iCalendar for iCampus users

Software Design Specification

2021.05.12.

Introduction to Software Engineering 41

TEAM 5 (iCalendar)

Team Leader	Suyoung Min
Team Member	Minseo Kim
Team Member	Chanjong Lee
Team Member	Taewoo Yoo
Team Member	Sumin Ham

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 iCalendar.

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 1 is the main reader. Additionally, professors, TAs, and team members in the Introduction to Software Engineering class can be the main readers, and defined to stakeholder.

1.2. Scope

This Design Specification is to be used by Software Engineering and Software Quality Engineering as a definition of the design to be used to implement iCalendar for iCampus users.

1.3. Objective

The main purpose of this Software Design Document is to provide a description of the technical design aspects for customizing and sharing calendar, iCalendar. This Software Design Document describes the software architecture and software design decisions for the implementation of iCalendar. It also provides an overview of the system to describe various aspects of the system. It further specifies the structure and design of some of the modules discussed in the SRS document and in addition, displays some of the use cases that have been transformed into sequential and activity diagrams, including the class diagrams which show how the programming team would implement the specific module. The intended audience of this document is, but not limited to, the stakeholders, developers, designers, and software testers of the iCalendar functionality.

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 sequence diagram and class diagram.
- **6. Protocol Design**: this chapter describes design of several protocols which used for communication of student client and icampus server.
- **7. Database Design**: this chapter describes database design using several ER diagrams and SQL DDL.
- **8. Testing Plan**: this chapter describes testing plan for iCalendar 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 criteria of this document and history of this document.

2. Introduction

The iCalendar project of team 5 is to develop and design a additional functionality used for customizing and sharing calendar with team project members, or friends who using icampus of Sungkyunkwan University. Many typical calenders tend to show all schedules without separating them. This trend is efficient when coordinating as a whole, but not appropriate when managing tasks on a task-by-task basis. So, iCalender give users a separated custom calenders that can use task-by-task. This design document presents the designs used or intended to be used in implementing the project. The designs described, follow the requirements specified in the Software Requirements Specifications document prepared earlier for the project.

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 is an acronym that stands for Unified Modeling Language, and is a standardized general-purpose modeling language published by Object Modeling Technology (OMG). UML is a modern approach to modeling and documenting software, and it is used to integrates multiple types of diagrams that have been used for object-oriented software design, designed and implemented to facilitate communication with both developers and system-related stakeholders by visualizing the system's structure during system development. Actually, it’s very popular business process modeling techniques. UML is a language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems. 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. It is based on diagrammatic representations of software components.

2.2.2. Use case Diagram

A diagram of a system's interaction with a user shows the service or function of the system from the user's point of view and the external elements associated with it. It indicates which features a user can use within the system, and the use of a use case diagram allows customers and developers to coordinate their opinions on the requirements. In short, it can be seen as a representation of the relationship between the user and the system. Use case diagrams are created when defining requirements for a project, analyzing detailed functions, and setting the scope of development.

2.2.3. Sequence Diagram

Sequence Diagram is a UML diagram that expresses the interactions between systems, objects, and classes over time. It is primarily a process of modeling the dynamic aspects of the system, which describes data exchanges, message sequences, etc. between each object to realize the use case.

2.2.4. Class Diagram

A class diagram 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 such as inheritance between classes can also be expressed, also it is a basic Diagram used in object-oriented design that expresses the static structure of a system. Because classes are the building block of objects, class diagrams are the building blocks of UML. The various components in a class diagram can represent the classes that will actually 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.

2.2.5. Context Diagram

The system context diagram 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). While it is also good to draw diagrams that represent the relationships between actors, it is also meaningful to draw diagrams to a wider extent. When you define the system you want to develop as an object and model it to give meaning to the relationship between the system and the actor, you give it the most basic meaning of the system, which is called a context diagram. A context diagram is typically included in a requirements document. It must be read by all project stakeholders and should be written in plain language, so the stakeholders can understand items. A system context diagram is often used early in a project to determine the scope under investigation. Therefore, A system context diagram represents all external entities that may interact with a system. The main software system is shown as a single process. Such a diagram pictures the system at the center, surrounded by all its External entities, interacting systems, and environments.

2.2.6. Entity Relationship Diagram

An ER (entity relationship) diagram is a conceptual model that represents a database as a relationship between entities and entities, which, unlike previous diagrams, is not included in

UML. At this point, each entity is a real-world object that targets tangible and intangible information and can be distinguished by one or more attributes. Relation refers to the correlation between each entity and is represented by a solid line.

2.3. Applied Tools

2.3 .1 Microsoft Paint

This is a tool that supports drawing text and figures. The program is simple in shape and convenient in manipulating various photo files. It is often used when parts of several pictures are edited into a single page or when tasks such as adding simple lines are needed.

2.4. Project Scope

The iCalender system is functionality to ease the management of schedule, like lectures, assignments, team project, or other personal jobs, and so on. It is also expected to share official announcements of classes with team members and sync project schedules and processes. This additional functionality can be added on current i-campus and using alarm function in LearningX. This system can provide new way of controlling schedules by using current icampus calendar functionality. Above all, we hope to provide a comfortable user experience along with the best possible pricing available to the users.

2.5. References

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

- Team 5, 2020 Spring. Software Design Document, SKKU.
- Appleton, Brad. A Software Design Specification Template. N.d.

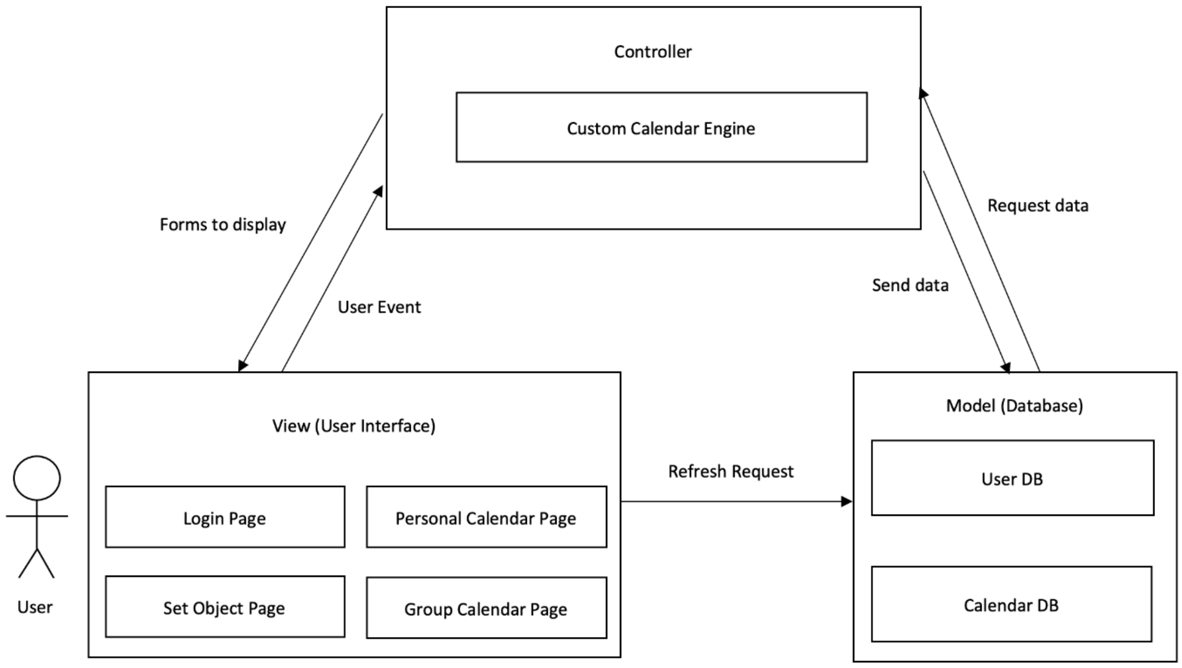
3. System Architecture – Overall

3.1. Objectives

Here in this chapter, we describe and show the organization of the system, ranging from the frontend design to the backend design of the application for the project.

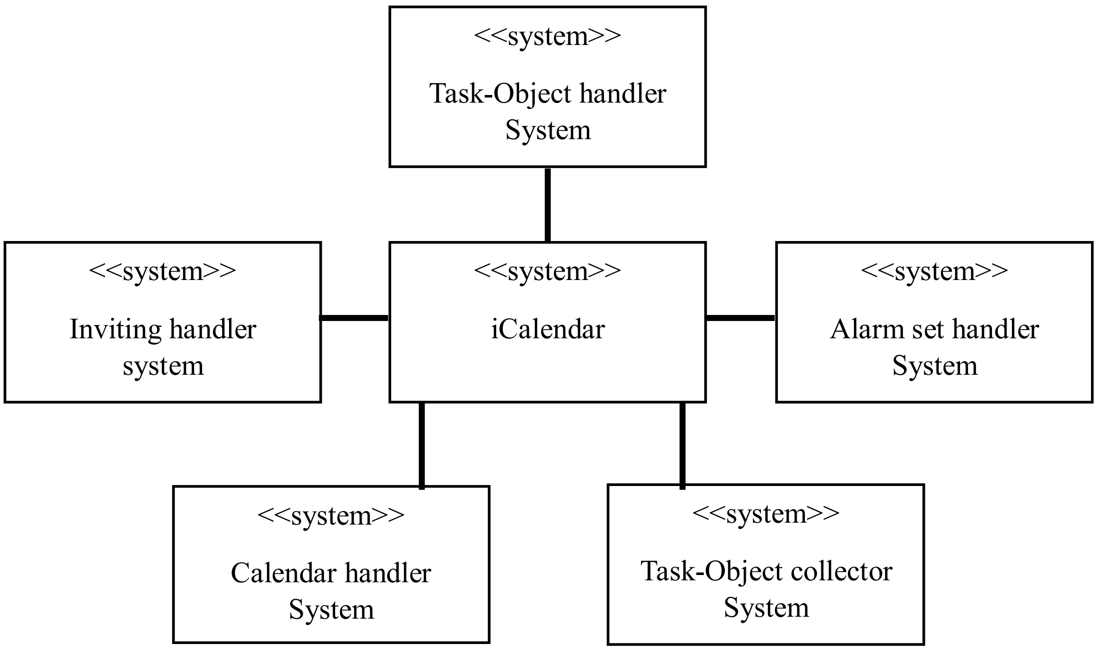
3.2. System Organization

This service is designed by applying the client - server model, front-end internet browser is responsible for all interactions with users, and the front-end functionality and back-end server send and receive data through HTTP communication. The back-end server stores data about the custom calendar, modifies it on demand from the front end, or forwards it to the front-end client. In addition, tasks such as invitations to calendar groups among users can also be borrowed and implemented by icampus. When an alarm setup request is received from a custom calendar, the back-end server allows the alarm to be set through the icampus function, which requests it to the LearningX server.



