies the flows of information between the system and external entities (i.e. actors). A context diagram is typically included in a requirements document as is read by the project stakeholders and thus should be written in natural language. The objective of the system context diagram is to focus attention on external factors and events that should be considered when developing a complete set of systems requirements and constraints. A system context diagram is often used early in development to determine the scope and it shows all external entities that may interact with a system. This diagram pictures the system at the center, with no details of its interior structure, surrounded by all its external entities, interacting systems, and environments.

* + 1. Entity Relationship Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database where each entity represents a software component. An entity set is a collection of such entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. Entity Relationship diagrams are used to sketch out the design of a database.

* 1. Applied Tools
     1. Microsoft PowerPoint

This is a tool that supports drawing text and figures. It is convenient to draw diagrams using various shapes. In addition, it is easy to edit because it works as full word-processor formatting (which is a tool for working with documents), graphic shapes with attached text for drawing diagrams and tables.

* + 1. Draw.IO

This is a web-based database modeling tool that lets you quickly and easily create Entity Relationship Diagrams (ERDs), Relational Schemas (Relational Diagrams)

* 1. Project Scope

The i-Campus lecture sub-system was built as a quality-of-life feature for when users wish to find important or necessary parts of a video. It can be used for the purpose of reviewing for an exam, assignment, etc. This system is based on a relational database and information that can be helpful for the lecture will be built on the server for the various lectures that exist in i-Campus.

* 1. References

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

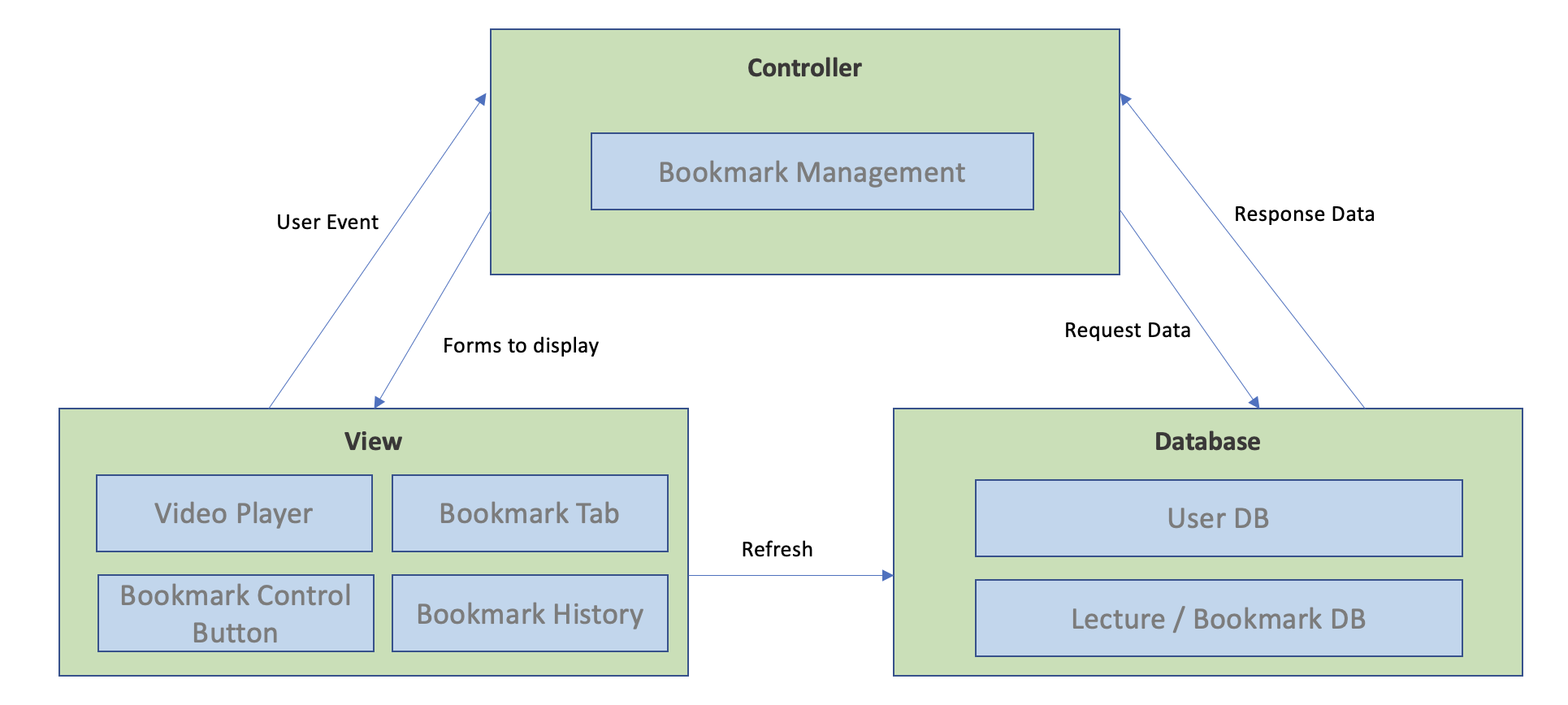
* Team 1, 2020 Spring. Software Design Document, SKKU.
* Appleton, Brad. A Software Design Specification Template. N.d.
* P. Burke, K. Martin, D. Longtin. Software Design Specification for a One Runway Airport/Air Traffic Controller Simulation  
  <<https://www.academia.edu/29811493/SOFTWARE_DESIGN_SPECIFICATION_FOR_A_ONE_RUNWAY_AIRPORT_AIR_TRAFFIC_CONTROLLER_SIMULATION>>.
* Instructure, inc. 2021 <<https://canvas.instructure.com/doc/api/users.html>>

1. System Architecture – Overall
   1. Objectives

In this chapter, we describe and show the organization of the system. Descriptions of the system range from the frontend design to the backend design.

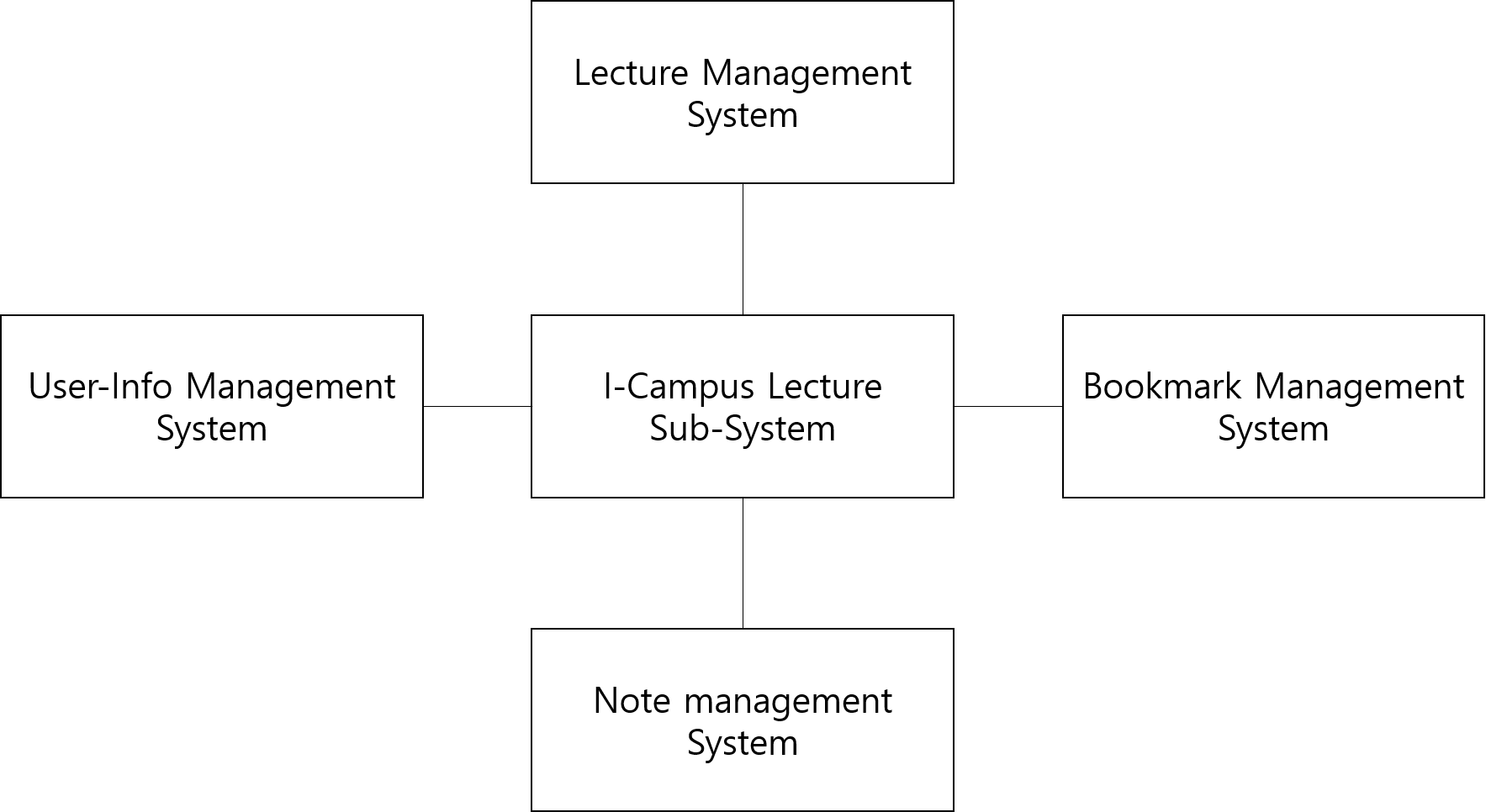
* 1. System Organization

This service is designed by applying the client - server model, frontend Application is responsible for all interactions with users, and the front-end application and back-end application send and receive data through HTTP communication based on JSON. The back-end application distributes design specification requests from the front-end to the controller, obtains the required object information from the database, processes it from the database, and delivers it to the JSON format.



