[Software Domain Analysis & Design]

M.A.X.I.M

(Matching Academy X Information Mapping)

- Elaboration Phase Iteration 2 (Final) Document -

Professor : Kristin, Chung
Team name : Pyramid
Team members :

201620560	신정호	0x01.leo@gmail.com
201720715	박주현	wngus1100@ajou.ac.kr
201720723	박수린	marble_giraffe@naver.com
201720724	서재명	kevindec18@ajou.ac.kr

Table of Contents

Content of Table

Content of Figure

1. VISION

1.1. Introduction

Most of Korean have a passion for study compared to other countries. Even if school offer lots of classes, students find another option for additional study. However, most students refer to their friends' opinion and a rumor to decide what kind of academy they will choose. Because of inaccurate information about the specific academy, students have trouble in studying. To solve this problem, recommendation system will play an important role in deciding proper academy for students.

1.2. Positioning

1.2.1. Business Opportunity

Over ten million of students spend most of their time in studying to increase their competitiveness in society. And the market is growing continuously regardless of the age. Academy wants lots of students to be enrolled, and students want to get accurate information for deciding the place to supplement their study. In this situation academy can help you build successful plan for additional study by recommending best options for you. Academy don't have to spend their time in advertising, and students don't need to spend their time in finding proper academy. Therefore, their concern can be solved at once.

1.2.2. Problem Statement

Usually, students try to find the academy which can help their study, but it is hard to get accurate information about the academy. In most case, students decide their academy according to their friends' opinion or rumors from other people. With the wrong information, students have to waste their time and money. Academy also have trouble in managing their students if improper students are in the class. They have to care about every students in the class, but it could be nuisance to manage in the perspective of the academy if the difference between students is too high. In the case of both academy and students, they have to waste money and time just because of inaccurate information.

1.2.3. Product position statement

For students who need to find an academy for their development, this system is a solution that prevents wasting time and money. Most of academy provides recommendation service just based on their own academy information. However, our service recommend academy to students based on both academy information and student's own profile information. Also, compared to other academy recommend system, our service will offer prediction for test score of each semester so that students can know how much he could be improved in advance.

1.3. Stakeholder description

1.3.1. Stakeholder Summary

- Academy : A person who wants to make a profit by promoting his own academy.
- Manager : A person who undertakes overall academy management.
- Teacher : A person who provides class syllabus and his own career.
- Student : A person who wants to get recommendation of the appropriate academies.

1.3.2. User Summary

- Manager : A person who manages the basic information of academy, class list and student's enrollment.
- Student : A person who registers profile and preference.
- Teacher : A person who provides class syllabus and manages student's test score.

1.3.3. Key high-level Goals and problems of the Stakeholders

high-level goals	priority	problem and concern	current solution
Want to get the appropriate academies list	High	It is difficult to find the appropriate academy for students just with the general information.	Recommend appropriate academies based on detailed information of the class and what student prefers to and provide the prediction of test score.
Want to supervise student's test score	Low	It is hard to recognize each student's distribution of test scores.	Manage each student's test score and visualize the distribution of test score
Benefit from promoting the academy to students	High	Academy waste money and time to promote.	Provide a platform to promote the academy.

<Table 1> Key high-level goals

1.3.4. User-level Goals

- Manager : Want to make a profit by promoting his own academy.
- Student : Want to get recommendation to the appropriate list of academies.
- Teacher : Want to manage syllabus and supervise their students' score.

1.3.5. User Environment

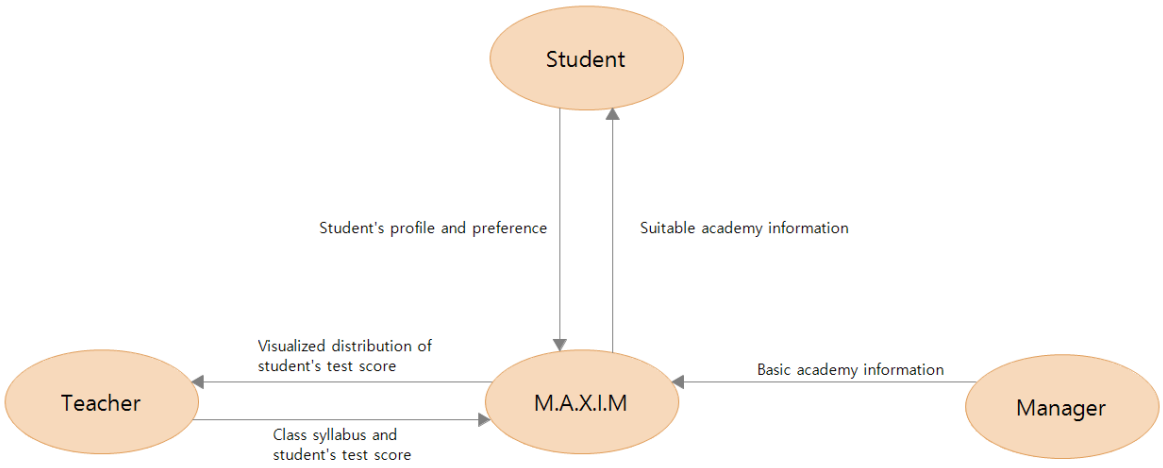
- User can access to a web browser.

1.4. Product Overview

1.4.1. Product Perspective

Users can use the M.A.X.I.M through web browser. User type1(Manager) will attracts students by registering the academy's information. User type2(teacher) will manage students' test score. User type3(Student) will choose the appropriate academy from the recommended list.

1.4.2. Context Diagram



<Figure 1> Context Diagram

1.4.3. Summary of Benefits

Supporting Feature	Stakeholder benefits
Functionally, the system will provide recommendation service	Provide information that suits for the student.
Support a platform to promote academy for free	Academy don't waste money on advertising.

<Table 2> Summary of benefits

1.4.4. Assumption and Dependencies

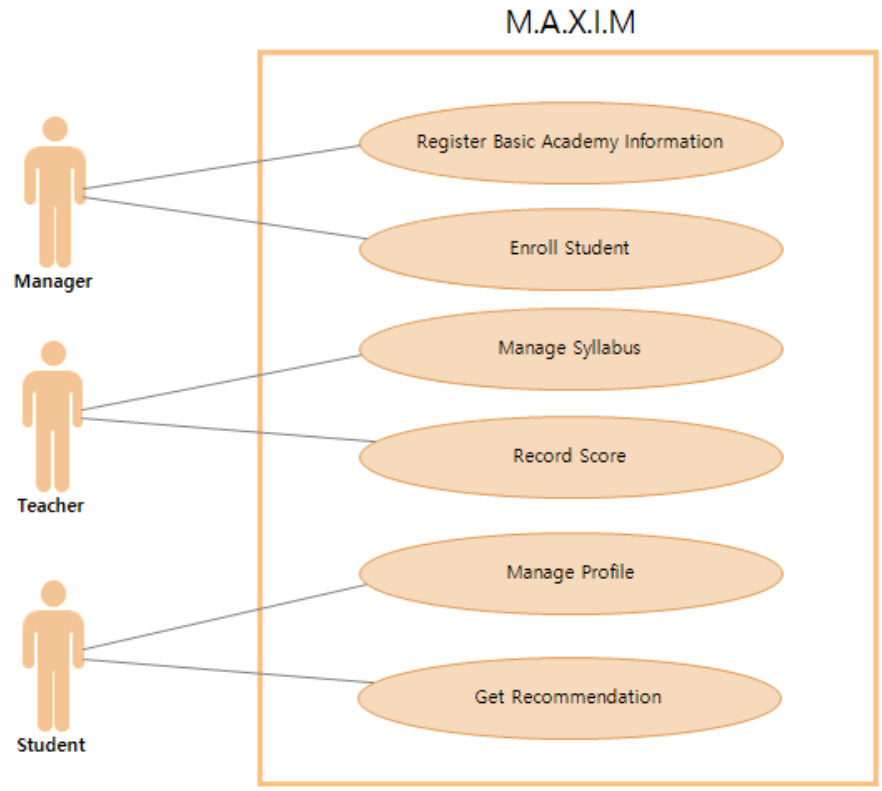
Users can access through the web browser. Encourage User type1(Manager) to register their information to the system. Make User type2(Teacher) to write class syllabus and student's test scores. Need User type3(Student) to register their profile to the system.

1.5. Summary of System Features

- Enroll student in the academy.
- Recommend academy lists with clustering algorithm.
- Visualize student's test score distribution.
- Authentication Process using personal code.

2. Requirements

2.1. Use Case Diagram



<Figure 2> Use case diagram

2.2. Use Case Brief

Register Basic Academy Information

- Manager manages basic information such as academy name, location and telephone number in the system.
- Manager manages detailed information about each class which is about name, subjects, tuition fee.

Enroll Student

- Manager selects students from the list.
- Manager stores information about the selected student's basic information if the student is already not registered.
- The manager saves what classes the student takes in the system.

Manage Syllabus

- Teacher manage his class time table.
- Teacher register course progress for the week in the system.
- Teacher manage test-name, schedule, pass-through score.

Record Score

- Teacher selects the test that he want to write down his students' grade.
- Teacher records the scores of the students who took the exam on the system.

Manage Profile

- Student manages basic human resources such as name, age, residence.
- Student answers the questions about which class style a student prefers more, which is provided by the system.
- Student manage the above preference.

Get Recommendation

- Student chooses the subject that he or she wants to take.
- Student inputs the own level of knowledge about subject that a student thinks for himself.
- Student enters the day of the week and available time when a student wants to take the class.
- System recommends classes that are likely to satisfy student, based on the student profile and input data.

2.3. Use Case Fully dressed

Use Case 1. Register Basic Academy Information

Use Case Name : Register Basic academy information
Scope : Register academy.
Level : user goal
Primary Actor : Manager

Stakeholders and Interest:

- Academy : Want to make a profit by promoting his own academy.
- Manager : Want to manage basic information of academy.

Precondition :

- User authentication.
- Manager should be aware of the ID given to the academy.

Success guarantee (postconditions) :

- The system updates the academy information.

Main Success Scenario :

1. Manager selects *Register Academy*.
2. Manager enters Academy ID to system.
3. The system shows register form to manager.
4. Manager registers basic information academy name, location, telephone number and office hour.
5. Manager enters the name of all classes in the academy.
6. Manager fills subjects and tuition fee, teacher career, level, class hour for each class.
7. Manager submits academy information to system.
8. The system gives Manager a notice that registration is successfully completed.

Extensions (Alternative Flows) :

*a. At any time, User's network connection fails.
1. System shows connection failure message to manager.

2. User tries to connect again.

*b. At any time, Server fails to respond.
1. System try to send request again.
2. Set timeout about server response.
2a. Over time out
1. Terminate system by time out.

2a. The academy registered before.
1. The system shows error message.
2. The system shows modification form.
3. Manager fills out modified information.
4. System uploads modified information and give a confirm message to Manager.

7a. The manager skips the element at least one.
1. The system shows submit error to manager.
2. System shows empty element space.

7b. The submit academy information failed.
1. Retry to submit.

Special Requirements :
- The response delay should be within 5 seconds in saving information in this system.

Technology and Data variation List :
- The contents of the form could be various.

Frequency of Occurrence :
- When academy first want to register in the system
- When manager want to modify academy information.

Open issues :
- Manager can fill in the wrong information.

Use Case 2. Enroll Student

Use Case Name : Enroll Student
Scope : M.A.X.I.M
Level : user goal
Primary Actor : Manager

Stakeholders and Interest :
- Manager : Want the information of the classes that students are taking to be managed efficiently.
- Teacher : Want to know which students are taking his classes.

Precondition :

- User authentication.
- Student agrees to offer their information to this system.
- The system already has a list of classes and information about teachers.
- Assume that there is some amount of data on the students who were previously taking classes.

Success guarantee (postconditions) :

- The system saves the basic information of a student and what classes he or she takes are in the system.
- The number of students in each class is correctly calculated and saved in the system.

Main Success Scenario :

1. Student arrives to manager and tells manager classes that he will takes.
2. Manager start a new *Enroll*.
3. The system shows the list of classes.
4. Manger selects the class from the list.
5. The system shows the list of students.
6. Manger selects the student from the list.
   Manager repeats 3-6 until indicates done.
7. Manager submits the form to the system.
8. The system shows success message.

Extensions (Alternative Flows) :

*a. At any time, User's network connection fails.
   1. System shows connection failure message to manager.
   2. User tries to connect again.

*b. At any time, Server fails to respond.
   1. System try to send request again.
   2. Set timeout about server response.
      2a. Over time out
      1. Terminate system by timeout.

4a. The student asks Manager to skip the class that he do not want to take.
   1. Manager removes the class requested from the list.

4b. Manager cannot find the name of classes that student wants to take on the list.
   1. The system signals error and rejects entry.
   2. Manager responds to the error :
      2a. Manager register new class name.
      2b. Manager registers basic information about class such as teacher, tuition fee, class time table.
      2c. The class list updated in the system.

6a. Manager cannot find the name of students he wants on the list.

1. The system signals error and rejects entry.
2. Manager responds to the error :

2a. Manager register new student name.

2b. Manager registers basic human resources about student such as name, age, residence, phone number.

2c. The student's profile registered in the system.

6b. Manager finds students with the same name.

1. The system shows list of the same name.
2. Manager selects the name of the student manager want to find.

Special Requirements :

- The response delay should be within 5 seconds in saving information in this system.

Technology and Data variation List :

- The contents of the form could be various.

Frequency of Occurrence :

- At the start of each semester.

Open issues :

- Manager can register student's information inaccurately.
- Manager can register a new class on the list that does not exist.
- Manager can register classes that the student did not pay.

Use Case 3. Manage Syllabus

Use Case Name : Manage Syllabus
Scope : M.A.X.I.M
Level : user goal
Primary Actor : Teacher

Stakeholders and Interest :

- Manager : Want teachers to fulfill their duty earnestly.
- Teacher : Want to manage the class syllabus and exam schedule efficiently by making progress systematically.

Precondition :

- User authentication.
- The system already has a list of classes.

Success guarantee (postconditions) :

- The system saves teacher's weekly class progress charts.
- Test schedules are stored in the system.
- The system manages the entire academy timetable based on class schedule entered

by teacher.

Main Success Scenario :

1. Teacher start a *Manage Syllabus.*
2. The system shows the list of classes.
3. Teacher selects the class from the list that he is going to teach.
4. The System shows the class time, week list.
5. Teacher chooses the week.
6. The system provides a form that includes week, contents, test schedule.
7. Teacher enters the weekly contents.
8. Teacher enters test name, selects date and writes down pass-through score.
   Teacher repeats 5-8 until indicates done.
9. The system updates the teacher's weekly class progress chart.
10. The system shows success message.

Extensions (Alternative Flows) :

*a. At any time, User's network connection fails.
   1. System shows connection failure message to manager.
   2. User tries to connect again.

*b. At any time, Server fails to respond.
   1. System try to send request again.
   2. Set timeout about server response.
      2a. Over time out
      1. Terminate system by timeout.

2a. Teacher cannot find the name of class he teaches.
   1. The system signals error and rejects entry.
   2. Teacher asks manager to update the class list.

4a. Teacher did not register his class hour yet.
   1. The system signals error and rejects entry.
   2. Teacher responds to the error :
      2a. Teacher register new class hour.
      2b. The class hour updated in the system's time table.

4b. Teacher want to modify his class time.
   1. Teacher selects 'update time table'
   2. Teacher enters modified class hour.
   3. The system updates modified time table.

8a. Teacher doesn't want to take the exam that week.
   1. Teacher leaves the test schedule form blank.

Special Requirements :

- The response delay should be within 5 seconds in saving information in this system.

Technology and Data variation List :
- The contents of the form could be various.

Frequency of Occurrence :
- At the start of each month.

Open issues :
- Each type of test that teachers want can be different.
- The teacher may conduct an exam-free class.

Use Case 4. Record Score

Use Case Name : Record Score
Scope : M.A.X.I.M
Level : user goal
Primary Actor : Teacher

Stakeholders and Interest :
- Manager : Want teachers to fulfill their duty earnestly.
- Teacher : Want to manage student's grades effectively and make sure that his class is going well.
- Student : Want to know how much he could be improved through the class.

Precondition :
- User authentication.
- The system already has a list of students taking the class.
- Regular Test should be executed.

Success guarantee (postconditions) :
- The students' grades recorded by the teacher are saved in the system.
- Visualized data distribution should be offered.

Main Success Scenario :
1. Teacher start a new *Record*.
2. The system shows teacher's class list.
3. Teacher selects class.
4. The system shows the students' list and test list who are taking his class.
5. Teacher selects the test that he wants to write down his students' grades.
6. Teacher chooses the student who has taken the exam from the list.
7. Teacher record the student's grade.
   Teacher repeats 6-7 until indicates done.

8. Teacher submits the form to the system.
9. The system shows success message.

Extensions (Alternative Flows) :

*a. At any time, User's network connection fails.
1. System shows connection failure message to manager.
2. User tries to connect again.

*b. At any time, Server fails to respond.
1. System try to send request again.
2. Set timeout about server response.
2a. Over time out
1. Terminate system by timeout.

2a. Teacher cannot find his class in the list.
1. Teacher asks manager to update the class list.

5a. Teacher take an exam that was not in syllabus.
1. System shows error message.
2. System redirect *Manage Syllabus.*

6-7a. The teacher want to skip recording score of special student.
1. Teacher checks "not taking the exam" in that special student's form.

Special Requirements :

- The response delay should be within 5 seconds in saving information in this system.
- The system uses students' grades anonymously to recommend academy to other students.

Technology and Data variation List :

- The contents of the form could be various.

Frequency of Occurrence :

- Most frequently used.

Open issues :

- The teacher may feel that simple quiz doesn't necessarily need to enter grades into the system.

Use Case 5. Manage Profile

Use Case Name : Manage Profile
Scope : M.A.X.I.M
Level : sub-function
Primary Actor : Student

Stakeholders and Interest:

- Student : Want to offer detailed information to get accurate recommendation.

Precondition:

- Student agrees to offer their information to this system.
- User authentication.

Success guarantee (postconditions):
- The system saves the student's information.

Main Success Scenario:
1. Student starts *Manage Profile.*
   If Student wants to register new Profile, Student conducts to steps 2-4.
2. The student starts to register new profile.
3. The system shows the form which includes human resources such as name, age, residence.
4. The student fills the form to the system.
5. The system shows the form which includes the preference of the number of students in class, career of teachers, tuition fee, and age distribution of students in class.
6. The student fills the form to the system and submits the system.
7. System gives Student a notice that registration is successfully completed.
8. The system shows profile.

Extensions (Alternative Flows) :

*a. At any time, User's network connection fails.
   1. System shows connection failure message to manager.
   2. User tries to connect again.

*b. At any time, Server fails to respond.
   1. System try to send request again.
   2. Set timeout about server response.
      2a. Over time out
      1. Terminate system by timeout

2a. Student already registered his profile.
   1. System disapproves Student's request.
   2. System shows a message that the profile is already registered.
      2a. Student presses the modify button if he wants to modify profile
         1. Student clicks Modify Profile button.
         2. The system shows modification form.
         3. Student fills out new profile in modification form.
         4. System uploads modified profile and give a confirm message to Student.

5a. Student do not want to express his preference.
   1. Student choose 'Don't care'.

6a. The student skips the question at least one.
   1. System shows error message.
   2. Return to the questions that student skipped.

6b. System fails to store student's profile.
    1. Retry to store profile.
    2. Set timeout about server response.
        2a. Over time out
            1. Terminate system by timeout.

Special Requirements :
- The response delay should be within 5 seconds in saving information in this system.

Technology and Data variation List:
- The contents of the form could be various.

Frequency of Occurrence :
- Most frequently used.

Open issues :
- Student can register their information inaccurately.
- Student can register same information again.
- Student can answer "Not certain" to all questions.

Use Case 6. Get Recommendation

Use Case Name : Get Recommendation
Scope : M.A.X.I.M
Level : User Goal
Primary Actor : Student

Stakeholders and Interest :
- Student : Want to be informed which class suits him best on the subject that he want to take.
- Academy : Wants to make a profit by promoting academy.

Precondition
- User authentication
- Students have already made a profile.

Success guarantee (postconditions) :
- Student knows his or her predicted grades.
- Student receives a list of recommendation classes.
- A list of recommended classes are saved in system with student id.

Main Success Scenario :
1. Student selects *Get Recommendation.*
2. System shows forms of the search window, level graphs, and timetable.
3. Student enters the subject he wants to take in the search window.
4. Student marks the own level of knowledge about the subject that a student thinks for himself on the graph.

5. Student marks the available days of the week and times on the displayed time table.
6. System shows recommendation list of classes that are likely to satisfy student, based on his original profile and input data.
7. Student select a class that looks best for him.
8. System shows basic information about the class and figures that can change when Student take this class.

   Student repeats 7-8 until indicates done

9. Student checks the interest about attractive academy for him.
10. System sends queries which includes student's basic information to academy manager who manages the class that student choose.

    Student repeats 9-10 until he reveals all interest about attractive academy.

11. Student ends recommendation.

Extensions (Alternative Flows) :

*a. At any time, User's network connection fails.
   1. System shows connection failure message to manager.
   2. User tries to connect again.

*b. At any time, Server fails to respond.
   1. System try to send request again.
   2. Set timeout about server response.
      2a. Over time out
      1. Terminate system by timeout.

1-5c. At any time, Student wants to change his profile : Include *Manage Profile*

4a. Student do not want to mark the own level.
   1. Student chooses 'I don't know'.

5a. Student is not sure at what time he is available.
   1. Student chooses 'I don't know'.

6a. System does not recognize the subject that student search for.
   1. System shows error message.
   2. System ask student to search again.
      2a. System fails again.
      1. System shows error message.
      2. Redirect to initial page.

6b. System fail to find any classes of student's available time.
   1. System shows error message.

2. System ask student to select another time.
   2a. System fails again.
   1. System shows error message.
   2. Redirect to initial page.

9a. Student cannot find any attractive interest in recommended class list.
   1. Student chooses “No interest for all”

Special Requirements :
- The response delay should be within 5 seconds in user authentication process.
- The response delay should be within 30 seconds in recommendation process.
- The recommended class list will be saved in system within 30 days, and then deleted in the system.
- The query will be saved in system within 30 days, and then deleted in the system.

Technology and Data variation List:
- System will be developed on web server and application.

Frequency of Occurrence :
- Most frequently used.

Open issues :
- Student does not like all the classes on the list.
- Student want to take several classes of other subject at the same time.
- Student often tries to get recommendation so that data storage could be overcapacitated.

2.4. Non-Function Requirements/ Supplementary Specification

2.4.1. Functionality

1. Log-in and Error handling
   - All errors are logged at persistent storage.
   - Authentication is required.
   - If an error or failure occurs, the previous page is re-loaded by the logged file.
2. Plugged rules
   - Server always can respond to user in 1 second.
3. Security
   - Information of user must be encapsulated by information manage system.

2.4.2. Usability

1. Human factor
   - User could identify academy information easily. So, the legibility of the dataset in this system is important.
   - Users should perform tasks with a maximum of 10 steps.

2.4.3. Reliability

- If the system being attacked, dataset will be transmitted to other connected SSH server.
- If user fails to log-in 5 times continuously, re-authentication through email should be needed.
- If the request to system fails twice, pop up the message for user to re-access.