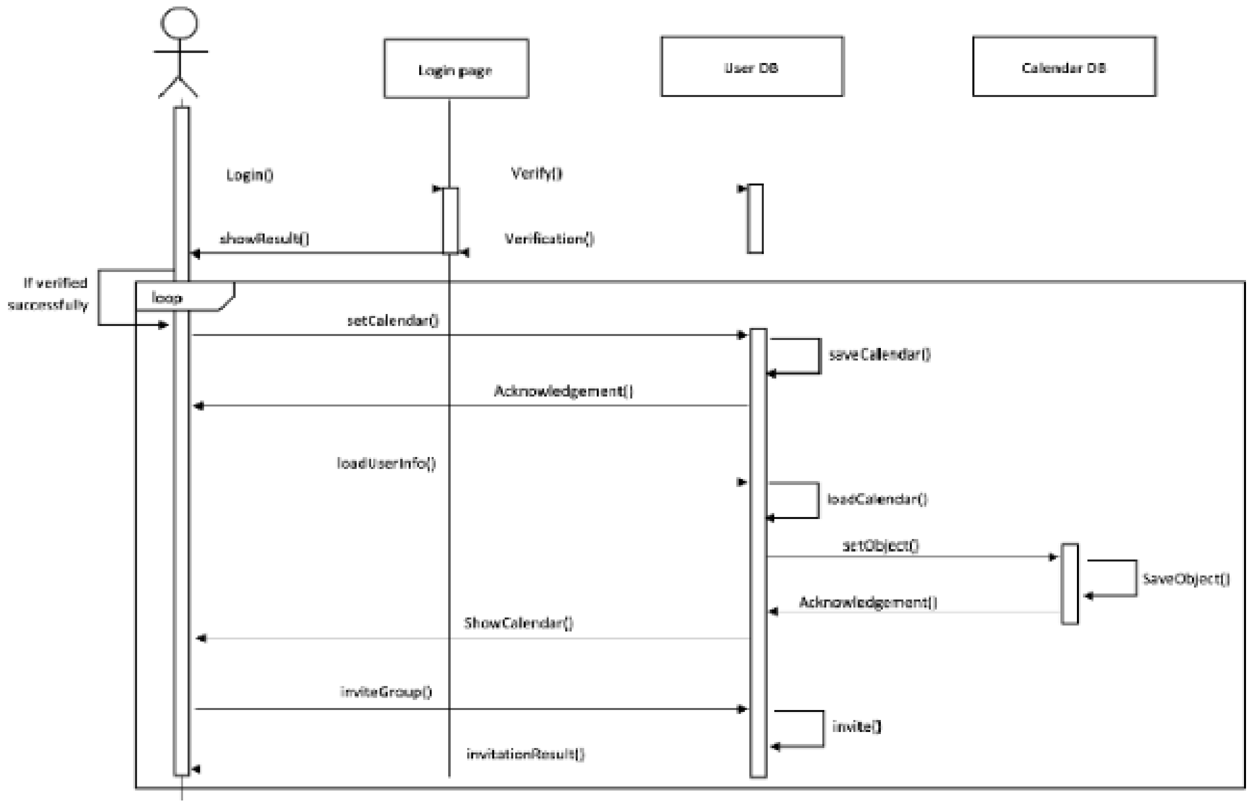
[Figure 1] Overall system architecture

3.2.1. Context Diagram



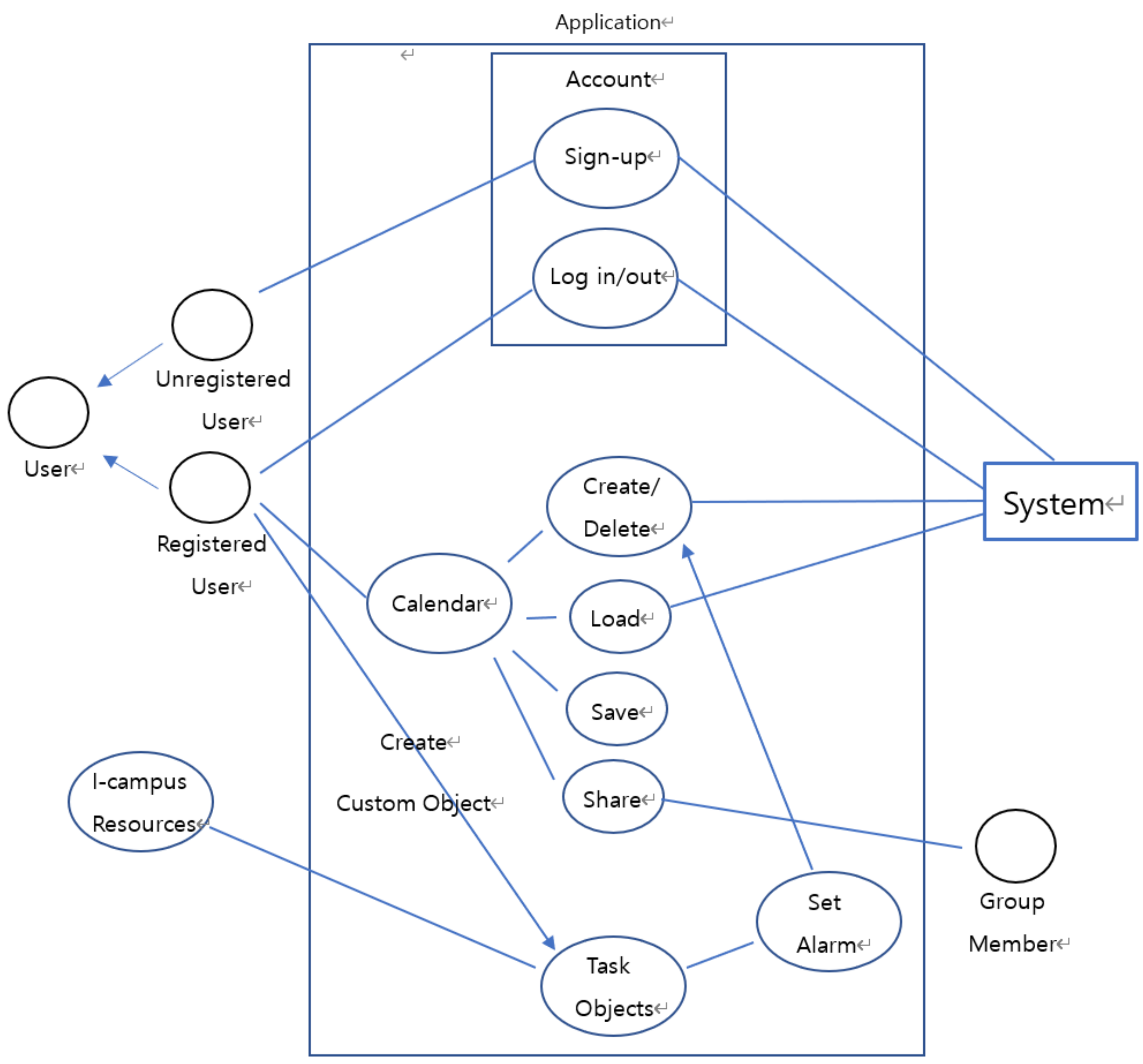
[Figure 2] Overall context diagram

3.2.2. Sequence Diagram



[Figure 3] Overall sequence diagram

3.2.3. Use Case Diagram



[Figure 4] Use case diagram

4. System Architecture – Frontend

4.1. Objectives

This chapter describes structure, attributes and function of the frontend system and describe the relation of each component.

4.2. Subcomponents

4.2.1. Profile

The profile classes handle basic user information. When user registration, users must link their profile with their I-campus account. After logging in the user, user can edit the user profile.

4.2.1.1. Attributes

These are the attributes that profile object has.

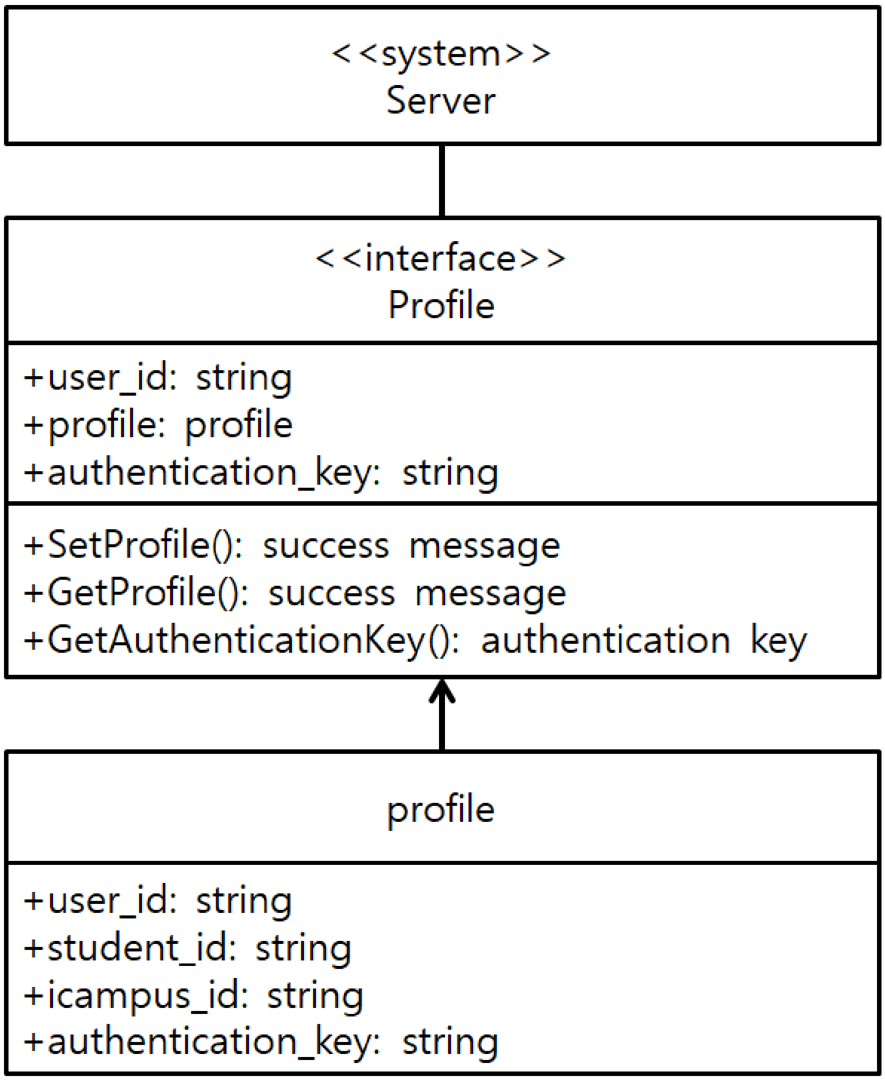
- **User id**: id of the user (email address)
- **Student id**: student id of the user
- **I-Campus id**: : I-Campus id of the user
- **Authentication key**: authenticate key the linkage with I-Campus system

4.2.1.2. Methods

These are the methods that profile class has.