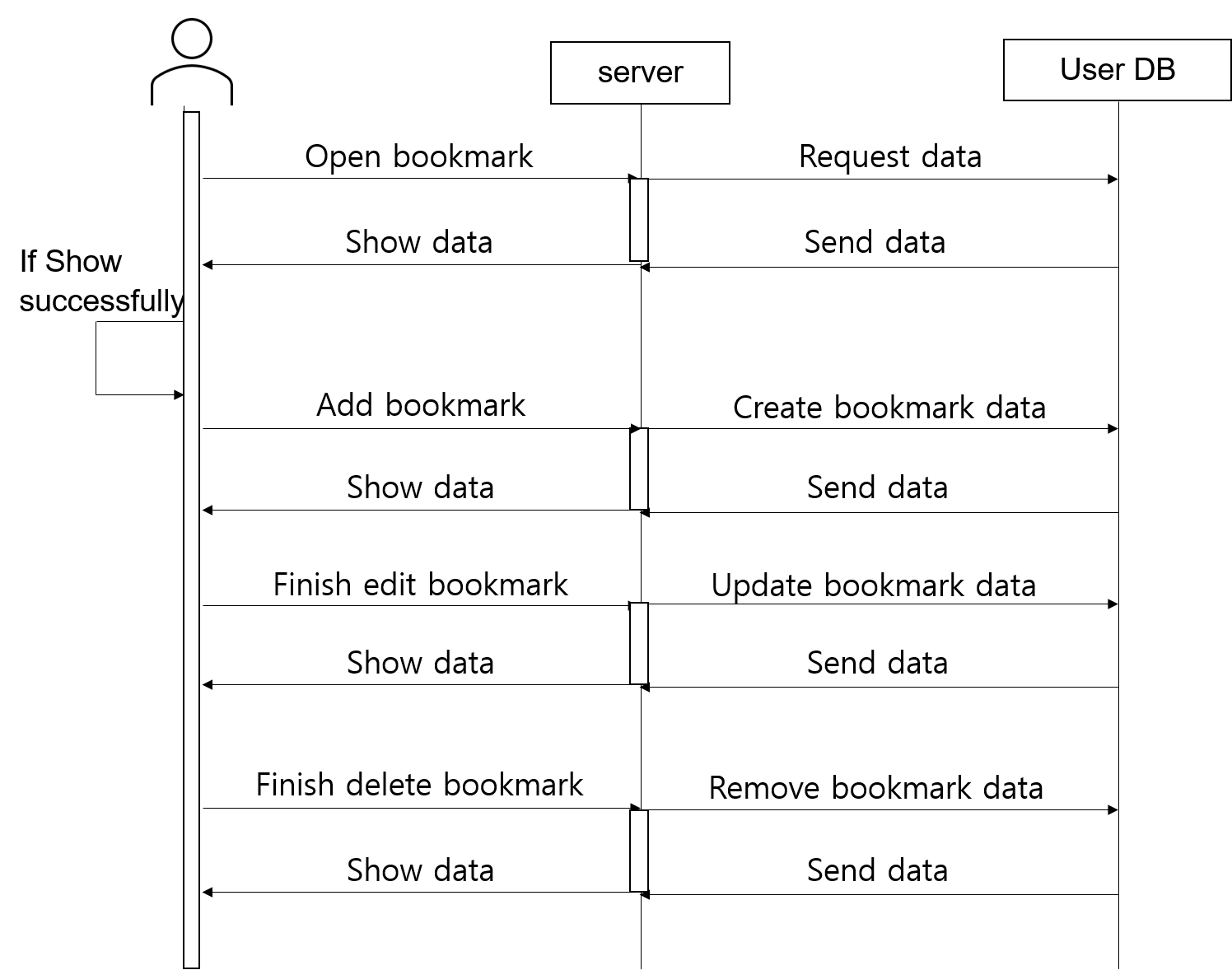
[Figure 1] Overall system architecture

* + 1. Context Diagram



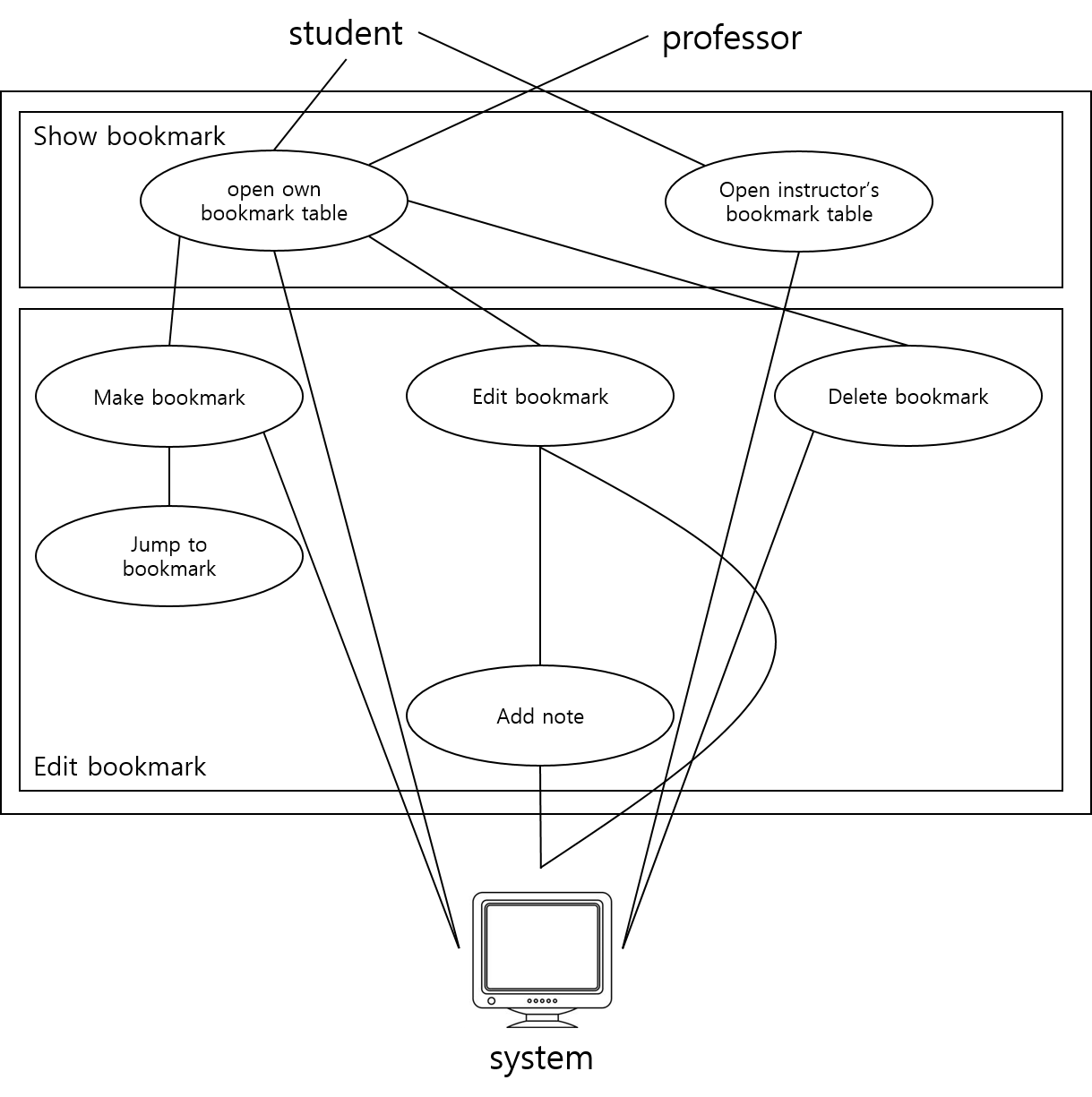
[Figure 2] Overall context diagram

* + 1. Sequence Diagram



[Figure 3] Overall sequence diagram

* + 1. Use Case Diagram



[Figure 4] Use case diagram

1. System Architecture – Frontend
   1. Objectives

This chapter describes the functionality, structure, and relationships of the components that make up the frontend.

