**MINISTRY OF EDUCATION AND TRAINING**

|  |
| --- |
| **FPT UNIVERSITY** |
| **Capstone Project Document** |
| **Course-Scheduling** |
|  |
| |  |  | | --- | --- | | **Course Scheduling Team** | | | **Group Member** | Nguyễn Sơn Hà – SE02618  Hoàng Minh Đức – SE02551  Lưu Thành Công – SE02543  Nguyễn Văn Hoài – SE90191 | | **Supervisor** | Nguyễn Tất Trung  Phạm Quang Dũng | | **Capstone Project code** | CourseScheduling |   Hanoi, 24/5/2015 Acknowledgements |

# Definitions and Acronyms

# Contents

[Acknowledgements 2](#_Toc421134246)

[Definitions and Acronyms 2](#_Toc421134247)

[Contents 3](#_Toc421134248)

[1 CHAPTER 1: INTRODUCTION 4](#_Toc421134249)

[1.1 Purpose 4](#_Toc421134250)

[1.2 Project Information 4](#_Toc421134251)

[1.3 The People 4](#_Toc421134252)

[1.3.1 Supervisor 4](#_Toc421134253)

[1.3.2 Team members 5](#_Toc421134254)

[1.4 Background 6](#_Toc421134255)

[1.5 Literature Review 6](#_Toc421134256)

[1.6 Ideas 6](#_Toc421134257)

[1.7 Proposal of System 6](#_Toc421134258)

[1.7.1 Our proposal system 6](#_Toc421134259)

[1.7.2 Introduction about our proposal system 6](#_Toc421134260)

[2 Chapter 2. Project Management Plan 7](#_Toc421134261)

[2.1 Purpose 7](#_Toc421134262)

[2.2 Project Organization 7](#_Toc421134263)

[2.2.1 Our Proposal System 7](#_Toc421134264)

[2.2.2 Roles and Responsibility 7](#_Toc421134265)

[2.3 Project Management Plan 9](#_Toc421134266)

[2.3.1 Task 9](#_Toc421134267)

[2.3.2 Meeting Minutes 9](#_Toc421134268)

[2.3.3 Coding Conventions 9](#_Toc421134269)

[2.3.4 Risk Management Plan 10](#_Toc421134270)

[2.3.5 Risk probability and impact 11](#_Toc421134271)

[2.3.6 Communication Plan 11](#_Toc421134272)

[3 CHAPTER 3. SOFTWARE REQUIREMENT SPECIFICATION 12](#_Toc421134273)

[3.1 Purpose 12](#_Toc421134274)

[3.2 Functional Requirements 12](#_Toc421134275)

[3.2.1 Use Cases Diagram 12](#_Toc421134276)

[3.2.2 Use Cases List 12](#_Toc421134277)

[3.2.3 Business Rules 14](#_Toc421134278)

[3.2.4 Use Cases 15](#_Toc421134279)

[3.3 Non-functional Requirements 27](#_Toc421134280)

[3.3.1 Security 27](#_Toc421134281)

[3.3.2 Maintainability & Extensibility 27](#_Toc421134282)

[3.3.3 Performance 27](#_Toc421134283)

[3.4 Entity Relationship Diagram 28](#_Toc421134284)

# CHAPTER 1: INTRODUCTION

## Purpose

The introduction chapter provides main information of project about team’s members and project’s objectives.

## Project Information

* Project name: **Course-Scheduling**
* Project code: **Course-Scheduling**
* Project group name **:**
* Product type: **Website Application**
* Timeline: **From 10th May 2015 to 30thAug 2015**

## The People

### Supervisor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-Mail** | **Title** |
| Supervisor | Nguyễn Tất Trung | 0904399139 | [*trungnt@fpt.edu.vn*](mailto:trungnt@fpt.edu.vn) | Lecturer |
| Supervisor | Phạm Quang Dũng | 0947732280 | [*dungpq1@fsoft.com.vn*](mailto:dungpq1@fsoft.com.vn) |  |