2.4.4. Performance

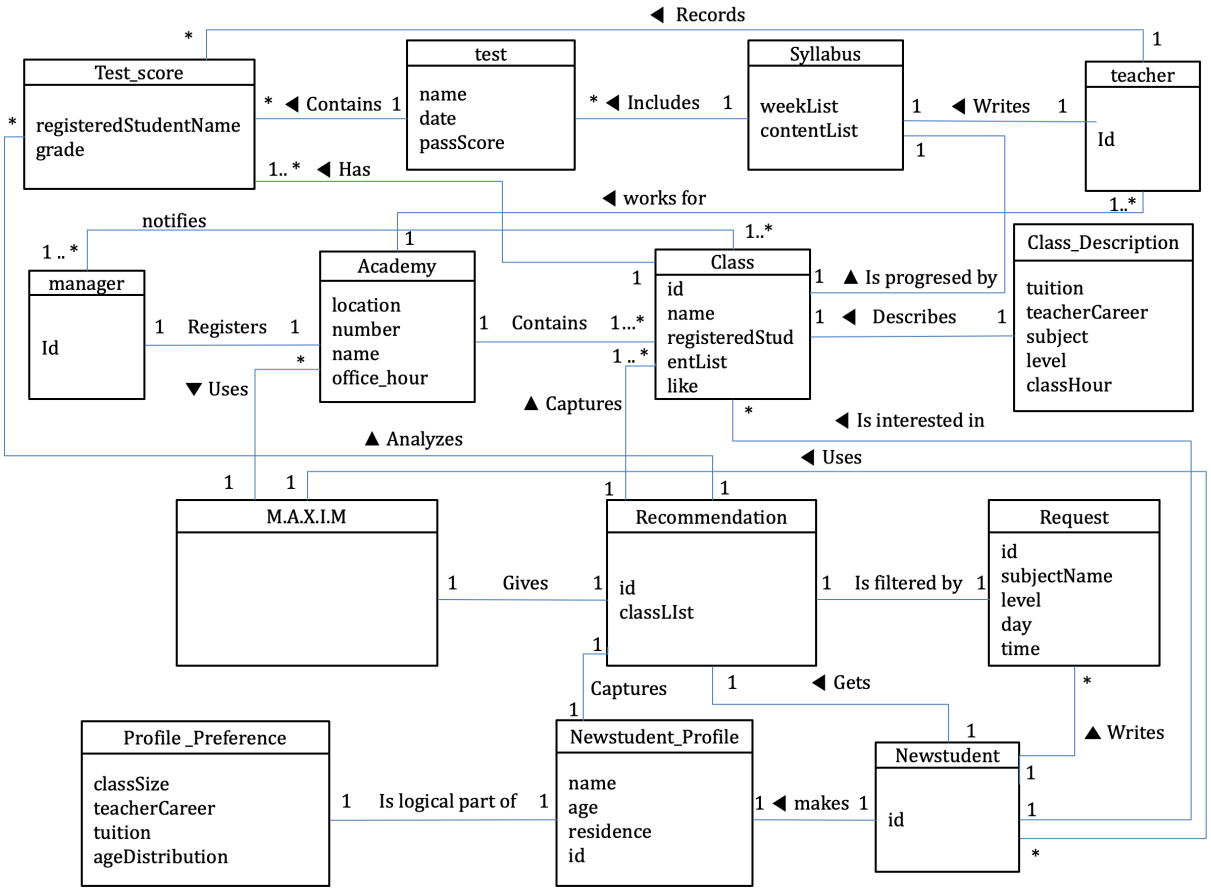
- The response time of M.A.X.I.M should be within 3 seconds.
- If the memory size of system exceeds 100GB, the oldest history will be deleted.
-

2.4.5. Supportability

1. Adaptability
   - M.A.X.I.M should be available in various web browsers, including Internet Explorer, Chrome, FireFox.
   - This system could be able to reflect database of various academies.
2. Configurability
   - This system can be able to apply requirements of all users.

3. Domain Model

3.1. Domain Model Diagram



<Figure 3> Domain Model

3.2. Domain Classes

Teacher	
Attribute	id
Association	One Teacher writes one Syllabus. One Teacher records zero or more Test_Score. One or more Teacher works for one Academy.
Syllabus	
Attribute	weekList, contentList
Association	One Syllabus includes zero or more Test. One Syllabus is written by Teacher. One Syllabus progresses one Class.
Test	
Attribute	name, date, passScore
Association	One Test contains zero or more Test_Score. Zero or more Test is included by Syllabus.
Test_Score	
Attribute	registeredStudentName, grade
Association	Zero or more Test_Score is contained by one Test. Zero or more Test_Score is analyzed by one Recommendation. Zero or more Test_Score is recorded by one Teacher. One ore more Test_Score is contained by one Class.
Manger	
Attribute	Id
Association	One or more Manger notify one or more Class. One or more Manager registers one Academy.
Academy	
Attribute	location, number, name, office_hour
Association	One Academy contains in one or more Class. One Academy is registered by one or more Manager. One Academy is worked by one or more Teacher. Zero or more Academy uses M.A.X.I.M
Class	
Attribute	id, name, registeredStudentList, like
Association	One Class is progressed by one Syllabus. One or more Class is notified by one or more Manger. One or more Class is contained in one Academy. One Class is described by one Class_Description. Zero or more Class interests one NewStudent. One or more Class is captured by Recommendation. One Class has one or more Test_Score
Class_Description	
Attribute	tuition, teacher career, subject, level, classhour
Association	One Class_Description describes one Class
Request	
Attribute	id, subjectName, level, day, time
Association	One Request is filtered by one Recommendation. Zero or more Request is written by one Student.
NewStudent	
Attribute	id
Association	One NewStudent is interested in zero or more Class One NewStudent writes zero or more Request.

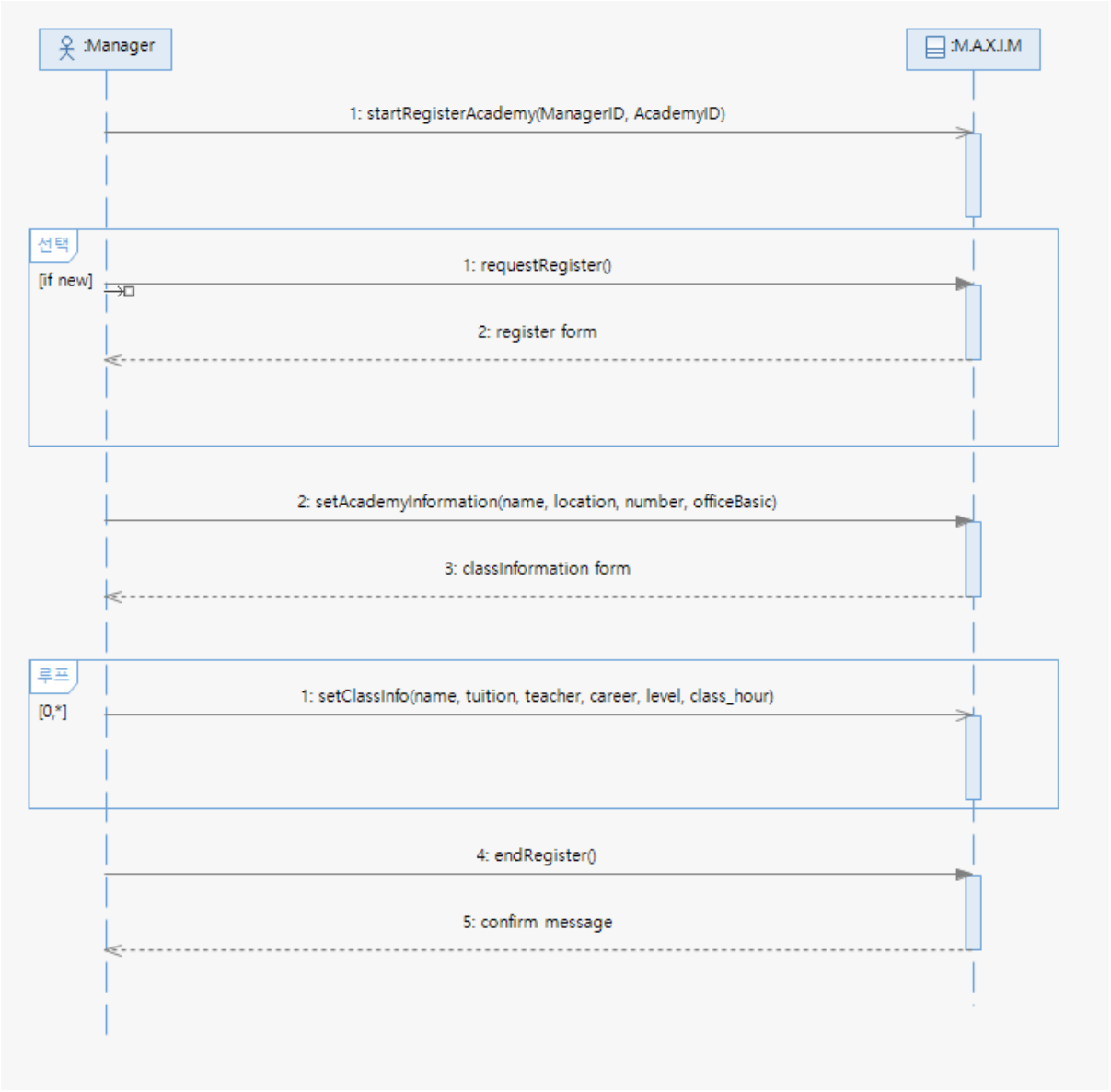
	One NewStudent makes one Student_Profile. One NewStudent gets one Recommendation. Zero or more NewStudent uses M.A.X.I.M
NewStudent_Profile	
Attribute	name, age, residence, id
Association	One NewStudent_Profile is made by one NewStudent. One NewStudent_Profile is been logical part of one Profile_Preference. One NewStudent_Profile is captured by one Recommendation.
Profile_Preference	
Attribute	classSize, teacherCareer, tuition, ageDistribution
Association	One Profile_Preference is logical part of one Student_Profile.
Recommendation	
Attribute	id, classList
Association	One Recommendation is filtered by one Request. One Recommendation is gotten by one NewStudent. One Recommendation captures one NewStudent_Profile. One Recommendation analyzes zero or more Test_Score. One Recommendation is given by one M.A.X.I.M.
M.A.X.I.M	
Attribute	None.
Association	One M.A.X.I.M. is used by zero or more Academy. One M.A.X.I.M. is used by zero or more NewStudent. One M.A.X.I.M gives one Recommendation.

<Table 3> Domain Classes

4. System Sequence Diagram

4.1. <Use Case 1> Register Basic Academy Information

4.1.1. System Sequence Diagram



<Figure 4> Register Basic Academy Information SSD

4.1.2. Operation Contract

Contract OC1. StartRegisterAcademy
Operation :
StartRegisterAcademy (managerID, academyID)
Cross Reference :
Register Basic Academy Information
Precondition :
- Manager instance mng was created.
- Manager is signed up & in.
- mng.ID was initialized.

Postcondition :

- Academy instance aca was created.
- aca. ID was initialized.
- aca was associated with mng.

<Table 4> Register Basic Academy Information OC1

Contract OC2. requestRegister
Operation :

requestRegister( )

Cross Reference :

Register Basic Academy Information

Precondition :

- managerID was checked by DBHandler.
- academyID was checked by DBHandler.
- Filling out Academy information in underway.

Postcondition :

- None.

<Table 5> Register Basic Academy Information OC2

Contract OC3. SetAcaInfo
Operation :

SetAcaInfo(name,location,number,office_hour)

Cross Reference :

Register Basic Academy Information

Precondition :

- Manager registered Acadmey to the system.
- Filling out Academy information in underway.

Postcondition :

- aca.name was initialized with name
- aca.location was initialized with location
- aca.number was initialized with number
- aca.office_hour was initialized with office_hour

<Table 6> Register Basic Academy Information OC3

Contract OC4. SetClassInfo
Operation :

SetClassInfo (name, tuition, teacher_career, level, class_hour)

Cross Reference :

Register Basic Academy Information

Precondition :

- Class instance cls was created.
- Class_Description instance cls_d was created.
- Filling out class information is underway.

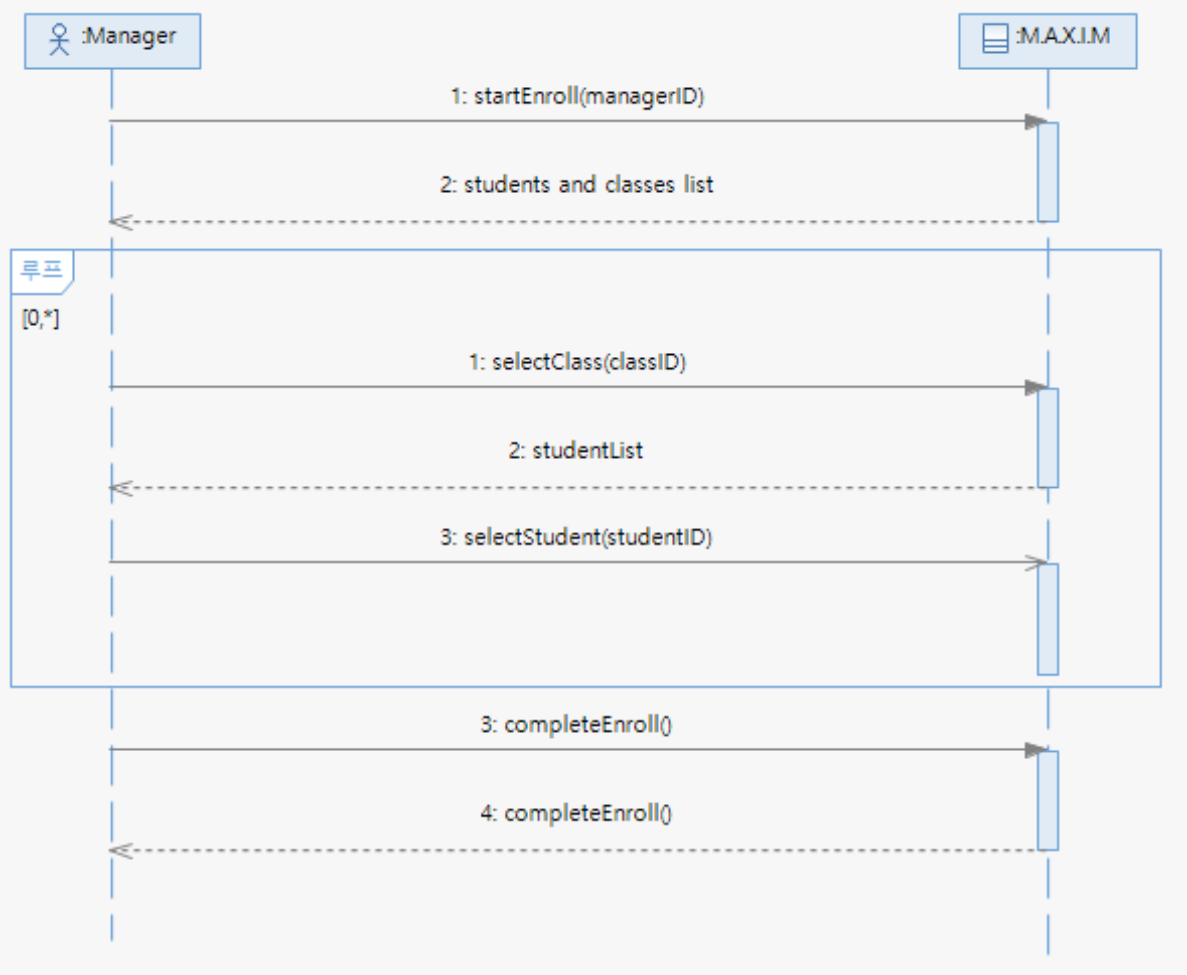
Postcondition :

- cls.name was initialized with name. - cls_d.tuition was initialized with tuition - cls_d.teacherCareer was initialized with teacher_career - cls_d.level was initialized with level - cls_d.classHour was initialized with class_hour - cls was associated with aca.

<Table 7> Register Basic Academy Information OC4

4.2. <Use Case 2> Enroll Student

4.2.1. System Sequence Diagram



<Figure 5> Enroll Student SSD

4.2.2. Operation Contract

Contract OC1. StartEnroll **Operation :** StartEnroll(managerID) **Cross Reference :** Enroll Student **Precondition :** - Manager is signed up & in. - Manager instance was associated with academy instance.

Postcondition :

- None.

<Table 8> Enroll Student OC1

Contract OC1. SelectClass

Operation :

SelectClass(classID)

Cross Reference :

Enroll Student

Precondition :

- class instance was created.

Postcondition :

- class instance was associated with Academy.

<Table 9> Enroll Student OC2

Contract OC3. SelectStudent

Operation :

SelectStudent(studentID)

Cross Reference :

Enroll Student

Precondition :

- There is Enroll underway.

Postcondition :

- class.registeredStudentList is added studentID.

<Table 10> Enroll Student OC3

Contract OC4. completeEnroll

Operation :

completeEnroll(class)

Cross Reference :

Enroll Student

Precondition :

- There is Enroll underway.

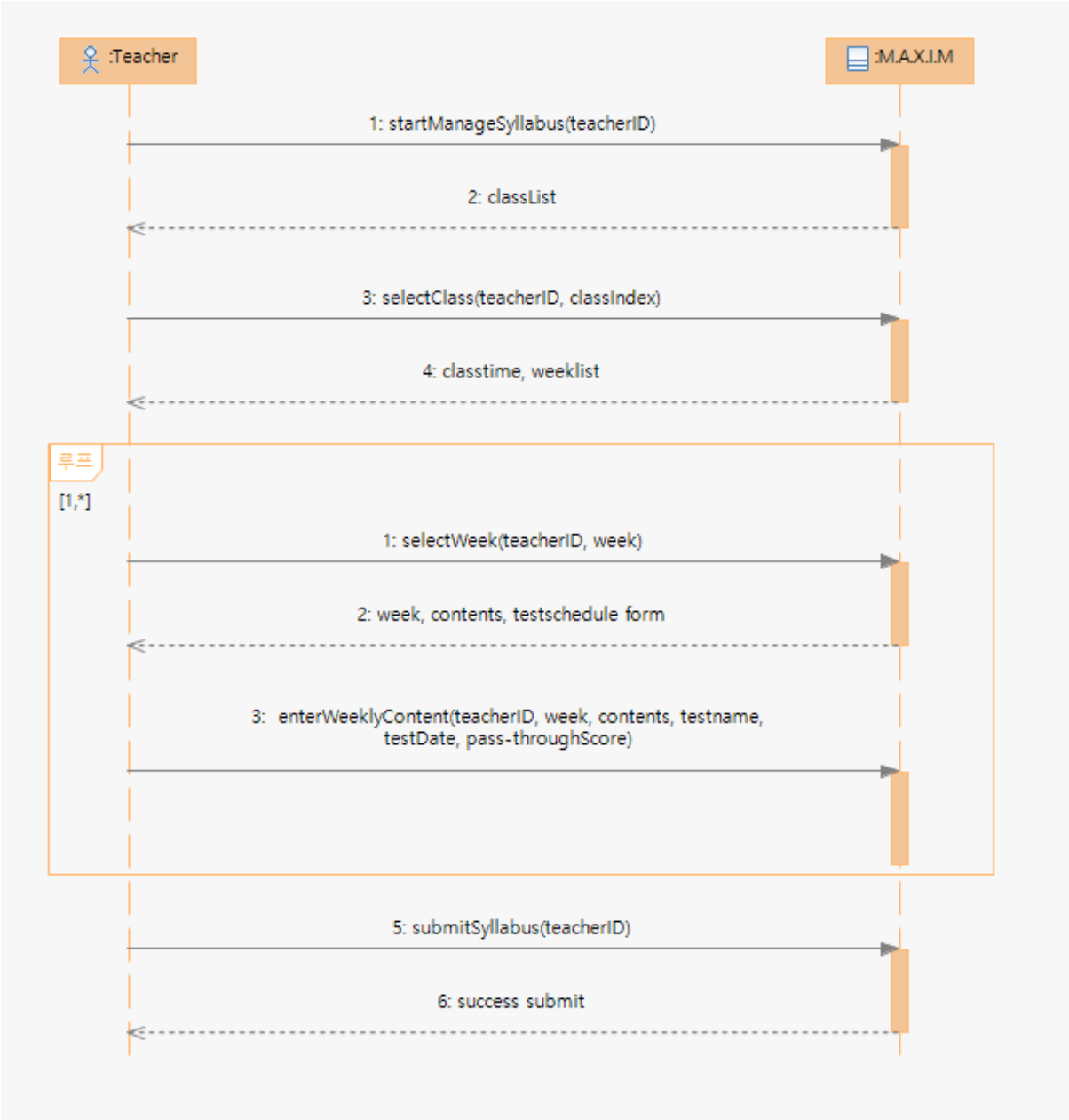
Postcondition :

- None.

<Table 11> Enroll Student OC4

4.3. <Use Case 3> Manage Syllabus

4.3.1. System Sequence Diagram



<Figure 6> Manage Syllabus SSD

4.3.2. Operation Contract

Contract OC1. StartManageSyllabus
Operation :
StartManageSyllabus(teacherID)
Cross Reference :
Manage Syllabus
Precondition :
- Teacher instance was created.
Postcondition :

- Syllabus instance syllabus was created.
- Syllabus was associated with teacher.

<Table 12> Manage Syllabus OC1

Contract OC2. selectClass
Operation :
selectClass(teacerID,classindex)
Cross Reference :
Manage Syllabus
Precondition :
- Class instance was created.

Postcondition :
- Syllabus was associated with class.

<Table 13> Manage Syllabus OC2

Contract OC3. SelectWeek
Operation :
SelectWeek(teacherID, week)
Cross Reference :
Manage Syllabus
Precondition :
- Managing Syllabus is underway.

Postcondition :
- None.

<Table 14> Manage Syllabus OC3

Contract OC4. enterWeeklyContent
Operation :
enterWeeklyContent(teacherID,week,contents, testname, testDate, pass-throughScore)
Cross Reference :
Manage Syllabus
Precondition :
- Managing Syllabus is underway.

Postcondition :
- Test instance was created.
- Syllabus was associated with test.
- test.name was initialized to testname.
- test.Date was initialized to testDate.
- test.pass-throughScore was initialized to pass-throughScore.