* 1. Subcomponents
     1. User controller

This shows the different permissions and functions depending on the user's type. The user's type in that course is automatically taken from the course api in Canvas.

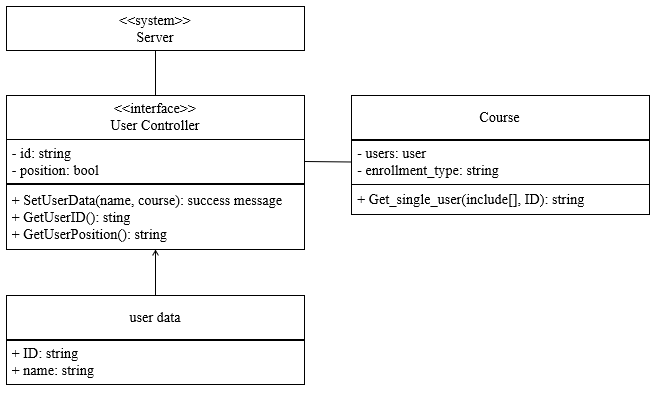
* + - 1. Attributes

These are the attributes that profile object has.

* **User id**: id of the user (email address)
* **Position**: The role of the user in the course (Instructor or learner).
  + - 1. Methods

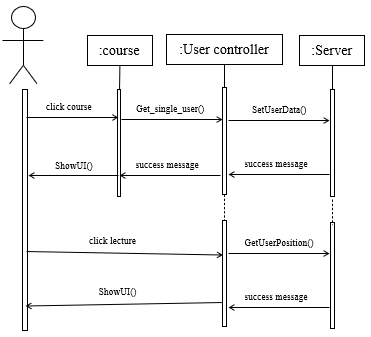
These are the methods that profile class has.

* SetUserData(name, course)
* GetUserID()
* GetUserPosition()
* ShowUI(position)
  + - 1. Class Diagram



[Figure 5] Class diagram – User Controller

* + - 1. Sequence Diagram



[Figure 6] Sequence diagram – User Controller

* + 1. Bookmark Controller

manages bookmark tables that manage the creation, modification, deletion, and other features of bookmarks.

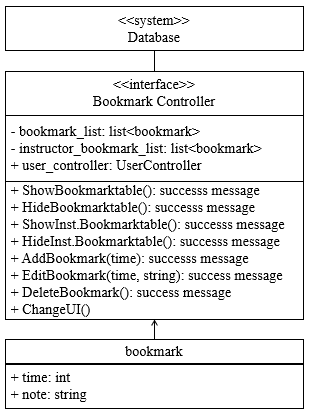
* + - 1. Attributes

These are the attributes that profile object has.

* **bookmark\_list**: List of bookmarks made by user.
* **instructor\_bookmark\_list**: List of bookmarks made by instructor of course. If user is intstructor, this list will be empty.
* **user\_controller**: It changes the flow according to user data.
  + - 1. Methods

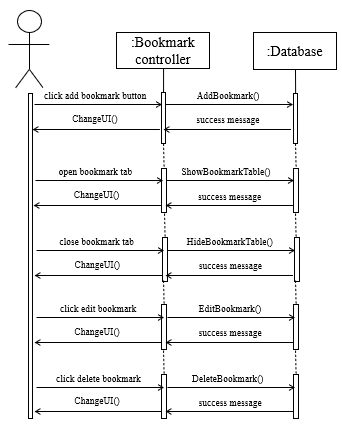
These are the methods that profile class has.

* ShowBookmarkTab()
* HideBookmarkTab()
* ShowInst.BookmarkTab()
* HideInst.BookmarkTab()
* AddBookmark(time)
* EditBookmark(time, string)
* DeleteBookmark()
  + - 1. Class Diagram



[Figure 7] Class diagram – Bookmark Controller

* + - 1. Sequence Diagram



[Figure 8] Sequence diagram – Bookmark Controller

* + 1. Lecture Controller

The bookmark flag created during bookmarking allows you to change the lecture directly by adjusting the lecture time and folding it.

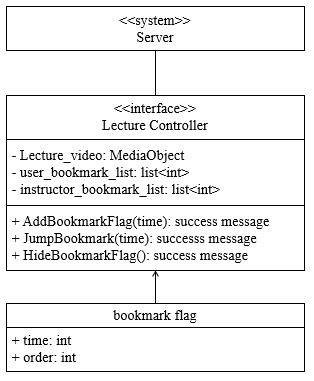
* + - 1. Attributes

These are the attributes that profile object has.

* **user\_bookmark\_list**: List of bookmark times made by user.
* **instructor\_bookmark\_list**: List of bookmark times made by instructor of course. If user is intstructor, this list will be empty.
* **Lecture\_video**: It contains all the information in the video in the MediaObject object.
  + - 1. Methods

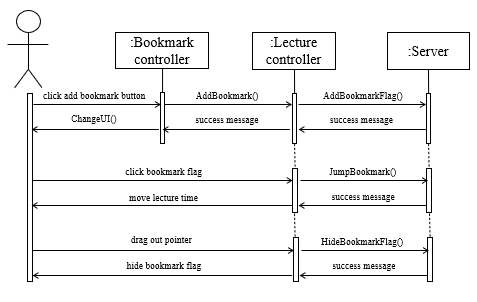
These are the methods that profile class has.

* AddBookmarkFlag(time)
* JumpBookmark(time)
* HideBookmarkFlag()
  + - 1. Class Diagram



[Figure 9] Class diagram – Lecture Controller

* + - 1. Sequence Diagram

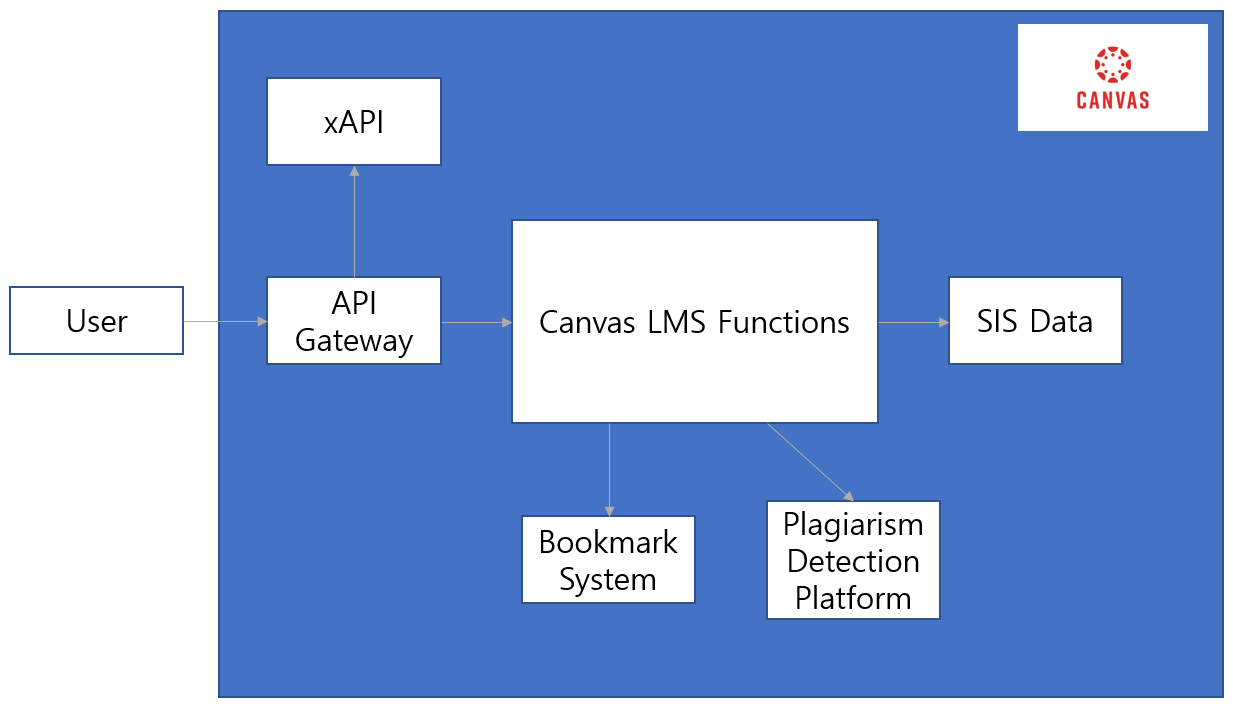


[Figure 10] Sequence diagram – Lecture Controller

1. System Architecture – Backend
   1. Objectives

This chapter describes the structure of the back-end system including DB and API.