*Table 1.1: Supervisor information*

### Team members

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Full name** | **Student code** | **Phone** | **E-mail** | **Role in Group** |
| 1 | **Nguyễn Sơn Hà** | SE02618 | 0945488693 | [*hansse02618@fpt.edu.vn*](mailto:hansse02618@fpt.edu.vn) | Leader |
| 2 | **Hoàng Minh Đức** | SE02551 | 0904634667 | [*duchmse02551@fpt.edu.vn*](mailto:duchmse02551@fpt.edu.vn) | Member |
| 3 | **Lưu Thành Công** | SE02543 | 01656083093 | [*congltse02543@fpt.edu.vn*](mailto:congltse02543@fpt.edu.vn) | Member |
| 4 | **Nguyễn Văn Hoài** | SE90191 | 01659029841 | [*hoainvse90901@fpt.edu.vn*](mailto:hoainvse90901@fpt.edu.vn) | Member |

*Table 1-2: Team member’s information*

## Background

## Literature Review

## Ideas

Scheduling Courses is always one of the hardest work for FPT University because of conflicts about time of teachers and the change of timetable in every semester. Our project will apply algorithms to do this work automatically by computer and also provide a tool to help them create timetable easer in the website.

## Proposal of System

### Our proposal system

### Introduction about our proposal system

# Chapter 2. Project Management Plan

## Purpose

This chapter describes about project team’s structure and plan of the project. All members will use this chapter as a guideline to work in this project.

## Project Organization

### Our Proposal System

This figure above describes the information and products flow lifecycle process model. Course-Scheduling project uses the Iterative and Incremental Software Process Model.

The Iterative and Incremental Software Process Model is most use when the scope of the project is big, the major requirements were defined clearly, some more detail will be added in time, and for the newbie group in software development. By using this software process model, we break down the developing system task into series of smaller tasks which be completed separately, evaluated, and subsequently re-worked until the system’s performance adequately. In addition, the iterative model is easier than other models when the issues are discovered. They are fed back to the team, and solutions found while the project is still in development.

### Roles and Responsibility

#### Organization Structure

|  |  |
| --- | --- |
| **Role** | **Responsibility** |
| Project Manager | Planning and defining scope, developing schedules, allocating resources, coordinating communication, generally responsible for keeping the team’s focus on main goal, and tries to keep the project team focused on the right goal at a time. |
| Technical Leader | Responsible for the underlying architecture for the software program, assigning tasks, mentoring people, reporting and updating regular associates RAG status. Also technical leader is a reference book for other software programmers. |
| Business Analyst | Eliciting requirement, analyzing and documenting requirements, communicate requirements, identifying solution and verifying solution meets the requirements, leads and coordinates use-case modeling. Outline the system’s functionality. |
| Test Leader | Responsible for test execution, including test set-up and test run, evaluation of test run and error recovery, defect logging and test results recording. |
| Developer | Involve to code product |
| Designer | Involve to design product |
| Tester | Involve to test product |

#### Project Team Members

|  |  |
| --- | --- |
| **Team Member** | **Role** |
| HaNS | Project Manager, Developer, Tester, Designer |
| DucHM | Technical Leader, Test Leader, Developer, Tester, Designer |
| CongLT | Business Analyst, Developer, Tester |
| HoaiNV | Business Analyst, Developer, Tester |

#### Tools and Techniques

* **Programming languages:** Java, HTML5, CSS3, JQuery.
* **Framework:** MVC Spring
* **Software Architecture:** Web Service (REST architecture).
* **Process Model:** Iterative and Incremental Software Process Model.
* **Version Control:** Github.
* **IDEs**: Spring Tool Suite
* **DBMS**: MySQL v5.5
* **UML tools:** Enterprise Architect v12, Astah Professional v6.9.0
* **Manage tools:**  Microsoft Project Professional 2013.
* **Web server:** Apache2.0.

## Project Management Plan

### Task

Refer to file Course-Scheduling.mpp

### Meeting Minutes

All meeting minutes will be written follow this template:

| ***Meeting/Project Name:*** | *CourseScheduling* | | | | |
| --- | --- | --- | --- | --- | --- |
| ***Date of Meeting:*** | *01/06/2015* | | ***Time: (Type)*** | *3 hours (Skype)* | |
| ***Meeting Called By:*** | *HaNS* | | ***Location:*** | *B109* | |
| ***Note Taker:*** | *DucHM* | | ***Time Keeper:*** | *CongLT* | |
| 1. Meeting Objective: | | | | | |
| Brainstorming all functions of systems | | | | | |
| 2. Attendance | | | | | |
| ***Name*** | ***Roles*** | ***E-mail*** | | | ***Phone*** |
| Nguyen Son Ha | Project Manager | [hansse02618@fpt.edu.vn](mailto:hansse02618@fpt.edu.vn) | | | 0945488693 |
| Hoang Minh Duc | Technical Leader, Test Leader | [duchmse02551@fpt.edu.vn](mailto:duchmse02551@fpt.edu.vn) | | | 0904634667 |
| Luu Thanh Cong | Business Analyst | [congltse02543@fpt.edu.vn](mailto:congltse02543@fpt.edu.vn) | | | 01656083093 |
| Nguyen Van Hoai | Business Analyst | [hoainvse90191@fpt.edu.vn](mailto:hoainvse90191@fpt.edu.vn) | | | 01659029841 |
| 3. Content: | | | | | |
|  | | | | | |

### Coding Conventions

Follow Java coding convention. See the reference for detail:

<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

### Risk Management Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Name** | **Root Cause** | **Avoidance plan** | **Contingency** |
| R1 | **Member**  **un-cooperating** | Conflict between team members. | Team building to connect member with each others. | Finding root conflict (social relationship, private conflict) |
| R2 | **Team member do not have enough appropriate skill, experience** | All member are students. | When developing project plan, estimating more time for study new technique. | Trying to contact with expertise to learn from their experience and get their advices. |
| R3 | **Changing in the way of executing** | Creative members. |  | Meeting for evaluating, and changing the way to execute the project if it's really better. |
| R4 | **Over deadline** | Low responsibility in working of members | Estimate time to plan task appropriate with ability of team members and hold daily meeting to keep track on project status day by day. | Other team members will support to help the task can complete in time. |
| R5 | **Low quality infrastructure of the internet** | Incident |  | Use other internet like 3G. |
| R6 | **Lack of real data** | Can not contact with people who can provide team the real data. |  | Create dummy data based on the knowledge about this business of team members. |

### Risk probability and impact

High

|  |  |  |
| --- | --- | --- |
|  | R5 |  |
|  | R1 | R4 |
| R3 | R2 | R6 |

Probability

Medium

Low

High

Medium

Low

Impact

*Figure 2.2. Risk probability and impact*

### Communication Plan

*Weekly meeting schedule:* By using Iterative and Incremental Process Model, Footprints Project system will be divided into a series of small tasks, each task will be assigned to team members by Technical Leader and depend on difficulty, and Technical Leader will assign deadlines for each task. We will have meeting every Tuesday and Thursday to keep track about our works in a week then discuss about the tasks we have to done in next week.

*Unscheduled meeting*: If any unplanned or important situation happened, we have to hold a meeting to find solution or discuss about what we need to do in this situation.

*Communication channel*: Our main communication channels are face-to-face meeting, e-mail and Facebook.

# CHAPTER 3. SOFTWARE REQUIREMENT SPECIFICATION

## Purpose

## Functional Requirements

### Use Cases Diagram

Admin, Staff, Teacher, Student

### Use Cases List

|  |  |
| --- | --- |
| Primary Actor | Use Cases |
| Staff, Teacher, Student | UC-01. Login |
| Staff, Teacher, Student | UC-02. Logout |
| Staff | UC-03. View All Staffs |
| Staff | UC-04. Add Staff |
| Staff | UC-05. Remove Staff |
| Staff | UC-06. Search Staffs |
| Staff | UC-08. View All Rooms |
| Staff | UC-09. Add Room |
| Staff | UC-10. Add List of Rooms |
| Staff | UC-11. Update Room |
| Staff | UC-12. Remove Room |
| Staff | UC-13. Search Rooms |
| Staff | UC-15. View All Semesters |
| Staff | UC-16. Add Semester |
| Staff | UC-17. Update Semester |
| Staff | UC-18. Search Semester |
| Staff | UC-20. View All Teachers |
| Staff | UC-21. Add Teacher |
| Staff | UC-22. Add List of Teachers |
| Staff | UC-23. Update Teacher |
| Staff | UC-24. Remove Teacher |
| Staff | UC-26. Search Teachers |
| Staff | UC-28. View All Students |
| Staff | UC-29. Add Student |
| Staff | UC-30. Add List of Students |
| Staff | UC-31. Update Student |
| Staff | UC-32. Remove Student |
| Staff | UC-34. Search Students |
| Staff | UC-36. View All Courses |
| Staff | UC-37. Add Course |
| Staff | UC-38. Add List of Courses |
| Staff | UC-39. Update Course |
| Staff | UC-40. Remove Course |
| Staff | UC-41. Search Courses |
| Staff | UC-43. View All Classes |
| Staff | UC-44. Add Class |
| Staff | UC-45. Add List of Classes |
| Staff | UC-46. Update Class |
| Staff | UC-47. Remove Class |
| Staff | UC-48. Search Classes |
| Staff | UC-51. Change Student’s Semester |
| Staff | UC-52. Update Student’s Learning Progress |
| Staff | UC-53. Student into Classes |
| Staff | UC-55. Create Timetable |
| Staff, Teacher | UC-56. Choose Teachers for Classes |
| Staff | UC-57. View All Timetable |
| Teacher, Student | UC-58. View Personal Timetable |
| Teacher | UC-59. View A Class Timetable |