<Table 15> Manage Syllabus OC4

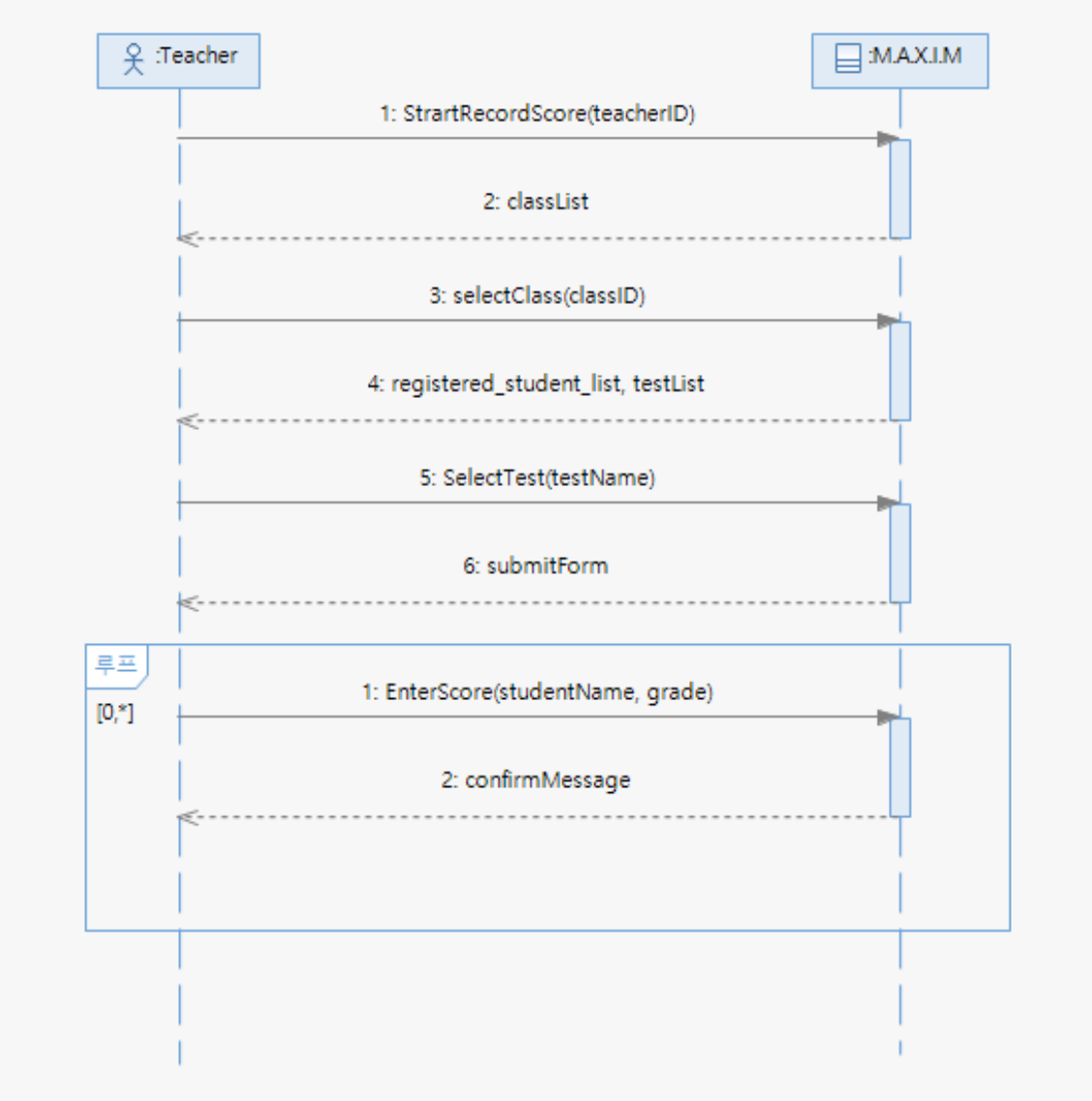
Contract OC5. submitSyllabus
Operation :
submitSyllabus(teacherID)
Cross Reference :

Manage Syllabus **Precondition :** - Managing Syllabus is underway. **Postcondition :** - None.

<Table 16> Manage Syllabus OC5

4.4. <Use Case 4> Record Score

4.4.1. System Sequence Diagram



<Figure 7> Record Score SSD

4.4.2. Operation Contract

Contract OC1. StartRecordScore **Operation :** StartRecordScore(teacherID) **Cross Reference :** Record Score

Precondition :

- Teacher is already signed up and in.

Postcondition :

- None.

<Table 17> Record Score OC1

Contract OC2. GetTestList

Operation :

GetTestList(classID)

Cross Reference :

Record Score

Precondition :

- Recording Score is underway.
- Class was already associated with Syllabus.

Postcondition :

- None.

<Table 18> Record Score OC2

Contract OC3. SelectTest

Operation :

SelectTest(classID, testname)

Cross Reference :

Record Score

Precondition :

- Recording Score is underway.
- Test was already created.

Postcondition :

- None.

<Table 19> Record Score OC3

Contract OC4. EnterScore

Operation :

EnterScore(studentName, grade)

Cross Reference :

Record Score

Precondition :

- Recording Score is underway.

Postcondition :

- Test_Score instance *ts* is created.
- *ts* is associated with Test.
- *ts*.registeredStudentName is initialized to studentName.
- *ts*.grade is initialized to grade.

<Table 20> Record Score OC4

4.5. <Use Case 5> Manage Profile

4.5.1. System Sequence Diagram

<Figure 8> Manage Profile SSD

4.5.2. Operation Contract

Contract OC1. StartManageProfile **Operation :** StartManageProfile(StudentID) **Cross Reference :** Manage Profile **Precondition :** - Student is signed up & in. **Postcondition :** - NewStudent_Profile instance *p* was created. - *p* was associated with NewStudent. - *p*.id was initialized to StudentID.

<Table 21> Manage Profile OC1

Contract OC2. SetProfile **Operation :** SetProfile(name, age, address) **Cross Reference :**

Manage Profile

Precondition :

- Student make up his profile.
- Managing Profile is underway.

Postcondition :

- *p*.name was initialized to name.
- *p*.age was initialized to age.
- *p*.residence was initialized to address.

<Table 22> Manage Profile OC2

Contract OC3. SetPreference

Operation :

SetPreference(numberOfStudents, CareerOfTeacher, tuitionFee, ageDistributionOfStudents)

Cross Reference :

Manage Profile

Precondition :

- Managing Profile is underway.

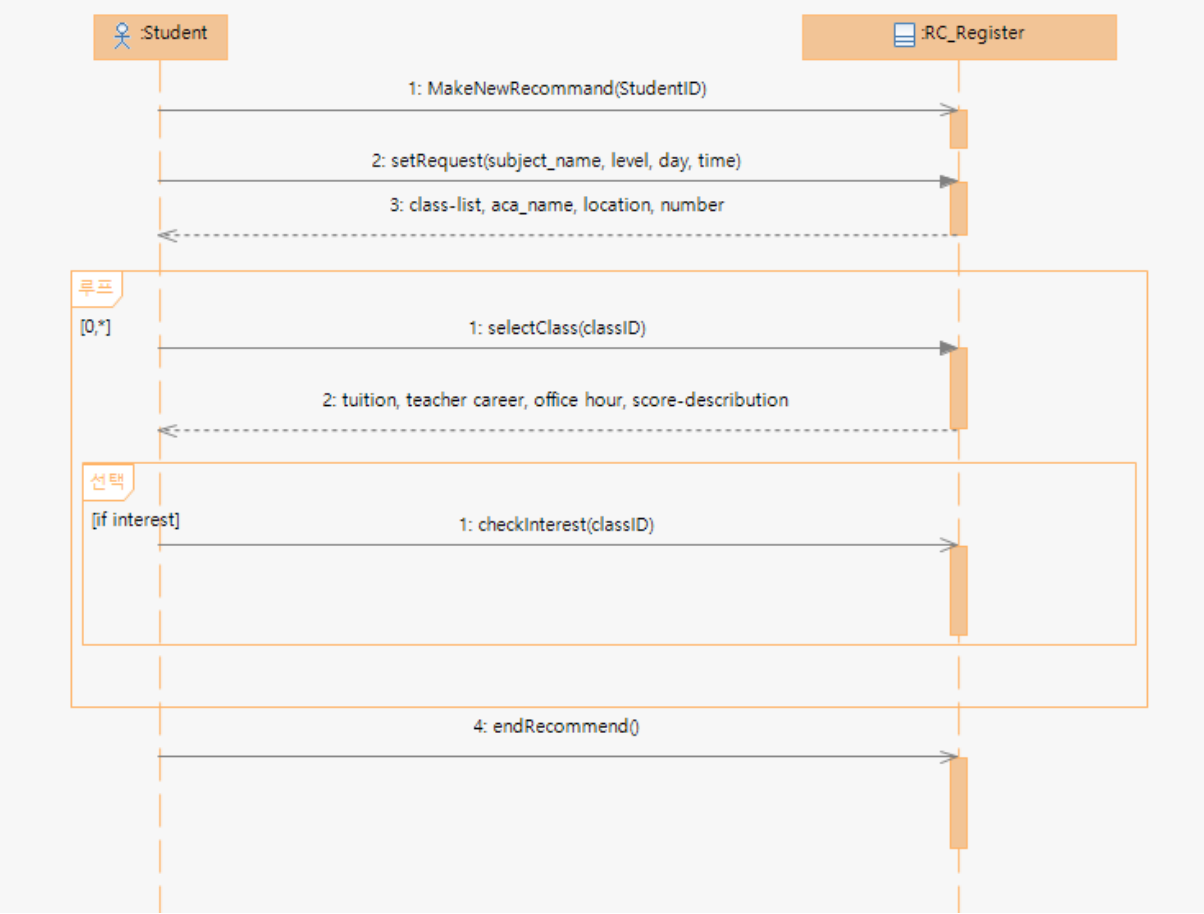
Postcondition :

- Profile_preference instance *pp* was created.
- *pp* was associated with *p*.
- *pp*.number_of_studnets was initialized to numberOfStudents.
- *pp*.career_of_teacher was initialized to CareerOfTeacher.
- *pp*.tuition was initialized to tuitionFee.
- *pp*.age_distribution was initialized to ageDistributionOfStudents.

<Table 23> Manage Profile OC3

4.6. <Use Case 6> Get Recommendation

4.6.1. System Sequence Diagram



<Figure 9> Get Recommendation SSD

4.6.2. Operation Contract

Contract OC1. MakeNewRecommendation **Operation :** MakeNewRecommendation(StudentID) **Cross Reference :** Get Recommendation **Precondition :** - M.A.X.I.M. was already created. **Postcondition :** - Request instance *rq* was created - Recommendation instance *r* was created. - *r* was associated with M.A.X.I.M. - *rq* and associated with *r*. - *rq*.id was randomly initialized. - *r*.id was initialized to StudentID.

<Table 24> Get Recommendation OC1

Contract OC2. SetRequest **Operation :**

SetRequest(subject_name, level, day, time)

Cross Reference :

Get Recommendation

Precondition :

- There is Recommendation underway.
- Profile_preference was already created.
- Newstudent_Profile was already created.
- Class was already created.
- Test_Score was already created.

Postcondition :

- *rq*.subject_name was initialized to subject_name.
- *rq*.level was initialized to level.
- *rq*.day was initialized to day.
- *rq*.time was initialized to time.
- *r* was associated with Class.
- *r* was associated with Newstudent_Profile
- *r* was associated with Test_Score.
- *r*.clsList was initialized to recommend classes by M.A.X.I.M.

<Table 25> Get Recommendation OC2

Contract OC3. SelectClass

Operation :

SelectClass(ClassID)

Cross Reference :

Get Recommendation

Precondition :

- There is Recommendation underway.

Postcondition :

- None.

<Table 26> Get Recommendation OC3

Contract OC4. CheckInterest

Operation :

CheckInterest(ClassID)

Cross Reference :

Get Recommendation

Precondition :

- There is Recommendation underway.
- Manager was already created.
- Manager was already associated with Class.

Postcondition :

- Class.like was updated.

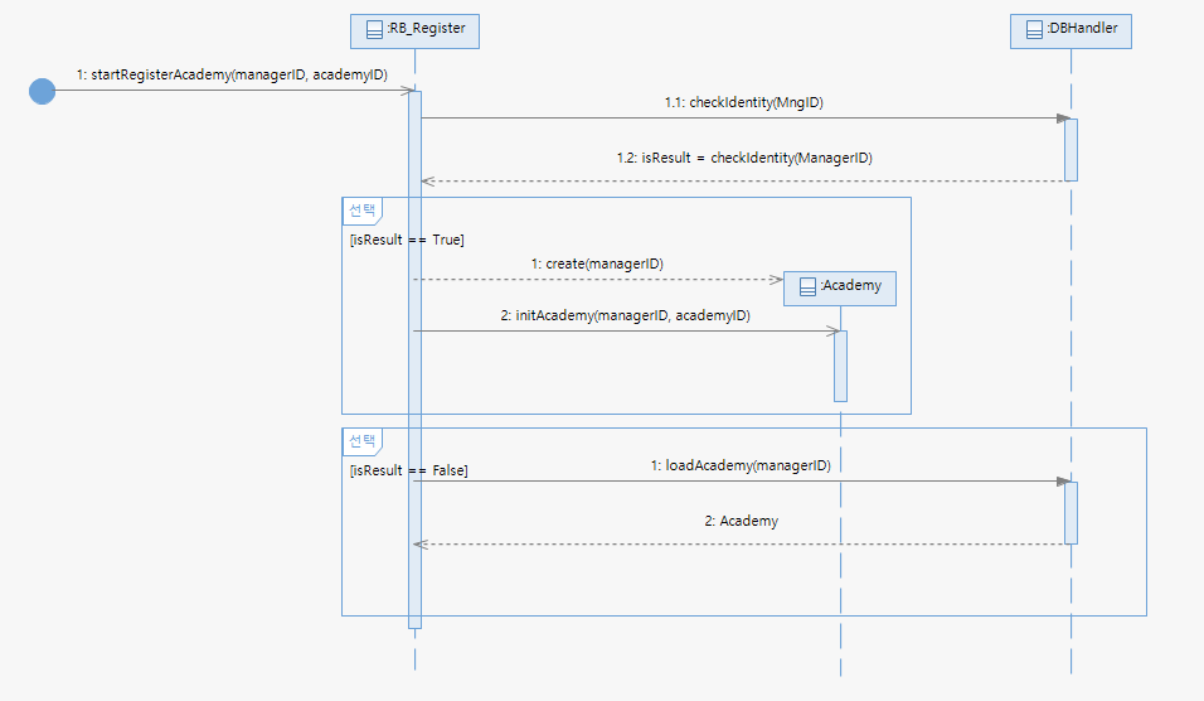
<Table 27> Get Recommendation OC4

5. Design Model

5.1. <Use Case 1> Register Basic Academy information Realization

5.1.1. Operation1 StartRegisterAcademy

5.1.1.1. Design Sequence Diagram



<Figure 10> <use case 1> Operation1. DSD

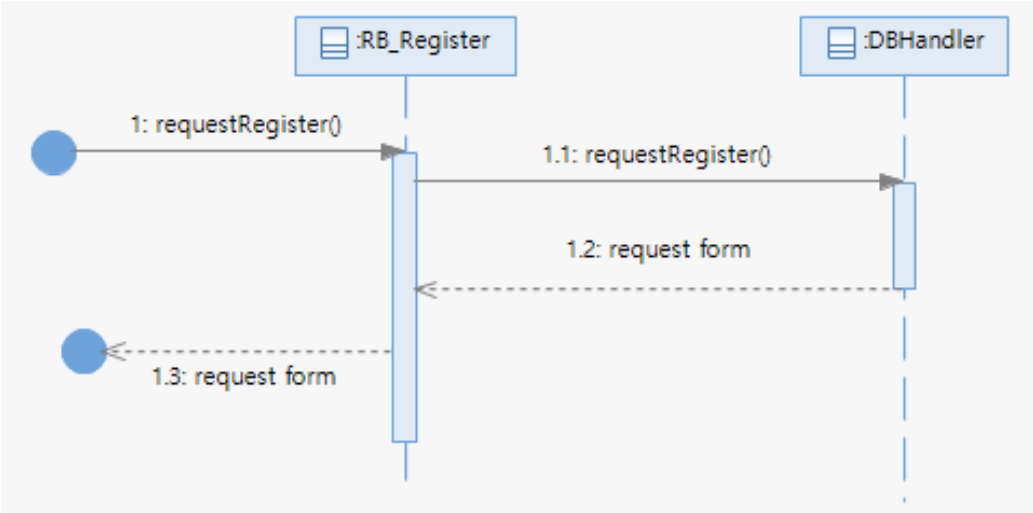
5.1.1.2. GRASP Pattern

Creator	RB_Register creates an Academy
Information Expert	None.
Low Coupling	RB_Register does not need to know how to check MangerID and AcademyID.
High Cohesion	RB_Register has only one responsibility about creating Academy.
Controller	RB_Register represents a handler of all system events occurred by Academy.

<Table 28> <use case 1> operation1. GRASP pattern

5.1.2. Operation2 RequestRegister

5.1.2.1. Design Sequence Diagram



<Figure 11> <use case1> Operation2. DSD

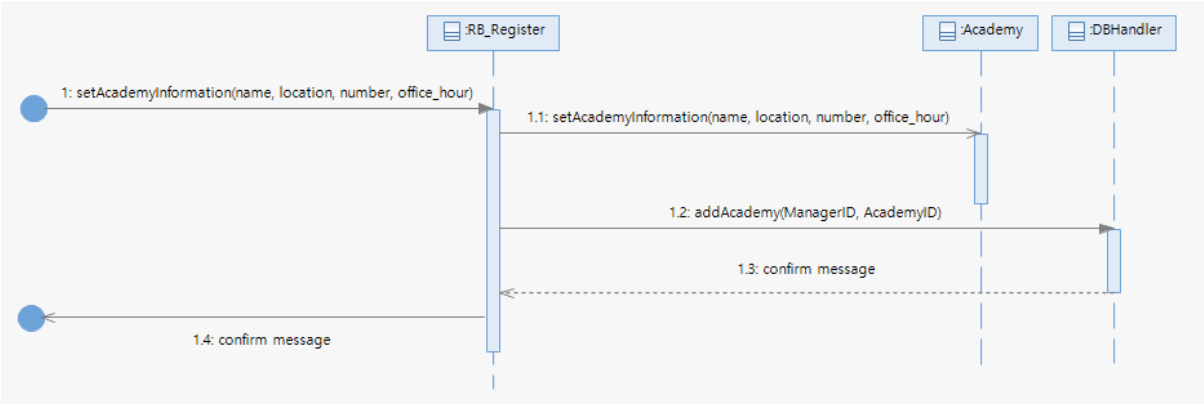
5.1.2.2. GRASP pattern

Creator	None
Information Expert	None
Low Coupling	RB_Register does not need to know how to request Register form
High Cohesion	RB_Register has only one responsibility about request Register form
Controller	RB_Register represents a handler of all system events occurred when manager starts registration.

<Table 29> <use case1> operation1. GRASP pattern

5.1.3. Operation3 SetAcaInfo

5.1.3.1. Design Sequence Diagram



<Figure 12> <use case1> Operation3. DSD

5.1.3.2. GRASP Pattern

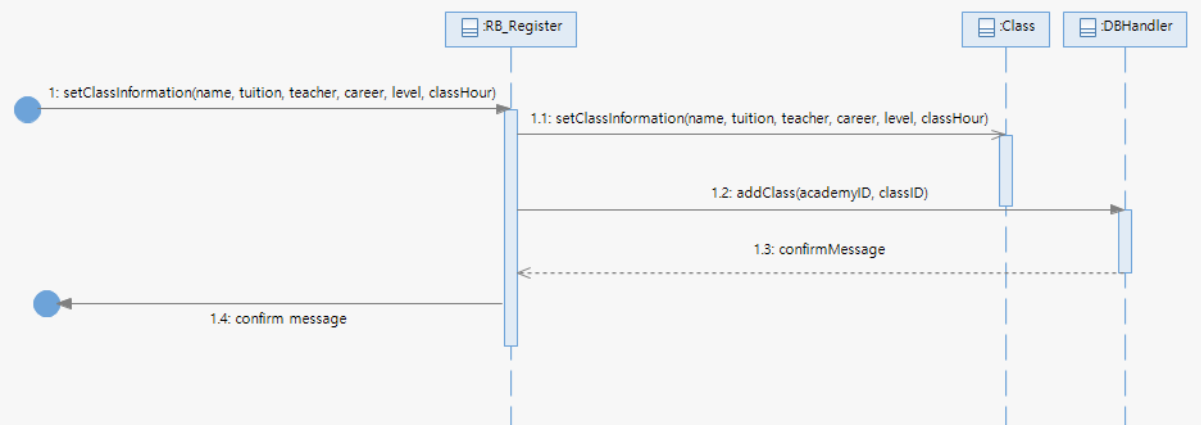
Creator	None
Information Expert	Academy knows the name, location, number and office_hour of Academy.

Low Coupling	RB_Register does not need to know how to set Academy Information to Academy instance.
High Cohesion	RB_Register has only one responsibility about setting and adding academy information to Academy.
Controller	RB_Register represents a handler of all system events occurred by Academy.

<Table 30> <use case 1> operation3. GRASP pattern

5.1.4. Operation4 SetClassInfo

5.1.4.1. Design Sequence Diagram



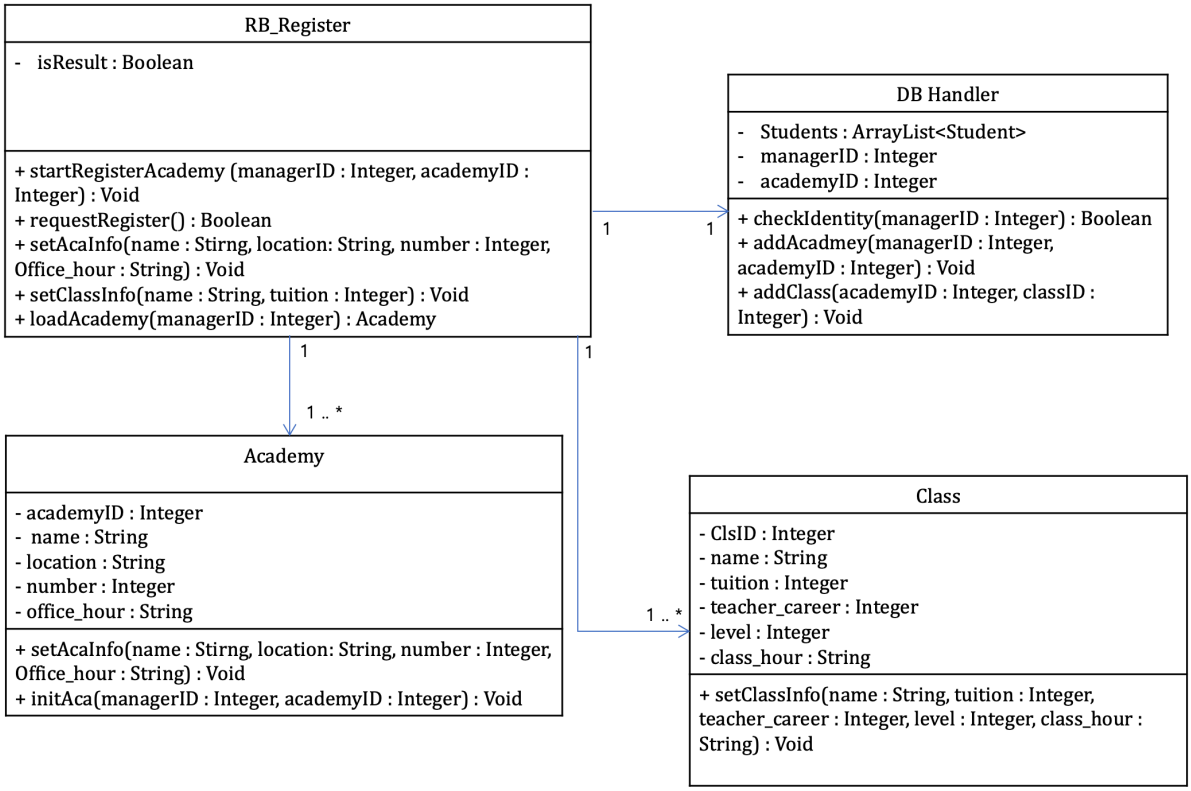
<Figure 13> <use case1> Operation4. DSD

5.1.4.2. GRASP Pattern

Creator	None
Information Expert	Class knows the name, tuition, teacher_career and level and class_hour of Class.
Low Coupling	RB_Register does not need to know how to set Class Information to Class instance.
High Cohesion	RB_Register has only one responsibility about setting and adding class information to Class.
Controller	RB_Register represents a handler of all system events occurred by Class.