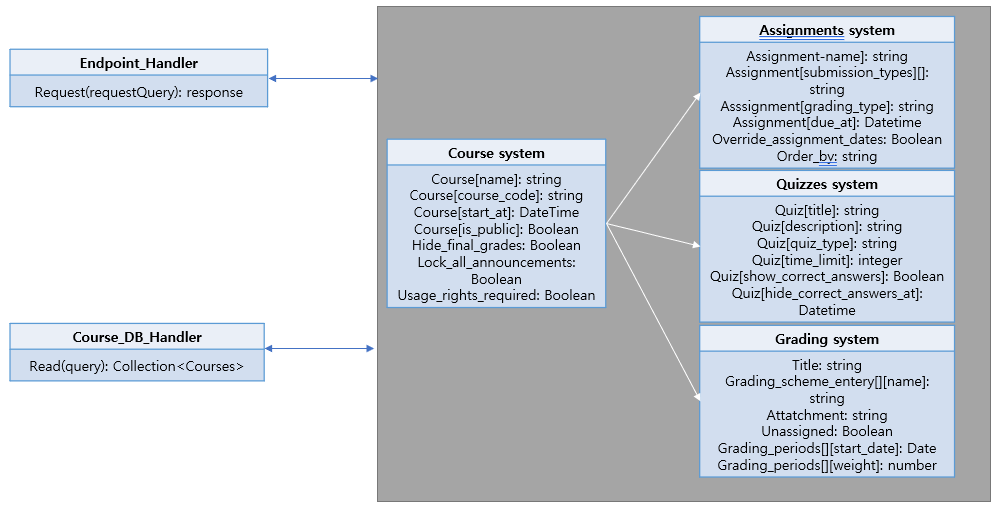
* 1. Overall Architecture



[Figure 11] Overall architecture

This is the overall architecture of canvas and our bookmark function. The API gateway receives the request from the user. There are xAPIs and functions in Canvas LMS such as Accounts, Assignments, Discussions, and quizzes. Bookmark function will be one of the functions in Canvas LMS and SIS(Student Information Services) data will be stored in the database. Plagiarism detection platform provides a standard way for LTI2 tool providers to seamlessly integrate plagiarism detection tools with Canvas.

* 1. Subcomponents
     1. Canvas LMS Functions
        1. Canvas LMS Functions – Course System



[Figure 12] Class diagram – Course system

* + - * 1. Endpoint Hander

Canvas LMS Functions – Course System

* + - * 1. Course DB Handler

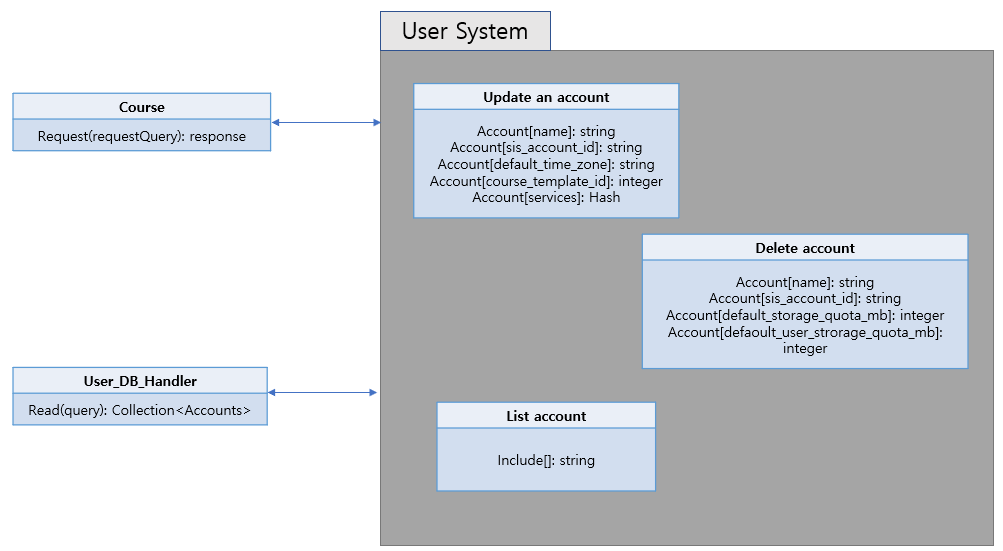
Interface to communicate with DB with course information. The information about each course is stored and it retrieves the course information to show it to the user.

* + - * 1. Course System

System for accessing course information. Users can list, create, delete, and update a course. There will be additional systems for assignments, quizzes and grading.

* + - 1. Canvas LMS Functions – User System

API gateway. Distribute requests form the user to the appropriate controller or API



[Figure 13] Class diagram – User system

* + - * 1. User DB Hander

Interface to communicate DB with user information. The information about user accounts is stored and gets the user information to show it to them.

* + - * 1. User System

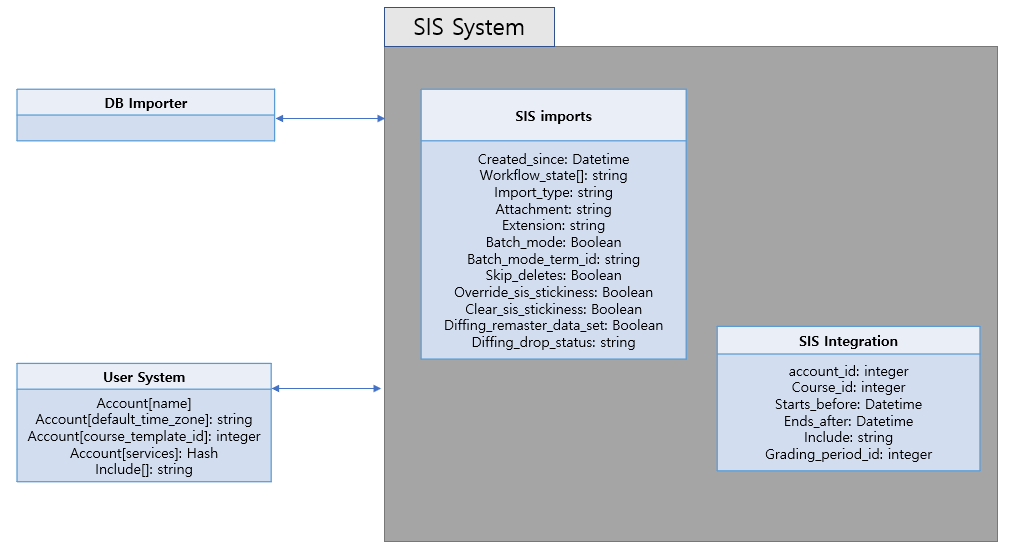
User can make a new account, update them and delete their account.

* + 1. xAPI

xAPI(Experience API) is a data and interface standard that lets software applications capture and share data on human performance. There are functions such as training progress data from a simulation, chatting with a mentor, and watching a video.

* + 1. SIS Data

LMS database gets data from DB Importer scripts and sends it to SIS Importer Scripts. Then this information turns into Roster files and is sent to Canvas LMS.



[Figure 14] Class diagram – SIS System

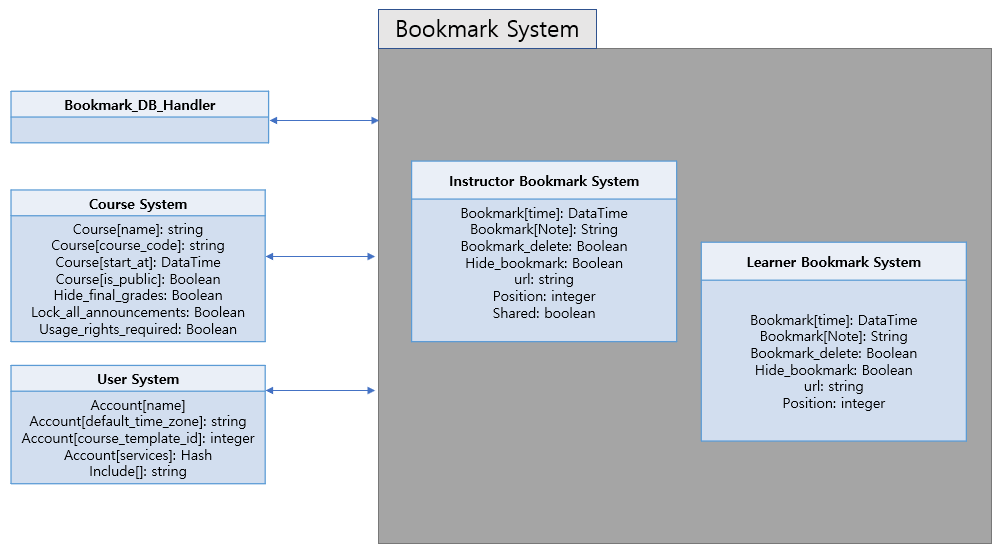
* + - 1. SIS imports

Imports data from Student Information Systems. Gets SIS import list, the current importing SIS import, import SIS data, and abort SIS imports.