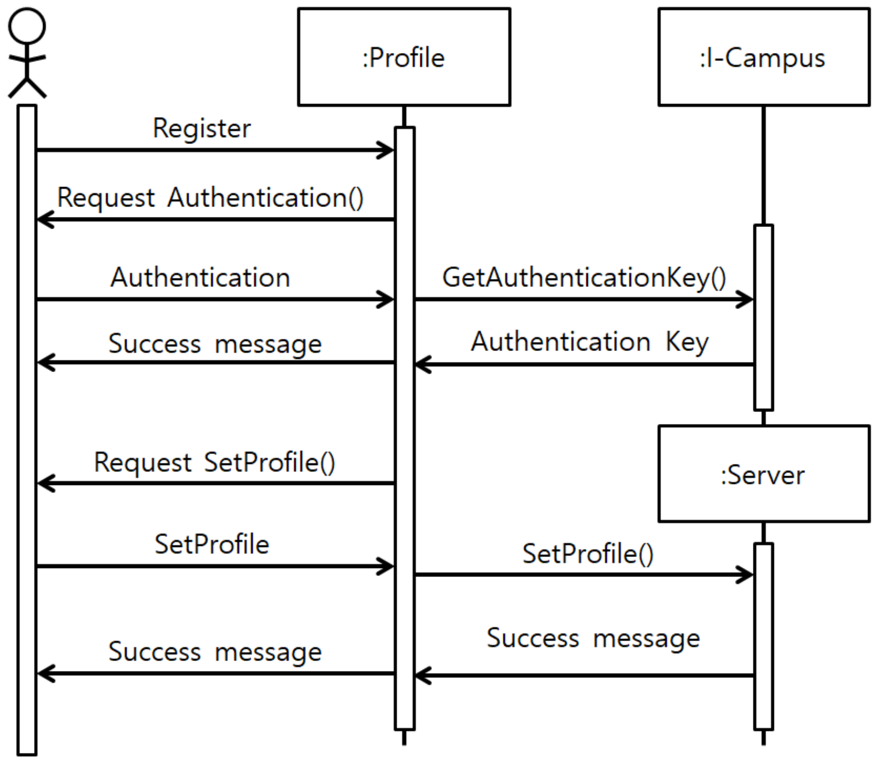
- SetProfile()
- GetProfile()
- GetAuthenticationKey()

4.2.1.3. Class Diagram


<<system>>
Server
<<interface>>
Profile
+user_id: string
+profile: profile
+authentication_key: string
+SetProfile(): success message
+GetProfile(): success message
+GetAuthenticationKey(): authentication key
profile
+user_id: string
+student_id: string
+icampus_id: string
+authentication_key: string

[Figure 5] Class diagram – Profile

4.2.1.4. Sequence Diagram



[Figure 6] Sequence diagram – Profile

4.2.2. Search

The custom calendar classes helps users create a calendar based on your I-Campus. After creating a custom calendar, users can add custom tasks.

4.2.2.1. Attribute

These are the attributes that calendar object has.

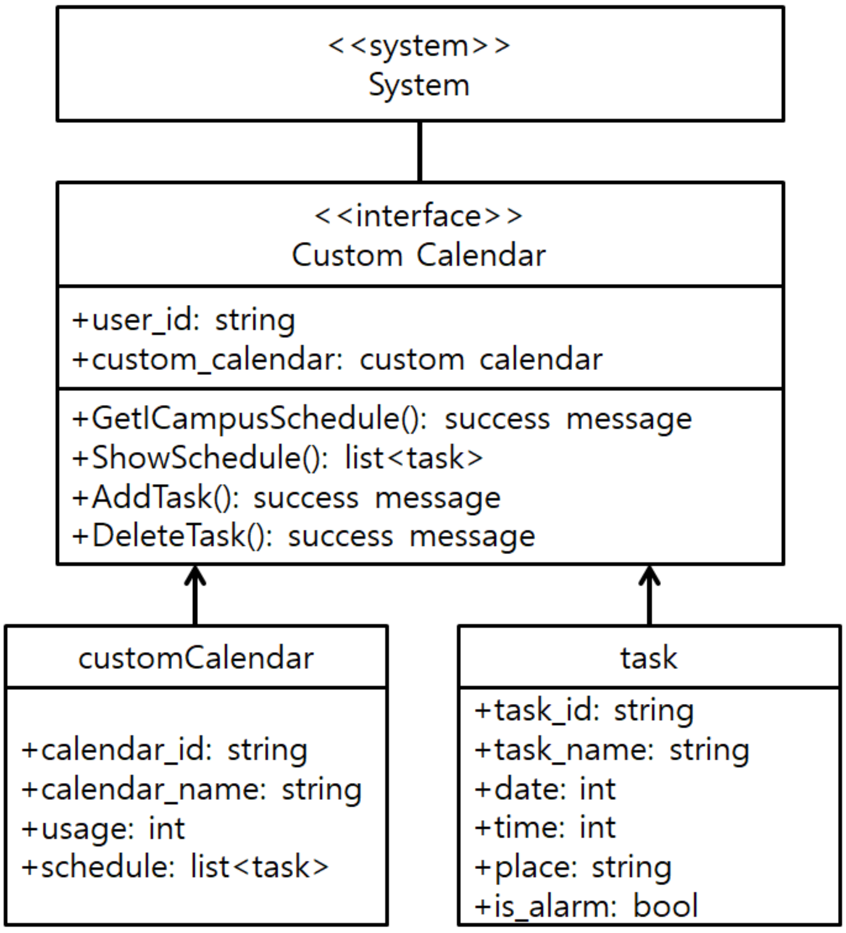
- **Calendar id**: unique id of the calendar
- **Calendar name**: name of the user-specified calendar
- **Usage**: usage of the calendar.
- **Schedule**: user’s schedule in calendar

4.2.2.2. Methods

These are the methods that custom calendar class has.

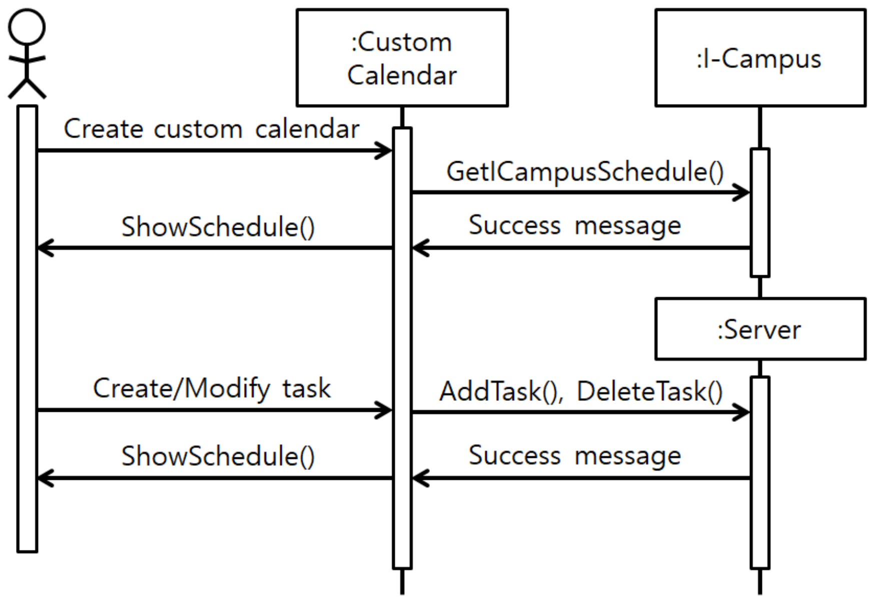
- GetICampusSchedule()
- ShowSchedule()
- AddTask()
- DeleteTask()

4.2.2.3. Class Diagram



[Figure 7] Class diagram – Custom Calendar

4.2.2.4. Sequence Diagram



[Figure 8] Sequence diagram – Custom Calendar

4.2.3. Task

The task classes deals with the results of the calendar. This class fetches the user's calendar from the server and presents it to the user. When a user creates a new task, this new task cache must be added to the server. Also, if the user wants to delete some tasks, the task object is removed.

4.2.3.1. Attributes

These are the attributes that task object has.