<Table 31> <use case 1> operation4. GRASP pattern

5.1.5. Combined DCD

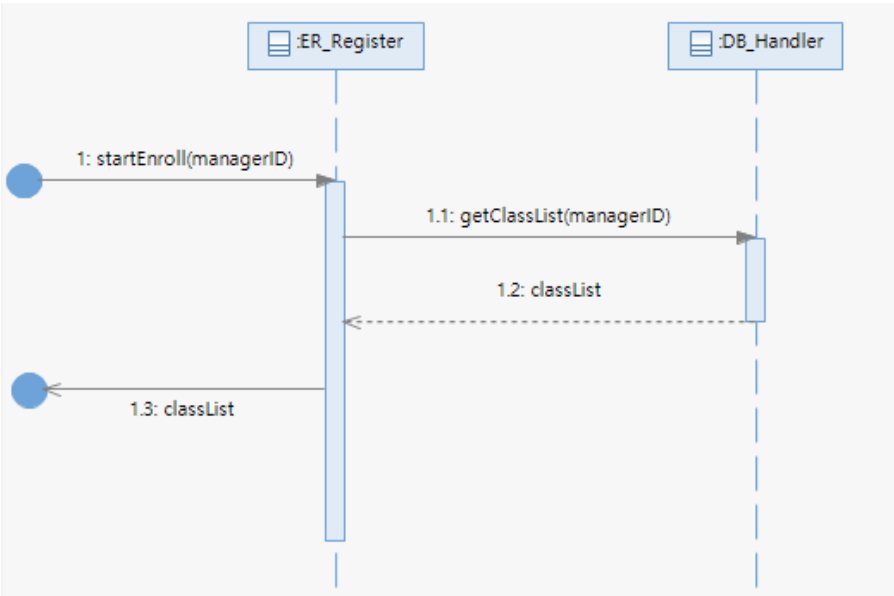


<Figure 14> <UseCase1> Combined DCD

5.2. <Use Case 2> Enroll Student Realization

5.2.1. Operation1 StartEnroll

5.2.1.1. Design Sequence Diagram



<Figure 15> <use case2> Operation1. DSD

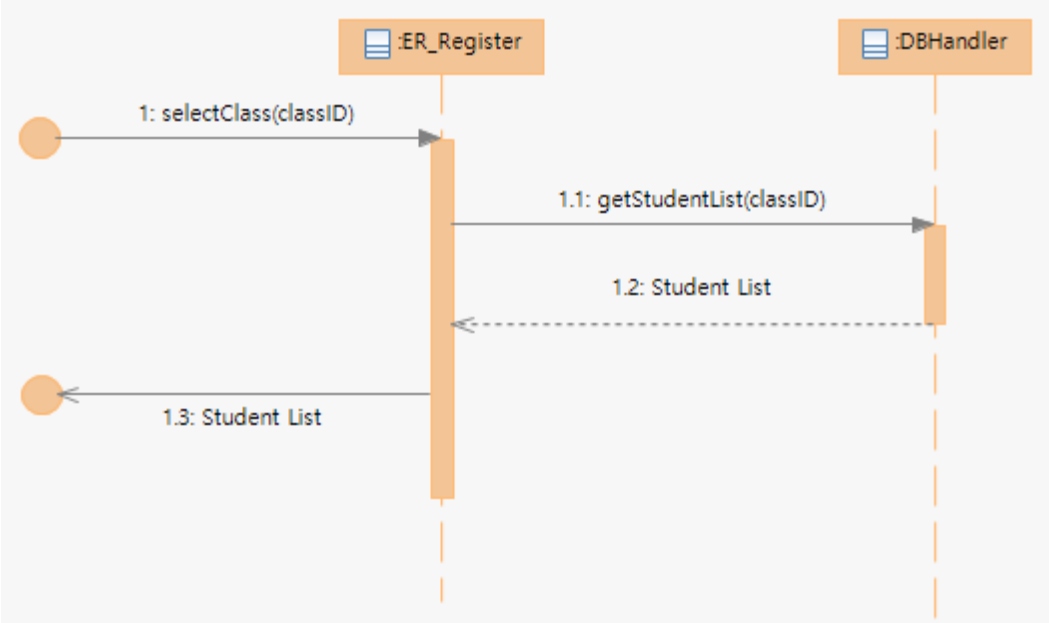
5.2.1.2. GRASP Pattern

Creator	None.
Information Expert	Manager knows the ManagerID.
Low Coupling	ER Register does not need to know how to bring classlist..
High Cohesion	ER Register has only one responsibility about getting classlist..
Controller	ER Register represents a handler of all system events occurred by Manager.

<Table 32> <use case 2> operation1. GRASP pattern

5.2.2. Operation2 SelectClass

5.2.2.1. Design Sequence Diagram



<Figure 16> <use case2> Operation2. DSD

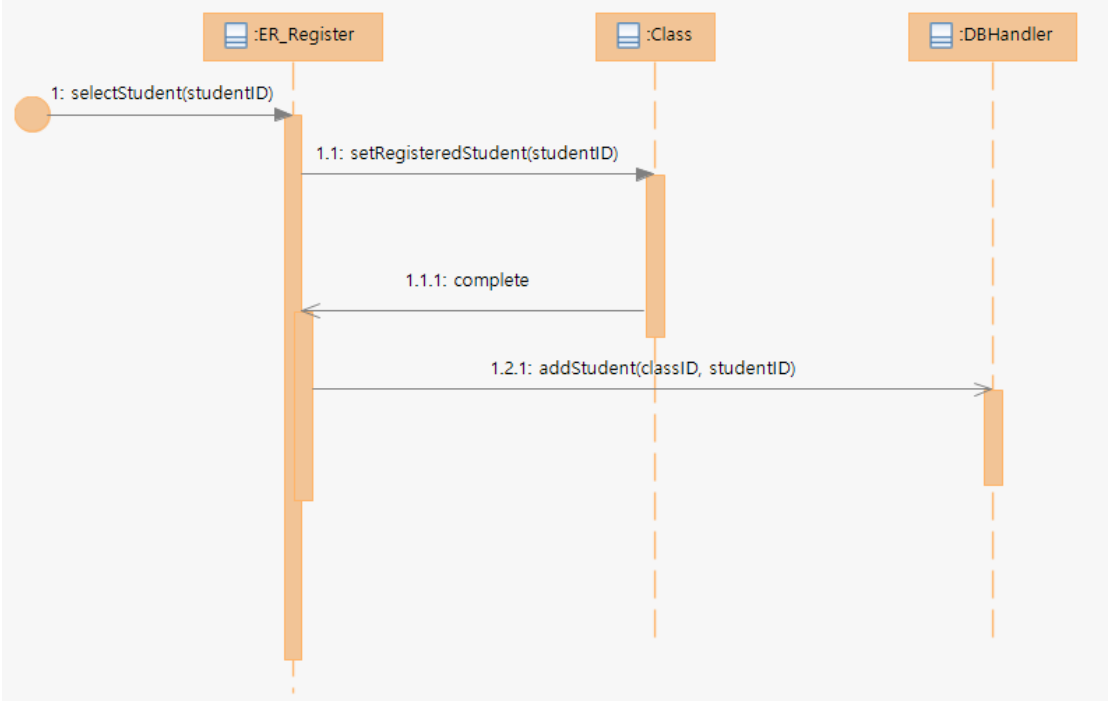
5.2.2.2. GRASP Pattern

Creator	None.
Information Expert	Manager knows the ClassID.
Low Coupling	ER_Register does not need to know how to bring studentlist..
High Cohesion	ER_Register has only one responsibility about getting studentlist..
Controller	ER_Register represents a handler of all system events occurred by Manager.

<Table 33> <use case 2> operation2. GRASP pattern

5.2.3. Operation3 SelectStudent

5.2.3.1. Design Sequence Diagram



<Figure 17> <use case2> Operation3. DSD

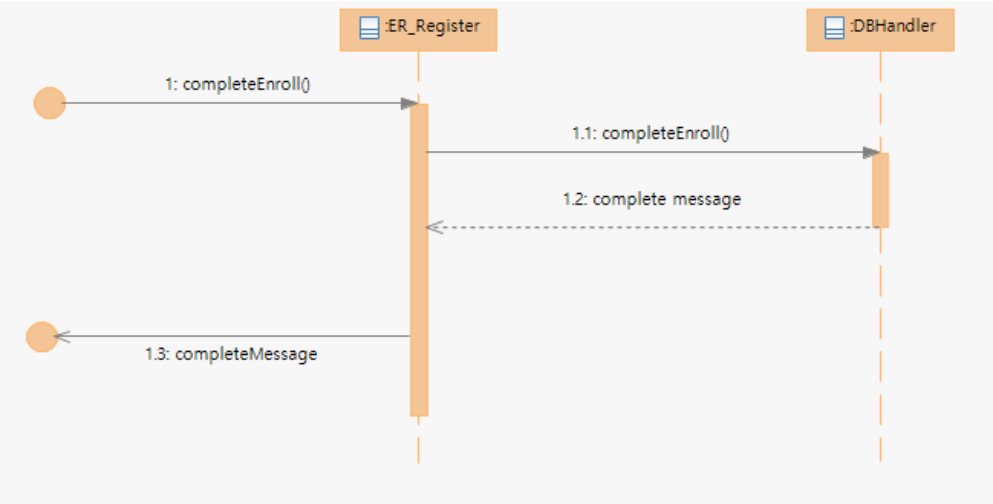
5.2.3.2. GRASP Pattern

Creator	None.
Information Expert	Manager knows the StudentID.
Low Coupling	ER_Register does not need to know how to set registered student.
High Cohesion	ER_Register has only one responsibility about registered student.
Controller	ER_Register represents a handler of all system events occurred by Manager.

<Table 34> <use case 2> operation3. GRASP pattern

5.2.4. Operation4 CompleteEnroll Realization

5.2.4.1. Design Sequence Diagram



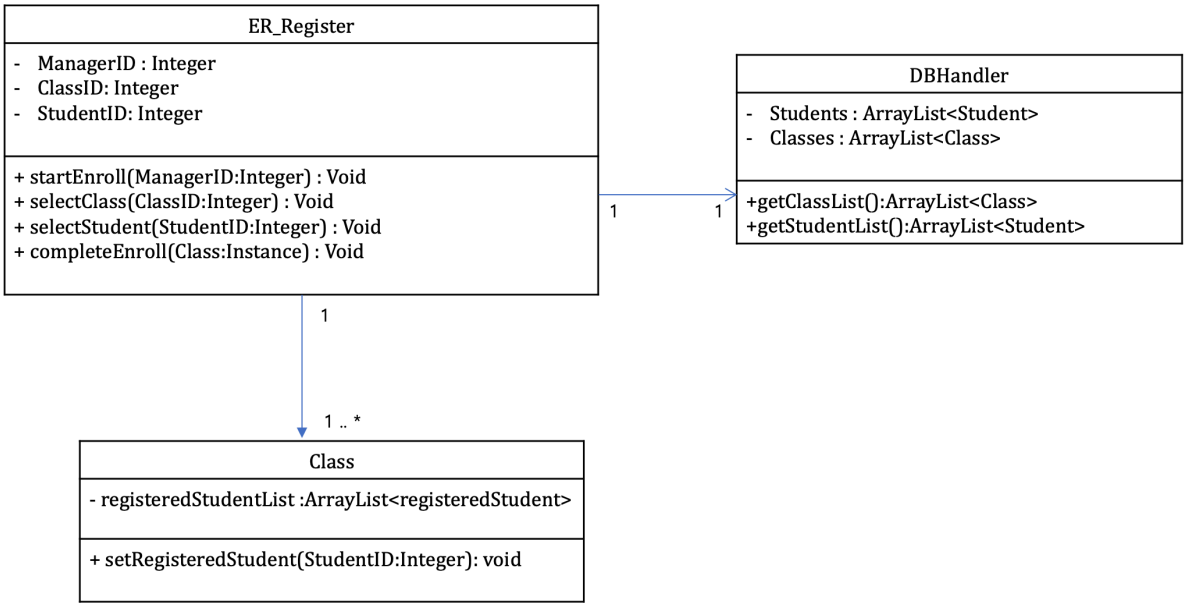
<Figure 18> <use case2> Operation4. DSD

5.2.4.2. GRASP Pattern

Creator	None.
Information Expert	Manager knows the ManagerID.
Low Coupling	ER_Register does not need to know how to update class
High Cohesion	ER_Register has only one responsibility about delivering class.
Controller	ER_Register represents a handler of all system events occurred by Manager.

<Table 35> <use case 2> operation4. GRASP pattern

5.2.5. Combined DCD

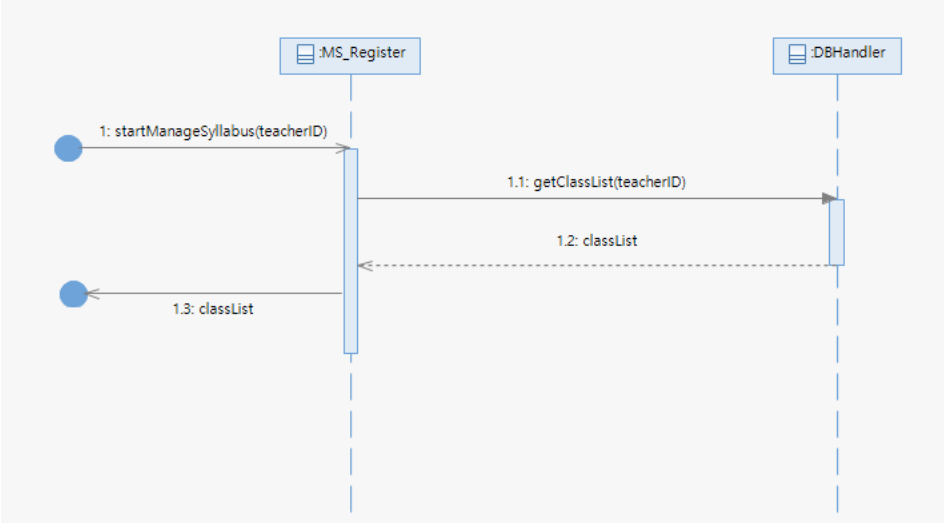


<Figure 19> <use case2> combined DCD

5.3. <Use Case 3> Manage Syllabus Realization

5.3.1. Operation1 StartManageSyllabus

5.3.1.1. Design Sequence Diagram



<Figure 20> <use case3> Operation1. DSD

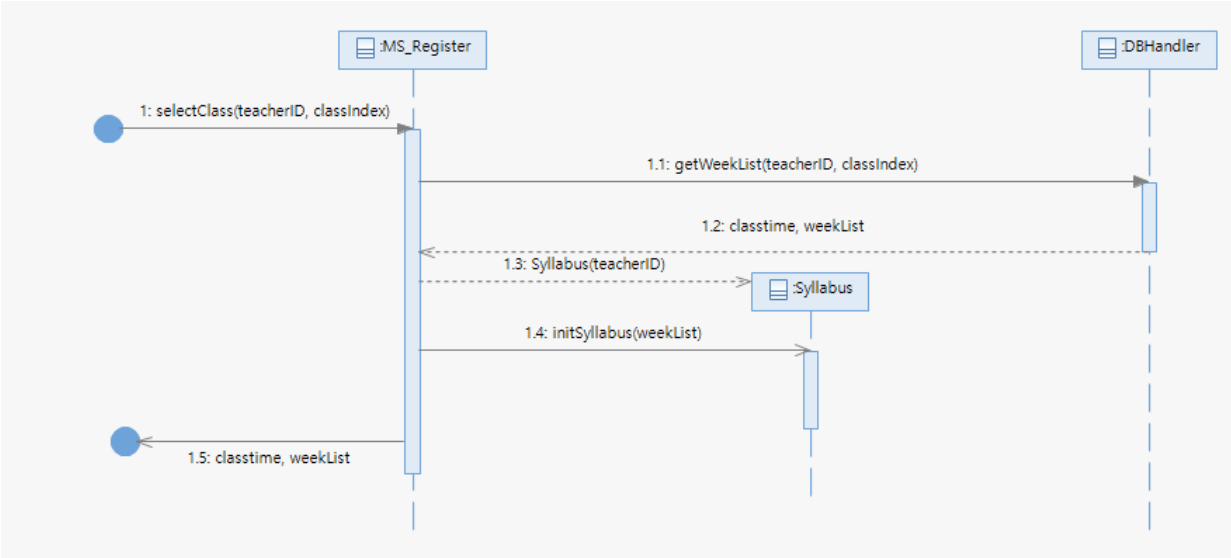
5.3.1.2. GRASP Pattern

Creator	None.
Information Expert	Teacher knows the teacherID.
Low Coupling	MS_Register does not need to know how to bring classlist.
High Cohesion	MS_Register has only one responsibility about getting classlist.
Controller	MS_Register represents a handler of all system events occurred by Teacher.

<Table 36> <use case3> operation1. GRASP pattern

5.3.2. Operation2 selectClass

5.3.2.1. Design Sequence Diagram



<Figure 21> <use case3> Operation2. DSD

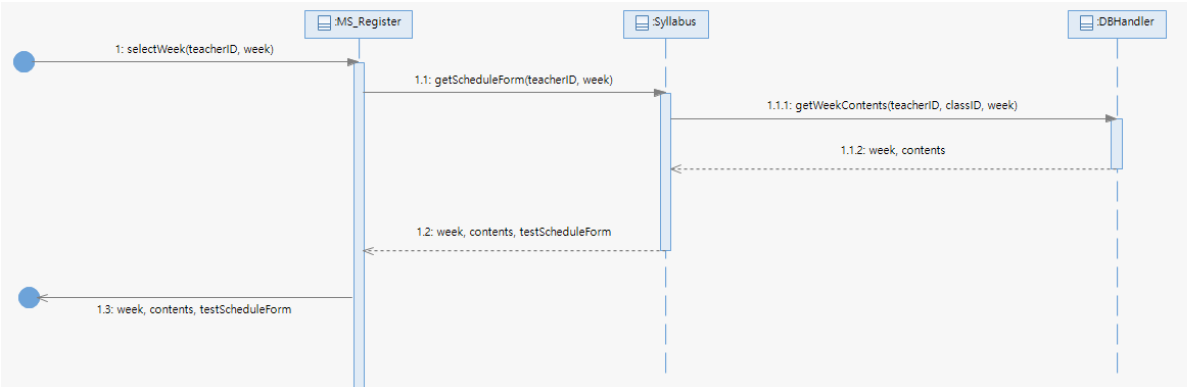
5.3.2.2. GRASP Pattern

Creator	None.
Information Expert	Teacher knows the teacherID and class index.
Low Coupling	MS_Register does not need to know how to bring class time and week List
High Cohesion	MS_Register has only one responsibility about getting time and week List
Controller	MS_Register represents a handler of all system events occurred by Teacher.

<Table 37> <use case3> operation2. GRASP pattern

5.3.3. Operation3 selectWeek

5.3.3.1. Design Sequence Diagram



<Figure 22> <use case3> Operation3. DSD

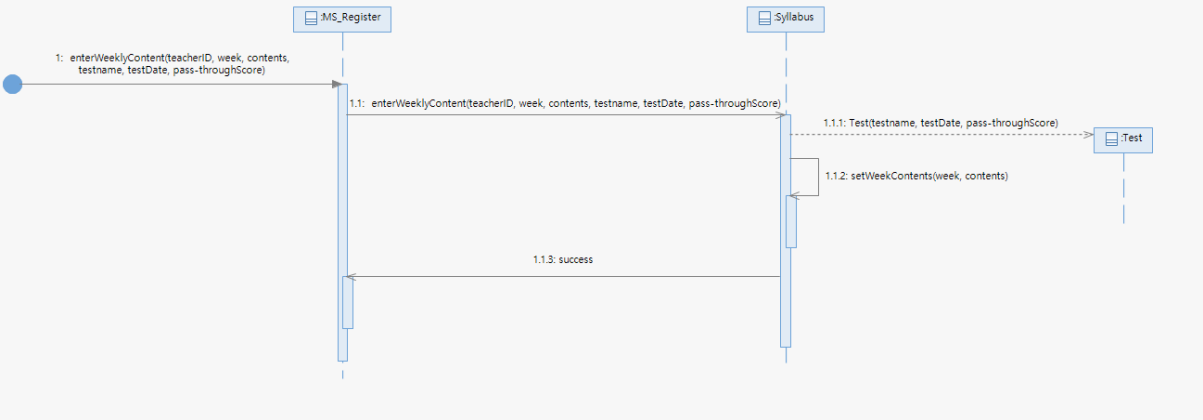
5.3.3.2. GRASP Pattern

Creator	None.
Information Expert	Teacher knows the teacherID and Week.
Low Coupling	MS_Register does not need to know how to bring week, contents and test Schedule Form
High Cohesion	MS_Register has only one responsibility about getting week, contents and test Schedule Form
Controller	MS_Register represents a handler of all system events occurred by Teacher.

<Table 38> <use case3> operation3. GRASP pattern

5.3.4. Operation4 enterWeeklyContent

5.3.4.1. Design Sequence Diagram



<Figure 23> <use case3> Operation4. DSD

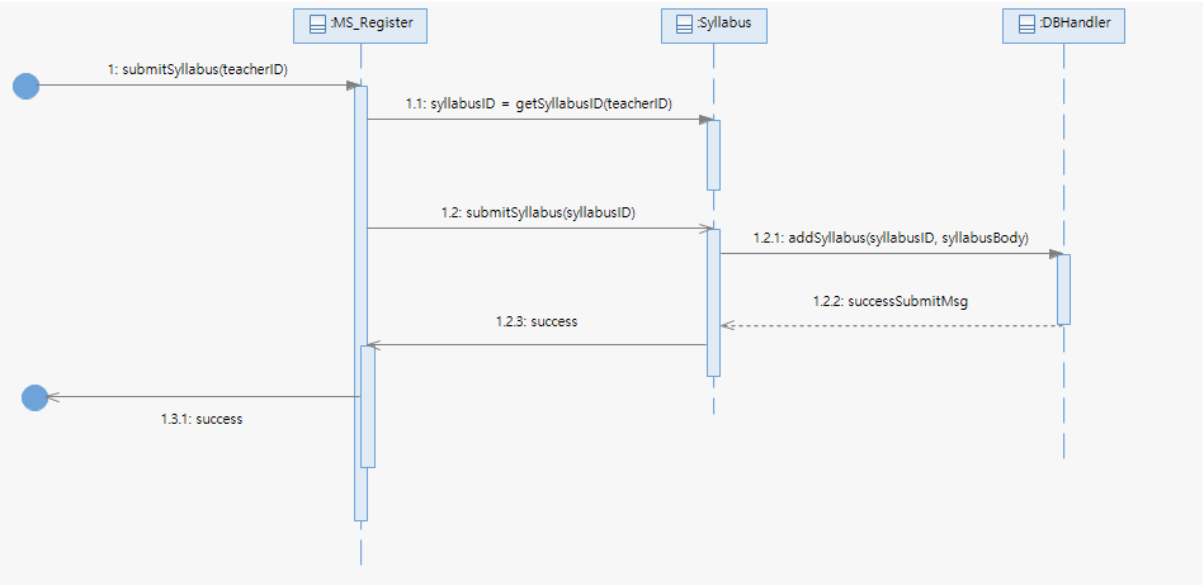
5.3.4.2. GRASP Pattern

Creator	None.
Information Expert	Teacher knows the contents of Week
Low Coupling	MS_Register does not need to know how to set contents in Syllabus.
High Cohesion	MS_Register has only one responsibility about setting requirement.
Controller	MS_Register represents a handler of all system events occurred by Teacher.