* + - 1. SIS Integration

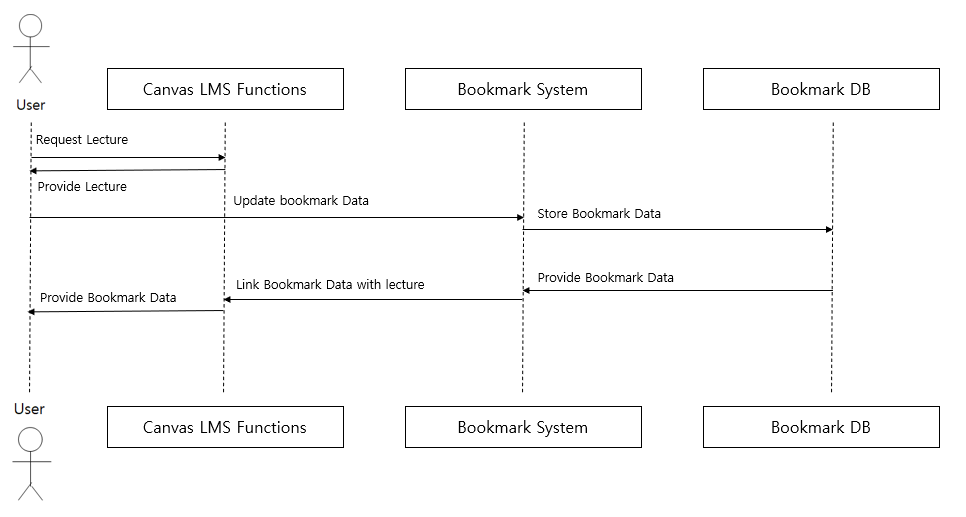
Includes helpers for integration with SIS Systems. It retrieves assignments enabled for grade export to SIS and disables assignments currently enabled for grad export to SIS.

* + 1. Bookmark System
       1. Class Diagram



[Figure 15] Class diagram – Bookmark System

* + - 1. Sequence Diagram



[Figure 16] Sequence diagram – Bookmark System

1. Protocol Design
   1. Objectives

In this chapter, we describe what structures are used for protocol design which are used for interactions between subsystems, especially between web or app applications (front-end) and the server (back-end) of the canvas-lms bookmark System. The chapter also describes how each interface is defined.

* 1. OAuth2
     1. Log-in
* Request

[Table 1] Table of Log-in request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Protocol | OAuth | |
| Request Parameters | client\_id | The client id for your registered application. |
| response\_type | The type of OAuth2 response requested. The only currently supported value is code. |
| redirect\_uri | The URL where the user will be redirected after authorization |

* Response

[Table 2] Table of Log-in response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | Access Token | Token for access |
| Message | Success message |
| Failure  Response Body | Message | Fail Message |

* 1. User
     1. Get User Profile
* Request

[Table 3] Table of get user profile request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | GET | |
| URI | /api/v1/users/:user\_id/profile | |
| Request Body | X | X |
| Header | Authorization | jwt token |

* Response

[Table 4] Table of get user profile response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | User Profile | User Profile Objects |
| Failure  Response Body | Message | Fail Message |

* + 1. Store User Data
       1. Description

Store arbitrary user data as JSON. Arbitrary JSON data can be stored for a User. A typical scenario would be an external site/service that registers users in Canvas and wants to capture additional info about them.

* + - 1. Request and Response
* Request

[Table 5] Table of store user data request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | POST | |
| URI | /api/v1/users/:user\_id/custom\_data | |
| Request Body | User Custom Data | Data Object |
| Header | Authorization | jwt token |

* Response

[Table 6] Table of store user data response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | User Custom Data | Data Object |
| Failure  Response Body | Message | Fail Message |

* 1. File
     1. Upload Lecture
* Request

[Table 7] Table of upload lecture request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | POST | |
| URI | /api/v1/courses/:course\_id/files | |
| Request Body | File | lecture file |
| Header | Authorization | jwt token |

* Response

[Table 8] Table of upload lecture response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | File | lecture file |
| Failure  Response Body | Message | Fail Message |

* + 1. Upload Assignment
* Request

[Table 9] Table of upload lecture request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | POST | |
| URI | /api/v1/courses/:course\_id/assignments/:assignment\_id/submissions/self/files | |
| Request Body | File | Assignment file |
| Header | Authorization | jwt token |

* Response

[Table 10] Table of upload lecture response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | File | Assignment file |
| Failure  Response Body | Message | Fail Message |

* 1. Bookmark
     1. Open User Bookmark
* Request

[Table 11] Table of Open User Bookmark request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | GET | |
| URI | /api/v1/bookmarks/:lecture\_id/:user\_id | |
| Request Body | X | X |
| Header | Authorization | jwt token |

* Response

[Table 12] Table of Open User Bookmark response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | List<Bookmark> | User Bookmark List |
| Failure  Response Body | Message | Fail Message |

* + 1. Open Instructor Bookmark
* Request

[Table 13] Table of Open Instructor Bookmark request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | GET | |
| URI | /api/v1/bookmarks/:lecture\_id/:instructor\_id | |
| Request Body | X | X |
| Header | Authorization | jwt token |

* Response

[Table 14] Table of Open Instructor Bookmark response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | List<Bookmark> | Instructor Bookmark List |
| Failure  Response Body | Message | Fail Message |

* + 1. Create User Bookmark
* Request

[Table 15] Table of Create User Bookmark request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | POST | |
| URI | /api/v1/bookmarks/:lecture\_id/:user\_id | |
| Request Body | Bookmark\_time | Bookmark Attribute |
| Bookmark\_note | Bookmark Attribute |
| Header | Authorization | jwt token |

* Response

[Table 16] Table of Create User Bookmark response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | Bookmark | Created User Bookmark Object |
| Failure  Response Body | Message | Fail Message |

* + 1. Create Instructor's Bookmark
* Request

[Table 17] Table of Create Instructor's Bookmark request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | POST | |
| URI | /api/v1/bookmarks/:lecture\_id/:instructor\_id | |
| Request Body | Bookmark\_time | Bookmark Attribute |
| Bookmark\_note | Bookmark Attribute |
| Header | Authorization | jwt token |

* Response

[Table 18] Table of Create Instructor's Bookmark response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | Bookmark | Created Instructor Bookmark Object |
| Failure  Response Body | Message | Fail Message |

* + 1. Edit Bookmark
* Request

[Table 19] Table of Edit Bookmark request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | PUT | |
| URI | /api/v1/bookmarks/:bookmark\_id | |
| Request Body | Bookmark\_time | Bookmark Attribute |
| Bookmark\_note | Bookmark Attribute |
| Header | Authorization | jwt token |

* Response

[Table 20] Table of Edit Bookmark response

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | Bookmark | Edited Instructor Bookmark Object |
| Failure  Response Body | Message | Fail Message |

* + 1. Delete Bookmark
* Request

[Table 21] Table of Delete Bookmark request

| **Attribute** | **Detail** | |
| --- | --- | --- |
| Method | PUT | |
| URI | /api/v1/bookmarks/:bookmark\_id | |
| Request Body | X | |
| Header | Authorization | jwt token |