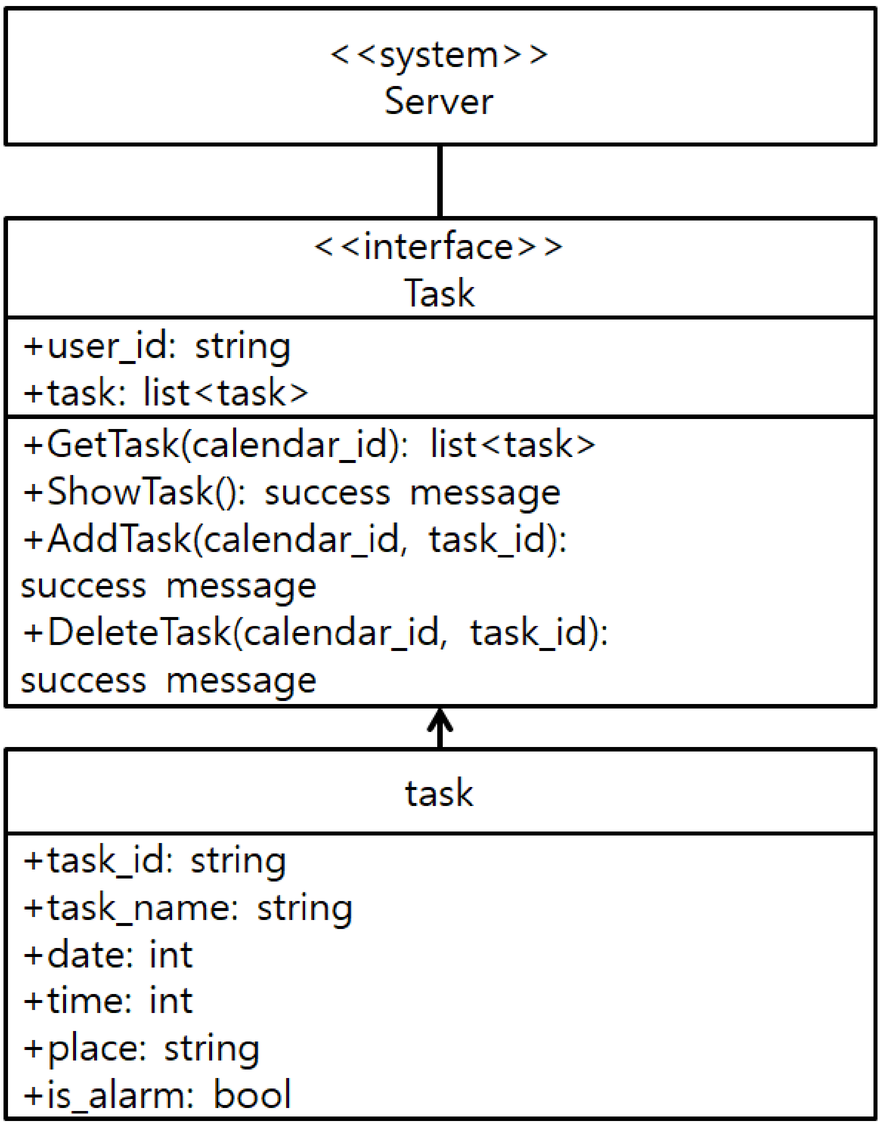
- **Task id**: unique id of the task.
- **Task name**: name of the task that user-specified
- **Date**: date of the task (Julian calendar)
- **Time**: time of the task (Pacific Standard Time)
- **place**: place of the task
- **Is alarm**: whether the task is alarmed or not

4.2.3.2. Methods

These are the methods that task class has.

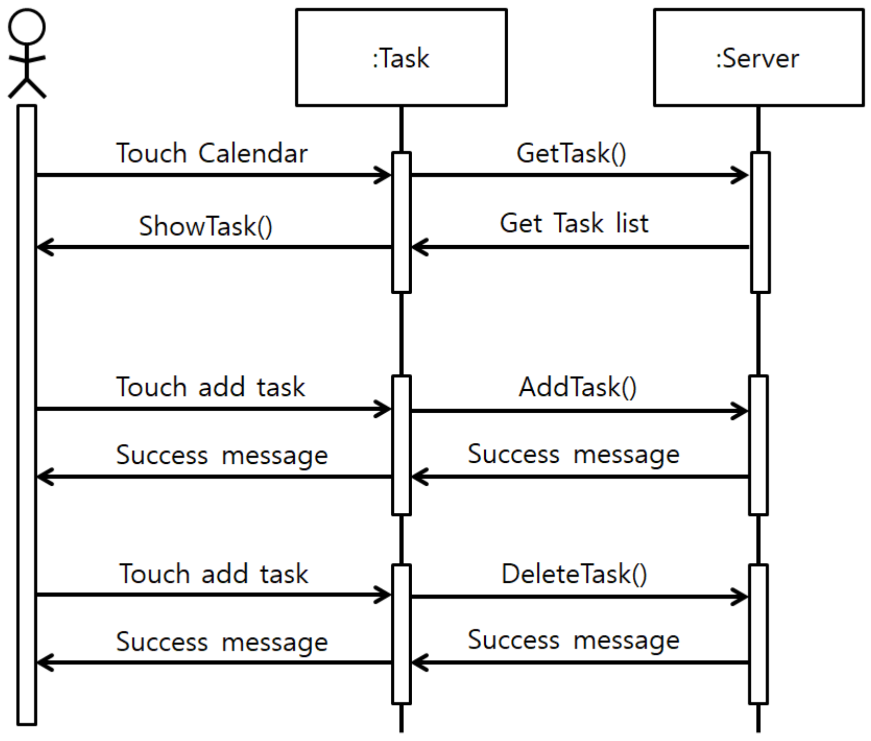
- GetTask()
- ShowTask()
- AddTask()
- DeleteTask()

4.2.3.3. Class Diagram


<<system>>
Server
<<interface>>
Task
+user_id: string
+task: list<task>
+GetTask(calendar_id): list<task>
+ShowTask(): success message
+AddTask(calendar_id, task_id): success message
+DeleteTask(calendar_id, task_id): success message
task
+task_id: string
+task_name: string
+date: int
+time: int
+place: string
+is_alarm: bool

[Figure 9] Class diagram – Task

4.2.3.4. Sequence Diagram



[Figure 10] Sequence diagram – Task

4.2.4 Group

The group class deals with sharing the calendar with other users. Users can share their calendars with other users. Users can also share the calendar by accepting invitations from other users.

4.2.4.1 Attributes

These are the attributes that group object has.

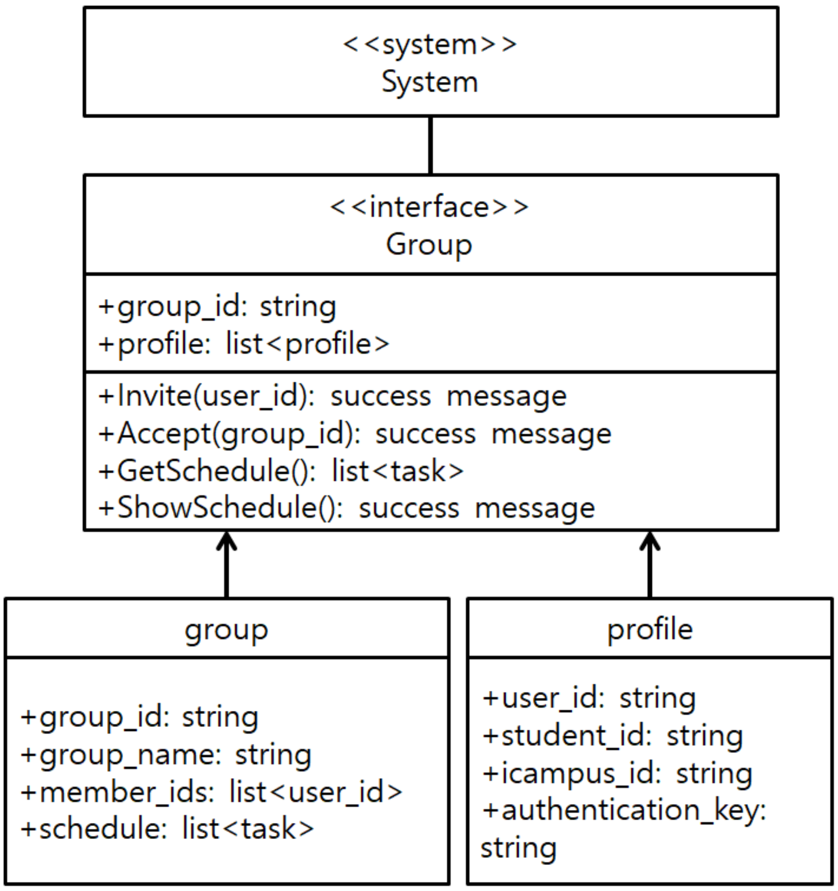
- **Group id**: id of the group
- **Group name**: name of the group
- **Member ids**: users id that joined the group
- **Schedule**: schedule of users who joined the group

4.2.4.2 Methods

These are the methods that group class has

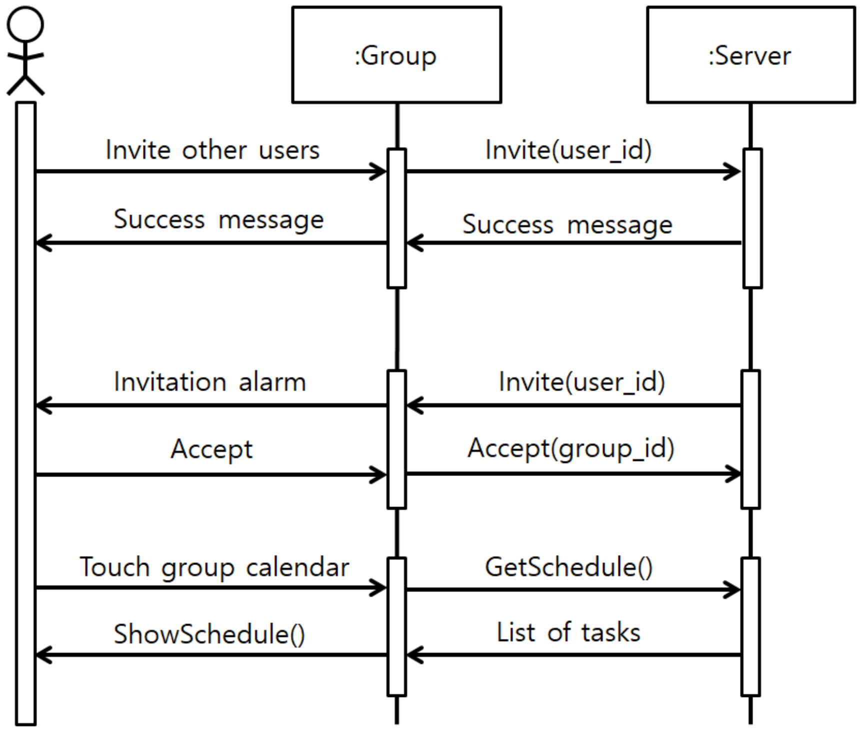
- Invite()
- Accept()
- GetSchedule()
- ShowSchedule()