### Business Rules

|  |  |
| --- | --- |
| **No** | **Description** |
| **B1** | The account must not be empty |
| **B2** | The account must be the account of mail |
| **B3** | The e-mail must not be empty |
| **B4** | The e-mail must be fpt university’s mail (fpt.edu.vn) |
| **B5** | Room’s code |
| **B6** | Room’s type |
| **B7** | Room’s capacity |
| **B8** | Semester’s Code |
| **B9** | Semester’s Name |
| **B10** | Semester’s Finish Date |
| **B11** | Student number of a class can not exceed 30 |

### Use Cases

#### Admin

##### UC-01. Login

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-01. Login | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff, Teacher, Student** | Secondary Actors: | N/A |
| Trigger: | Actor wants to login to Course Scheduling. | | |
| Description: | Actor logins to website to use its function. | | |
| Pre-conditions: | Actor has a fpt’s mail (…@fpt.edu.vn) and this account was registered in server. | | |
| Post-conditions: | Actor logins successfully. | | |
| Normal Flow: | **UC – 01. Login.**   1. Actor clicks “Login” button. 2. A dialog from google will be shown. 3. Actor choose an account they will use to login. 4. System indicates this account. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | * 1. **Account Not Found Exception**  1. Course-Scheduling will show a dialog to inform the account does not exist. | | |
| Priority: | Low | | |
| Frequency of Use: | Medium | | |
| Business Rules: | **N/A** | | |
| Other Information: | N/A | | |
| Assumptions: | N/A | | |

##### UC-02. Logout

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-01. Logout | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff, Teacher, Student** | Secondary Actors: | N/A |
| Trigger: | Actor wants to logout to Course Scheduling. | | |
| Description: | Actor logouts to website after using its function. | | |
| Pre-conditions: | 1. Actor logged in successfully. | | |
| Post-conditions: | Actor logouts successful. | | |
| Normal Flow: | **UC – 02. Logout.**   1. In main page, Actor clicks “Logout” button. 2. Actor logout. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | N/A | | |
| Priority: | Low | | |
| Frequency of Use: | Low | | |
| Business Rules: | **N/A** | | |
| Other Information: | N/A | | |
| Assumptions: | N/A | | |

##### UC-03. View All Staffs

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-03. View All Staffs | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the staffs. | | |
| Description: | Actor uses this function to see all the staffs. | | |
| Pre-conditions: | **PRE – 03.01.** Actor is logged to Course-Scheduling Website.  **PRE – 03.02.** Actor is set as the admin of the Website. | | |
| Post-conditions: | **POST – 03.01.** Course-Scheduling shows list of all staffs. | | |
| Normal Flow: | **UC – 03. View All Staffs**   1. Actor clicks to menu “*Staffs Management*”. 2. Course-Scheduling shows list of all Staffs. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-04. Add Staff

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-04. Add Staff | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new staff to use the web. | | |
| Description: | Actor uses this function to add new staff to use the web. | | |
| Pre-conditions: | **PRE – 04.01.** Actor is logged to Course-Scheduling Website.  **PRE – 04.02.** Actor is set as the staff of the Website.  **PRE – 04.03.** Actor accessed to “*Staffs Management*” page. | | |
| Post-conditions: | **POST – 04.01.** The staff was added will be shown in list of all staffs of Staff page. | | |
| Normal Flow: | **UC – 04. Add Staff**   1. Actor clicks to button "Add Staff”. 2. The actor enters information about new staff: “Account”, “Name”, “E-mail”. 3. Actor click “Save” button. 4. Course-Scheduling will validate all information, if it’s correct, the staff will be added to database. 5. Course-Scheduling redirects to Staff page with message “*Add Staff Successful.*”. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **B1, B2, B3, B4** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-05. Remove Staff