* Response

[Table 22] Table of Delete Bookmark response

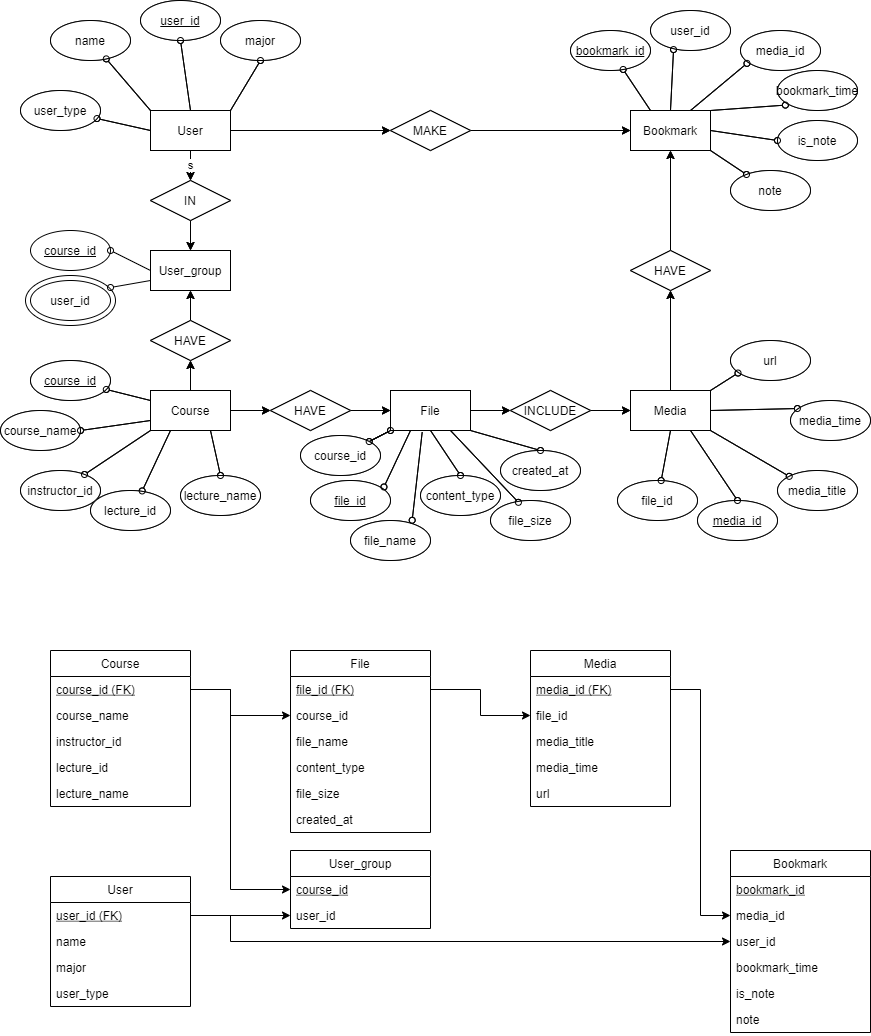
| **Attribute** | **Detail** | |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | HTTP error code = 400 | |
| Success  Response Body | Messege | Success Messege |
| Failure  Response Body | Message | Fail Message |

1. Database Design
   1. Objectives

This section describes the system data structures and how these are to be represented in a database. It first identifies entities and their relationship through ER-diagram (Entity Relationship diagram). Then, it generates Relational Schema and SQL DDL (Data Description Language) specification.

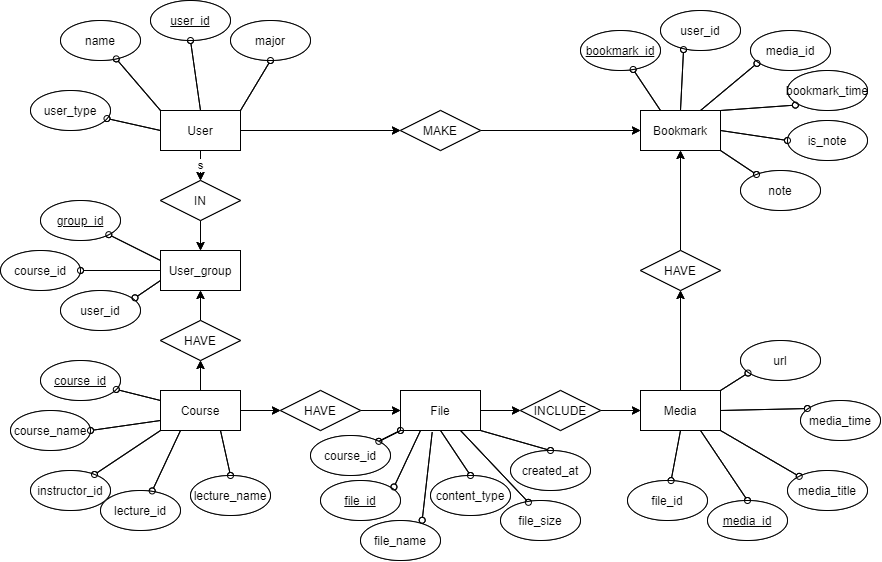
* 1. ER Diagram

The system consists of four entities; User, Course, File, Media, Bookmark. ER-diagrams express each entity as a rectangular and their relationship as a rhombus. When an entity has multiple relationships with other entities, tridents (three line) are used to indicate it. When an entity has just one relationship with another entity, the cross (two line) is used to indicate it. The attribute of an entity is expressed as an ellipse. The unique attribute which uniquely identifies an entity is underlined. If an entity has a number of same attributes, that attribute is expressed as an ellipse with a double border line.



[Figure 17] ER-diagram

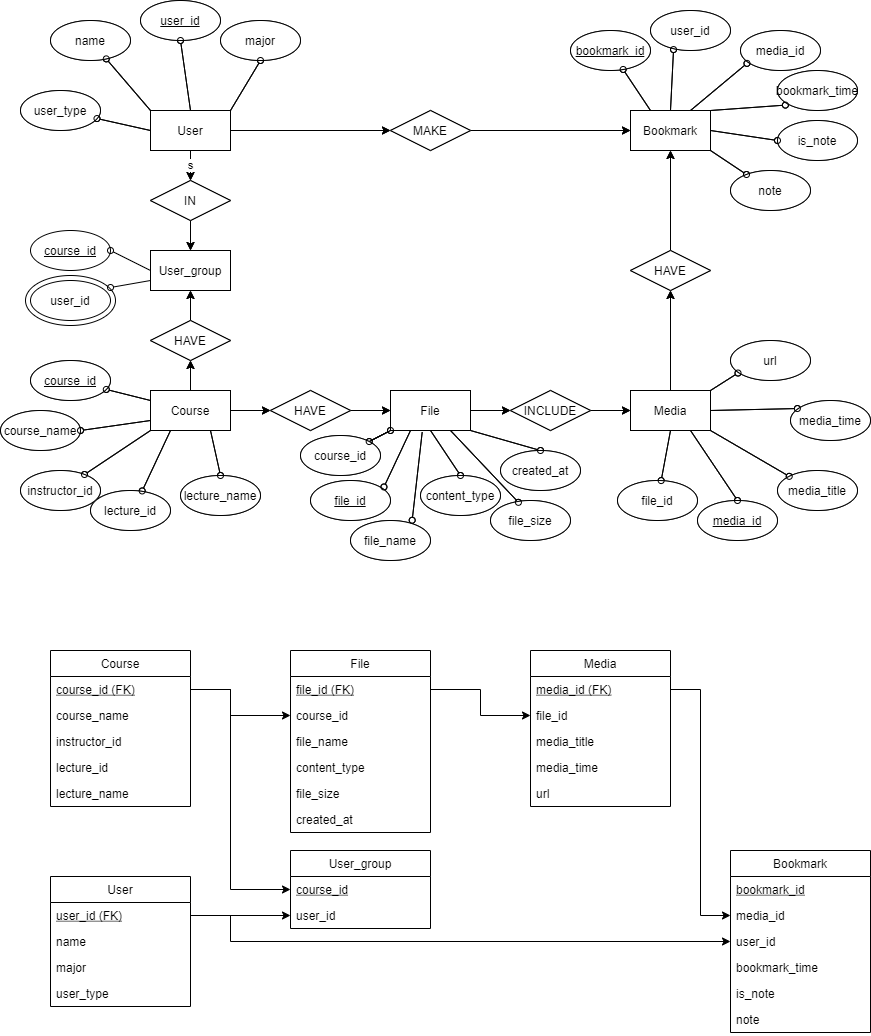
* + 1. Entities
       1. User



[Figure 18] ER diagram, Entity, User

User entity represents the user of the i-Campus lecture sub-system. It consists of the user’s information which include user\_id, name, major, user\_type, and user\_id attribute as the primary key. It can make multiple bookmarks and belong to multiple user\_groups.