4.2.4.3 Class Diagram



[Figure 11] Class diagram – Group

4.2.4.4 Sequence Diagram



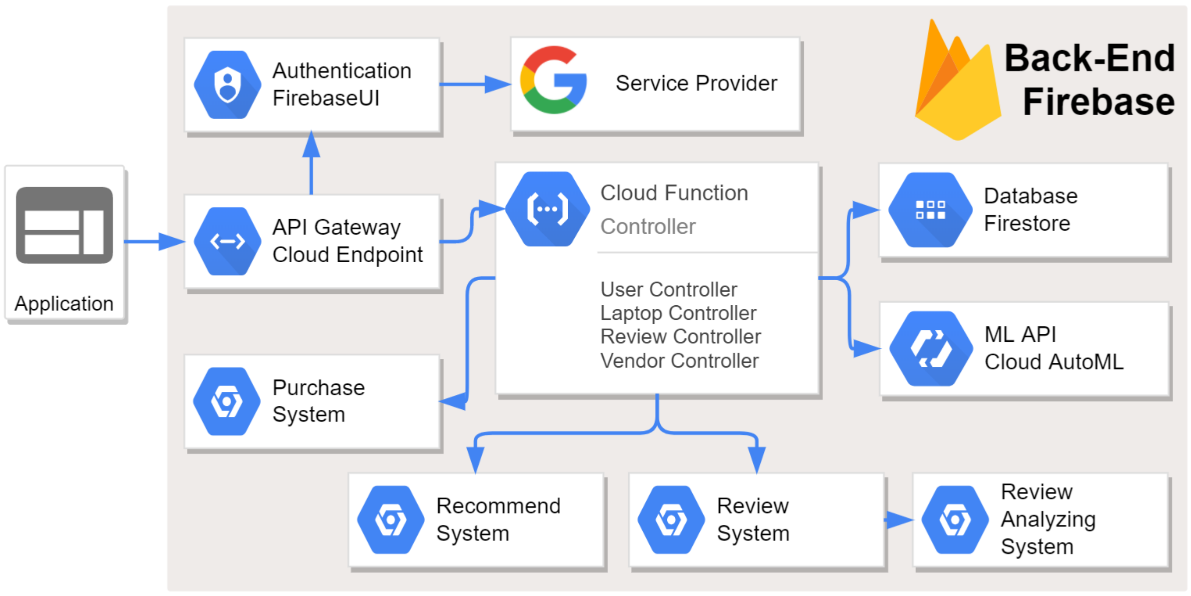
[Figure 12] Sequence diagram – Group

5 System Architecture – Backend

5.1 Objectives

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

5.2 Overall Architecture

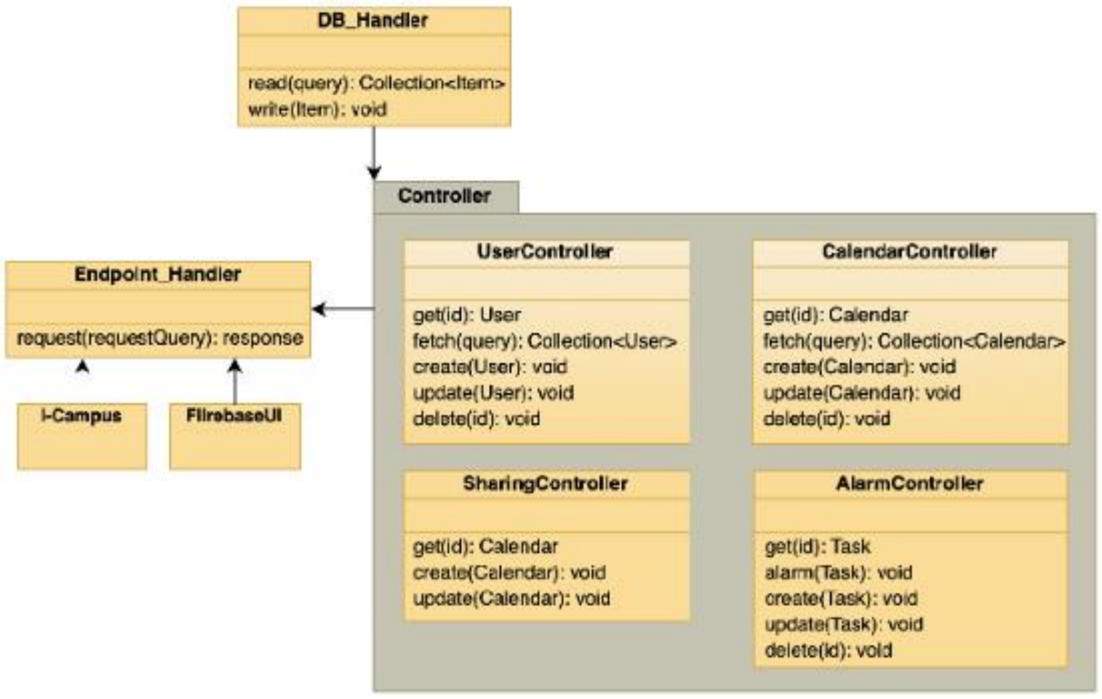


[Figure 13] Overall architecture

The overall architecture of the system is as above. The API gateway (Request Handler) receives the request from the front-end and distributes it to the appropriate cloud function (Controller / Manager). In this process, function that use external API such as authentication are handled. Other requests (processed internally) sent to the corresponding controller. The controller interacts with the cloud (Database / Notification) and processes request.

5.3 Subcomponents

5.3.1 Cloud Function



[Figure 14] Class diagram – Cloud functions

5.3.1.1 Endpoint Handler Class

API gateway. Distribute requests from the frontend to the appropriate controller or API.

5.3.1.2 FirebaseUI

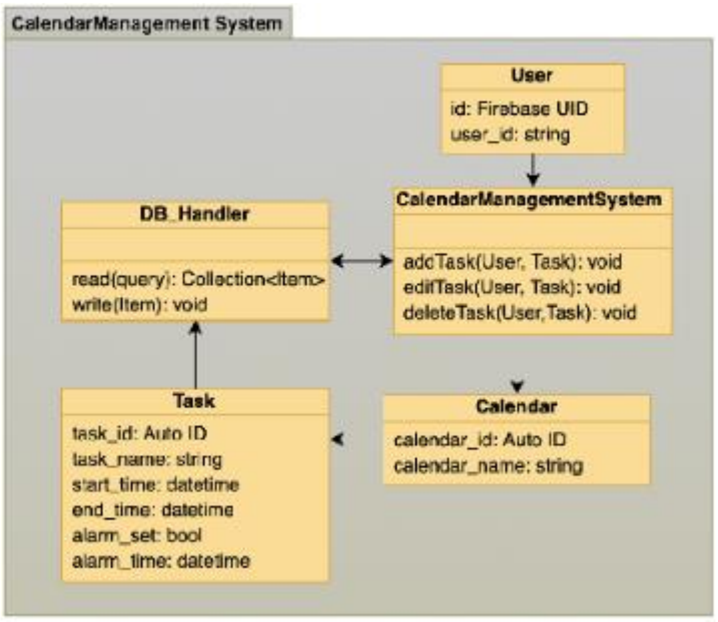
Class to implement authentication through FirebaseUI

5.3.1.3 DB_Handler Class

Interface to communicate with DB. It imports existing calendar information from i-Campus, and it creates, modifies, or imports calendar and task objects into the DB.

5.3.2 Calendar Management

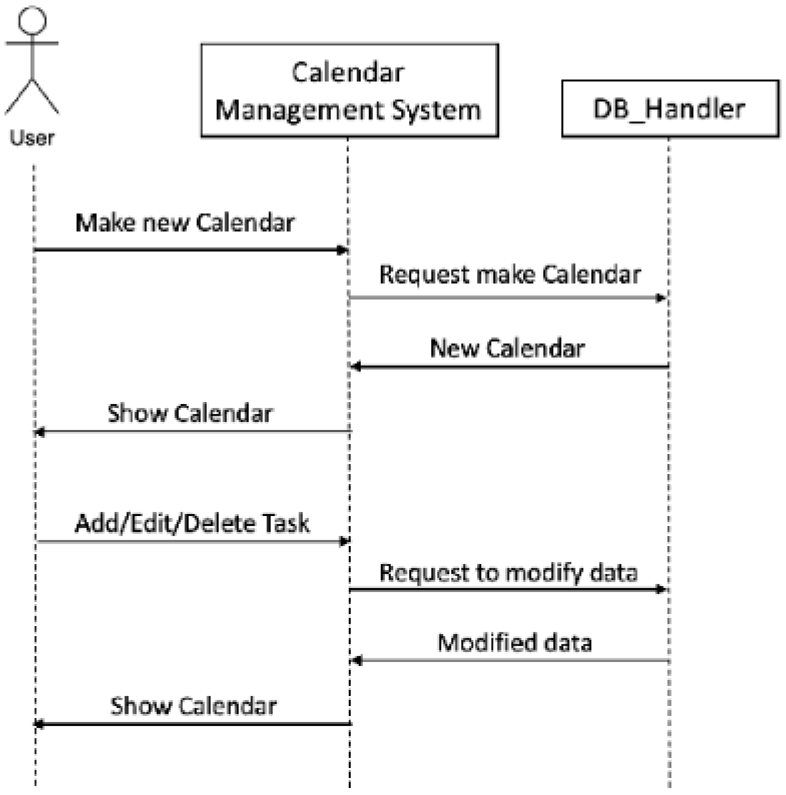
5.3.2.1 Class Diagram



[Figure 15] Class diagram – Review system

- Class Description
  - ✓ **Calendar Management Class**: This is an interface for calendar management. Called when modifications occur, such as adding a new calendar or adding or changing a schedule.