<Table 39> <use case3> Operation4. GRASP pattern

5.3.5. Operation5 submitSyllabus

5.3.5.1. Design Sequence Diagram



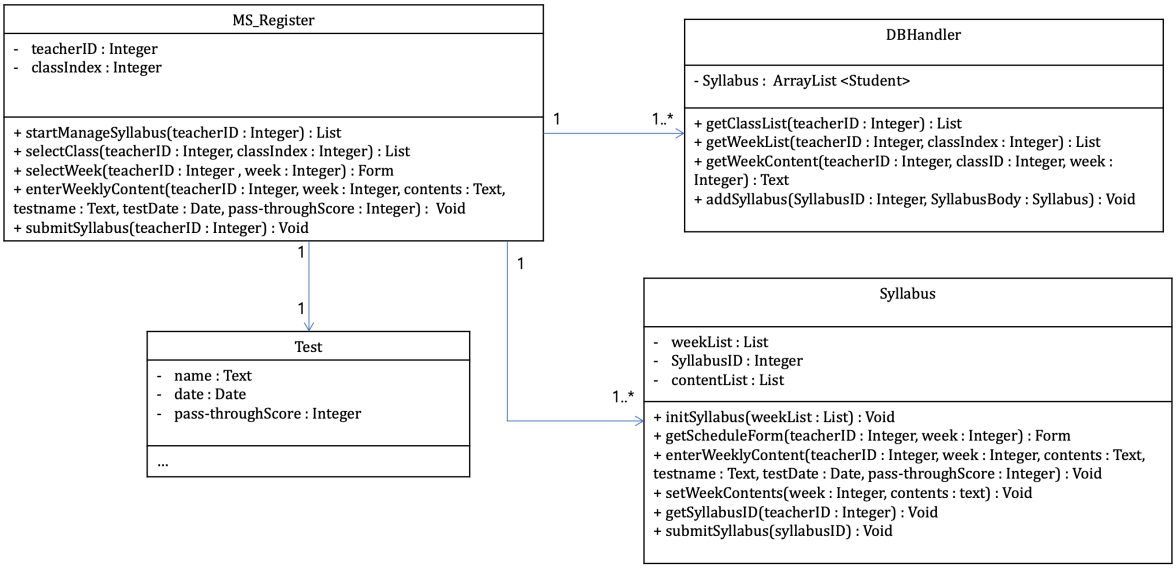
<Figure 24> <use case3> Operation5. DSD

5.3.5.2. GRASP Pattern

Creator	None.
Information Expert	None.
Low Coupling	MS Register does not need to know how to update Syllabus.
High Cohesion	MS Register has only one responsibility about updating requirement.
Controller	MS Register represents a handler of all system events occurred by Teacher.

<Table 40> <use case3> operation5. GRASP pattern

5.3.6. Combined DCD

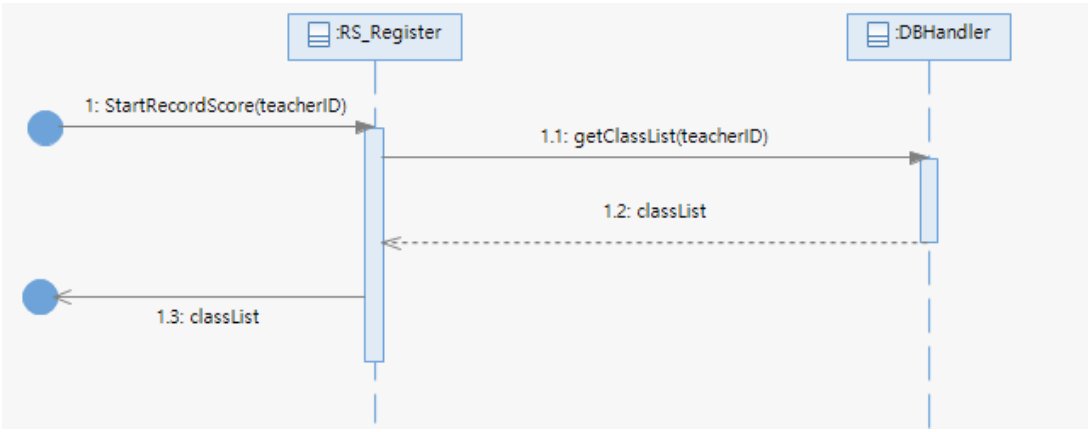


<Figure 25> <use case3> combined DCD

5.4. <Use Case 4> Record Score

5.4.1. Operation1 StartRecordScore

5.4.1.1. Design Sequence Diagram



<Figure 26> <use case4> Operation1. DSD

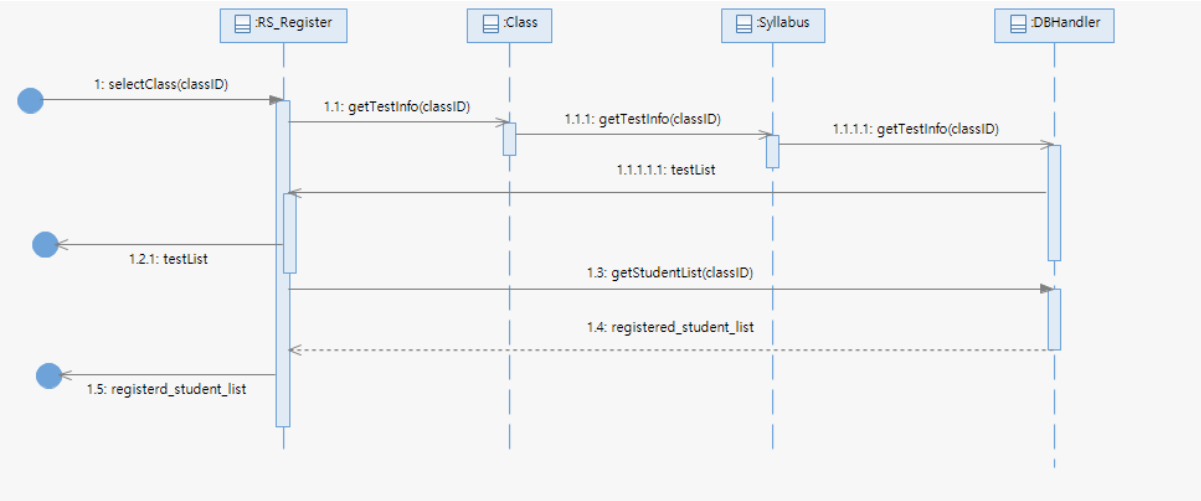
5.4.1.2. GRASP Pattern

Creator	None.
Information Expert	Teacher knows the teacher ID.
Low Coupling	RS_Register does not need to how to get class list.
High Cohesion	RS_Register has responsibility about getting class list.
Controller	RS_Register represents a handler of all system events occurred by Teacher.

<Table 41> <use case 4> operation1. GRASP pattern

5.4.2. Operation2 SelectClass

5.4.2.1. Design Sequence Diagram



<Figure 27> <use case4> Operation2. DSD

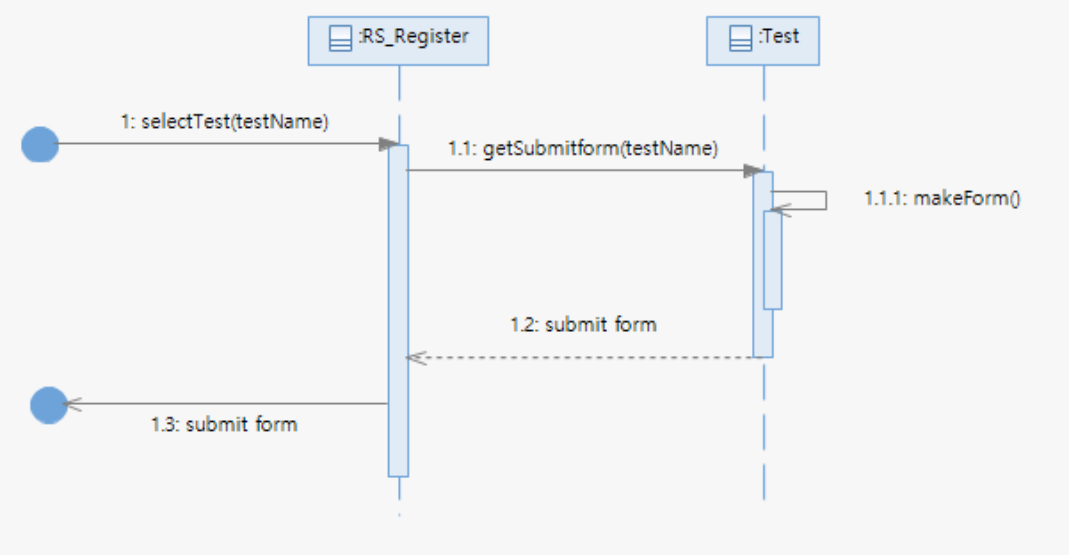
5.4.2.2. GRASP Pattern

Creator	None.
Information Expert	Teacher knows the Class ID.
Low Coupling	RS_Register does not need to how to get test list and registered student list.
High Cohesion	RS_Register has responsibility about getting test list and registered student list.
Controller	RS_Register represents a handler of all system events occurred by Teacher.

<Table 42> <use case 4> operation2. GRASP pattern

5.4.3. Operation3 SelectTest

5.4.3.1. Design Sequence Diagram



<Figure 28> <use case4> Operation3. DSD

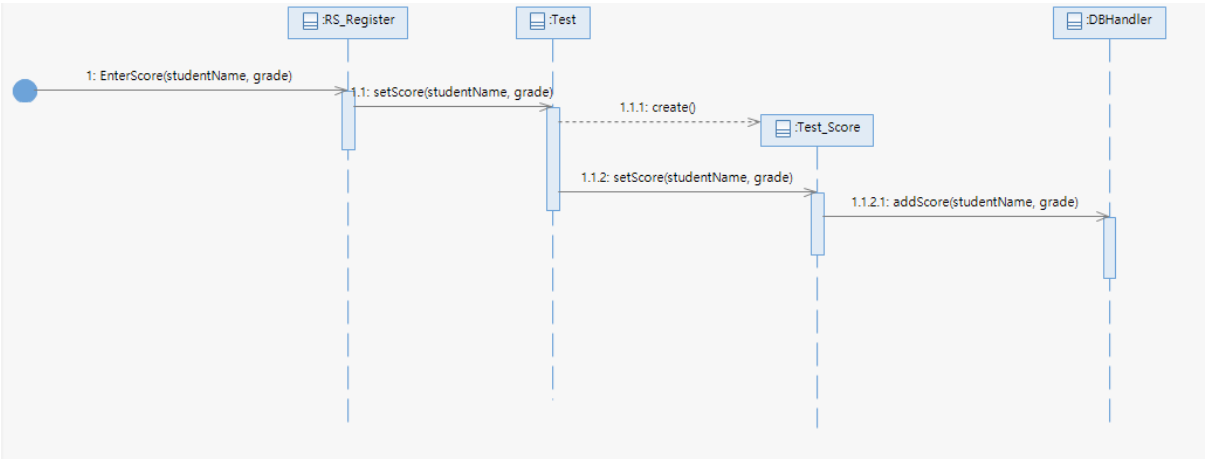
5.4.3.2. GRASP Pattern

Creator	None.
Information Expert	Teacher knows the test name.
Low Coupling	RS_Register does not need to how to make submit form.
High Cohesion	RS_Register has responsibility about getting submit form.
Controller	RS_Register represents a handler of all system events occurred by Teacher.

<Table 43> <use case 4> operation3. GRASP pattern

5.4.4. Operation4 EnterScore

5.4.4.1. Design Sequence Diagram



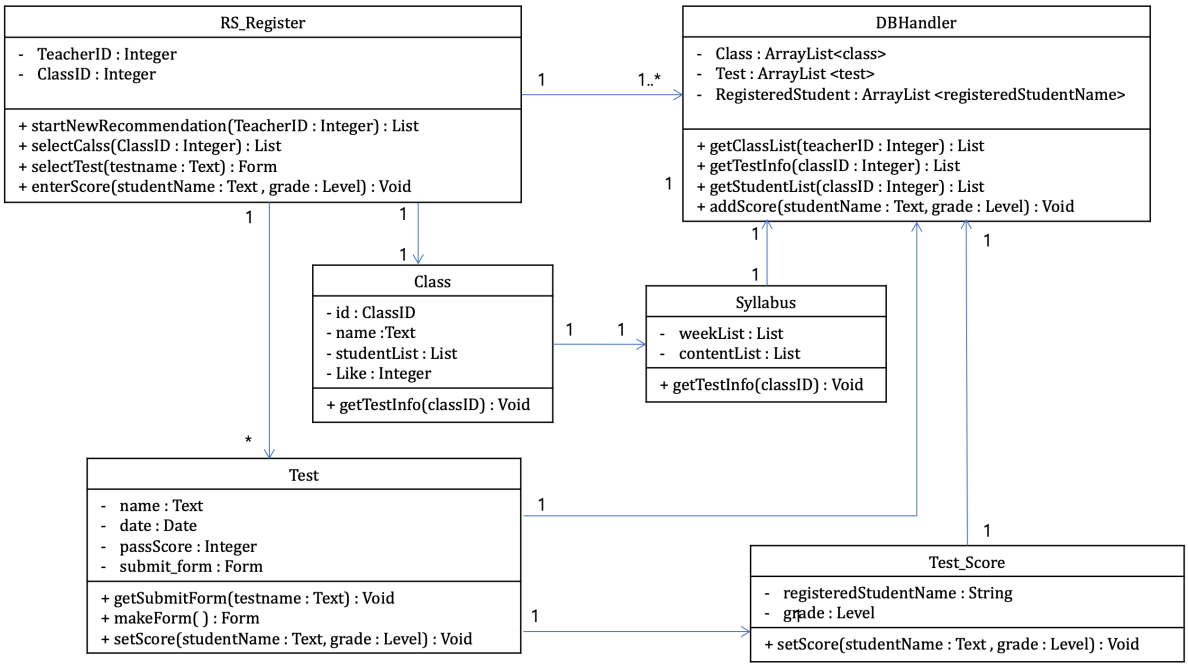
<Figure 29> <use case4> Operation4. DSD

5.4.4.2. GRASP Pattern

Creator	Test creates a Test_Score.
Information Expert	Teacher knows the student name and grade.
Low Coupling	RS_Register does not need to how to set score to Test_score.
High Cohesion	RS_Register has responsibility about recording score.
Controller	RS_Register represents a handler of all system events occurred by Teacher.

<Table 44> <use case 4> operation4. GRASP pattern

5.4.5. Combined DCD

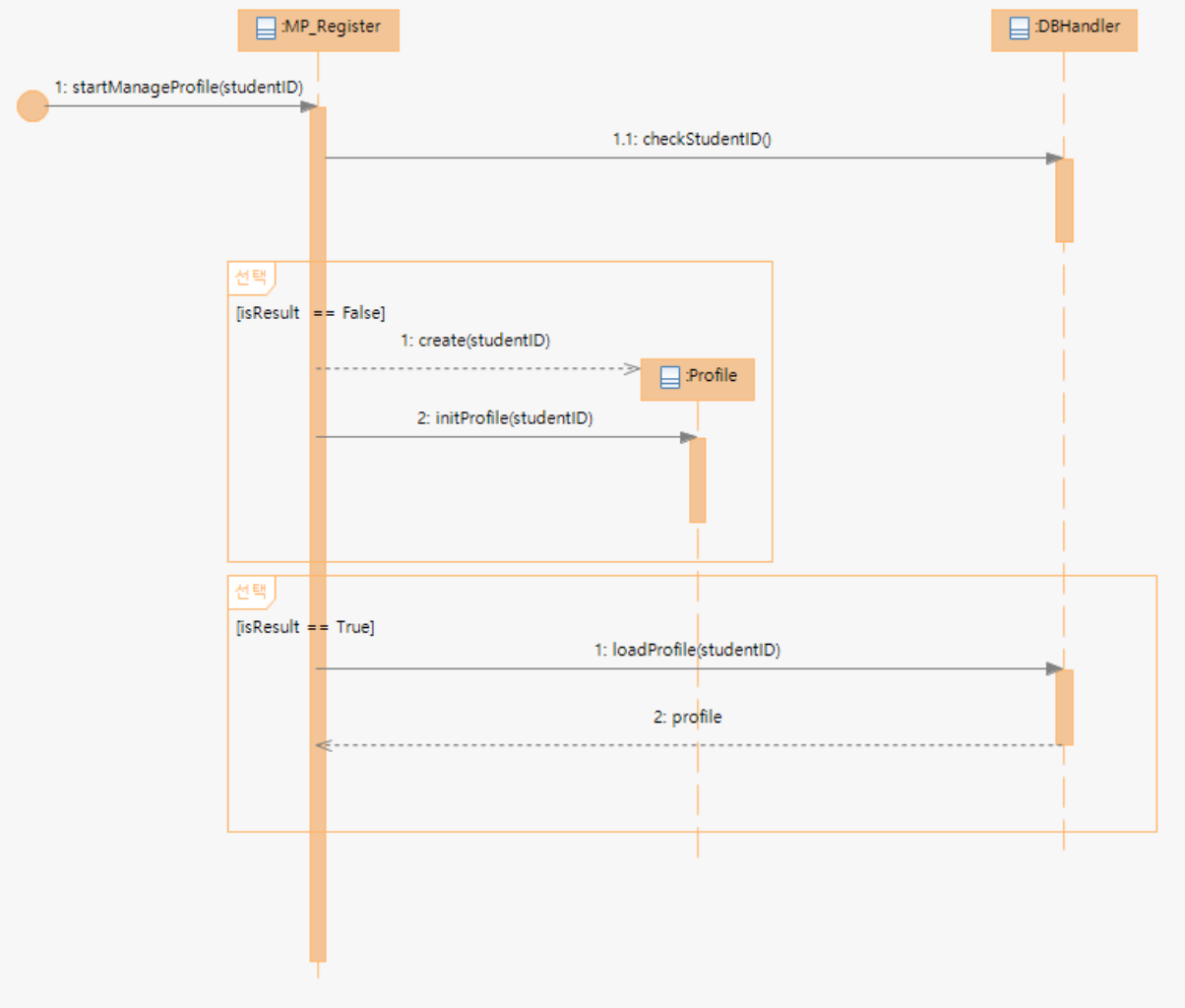


<Figure 30> <use case4> combined DCD

5.5. <Use Case 5> Manage Profile Realization

5.5.1. Operation1 StartManageProfile

5.5.1.1. Design Sequence Diagram



<Figure 31> <use case 5> Operation1. DSD

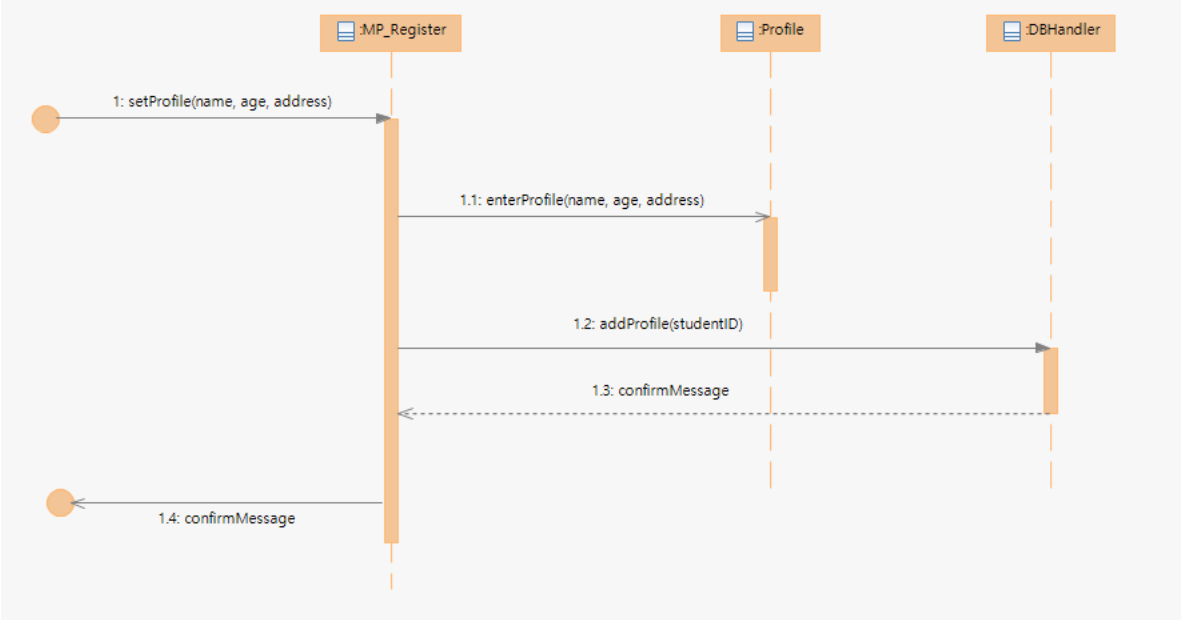
5.5.1.2. GRASP Pattern

Creator	MP_Register creates a Student_profile.
Information Expert	Student knows the ID of Student.
Low Coupling	MP_Register does not need to know how to check StudentID.
High Cohesion	MP_Register has only one responsibility about creating Student_profile.
Controller	MP_Register represents a handler of all system events occurred by Student.

<Table 45> <use case 5> operation1. GRASP pattern

5.5.2. Opearation2 SetProfile

5.5.2.1. Design Sequence Diagram



<Figure 32> <use case 5> operation2. DSD

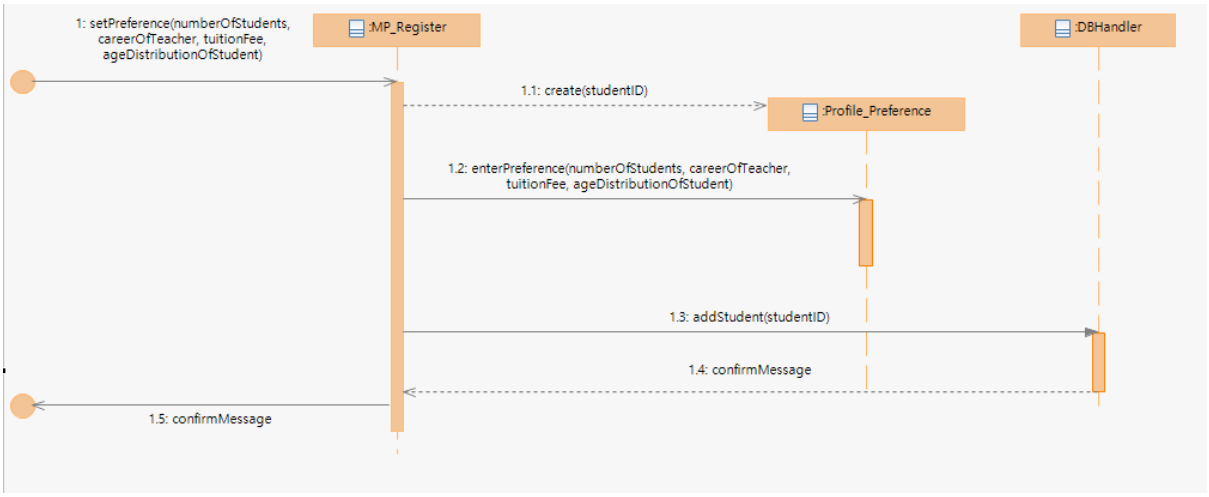
5.5.2.2. GRASP Pattern

Creator	None.
Information Expert	Student knows the profile_information.
Low Coupling	MP_Register does not need to how to set profile to Student_Profile instance.
High Cohesion	MP_Register has responsibility about setting and adding profile_information to profile.
Controller	MP_Register represents a handler of all system events occurred by Student.