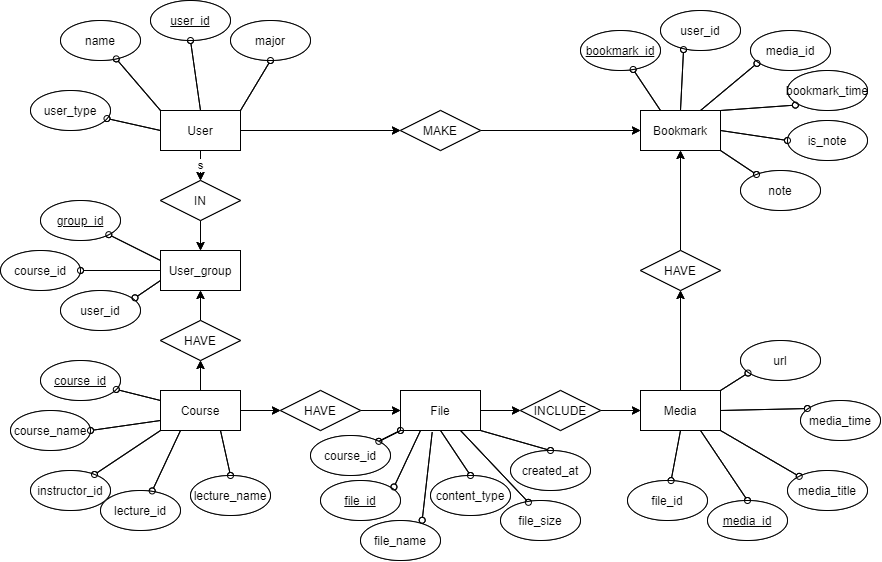
* + - 1. User Group



[Figure 19] ER diagram, Entity, Search History

The user\_group entity represents the group of users who watch the corresponding lecture. It consists of course\_id and user\_id. The course\_id attribute is the primary key and user\_id is a multi-value attribute.

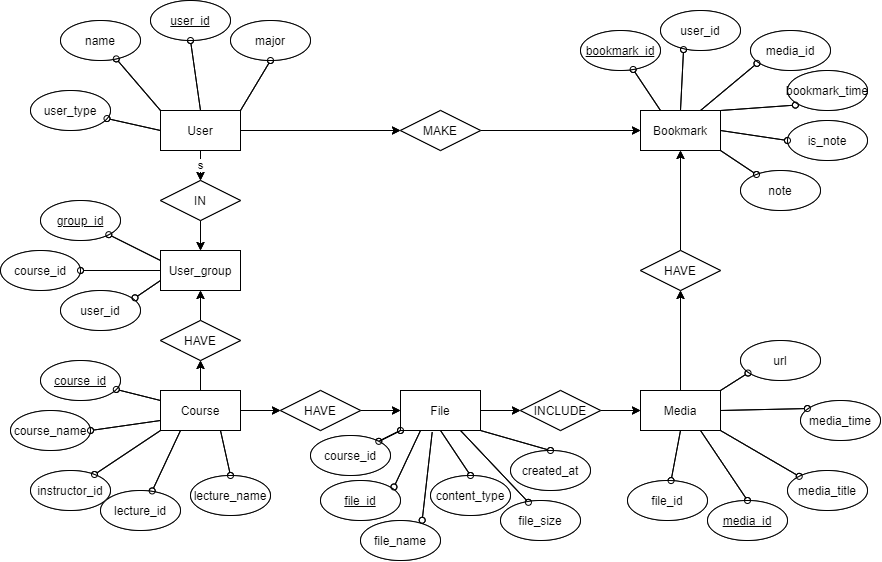
* + - 1. Course



[Figure 20] ER diagram, Entity, Cart

Course entity represents the lecture of the i-Campus lecture sub-system. It consists of the information of the lecture where course\_id, course\_name, instructor\_id, lecture\_id, lecture\_name and course\_id attribute is the primary key. It can have multiple files and one user\_group.

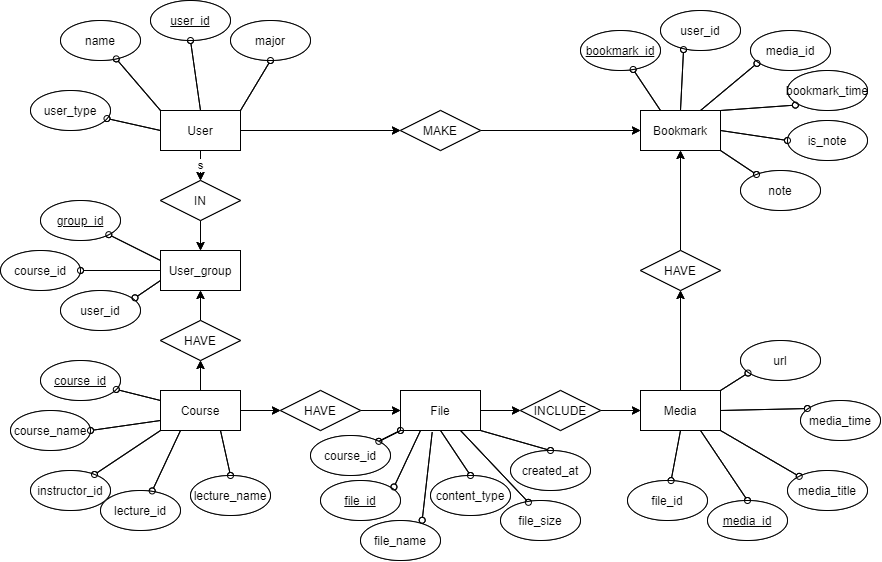
* + - 1. File



[Figure 21] ER diagram, Entity, Laptop

File entity represents the files including the lectures uploaded by the instructor. It consists of file\_id, course\_id, tile\_name, content\_type, file\_size, created\_at, and file\_id which is the primary key. It can belong to one course.

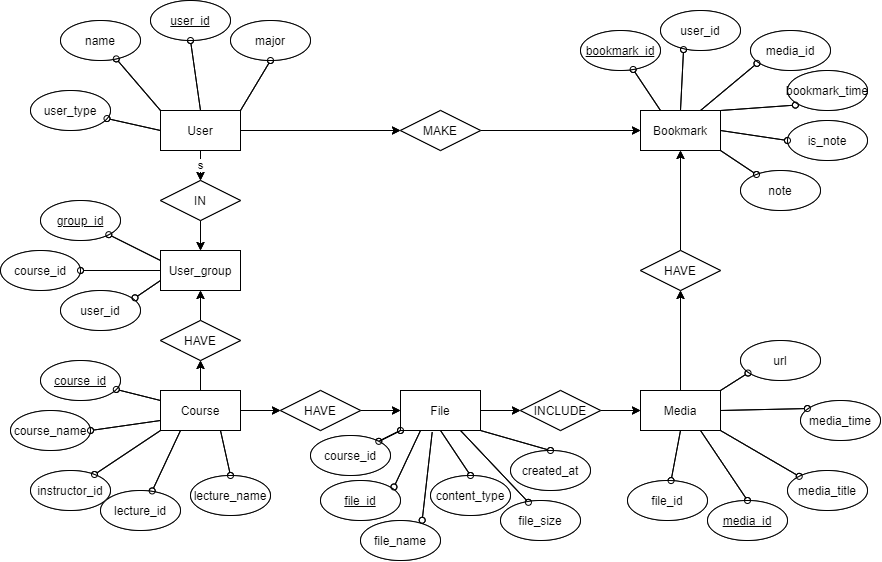
* + - 1. Media



[Figure 22] ER diagram, Entity, Laptop

Media entity represents the lecture video uploaded by the instructor. It consists of file\_id, url, media\_id, media\_title, media\_time, and media\_id attribute, which is primary key. It can have multiple bookmarks.