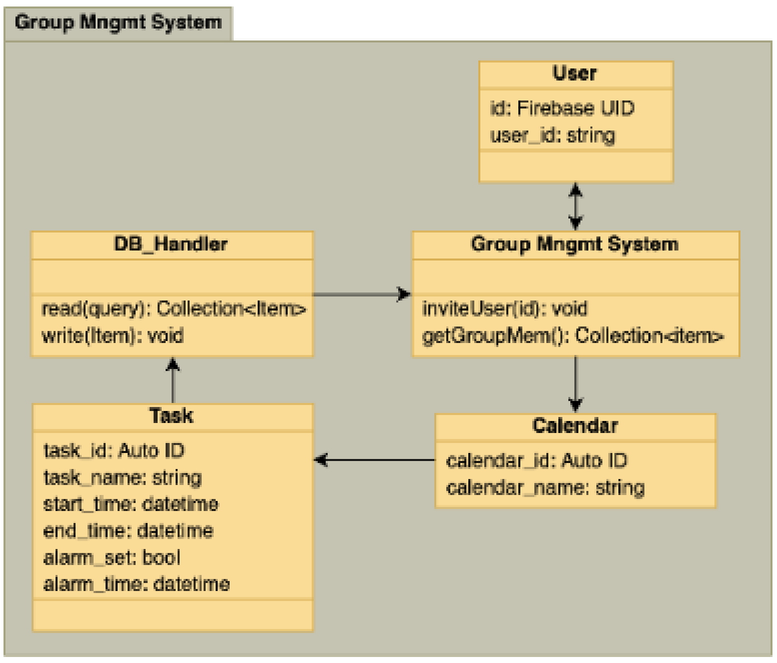
5.3.2.2 Sequence Diagram



[Figure 16] Sequence diagram – Calendar Management system

5.3.3 Group Management System

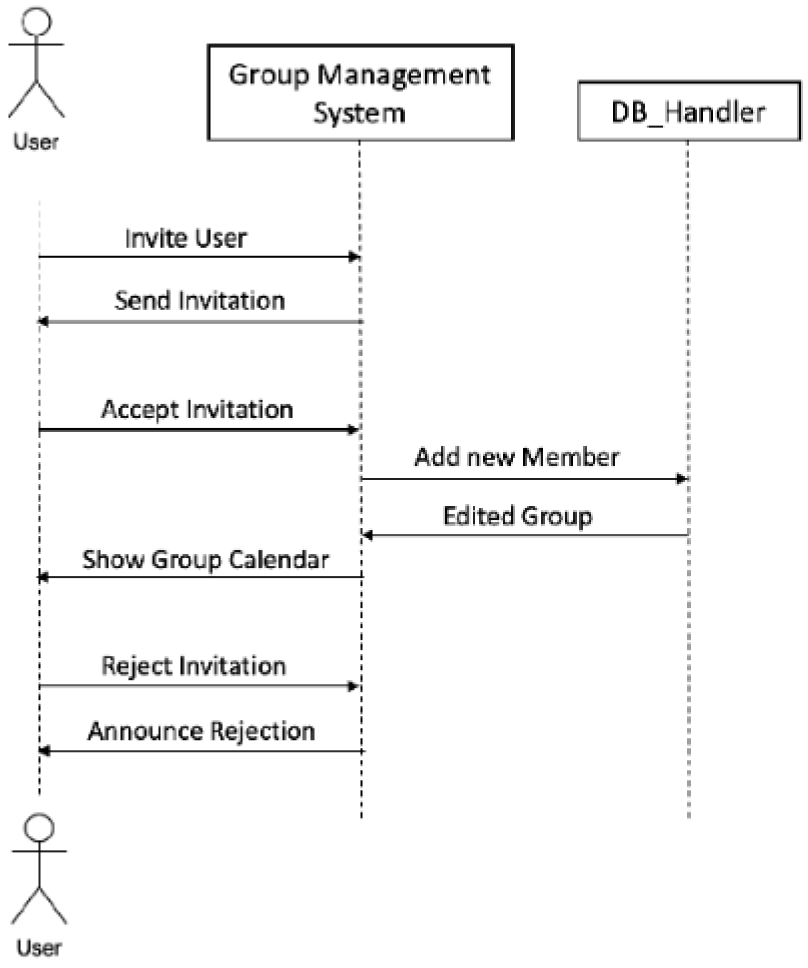
5.3.3.1 Class Diagram



[Figure 17] Class diagram – Group Management system

- Class Description
  - ✓ **Group Management Class**: It is an interface for Group calendar management. Called when a modification to a new calendar is added or when a new group member is invited.

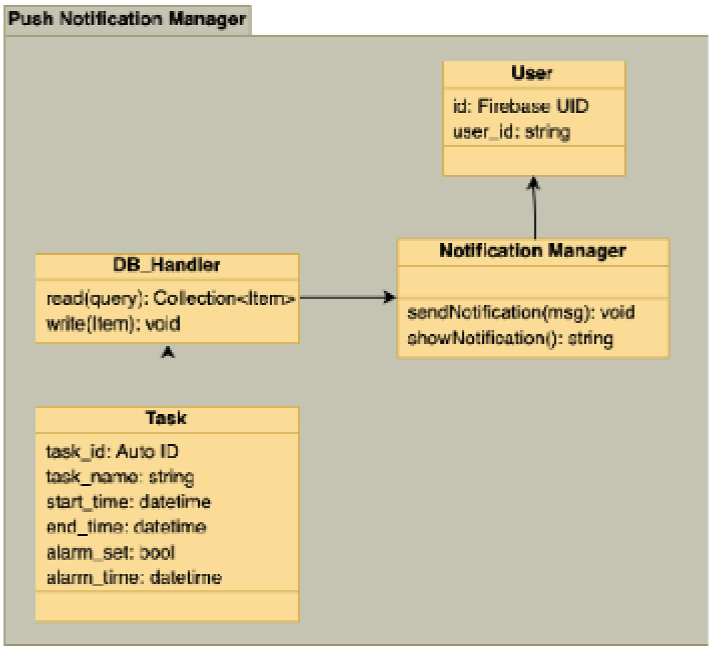
5.3.3.2 Sequence Diagram



[Figure 18] Sequence diagram – Group Management system

5.3.4 Push Notification Manager

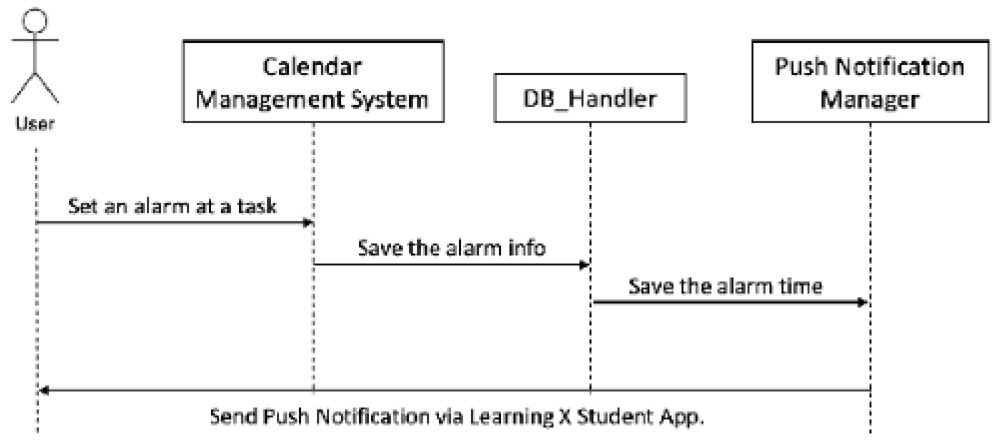
5.3.4.1 Class Diagram



[Figure 19] Class diagram – Push Notification Manager Class

- Class Description
  - ✓ **Push Notification Manager Class**: Interface for sending push notification. When a user sets a notification in Task, Push Notification Manager Class is called at that time and Push Notification is sent Notification through the Learning X Student App of the user's.

5.3.4.2 Sequence Diagram


User
Calendar Management System
DB_Handler
Push Notification Manager
Set an alarm at a task
Save the alarm info
Save the alarm time
Send Push Notification via Learning X Student App.

[Figure 20] Sequence diagram – Push Notification Manager Class

6 Protocol Design

6.1 Objectives

This chapter describes what structures are used for the protocols which are used for interaction between each subsystem, especially iCalendar application and the server. Also, this chapter describes how each interface is defined.

6.2 HTTP

The Hypertext Transfer Protocol (HTTP) is an application layer protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen in a web browser.

6.3 OAuth

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. This mechanism is used by companies such as Amazon, Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites.

6.4 Authentication

6.4.1 Signup

- Request

[Table 1] Table of signup request

Attribute	Detail	
Protocol	OAuth, HTTP	
Request Body	iCampus ID	User's iCampus ID

Attribute	Detail	
	Request Token	Token for OAuth
	User	Basic information of the user
Attribute	**Detail**	
Method	POST	
URI	/signup	
Parameter	User	Information about the user : ID, password, iCampus ID
Header	Authorization	User authentication

- Response

[Table 2] Table of signup response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400 (Bad Request)	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Success	Fail
	Message	Success message

6.4.2 Login

- Request

[Table 3] Table of login request

Attribute	Detail	
Protocol	OAuth, HTTP	
Request Body	iCampus ID	User’s iCampus ID
	Request Token	Token for OAuth
	User	Basic information of the user
Attribute	**Detail**	
Method	POST	
URI	/login	
Parameter	User	ID, password
Header	Authorization	User authentication

- Response

[Table 4] Table of login 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

6.5 Custom Calendar Management

6.5.1 Create Custom Calendar

- Request

[Table 5] Table of create custom calendar request

Attribute	Detail	
Method	POST	
URI	/user/:id/create_calendar	
Parameter	Calendar	Information about the calendar
Header	Authorization	User authentication

- Response

[Table 6] Table of create custom calendar response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.5.2 Modify Custom Calendar

- Request (get profile)

[Table 7] Table of modify custom calendar request

Attribute	Detail	
Method	GET	
URI	/user/:id/calendar_id/modify	
Request Body	Calendar	Object ID of calendar, updated information about the calendar
Header	Authorization	User authentication

- Response

[Table 8] Table of modify custom calendar response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	User	Success message
Failure Response Body	Message	Fail message

6.5.3 Delete Custom Calendar

- Request

[Table 9] Table of delete custom calendar request

Attribute	Detail	
Method	DELETE	
URI	/user/:id/calendar_id/delete	
Request Body	Calendar	Object ID of calendar

- Response

[Table 10] Table of delete custom calendar response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.5.4 Get Calendar List

- Request(get history)

[Table 11] Table of get calendar list request

Attribute	Detail	
Method	GET	
URI	/user/:id/calendars	
Parameters	-	-
Header	Authorization	User authentication

- Response

[Table 12] Table of get calendar list response

Attribute	Detail
Success Cod e	200 OK
Failure Code	HTTP error code = 404 (Not Found)

Attribute	Detail	
Success Response Body	Calendars	List of calendars that the user has created or got invited for by the other users

6.5.5. Get Custom Calendar

- Request(get history)

[Table 13] Table of get custom calendar request

Attribute	Detail	
Method	GET	
URI	/user/:id/calendar_id	
Parameters	-	-
Header	Authorization	User authentication

- Response

[Table 14] Table of get custom calendar response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Calendar, Task object	Information about the calendar and task objects included in that calendar (including object IDs)
Failure Response Body	Message	Fail message

6.6 Group Sharing

6.6.1 Send Invitation

- Request

[Table 15] Table of send invitation request

Attribute	Detail	
Method	POST	
URI	/user/:id/send_invitation	
Parameter	User, Invitee, Invitation	ID of the user and the invitee, message for the invitation
Header	Authorization	User authentication

- Response

[Table 16] Table of send invitation response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 403 (Forbidden)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.6.2 Check Invitation

- Request

[Table 17] Table of check invitation request

Attribute	Detail	
Method	GET	
URI	/user/:id/check_invitation	
Parameter	-	-
Header	Authorization	User authentication