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-05. Remove Staff | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a staff. | | |
| Description: | Actor uses this function to remove a staff. | | |
| Pre-conditions: | **PRE – 05.01.** Actor is logged to Course-Scheduling Website.  **PRE – 05.02.** Actor is set as the staff of the Website.  **PRE – 05.03.** Actor accessed to “*Staffs Management*” page. | | |
| Post-conditions: | **POST – 05.01.** The staff was removed must not be shown in list of all staffs of Staff page. | | |
| Normal Flow: | **UC – 05. Remove Staff**   1. Actor clicks “Delete” of a staff in list of staffs. 2. Course-Scheduling asks for confirmation to remove this staff. 3. Actor chooses “Accept” 4. Course-Scheduling removes this staff in database. 5. Course-Scheduling redirects to Staff page with message “*Remove Staff Successful.*”. | | |
| Alternative Flows: | **AF – 05.01. At step 5, Actor chooses “Decline”**   1. Actor chooses “Decline” in the dialog. 2. Course-Scheduling redirects to Staff page. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-06. Search Staffs

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-06. Search Staff | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific staff. | | |
| Description: | Actor uses this function to search for a specific staff in the website. | | |
| Pre-conditions: | **PRE – 06.01.** Actor is logged to Course-Scheduling Website.  **PRE – 06.02.** Actor is set as the staff of the Website.  **PRE – 06.03.** Actor accessed to “*Staffs Management*” page. | | |
| Post-conditions: | **POST – 06.01.** The staffs have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 06. Search** **Staff**   1. Actor enters information about the staff that they want to find in search box. 2. Course-Scheduling shows list of staffs that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-07. View All Buildings

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-07. View All Buildings | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all buildings. | | |
| Description: | Actor uses this function to see all buildings. Actor can access to “*Buildings Management*” page in many ways:  . In “*Main*” page, click to menu “*Buildings Management*”.  . In “*General Management*” pages (*“Staffs Management”, “Rooms Management”, “Specialized Management”, “Semesters Management”, “Departments Management”, “Programs Management”, “Buildings Management”)* click to tab “Building” in sidebar. | | |
| Pre-conditions: | **PRE – 07.01.** Actor is logged to Course-Scheduling Website.  **PRE – 07.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 07.01.** Course-Scheduling shows list of all buildings. | | |
| Normal Flow: | **UC – 07. View All Staffs**   1. Actor clicks to menu “*Buildings Management*”. 2. Course-Scheduling shows list of all *Buildings*. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-08. Add Building

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-08. Add Room | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new room. | | |
| Description: | Actor uses this function to add new room. | | |
| Pre-conditions: | **PRE – 08.01.** Actor is logged to Course-Scheduling Website.  **PRE – 08.02.** Actor is set as the staff of the Website.  **PRE – 08.03.** Actor accessed to “*Buildings Management*” page. | | |
| Post-conditions: | **POST – 08.01.** The room was added will be shown in list of all rooms in “*Buildings Management*” page. | | |
| Normal Flow: | **UC – 08. Add Building**   1. In Building page, Actor clicks to button "Add Building”. 2. The actor enters information about new Building: “Code. 3. Actor clicks save “button”. 4. Course-Scheduling will validate all information, if it’s correct, the Room will be added to database. 5. Course-Scheduling redirects to “*Buildings Management*” with message “*Add* Building *Successful.*”. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **08.01 Building Existed Exception (Duplicate code).**  . Course-Scheduling redirects to “*Buildings Management”* with error message *“This Building’s code existed! Please Try Again!”.*  **08.02 Code input is empty.**  . Course-Scheduling show message “*Your Code can not be empty!!!*” | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-09. Remove Building