<Table 46> <use case 5> operation2. GRASP pattern

5.5.3. Opearation3 SetPreference

5.5.3.1. Design Sequence Diagram



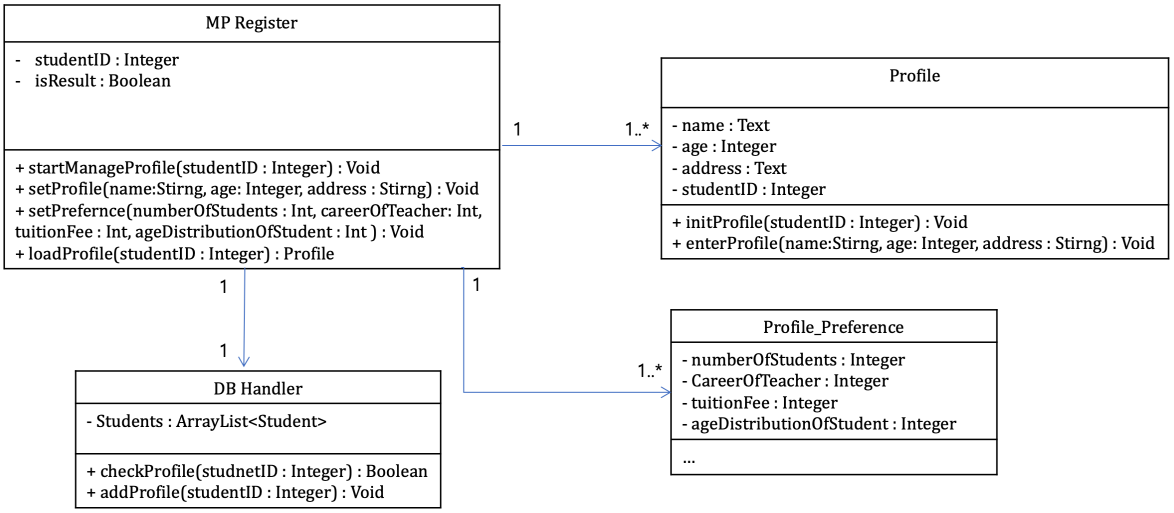
<Figure 33> <use case 5> operation3. DSD

5.5.3.2. GRASP Pattern

Creator	MP_Register creates a Profile_preference.
Information Expert	Profile knows the preference_information.
Low Coupling	MP_Register does not need to how to set preference to Profile_preference instance.
High Cohesion	MP_Register has responsibility about creating Profile_preferece. MP_Register has responsibility about setting and adding profile_information to profile.
Controller	MP_Register represents a handler of all system events occurred by Student.

<Table 47> <use case 5> operation3. GRASP pattern

5.5.4. Combined DCD

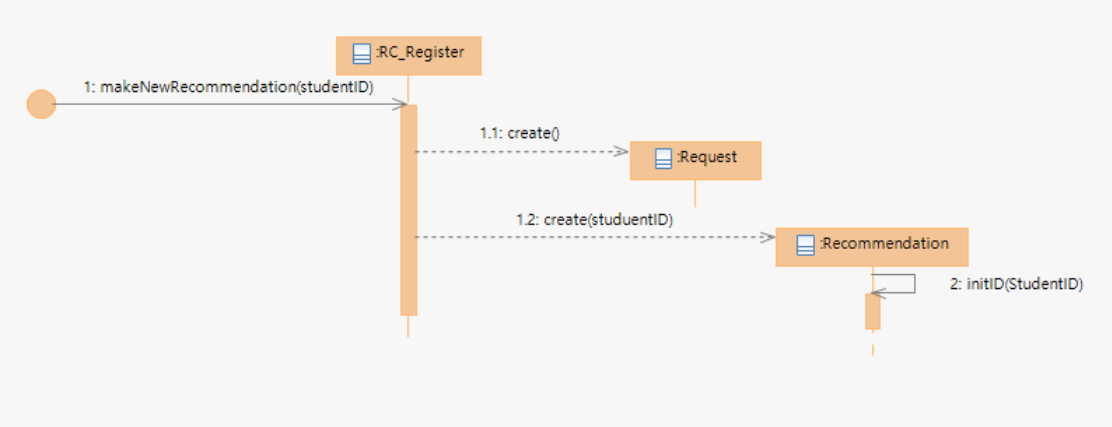


<Figure 34> <use case 5> combined DCD

5.6. <Use Case 6> Get Recommendation Realization

5.6.1. Operation1 makeNewRecommendation

5.6.1.1. Design Sequence Diagram



<Figure 35> <use case 6> operation1 DSD

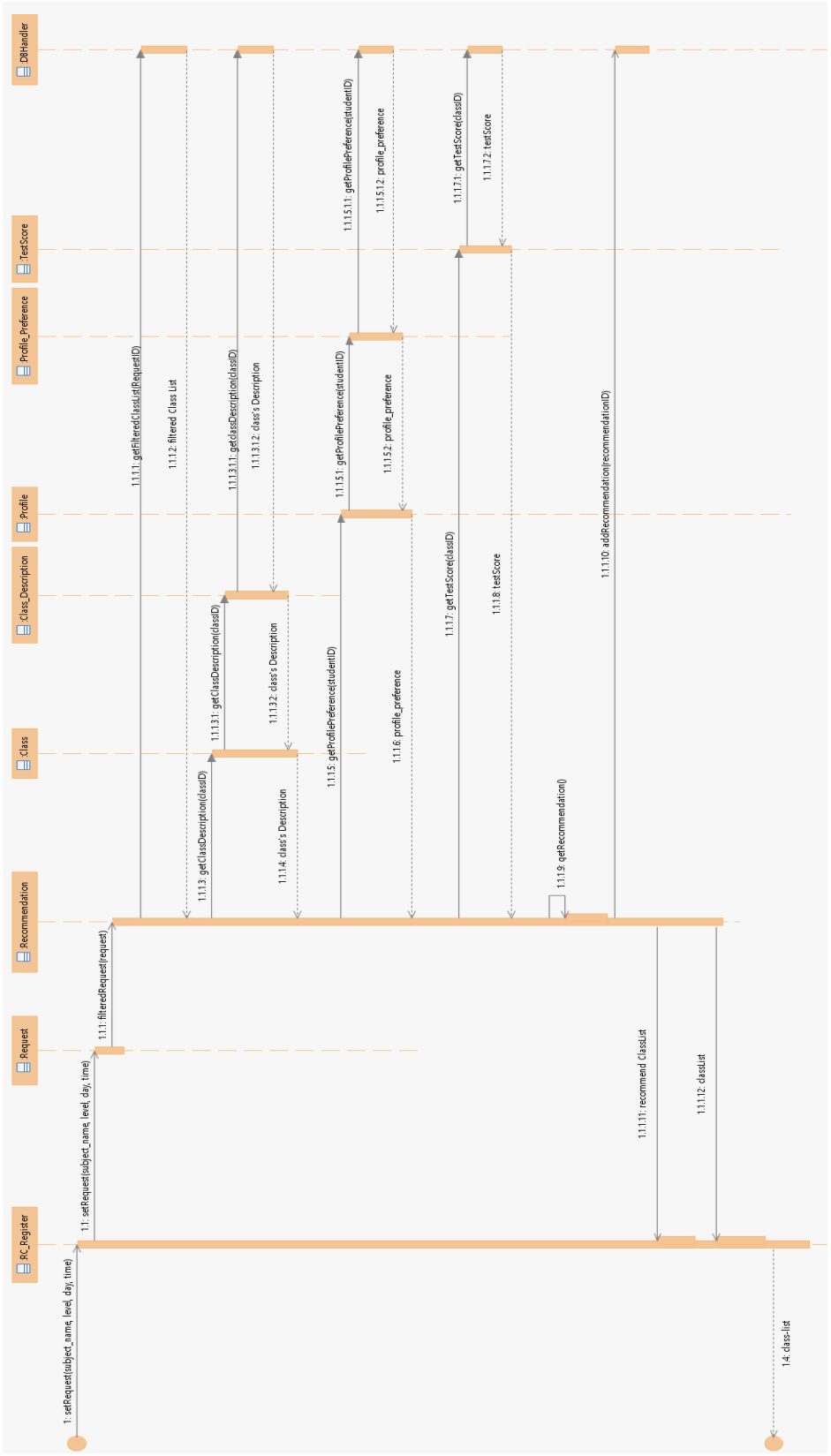
5.6.1.2. GRASP Pattern

Creator	RC_Register creates Request. RC_Register creates Recommendation.
Information Expert	Student knows Student ID.
Low Coupling	RC_Register does not need to know how to initialize RequestID.
High Cohesion	RC_Register has responsibility about creating Request and Recommendation.
Controller	RC_Register represents a handler of all system events occurred by Student.

<Table 48> <use case 6> operation1. GRASP pattern

5.6.2. Operation2 SetRequest

5.6.2.1. Design Sequence Diagram



<Figure 36> <use case 6> operation2. DSD

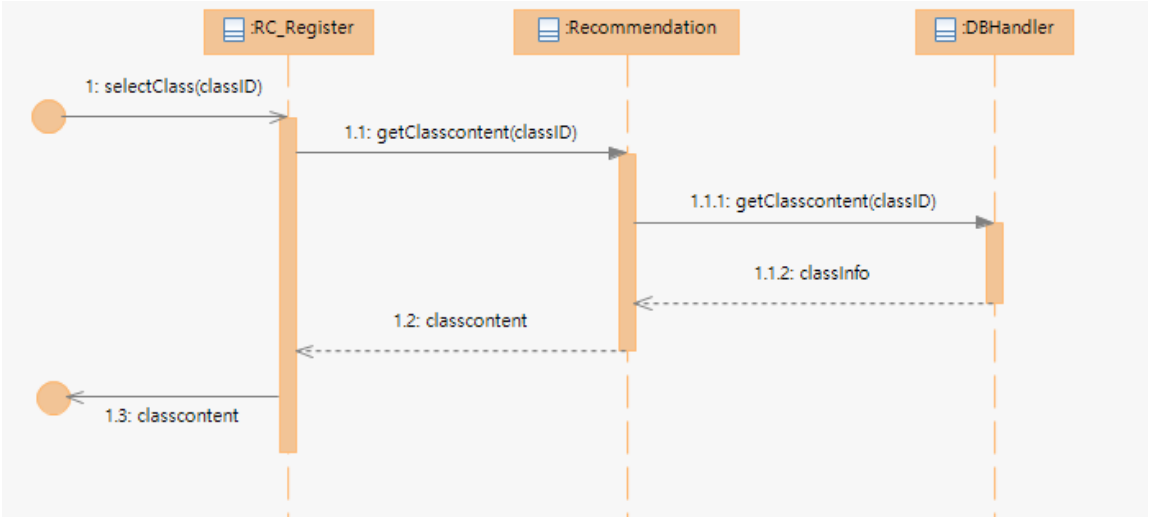
5.6.2.2. GRASP Pattern

Creator	None.
Information Expert	Recommendation knows classID.
Low Coupling	RC_Register does not need to know how to recommend class.
High Cohesion	RC_Register has only one responsibility about setting Request.
Controller	RC_Register represents a handler of all system events occurred by Student.

<Table 49> <use case 6> operation2. GRASP pattern

5.6.3. Operation3 selectClass

5.6.3.1. Design Sequence Diagram



<Figure 37> <use case 6> operation3. DSD

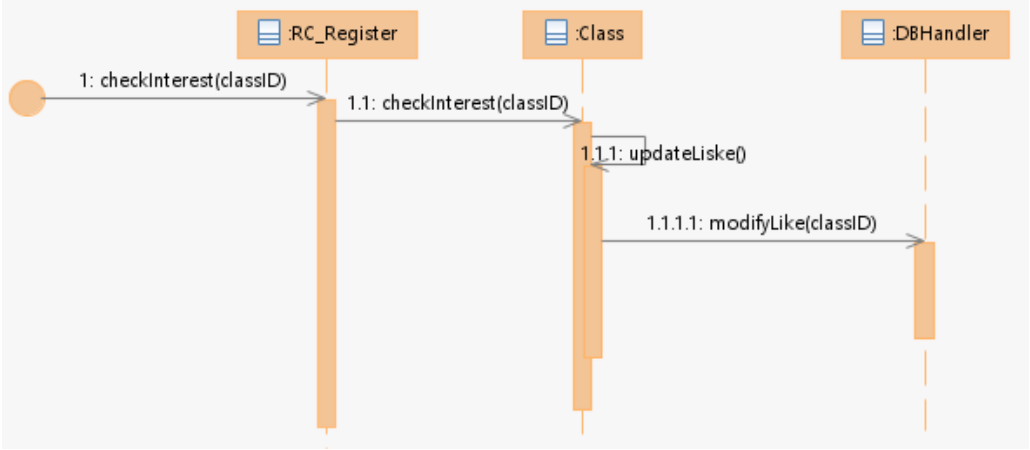
5.6.3.2. GRASP Pattern

Creator	None.
Information Expert	Recommendation knows classID.
Low Coupling	RC_Register does not need to know how to get Classlist.
High Cohesion	RC_Register has only one responsibility about getting class information.
Controller	RC_Register represents a handler of all system events occurred by Student.

<Table 50> <use case 6> operation3. GRASP pattern

5.6.4. Operation4 checkInterest

5.6.4.1. Design Sequence Diagram



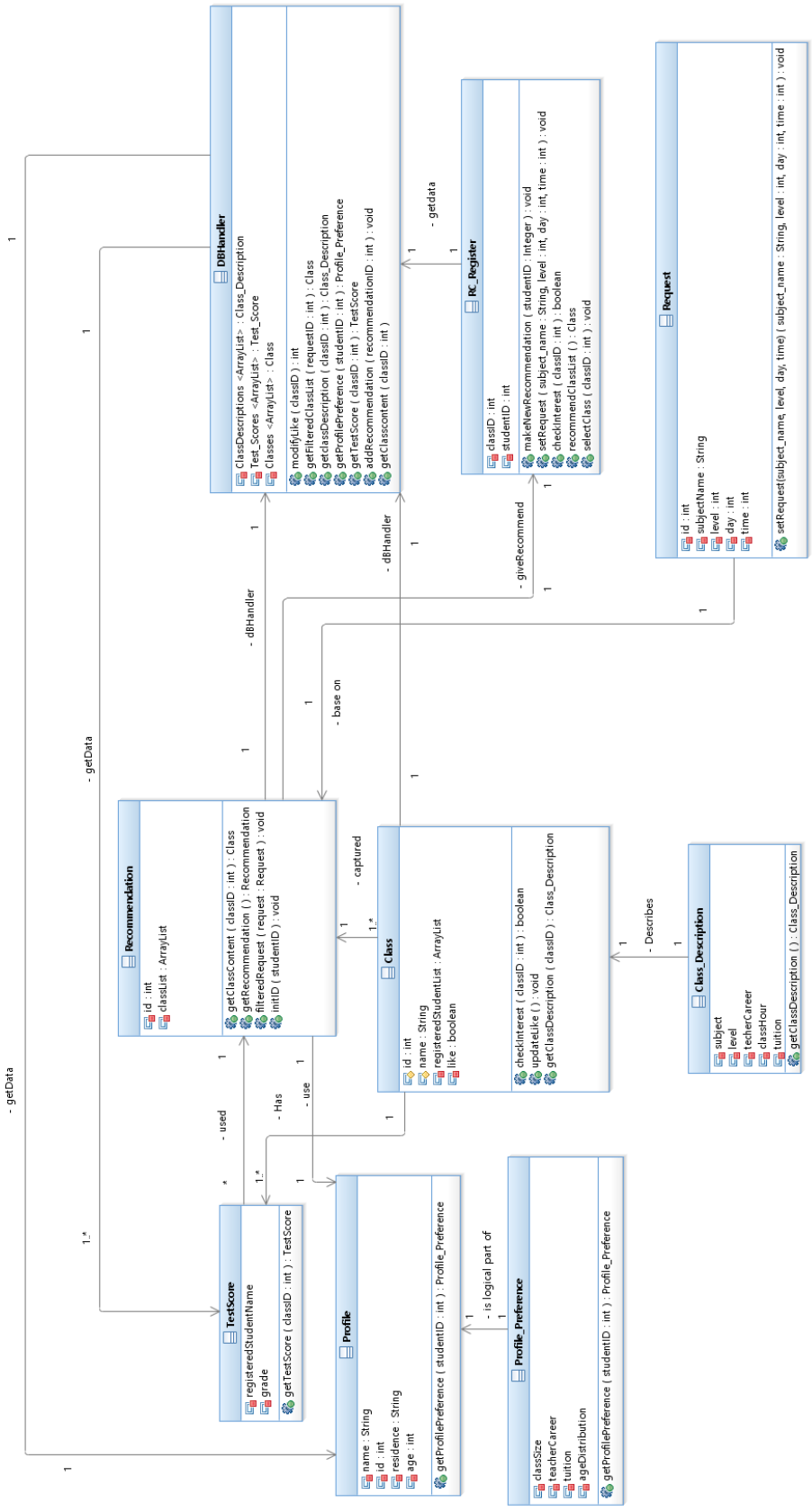
<Figure 38> <use case 6> operatio4. DSD

5.6.4.2. GRASP Pattern

Creator	None.
Information Expert	None.
Low Coupling	RC_Register does not need to know how to change like.
High Cohesion	RC_Register has only one responsibility about changing class like.
Controller	RC_Register represents a handler of all system events occurred by Student.

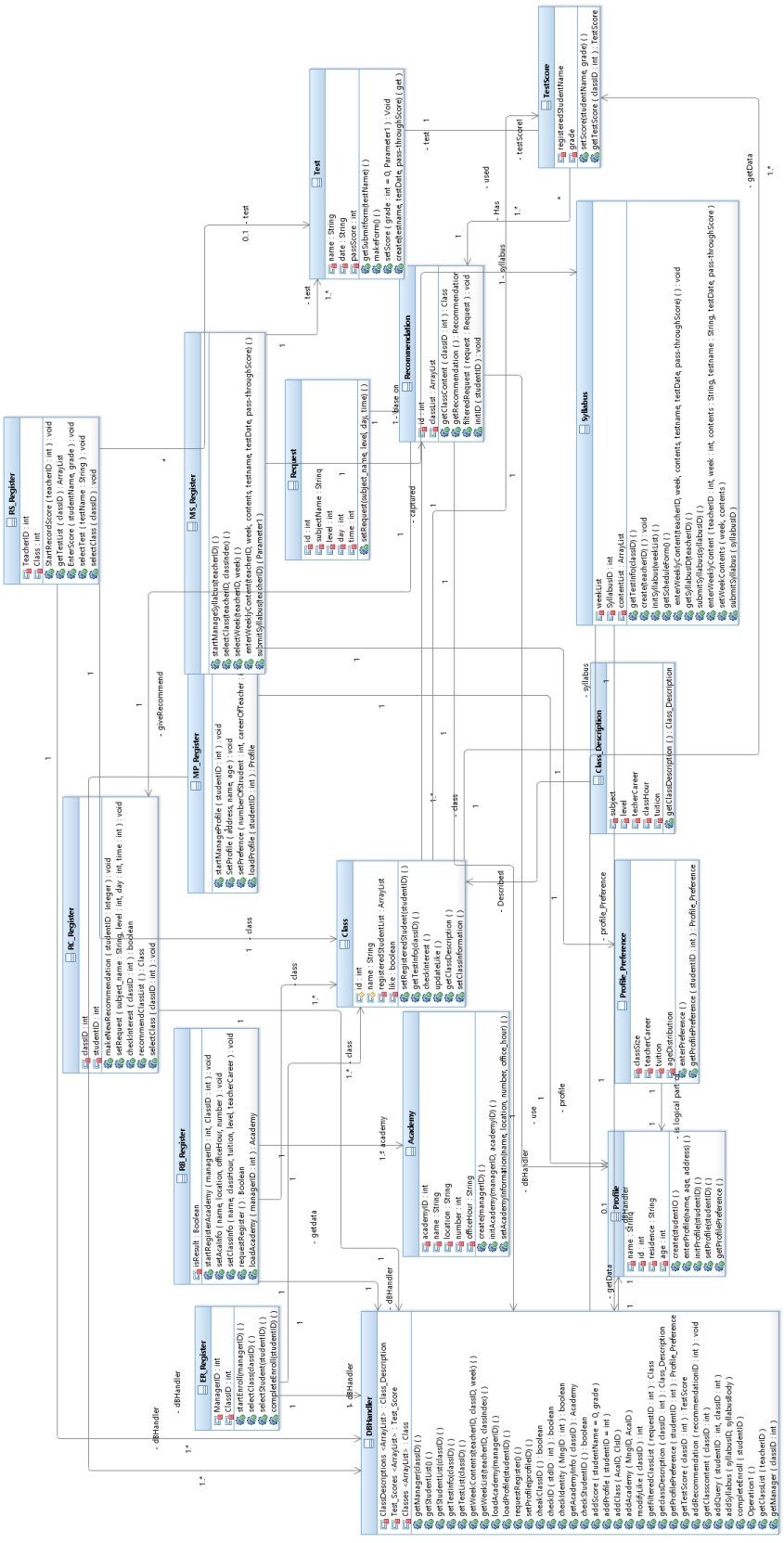
<Table 51> <use case 6> operation4. GRASP pattern

5.6.5. Combined DCD



<Figure 39> <use case 6> combined DCD

6. Design Class Diagram of the System



<Figure 40> Design Class Diagram of the System

7. Architecture

7.1. Introduction

We made specific architecture views for our system. Each of architectural views explains the reason why we construct architecture using this way. Also it offers to the user that the major ideas in the system. We chose logical view, deployment view, use case view and data view. We draw architecture based on these views. When we made various kind of view, it summarized the key architectural decisions.

- **logical view** : Conceptual organization of the software. This is consist of four layers. Also summarizes the functionality of the major software elements, such as each subsystem.
- **Deployment View** : Physical deployment of processes and components to processing nodes, and the physical network configuration between nodes.
- **Use case View** : Summary of the most architecturally significant use cases and their non-functional requirements.
- **Data View** : Overview of the data flows, persistent data schema, the schema mapping from objects to persistent data, the mechanism of mapping from objects to a database, and flow of what visits through the layers.

In conclusion, each view includes a discussion of motivation, which may help you when you need to modify the architecture.