- Response

[Table 18] Table of check invitation response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Invitation List	Shows a list of invitations to the user from the other users
Failure Response Body	Message	Fail message

6.6.3 Accept/Reject Invitation

- Request

[Table 19] Table of accept/reject invitation request

Attribute	Detail
Method	POST
URI	/user/:id/check_invitation/initation_id

Attribute	Detail	
Request Body	Invitation, acceptance/rejection	Object ID of invitation, information about whether the user accepted/rejected the invitation
Header	Authorization	User authentication

- Response

[Table 20] Table of accept/reject invitation response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 403 (Forbidden)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.7 Task Object Management

6.7.1 Create Task Object

- Request

[Table 21] Table of create task object request

Attribute	Detail	
Method	POST	
URI	/user/:id/calendar_id/create_task	
Request Body	Calendar, Task object	Object ID of calendar, information about the task object

Attribute	Detail	
Header	Authorization	User authentication

- Response

[Table 22] Table of create task object response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.7.2 Edit Task Object

- Request (add item to cart)

[Table 23] Table of edit task object request

Attribute	Detail	
Method	POST	
URI	/user/:id/calendar_id/edit_task/object_id	
Paramet er	Calendar, Task O bject	Object ID of calendar and task object, updated information about t he task object
Header	Authorization	User authentication

- Response

[Table 24] Table of edit task object response

Attribute	Detail	
Success Code	200 OK = 404 (Not Found)	
Failure Code	HTTP error code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.7.3 Delete Task Object

- Request

[Table 25] Table of delete task object request

Attribute	Detail	
Method	DELETE	
URI	/user/:id/calendar_id/delete_task/object_id	
Request Body	Calendar, Task Object	Object ID of calendar and task object

- Response

[Table 26] Table of delete task object response

Attribute	Detail
Success Code	200 OK
Failure Code	HTTP error code = 404 (Not Found)

Attribute	Detail	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.7.4 Get Task Object

- Request

[Table 27] Table of get task object request

Attribute	Detail	
Method	GET	
URI	/user/:id/calendar_id/object_id	
Parameters	-	-
Header	Authorization	User authentication

- Response

[Table 28] Table of get task object response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Task object	Information about the task object(including object ID)
Failure Response Body	Message	Fail message

6.8 Task Alarm Management

6.8.1 Update Alarm Information (for 'Learning X')

- Request

[Table 29] Table of update alarm information request

Attribute	Detail	
Method	POST	
URI	-	
Parameters	User, Task object	User ID, alarm information about the user's task object
Header	Authorization	User authentication

(Note : Operation takes place automatically every time the user create/edit/deletes a task obje
ct)

- Response

[Table 30] Table of update alarm information response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400 (Bad Request)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

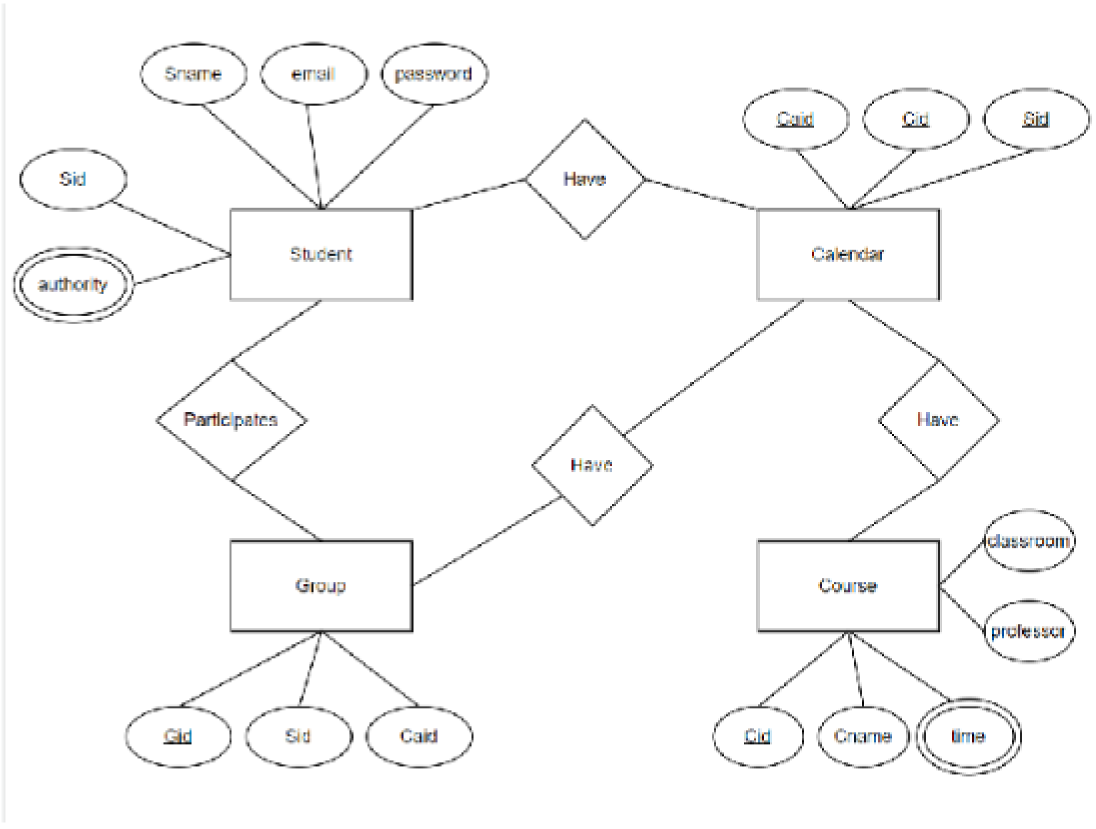
7 Database Design

7.1 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.

7.2 ER Diagram

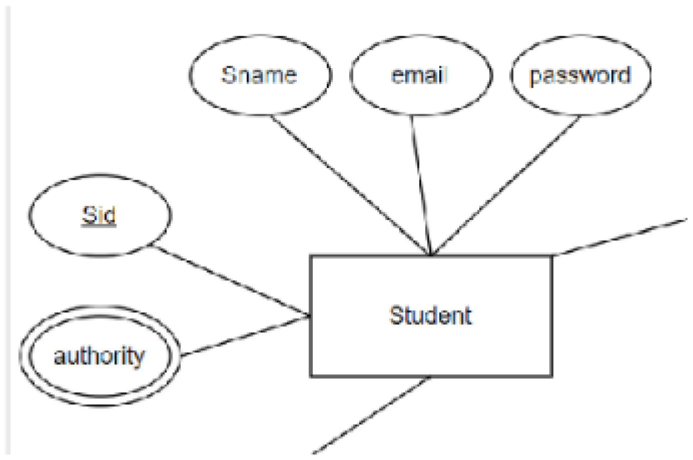
In I-Calendar, there are four entities; Student, Calendar, Group, Course. ER-diagram expresses each entities as rectangle and their relationship as diamond. All of the attributes are expresses as single circles, and primary keys are expressed with an underline. Attributes that have multiple values are expresses as double concentric circles.



[Figure 21] ER-diagram

7.2.1 Entities

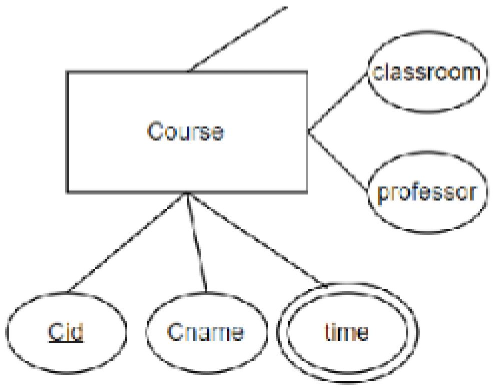
7.2.1.1 Student



[Figure 22] ER diagram, Entity, Student

Student Entity represents user(students) of I-Calendar. It consists Student ID(Primary key), Authority(which contains the information whether the student has an authority to access to groups when checking out the calendars of the members), Student name, E-mail address, Password.

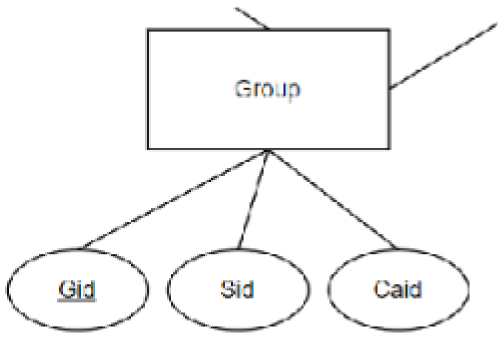
7.2.1.2 Course



[Figure 23] ER diagram, Entity, Course

Course Entity represents courses. It consists Course ID(Primary Key), Course name, Time(Time has two values since courses span two times), Classroom, Professor.

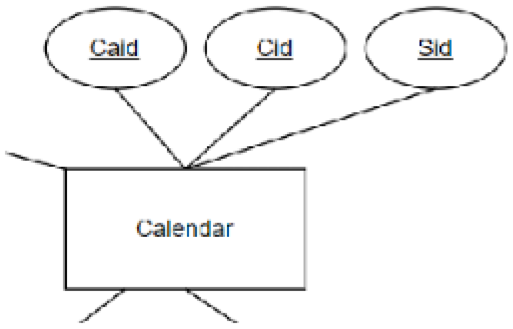
7.2.1.3 Group



[Figure 24] ER diagram, Entity, Group

Group entity represents the groups that students participate. It contains Group ID(Primary Key), Student ID(Foreign Key), Calendar ID(Foreign Key).