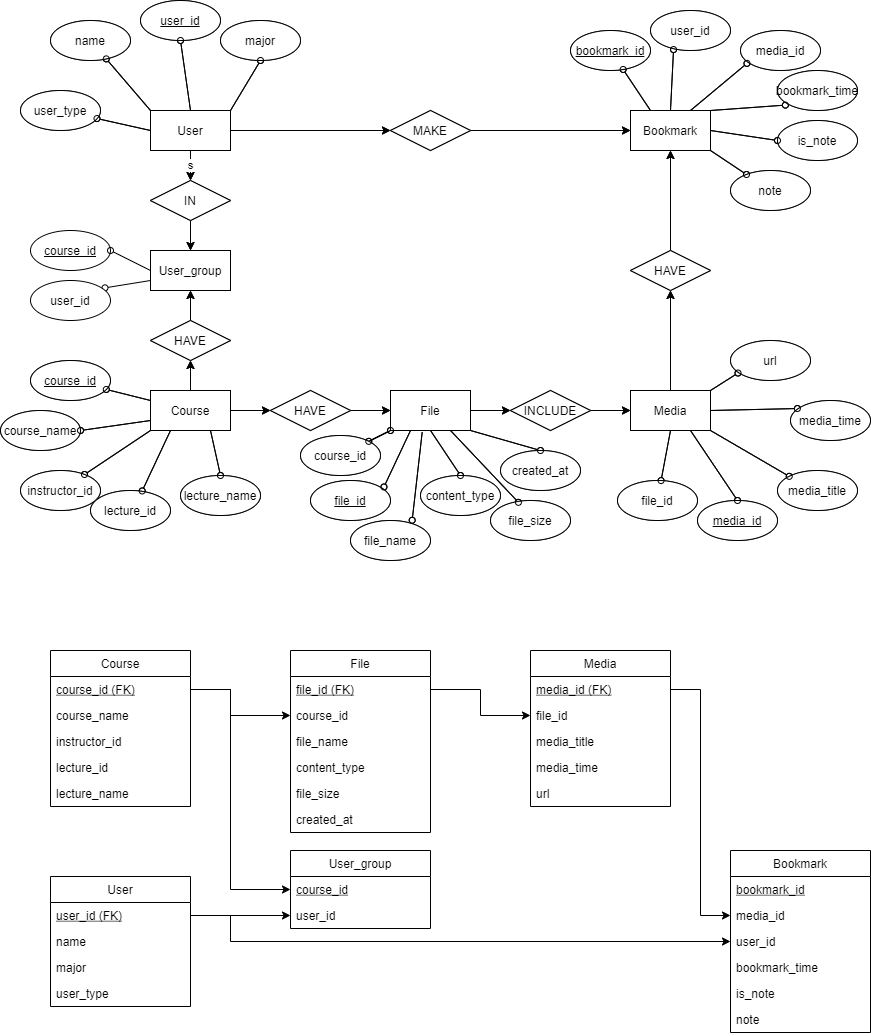
* + - 1. Bookmark



[Figure 23] ER diagram, Entity, Laptop

Bookmark entity represents bookmarks in a lecture video. It consists of bookmark\_id, user\_id, media\_id, bookmark\_time, is\_note, note, and bookmark\_id is the primary key. The user\_id represents the foreign key of the user entity representing the bookmark author, and the media\_id represents the foreign key of the media representing the lecture to which the bookmark belongs. The bookmark\_time refers to the time when the bookmark jumps in the media or indicates the position. is\_note represents whether the bookmark has a note element. Note means the contents of the memo written on the bookmark.

* 1. Relational Schema



[Figure 24] Relational Schema

* 1. SQL DDL
     1. User

CREATE TABLE User

(

user\_id INT NOT NULL,

name CHAR(10) NOT NULL,

major CHAR(30) NOT NULL,

user\_type CHAR(20) NOT NULL,

PRIMARY KEY (user\_id)

FOREIGN KEY (user\_id) REFERENCES User\_group(user\_id),

FOREIGN KEY (user\_id) REFERENCES Bookmark(user\_id),

);

* + 1. User\_group

CREATE TABLE User\_group

(

course\_id INT NOT NULL,

user\_id INT NOT NULL,

PRIMARY KEY (user\_id)

);

* + 1. Course

CREATE TABLE Course

(

course\_id INT NOT NULL,

course\_name CHAR(20) NOT NULL,

instructor\_id INT NOT NULL,

lecture\_id INT NOT NULL,

lecture\_name CHAR(20) NOT NULL,

PRIMARY KEY (course\_id)

FOREIGN KEY (course\_id) REFERENCES File(course\_id),

FOREIGN KEY (course\_id) REFERENCES User\_group(course\_id)

);

* + 1. File

CREATE TABLE File

(

file\_id INT NOT NULL,

course\_id INT NOT NULL,

file\_name CHAR(20) NOT NULL,

content\_type CHAR(20) NOT NULL,

file\_size INT NOT NULL,

created\_at INT NOT NULL,

PRIMARY KEY (file\_id).

FOREIGN KEY (file\_id) REFERENCES Media(file\_id)

);

* + 1. Media

CREATE TABLE Media

(

media\_id INT NOT NULL,

file\_id INT NOT NULL,

media\_title CHAR(20) NOT NULL,

media\_time INT NOT NULL,

url CHAR(20) NOT NULL,

PRIMARY KEY (media\_id),

FOREIGN KEY (media\_id) REFERENCES Bookmark(media\_id)

);

* + 1. Bookmark

CREATE TABLE Bookmark

(

bookmark\_id INT NOT NULL,

media\_id INT NOT NULL,

user\_id INT NOT NULL,

bookmar\_time INT NOT NULL,

is\_note BOOLEAN NOT NULL,

note CHAR(20) NOT NULL,

PRIMARY KEY (link\_to\_vendor, laptop\_model),

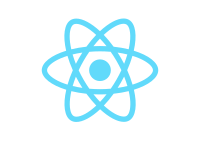
FOREIGN KEY (laptop\_model) REFERENCES Laptop(laptop\_model)

);

1. Development Plan
   1. Objectives

This chapter illustrates the technologies and environment for the development of the application.

* 1. Frontend Environment
     1. React



[Figure 34] React logo

It is a raster graphics editor developed and published by Adobe Inc. for Windows and macOS. This program would provide aesthetical layout and icons for improved user experience during our project.

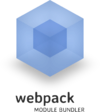
* + 1. jQuery



[Figure 36] jQuery logo

jQuery is an open source-based JavaScript library designed to simplify client-side manipulation of HTML. It has been a long time since it was developed, and especially because of the framework that employed virtual DOMs such as React and Vue.js, it is not as popular as before, but is more useful when producing static and light web pages.

* + 1. Webpack



[Figure 36] Webpack logo

Webpack is an open-source JavaScript module bundler. It converts front-end assets such as JS, HTML, CSS, and images. It is used for fast loading speed and high performance of web applications.

* 1. Backend Environment
     1. RubyMine

텍스트이(가) 표시된 사진

자동 생성된 설명

[Figure 37] RubyMine logo

RubyMine is a full-featured IDE that provides essential tools for Ruby and Ruby on Rails development out of the box. It offers smart code completion and analysis, easy code navigation, safe automated refactorings, an interactive debugger, JavaScript & HTML/CSS, automatic deployment, Git workflow support, and many other tools all integrated together in a highly customizable, productive, user-friendly environment.

* + 1. Redis

텍스트, 클립아트이(가) 표시된 사진

자동 생성된 설명

[Figure 38] Redis logo

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker. Redis provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes, and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions, and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

* + 1. Github



[Figure 39] Github logo

GitHub is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project.

* 1. Constraints

The system is designed and implemented based on this document. Basically, the following matters shall be followed and other matters shall be governed by the developer in a way that does not deviate from the ethics of software engineering:

* When bookmarks are created or deleted, they shall be applied within 3 seconds.
* Reduce page load speed difference from existing to less than 1 second after bookmark function is added.
* It provides enough space to store bookmark data for all learners for at least one year.
* Made by considering system development costs, maintenance costs, and scalability.
* Optimize source code to reduce system efficiency and operating time.
* Specify the modifications accurately so that they do not violate the Canvas LMS license.
* Make all versions supported by Canvas LMS compatible.
* All developments are open source based and should be made available on all devices that are not subject to operating system restrictions in Canvas LMS.
  1. Assumptions and Dependencies

All systems in this document are designed and implemented as internal features of the Canvas LMS system. Therefore, the system's constraints are the same as those of Canvas LMS and its mobile application Learning-X. For PC versions, it is assumed to be available on search engines such as Safari (MacOS 11.0 or later), Google Chrome, and Firefox. Other search engines may have login or video viewing problems. For mobile devices, the system should work on the Learning X application operating basics of iPhone iOS 13.0 or later, iPad OS 13.0 or later, and Android version 5.0 or later. Other devices may have application usage problems.

1. Supporting Information
   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).

* 1. Document History

[Table 25] Document History

| **Date** | **Version** | **Description** | **Writer** |
| --- | --- | --- | --- |
| 2021/05/10 | 0.1 | Style and overview | Jisun Lee |
| 2021/05/12 | 1.0 | Addition of 1 | Krogross Ryan |
| 2021/05/13 | 1.1 | Addtion of 2, 3 | Dongjun Lee |
| 2021/05/14 | 1.2 | Addition of 7 | Jisun Lee |
| 2021/05/14 | 1.3 | Addition of 6 | Seongwoo Kim |
| 2021/05/14 | 1.4 | Addition of 4 | Deokjae Kang |
| 2021/05/14 | 1.5 | Addition of 5 | Chihyun Lee |
| 2021/05/15 | 1.6 | Addition of 8.2 | Deokjae Kang |
| 2021/05/15 | 1.7 | Addition of 8.3 | Seongwoo Kim |
| 2021/05/16 | 1.8 | Addition of 8.4, 8.5 | Dongjun Lee |
| 2021/05/16 | 1.9 | Fix grammar error | Krogross Ryan |
|  |  |  |  |
|  |  |  |  |