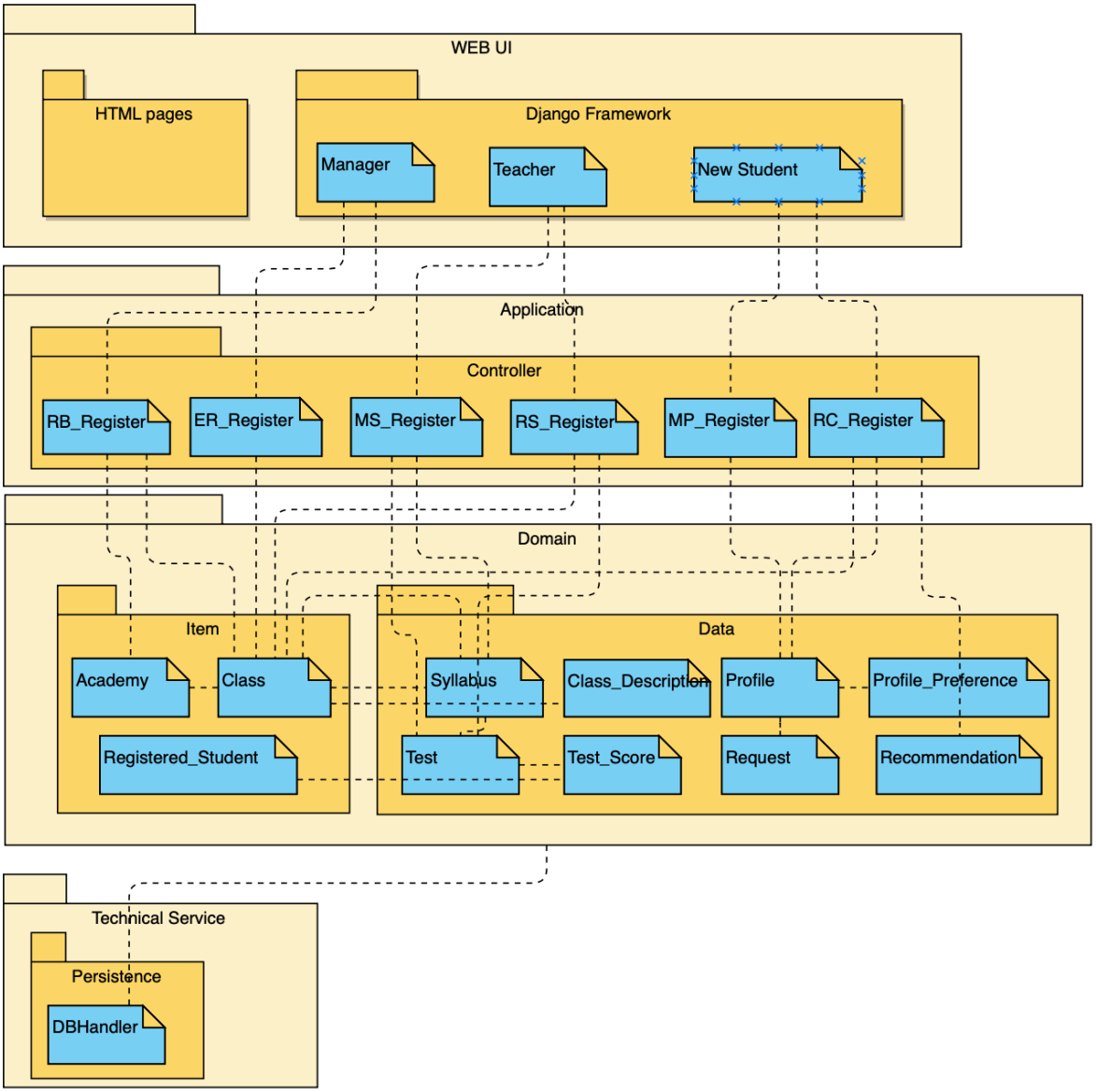
7.2. Architectural Factor

Factor	**Measures and quality scenarios**	**Variability (current flexibility and future evolution)**	**Impact of factor(and its variability) on stakeholders, architecture and other factors**	**Priority for Success**	**Difficulty or Risk**
Functionality - quality of program					
Error Handling.	All errors are logged at persistent storage. And if an error or failure occurs, the previous page is re-loaded by logged file.	**Current flexibility :** Our system says local client-side simplified services are acceptable until reconnection is possible. **Future evolution :** within 1 year, client-side mass storage solutions will be cheap and effective for logging.	Student who is using M.A.X.I.M. system really dislike connection fail.	H	H
Functionality - security					

Maintaining login time which provides convenience could be accessed by other users.	Set logout automatically after a certain period of time in public place.	**Current flexibility :** If there is no working activity for a certain period of time, set logout automatically through setting the time of expiration **Future evolution :** None	Cannot guarantee the reliability of data when information is changed by a third party. Accurate recommendation cannot be performed unless the reliability of class information, class Description and Test score are guaranteed.	M	H
Reliability - Recoverability					
Recovery when HTTP Connection failure	When a HTTP connection fails, re-establish connectivity with it within 5 second of its detected availability.	**Current flexibility :** Local Client -side simplified services are acceptable and desirable until reconnection is possible. **Future evolution :** Quick HTTP access recovery keeps application users.	User hates HTTP connection failure because it interrupts reservation Management: Need to improve connection process constantly	H	M
Send push message Again when sending was failure	When sending push message failed, server need to wait until can send push message to client(this can be within 5 second)	**As above**	Affects to the architecture a lot. When sending push message failed, it is likely to be omitted information and it may make errors	H	H
Performance					
Application must be available at all time.	Except for the server check time, services must be available 24 hours.	**Current flexibility** : Only during the time that the system manages the server, users can access it. **Future Evolution :** System ensures to user can access server and use service at any time by AWS.	Always user can wants to use the system. Low impact on design.	H	L

Supportability – adaptability					
Adaptability on various web API version.	This system is applicable on various web API version.	**Current flexibility** : System supports chrome, safari, Internet Explorer etc. **Future Evolution :** None.	Small impact on design.	L	L
Support the change of current Database type such as from SQL DB to other DB.	When there is DB change, all of function should be kept equally.	**Current flexibility** : System maintain the user's request record when DB changed. **Future Evolution :** System need to ensure the list of classes and user's profile is saved.	Lot impact on architecture. Database access failure makes system doesn't work and user can't get recommendation.	H	H

7.3. Logical view

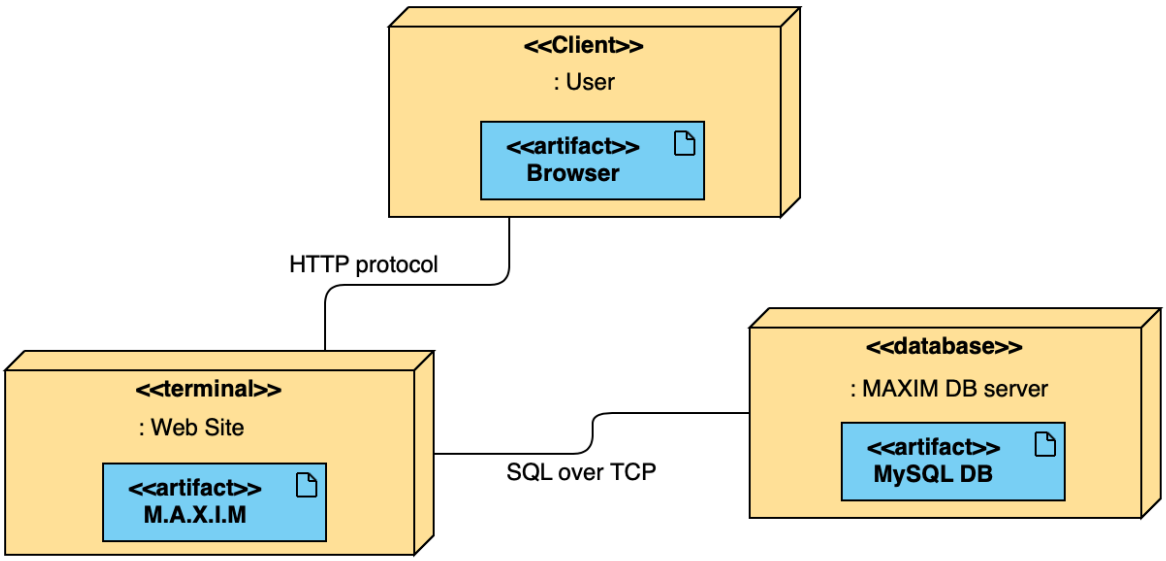


<Figure 41> logical view

7.4. Process view

Process view illustrates the processes and threads of the system. However, our system is not a multi-process or multi-thread system. So process view is non-applicable.

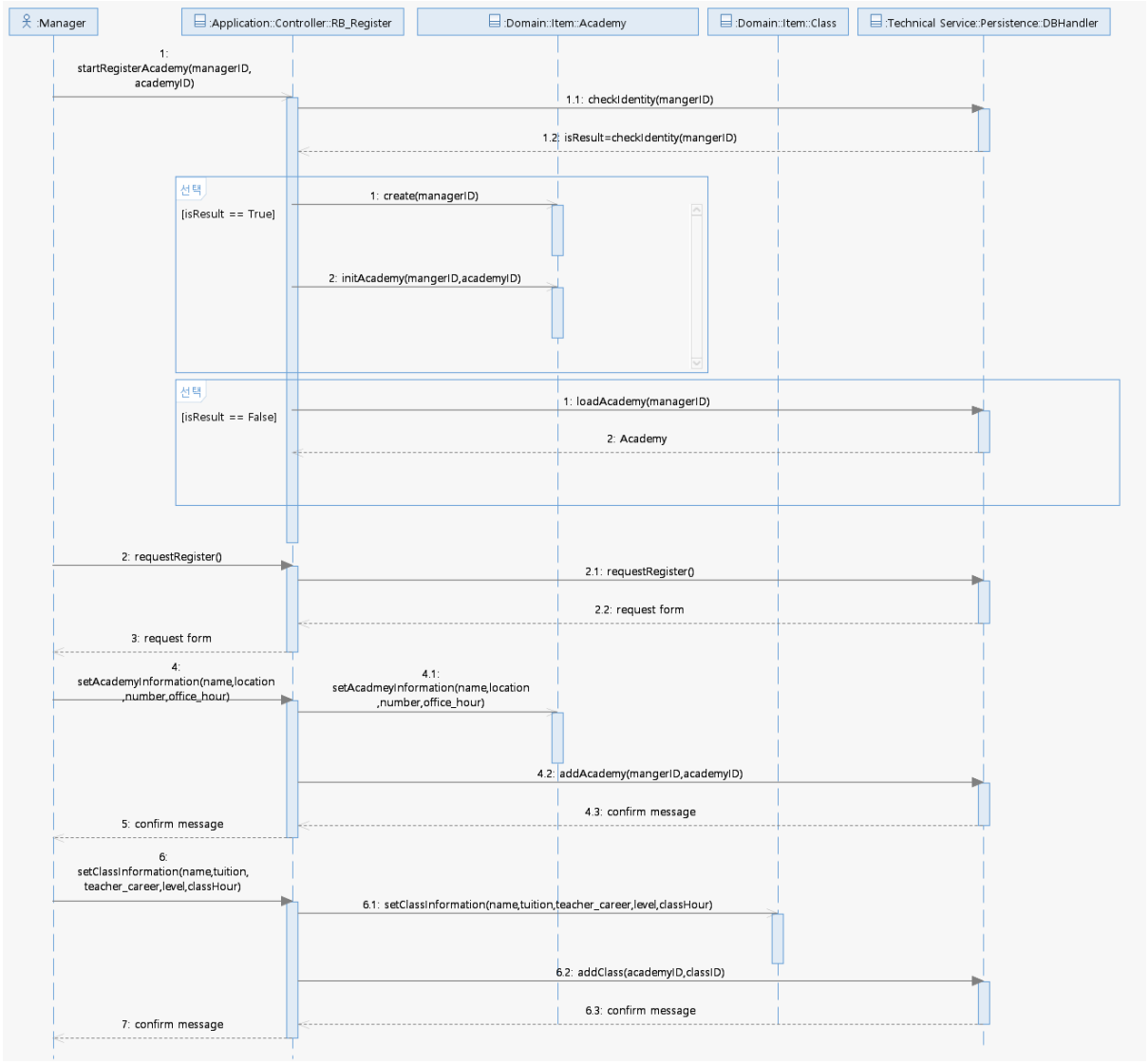
7.5. Deployment view



<Figure 42> Deployment View

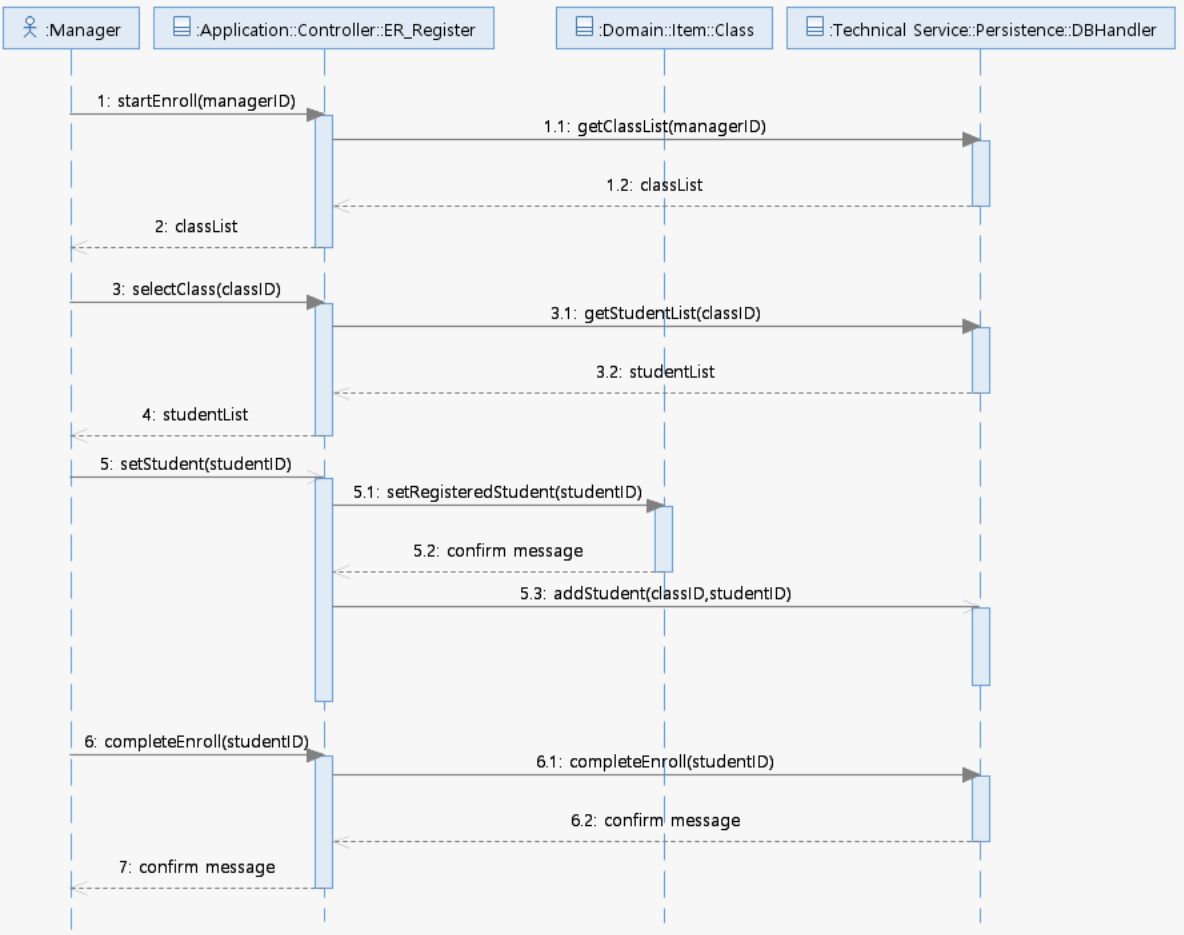
7.6. Use case view

7.6.1. <use case1> Register Basic Academy Information



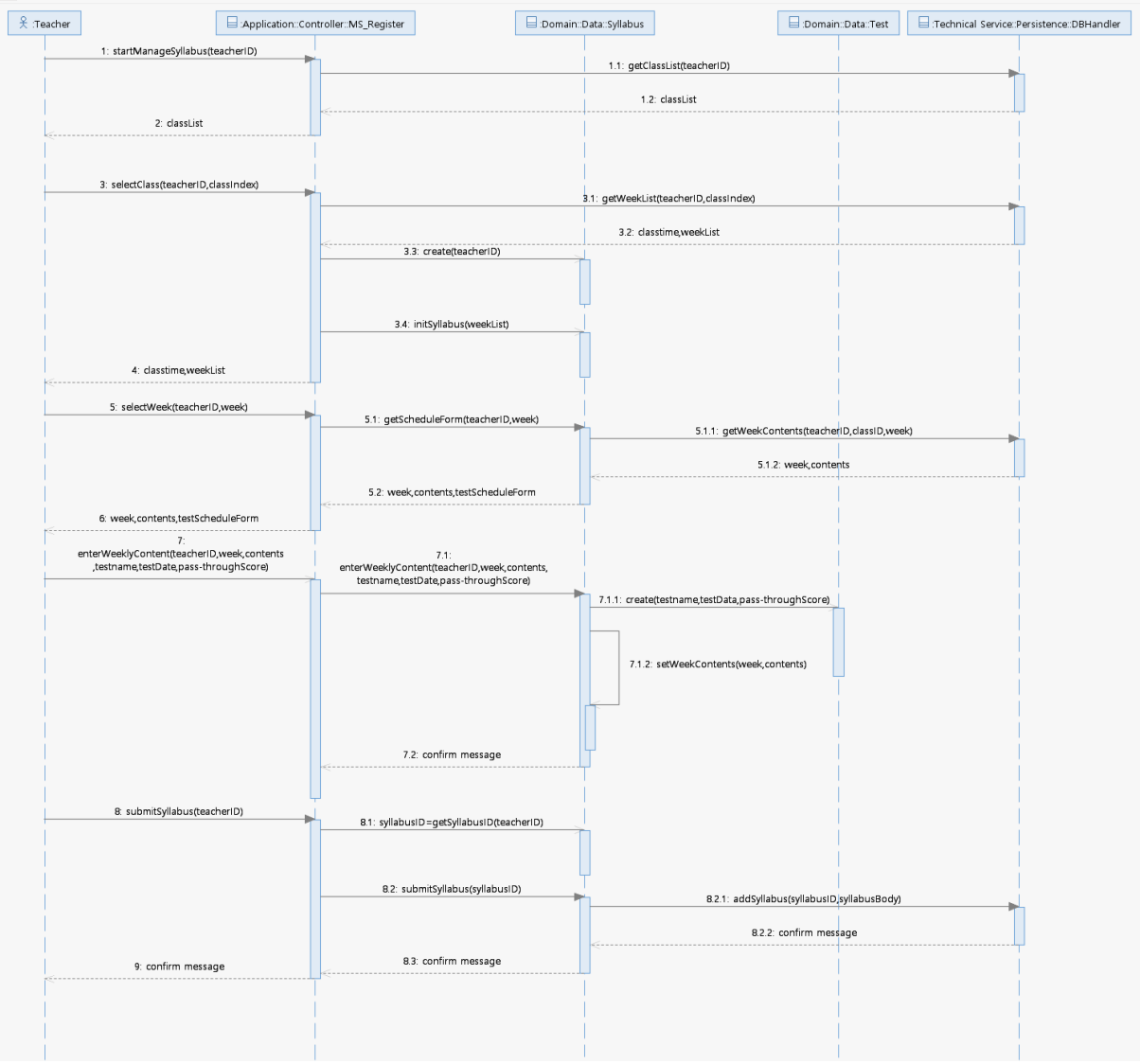
<Figure 43> Register Basic Academy Information use case view

7.6.2. <use case2> Enroll Student



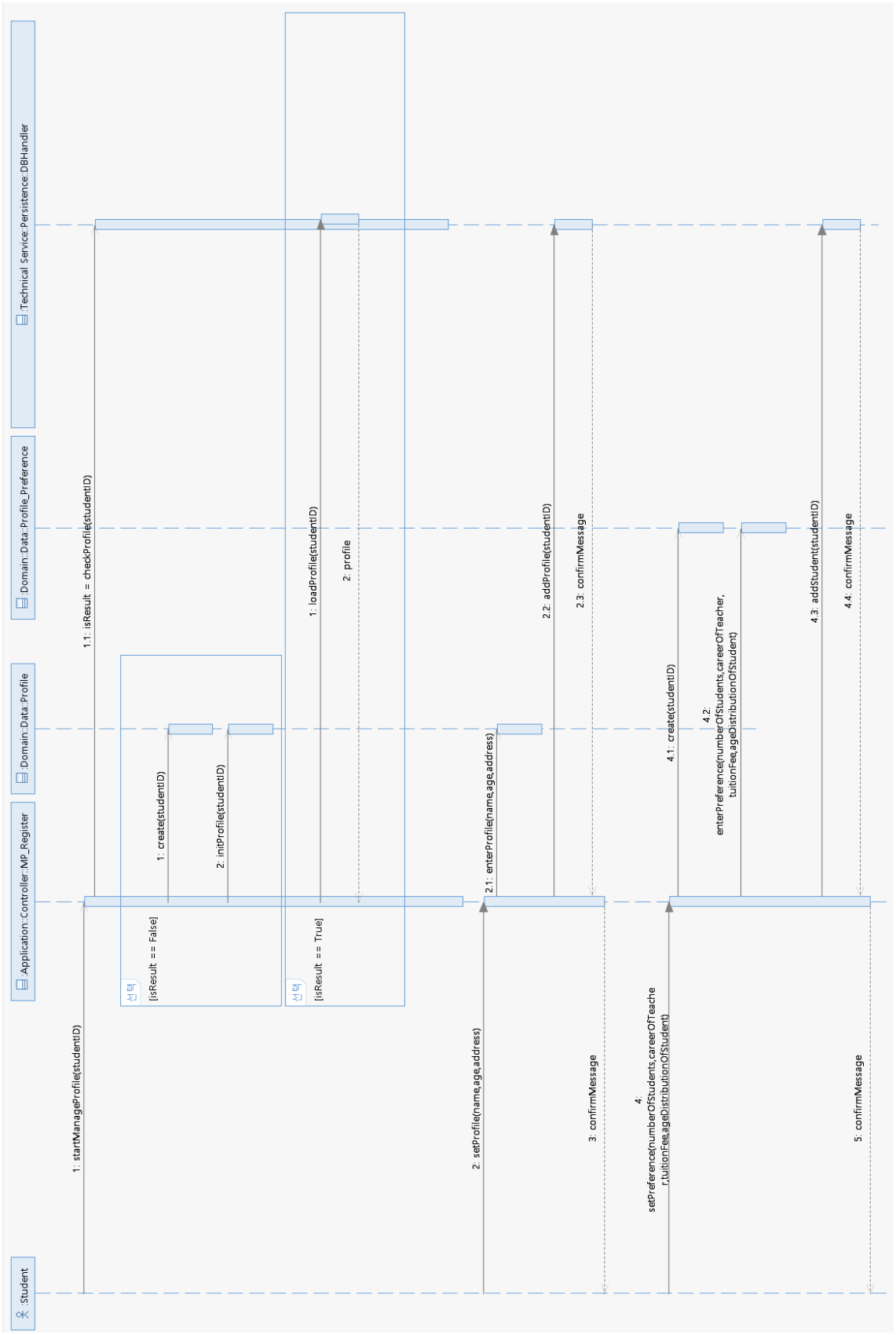
<Figure 44> Enroll Student use case view

7.6.3. <use case3> Manage Syllabus



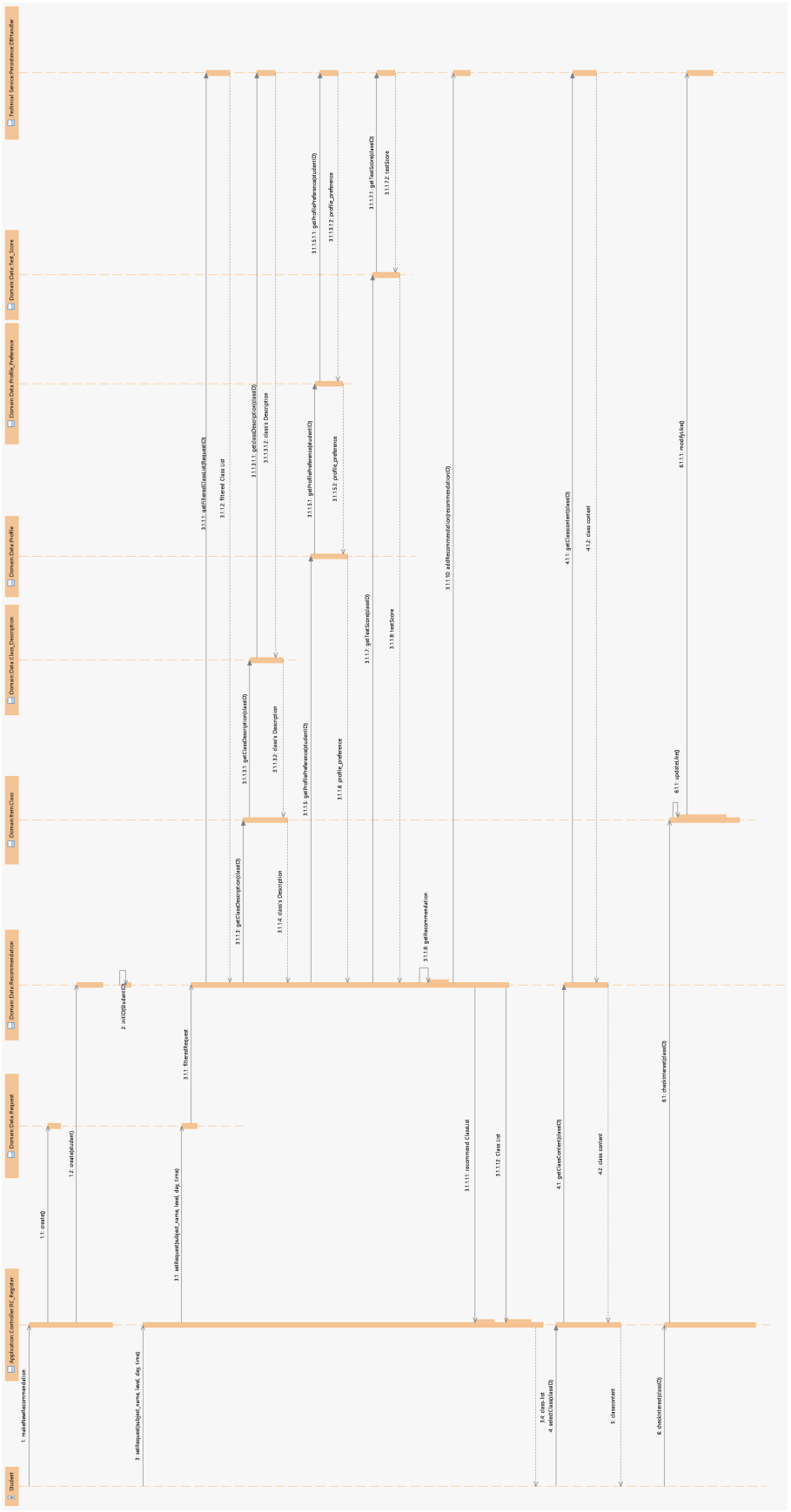
<Figure 45> Manage Syllabus use case view