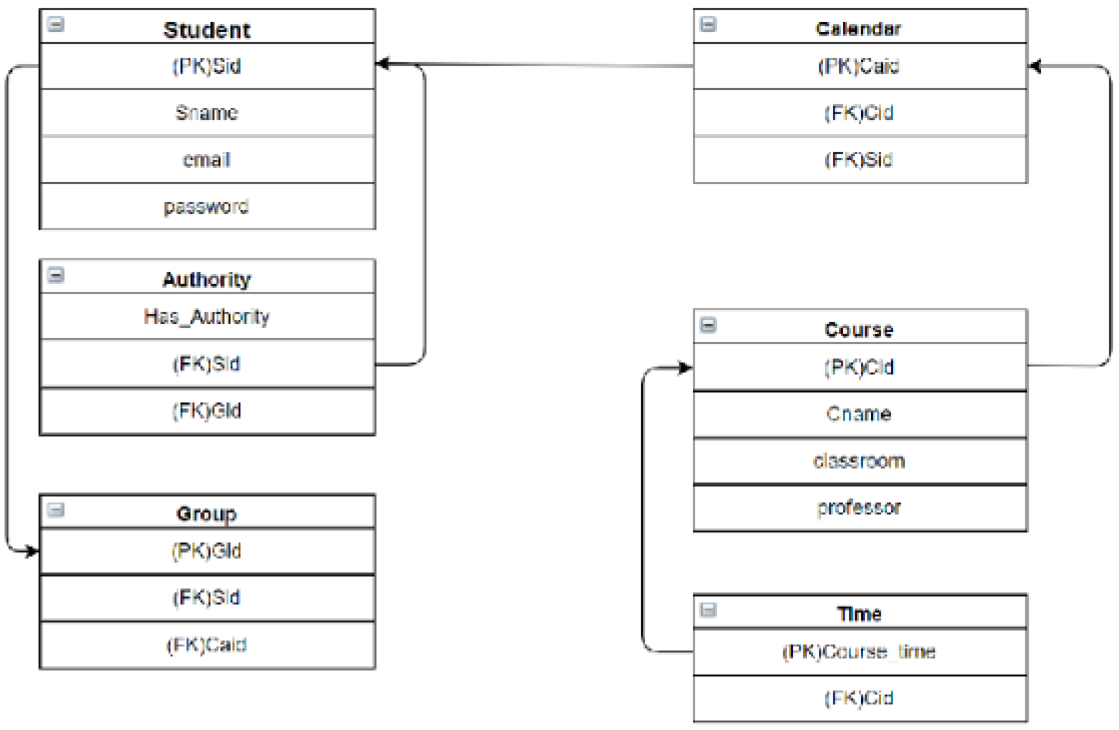
7.2.1.4 Calendar



[Figure 25] ER diagram, Entity, Calendar

Calendar entity represents informations about calendar. It contains Calendar ID(Primary Key), Course Id(Foreign Key), Student Id(Foreign Key).

7.3 Relational Schema



[Figure 26] Relational Schema

7.4 SQL DDL

7.4.1 Student

```
CREATE TABLE Student

(

  Sid INT NOT NULL,

  Sname CHAR(10) NOT NULL,

  email CHAR(30) NOT NULL,

  password CHAR(30) NOT NULL,

  PRIMARY KEY (Sid)

);
```

7.4.2 Course

```
CREATE TABLE Student

(

  Sid INT NOT NULL,

  Sname CHAR(10) NOT NULL,

  email CHAR(30) NOT NULL,

  password CHAR(30) NOT NULL,

  PRIMARY KEY (Sid)

);
```

7.4.3 Calendar

```
CREATE TABLE Calendar

(

  Caid INT NOT NULL,

  FOREIGN KEY (Cid) REFERENCES Course(Cid),

  FOREIGN KEY (Sid) REFERENCES Student(Sid),

  PRIMARY KEY (Caid)

);
```

7.4.4 Group

```
CREATE TABLE Group
(
  Gid INT NOT NULL,
  FOREIGN KEY (Sid) REFERENCES Student(Sid),
  FOREIGN KEY (Caid) REFERENCES Calendar(Caid),
  PRIMARY KEY (Gid)
);
```

7.4.5 Authority

```
CREATE TABLE Authority
(
  Has_authority BINARY NOT NULL,
  PRIMARY KEY (Has_authority, Sid),
  FOREIGN KEY (Sid) REFERENCES Student(Sid),
  FOREIGN KEY (Gid) REFERENCES Group(Gid)
);
```

7.4.6 Time

```
CREATE TABLE Time
(
  Course_time TIME NOT NULL,
  PRIMARY KEY (Course_time),
  FOREIGN KEY (Cid) REFERENCES Course(Cid)
);
```

8 Testing Plan

8.1 Overall

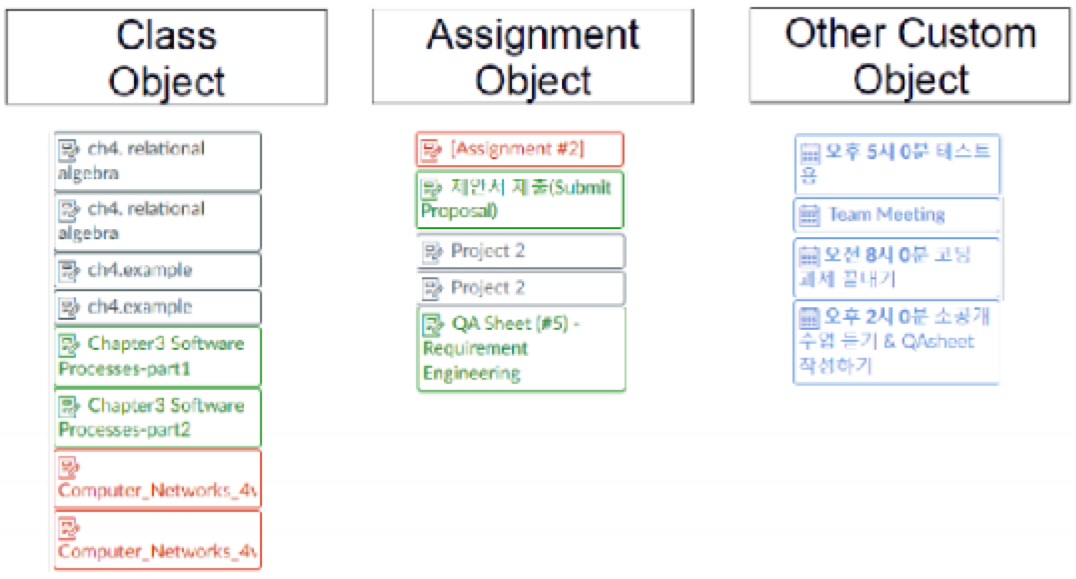
This section establishes a plan for system testing to determine if the system is operating as intended to identify and analyze the system for defects after completion of the system. During the development of the 'I-Calendar' system, testing is divided into three stages. It is divided into Developing Testing, Release Testing, and User Testing.

8.2 Development Testing

Development testing is aimed for synchronizing application of broad spectrum of defect prevention and detection strategies for reducing software development risk.

8.2.1 Performance

Since customized calendar and group sharing are main tasks in I-Calendar, it is crucial for developers to focus on those. The program should make customized calendar objects shown as figure below properly. They should contain informations from the courses of the student according to database. After that, the program should share that calendar with the groups that the student is actually in.



[Figure 27] Example of calendar object

8.2.2 Reliability

The system operating safely without failure. Since we are upgrading the functionality of existing calendar app, I-Calendar components and units that integrated to the already existing system should operate and represented correctly. Therefore, we need to proceed development testing from integrating stage and check failure iteratively.

8.2.3 Security

When the system access to the database, possible unwanted visitors can access another information whether intended or not. Regardless of the value of the information, it should be protected. In order to achieve that, developers should concern about database access authority issues for that.

8.3 Release Testing

Release testing is the process of testing a particular release of a system that is intended for use outside of the development team. The primary goal of the release testing process is to convince the customer of the system that it is good enough for use. The test will be focused on system specification, it will be conducted in black-box testing process.

8.4 User Testing

User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing. Since I-Calendar aims at comforting the use of the calendar. User testing will be conducted in beta testing, which release software that is made available to users to allow them to experiment and to raise problems that they discover with the system developers.

8.5 Testing Case

Testing case would be set according to 3 fundamental aspects of the application– function(interaction), performance, and security. Tasting cases should set up possible scenario and realistic situation that can proceed beta test in user testing phase. This phase will be proceeded iteratively since we can get feedback by users by beta testing in user testing phase.

9 Development Plan

9.1 Objectives

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

9.2 Frontend Environment

9.2.1 JavaScript

[Figure 28] JavaScript Logo

In this project, there is no actual implementation. However, if implemented, it would be to implement additional features on the icampus web page running on a Chrome browser. Then the most appropriate language could be JavaScript.

The iCampus feature already has a calendar implementation. Therefore, this project will place more importance on the back end than on the front end.

9.3 Backend Environment

9.3.1 Github

[Figure 29] Github logo

It is a useful environment for sharing project-related documents such as Proposals, Requirements, Design Documentation, etc. on GitHub while providing synchronization capabilities that allow team members to make simultaneous code changes. Our team members are using GitHub for developing iCalendar and controlling version of it.

9.3.2 FireBase

[Figure 30] Firebase logo

It supports development of mobile and web applications by providing various features such as cloud storage, real-time database, machine learning kit, etc. Among them, we would use real-time database feature for managing data of users, laptops, and etc. Thanks to real-time database, data is synchronized across all the clients connected to this database. It means all students can share a single real-time database instance and receive updated data with automated update.

9.4 Constraints

The system will be designed and implemented based on the contents mentioned in this document. Other details are designed and implemented by selecting the direction preferred by the developer, but the following items are observed.

- Custom Calendar getting speed should not exceed 4 seconds.
- Avoid using technology or software that requires a separate license or pays for royalty. (Exclude this provision if this is the only technology or software that the system must require.)
- Customizing Calendar is must friendly to students who using this functionality in more convenient way.
- Use open source software whenever possible
- Consider the system cost and maintenance cost
- Consider future scalability and availability of the system
- Optimize the source code to prevent waste of system resources
- Consider future maintenance and add sufficient comments when writing the source code

9.5 Assumptions and Dependencies

All systems in this document are written on the assumption that they are designed and implemented based on Chrome browser. Therefore, all contents are written based on the i-campus server system that have front-end at http communication environment.

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 9] Document History

Date	Version	Description	Writer
2021/05/05	0.1	Style and overview	Suyoung Min
2021/05/05	0.2	Addition of 1,2,3	Suyoung Min
2021/05/06	1.0	Addition of 7, 8	Sumin Ham
2021/05/06	1.1	Addition of 6	Chanjong Lee
2021/05/06	1.2	Addition of 5	Minseo Kim
2021/05/06	1.3	Addition of 4	Taewoo Yoo
2021/05/08	1.4	Addition of 9,10	Suyoung Min
2021/05/10	1.5	Revision of 7.2	Sumin Ham
2021/05/10	1.6	Revision of 4.2	Taewoo Yoo
2021/05/11	1.7	Revision of 6	Chanjong Lee
2021/05/11	1.8	Revision of 9	Suyoung Min
2021/05/12	1.9	Revision of 5.3	Minseo Kim
2021/05/12	1.9	Revision of structure	Chanjong Lee