7.6.4. <use case5> Manage Profile



<Figure 46> Manage Profile use case view

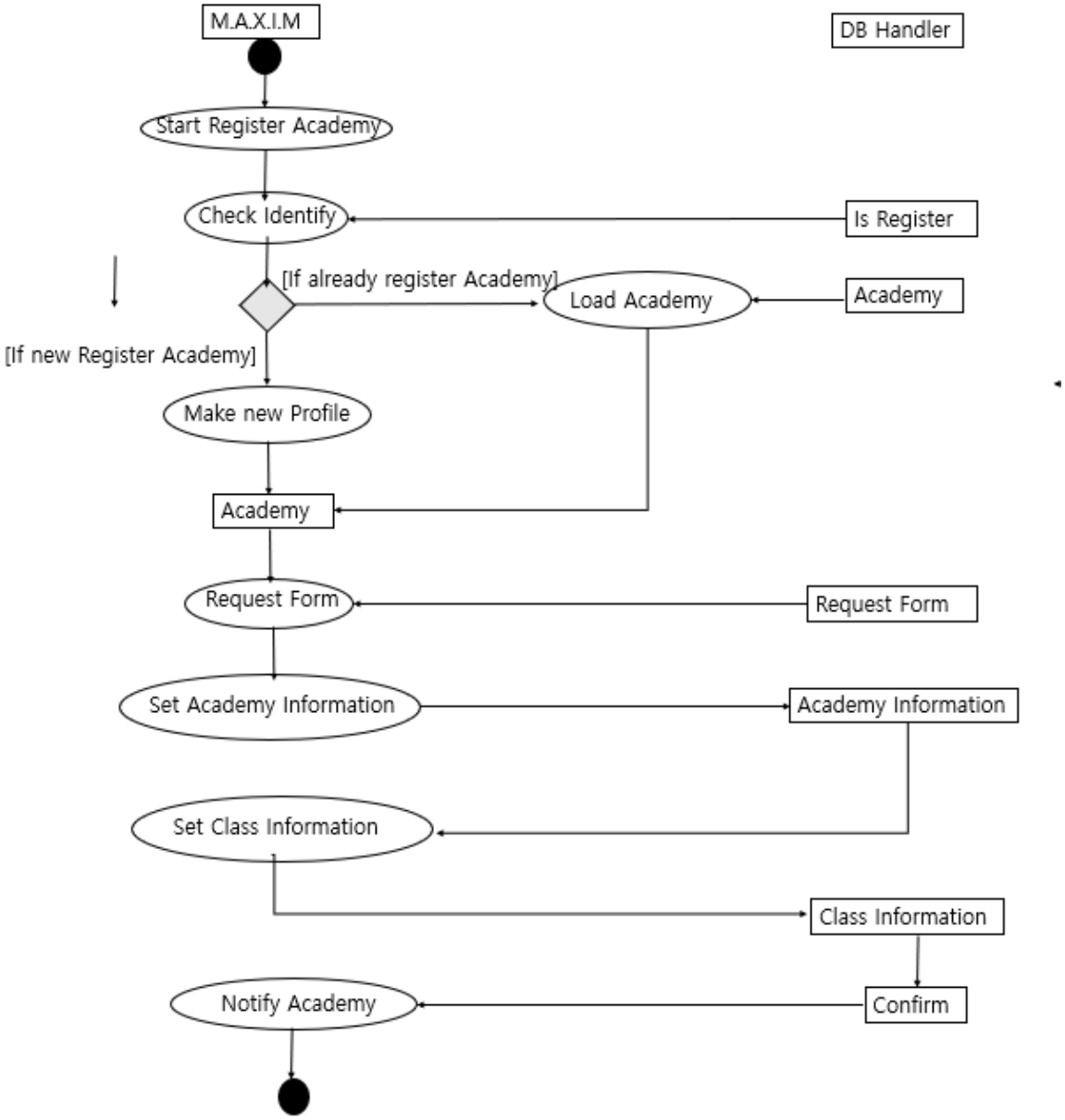
7.6.5. <use case6> Get Recommendation



<Figure 47> Get Recommendation use case view

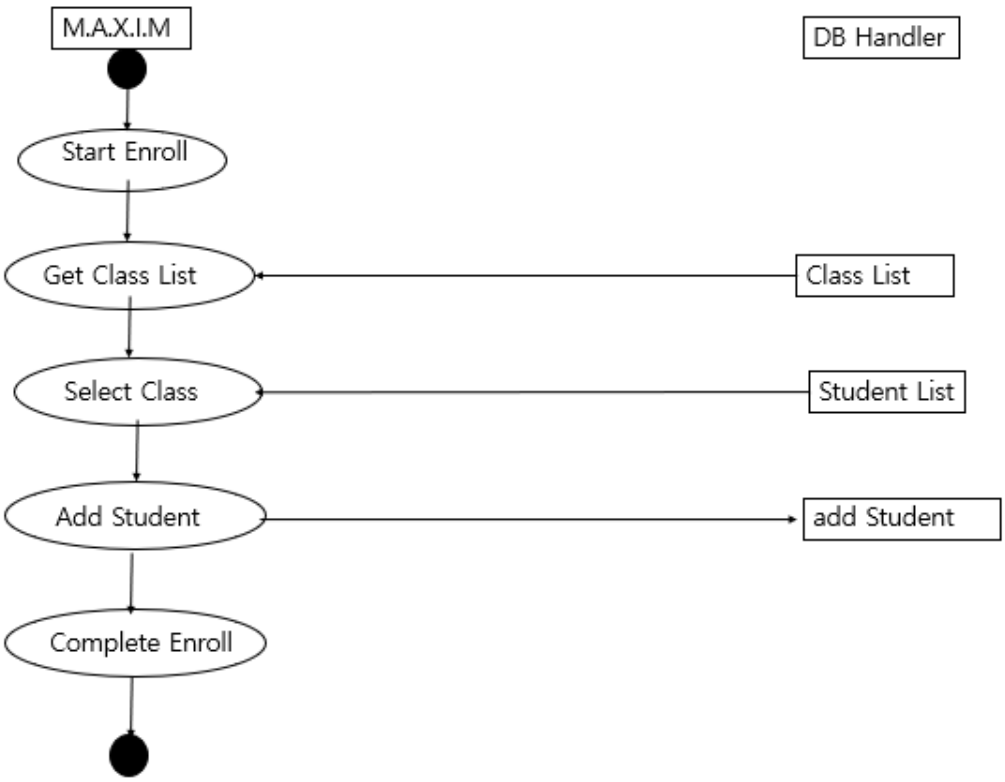
7.7. Data view

7.7.1. <use case1> Register Basic Academy Information



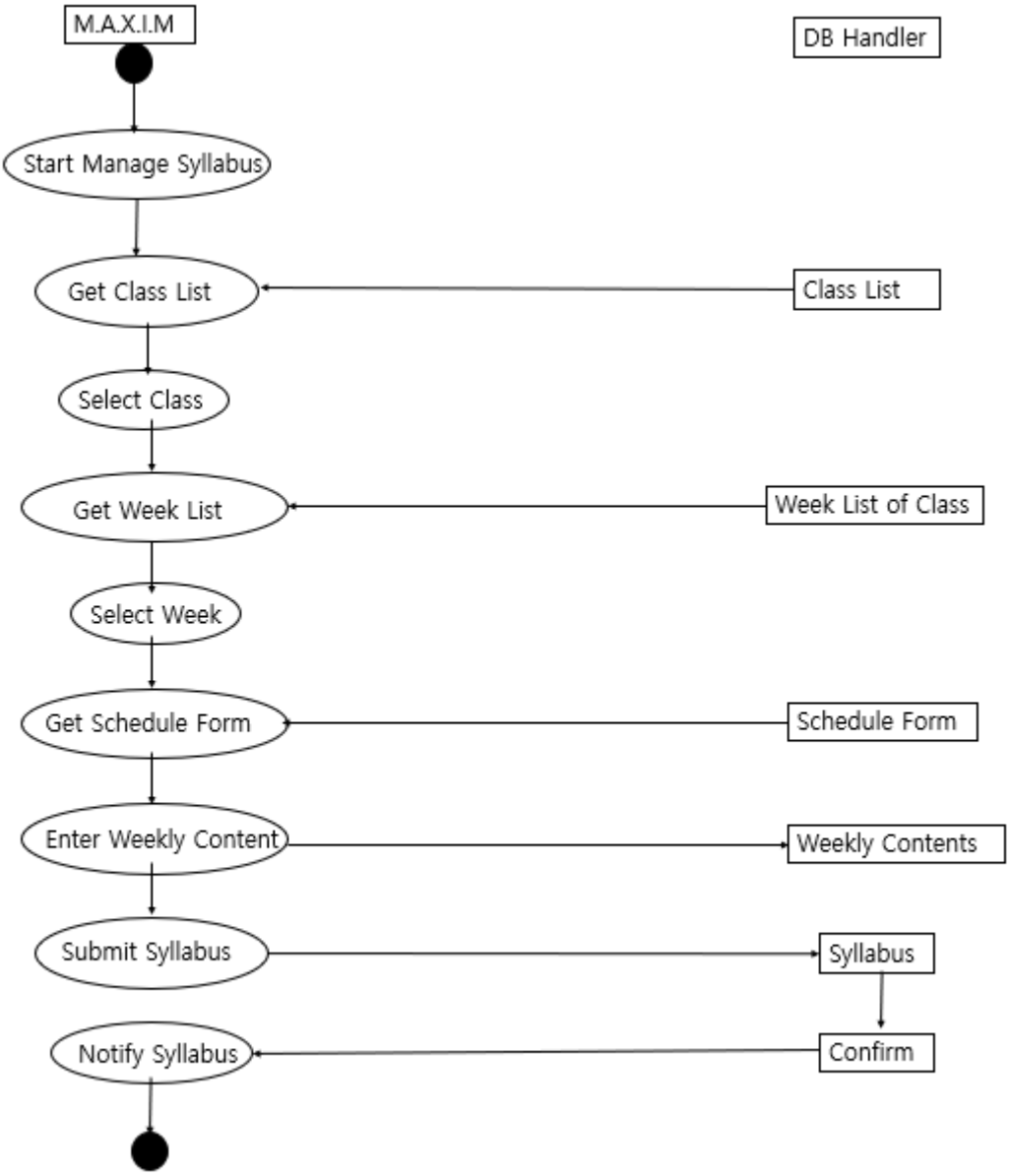
<Figure 48> Register Basic Academy Information data view

7.7.2. <use case2> Enroll Student



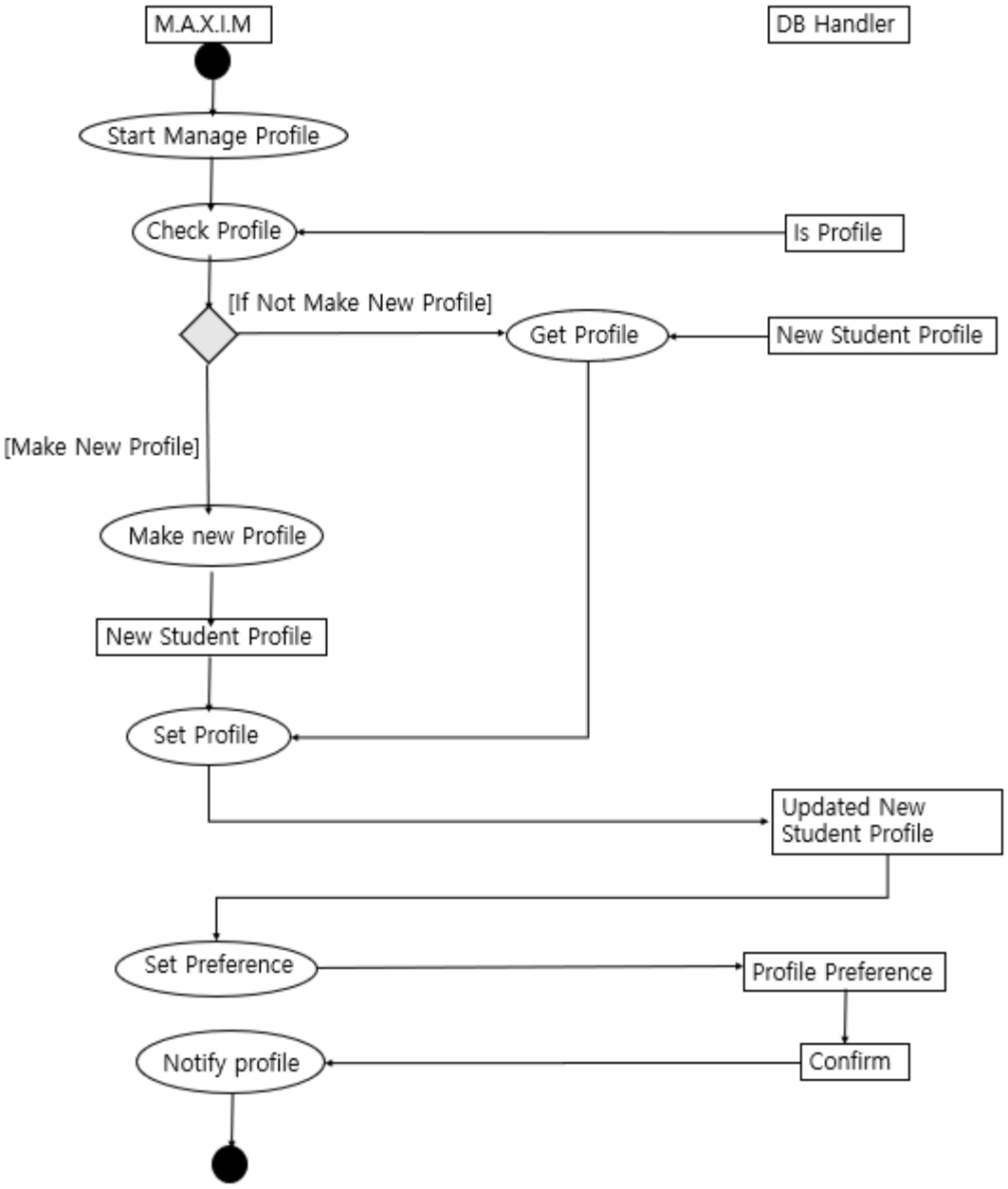
<Figure 49> Enroll Student data view

7.7.3. <use case3> Manage Syllabus



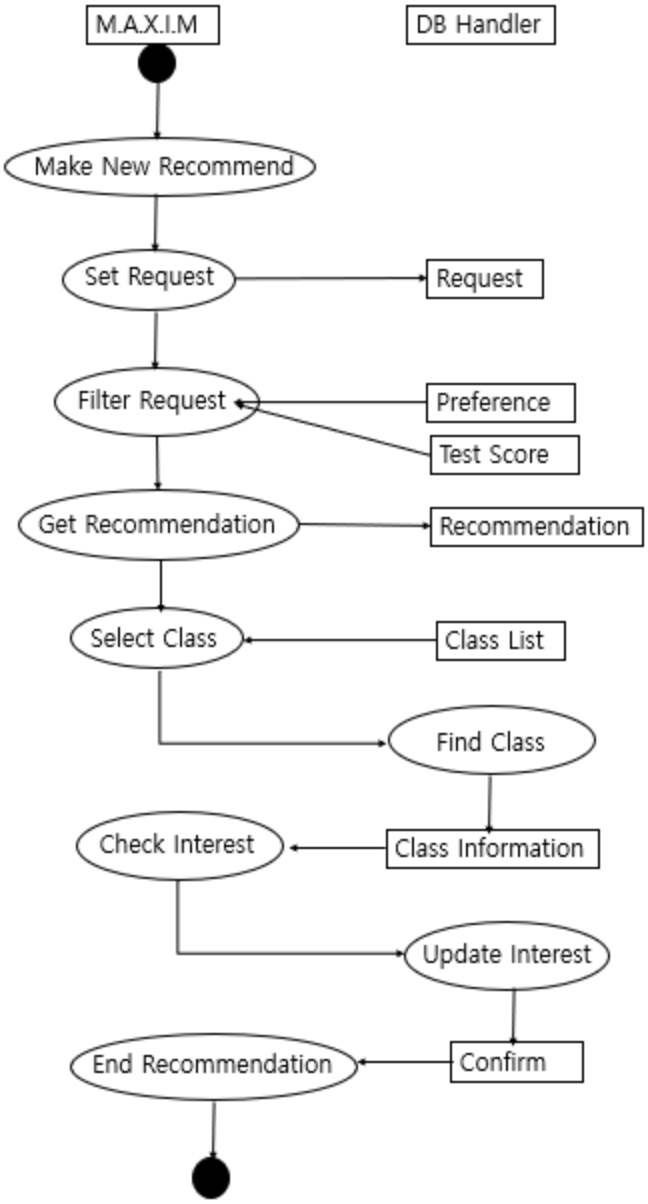
<Figure 50> Manage Syllabus data view

7.7.4. <use case5> Manage Profile



<Figure 51> Manage Profile data view

7.7.5. <use case6> Get Recommendation



<Figure 52> Get recommendation data view

8. Conclusion

8.1. Objectives

Develop 'M.A.X.I.M' system. In this system, Students are able to get recommended classes to develop their academic level. Also, make a platform which connects students and academy directly so that they can share their information. Through the 'M.A.X.I.M' system, academy manager could register their academy information, and teachers in the academy could manage class syllabus and student's test score data.

8.2. Current statement of Design & Implementation

- Design is 90% done.
- Implementation is 50% done.
  - UC6. Get Recommendation
- Implementation in detail
  - 'M.A.X.I.M' system is implemented based on Web server & client.
  - Students can register their profile in 'M.A.X.I.M' system. When they register their profile in our system, they could get recommendation of classes.
  - Managers in academy can register their academy information such as name, location, number in 'M.A.X.I.M' system.
  - 'M.A.X.I.M' system use machine learning for class recommendation to students.
  - Students can see the recommended class list which will be helpful for their study.
  - 'M.A.X.I.M' system use "k-means clustering" algorithm to train our recommendation model.

8.3. Additional Work

Applying appropriate patterns would be worthwhile to refine the design of our system. In addition, current implementation is the part for only recommendation. So, it would be better if system offer some prediction of student's score after he took a specific class. Also, even though we made the dataset by ourselves, the procedure of make dataset should be collected by lots of academies. Not only this, but also we did not offer a registration which connects student and academy. If there exists some registration process, then both students and academy could feel convenience. Lastly, in order to offer appropriate classes for students, we need to collect lots of data to train our recommendation model.

8.4. Learnt through this project

Through this project, we have learned how to develop a object-oriented program step-by-step. In the process of analyzing requirement, we could decide the scenarios in detail. Especially, Use-case Diagram, Use-case List, and Use-case Text were so helpful. Unlike other projects all we have done before, we could apply GRASP and GoF patterns to optimize the design of the program. As a result of this, we could develop the program efficiently and logically. Overall, this project let us know how to apply patterns in the OOAD system as well as appropriate procedure.

9. Reference

9.1. Books

Applying UML and Patterns, 3rd edition (2005), C. Larman, Prentice Hall.

10. Appendix

A. Glossary

Term	Definition and information
Manager	A person who provides academy name, location, phone number, tuition fee and manages student enrollment.

New Student	A person who register own profile and looking for academy to study.
Registered Student	A person who has already been enrolled in an academy.
Profile	Base information used by the recommendation system.
Recommendation	Offering suitable information for students, based on your profile and supplementary information.
Like	The degree to which the recommended students expressed interest in the class.
Teacher	A person who provides class syllabus and manages student's test score.
Enroll	Entering student human resources and storing information about the classes that the student takes in the system.
Syllabus	A summarized weekly progress contents of each lesson.
Score	The grades of students managed by the teacher after the exam.

<Table 52> Glossary

B. Source Code

The source code will be attached separately through the zip file.

11. Revision History

Version	Date	Description
Inception ver.0.0.0	3.15	Propose each team member's ideas Share team members' role
Inception ver.1.0.0	3.18	Confirm theme of project
Inception ver.1.0.1	3.22	Set up inception phase document
Inception ver.1.0.2	3.24	Combine team members' work Update Inception phase Document
Inception ver.1.1.0	3.25	Modify Use Case Diagram
Inception ver.1.1.1	3.26	Do mini presentation about inception
Inception ver.1.2.0	3.27	Have a meeting with TA Modify the direction of 'Get Recommendation' Modify the direction of 'Enroll Student'
Inception Draft	3.29	Frist Draft Submission
Inception ver.1.2.1	4.2	Do presentation about inception
Inception ver.1.3.0	4.5	Have a meeting with professor Modify the direction of 'Register Academy'
Elaboration ver.1.0.0	4.9	Update Fully Dressed of all Use cases Modify full dressed of 'Enroll Student'
Elaboration ver.1.1.0	4.14	Domain Modeling, mini SSD, OC
Elaboration ver.1.1.1	4.26	Update SSD, OC
Elaboration ver.1.2.0	4.28	Update Operation Contract of 'Get Recommendation'
Elaboration ver.1.3.0	5.1	Complement Fully Dressed Update Domain Modeling
Elaboration ver.1.3.1	5.2	Modify as a whole Design Domain Diagram, Design Class Diagram
Elaboration ver 1.3.2	5.6	Set up mini presentation document
Elaboration ver 1.3.3	5.21	Revise version of Elaboration1.

		Share team members' role
Elaboration ver 2.0.0	5.23	Complement as a whole.
Elaboration ver 2.0.1	5.31	Combine team members' work
Elaboration ver 2.0.2	6.2	Modify logical view, use case view, data view Update Get recommendation design sequence diagram
Elaboration ver 2.0.3	6.5	Have a meeting with TA
Elaboration ver 2.0.4	6.6	Modify N+1 View and update Design Class diagram. Complement as a whole.
Elaboration ver 2.0.4	6.7	Do submission.

<Table 53> Revision History