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-09. Remove Building | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a building. | | |
| Description: | Actor uses this function to remove a building. | | |
| Pre-conditions: | **PRE – 09.01.** Actor is logged to Course-Scheduling Website.  **PRE – 09.02.** Actor is set as the staff of the Website.  **PRE – 09.03.** Actor accessed to “*Buildings Management*” page. | | |
| Post-conditions: | **POST – 09.01.** The building was removed must not be shown in list of all buildings of “*Buildings Management*” page. | | |
| Normal Flow: | **UC – 09. Remove Building**   1. Actor clicks to “Delete” in the row of a building in list of buildings. 2. Course-Scheduling asks for confirmation to remove this building. 3. Actor chooses “Accept” 4. Course-Scheduling removes this building in database. 5. Redirect to “*Buildings Management*” page with message: “*Remove Building successfully*”. | | |
| Alternative Flows: | **AF – 09.01. At step 3, Actor chooses “Decline”**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-10. Search Buildings

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-10. Search Buildings | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific building. | | |
| Description: | Actor uses this function to search for a specific building. | | |
| Pre-conditions: | **PRE – 10.01.** Actor is logged to Course-Scheduling Website.  **PRE – 10.02.** Actor is set as the admin or staff of the Website.  **PRE – 10.03** Actor accessed to “*Buildings Management*” page. | | |
| Post-conditions: | **POST – 10.01.** The buildings have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 10. Search** **Buildings**   1. Actor enters information about the building that they want to find in the search box. 2. Course-Scheduling shows list of buildings that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-11. View All Rooms

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-11. View All Rooms | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the rooms. | | |
| Description: | Actor uses this function to see all the rooms. Actor can access to “*Rooms Management*” page in many ways:  . In “*Main*” page, click to menu “*Rooms Management*”.  . In “*General Management*” pages (*“Staffs Management”, “Rooms Management”, “Specialized Management”, “Semesters Management”, “Departments Management”, “Programs Management”, “Buildings Management”)* click to tab “*Rooms*” in sidebar. | | |
| Pre-conditions: | **PRE – 11.01.** Actor is logged to Course-Scheduling Website.  **PRE – 11.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 11.01.** Course-Scheduling shows list of all rooms. | | |
| Normal Flow: | **UC – 11. View All Rooms**   1. Actor clicks to menu “*Rooms Management*”. 2. Course-Scheduling shows list of all Rooms. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-12. Add Room

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-12. Add Room | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new room. | | |
| Description: | Actor uses this function to add new room. | | |
| Pre-conditions: | **PRE – 12.01.** Actor is logged to Course-Scheduling Website.  **PRE – 12.02.** Actor is set as the staff of the Website.  **PRE – 12.03.** Actor accessed to “*Rooms Management*” page. | | |
| Post-conditions: | **POST – 12.01.** The room was added will be shown in list of all rooms in “*Rooms Management*” page. | | |
| Normal Flow: | **UC – 12. Add Room**   1. In Room page, Actor clicks to button "Add Room”. 2. The actor enters information about new room: “Building”, “Code”, “Capacity” and some specific “course” will be learnt in this room. 3. Actor clicks save “button”. 4. Course-Scheduling will validate all information, if it’s correct, the Room will be added to database. 5. Course-Scheduling redirects to Room page with message “*Add Room Successful.*”. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **12.01 Room Existed Exception (Duplicate code)**  . Course-Scheduling redirects to “*Rooms Management*” page with error message “*This Room’s code existed! Please try again!*”.  **12.02 Room’s code input is empty.**  . Course-Scheduling shows message *“Your Code can not be empty!!!”.*  **12.03 Room’s capacity input is empty.**  . Course-Scheduling shows message *“Your Capacity can not be empty!!!”.* | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **B5, B6, B7** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-13. Add List of Rooms

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-13. Add List of Rooms | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add List of new room. | | |
| Description: | Actor uses this function to add List of new room. | | |
| Pre-conditions: | **PRE – 13.01.** Actor is logged to Course-Scheduling Website.  **PRE – 13.02.** Actor is set as the staff of the Website.  **PRE – 13.03.** Actor accessed to “*Rooms Management*” page. | | |
| Post-conditions: | **POST – 13.01.** The rooms were added will be shown in list of all rooms of “*Rooms Management*” page. | | |
| Normal Flow: | **UC – 13. Add List of Rooms**   1. In Room page, Actor clicks to button "Add List of Rooms”. 2. Course-Scheduling shows a dialog with an input to enter the link of excel file contains all information about rooms. 3. Actor enters the link of file. 4. Actor choose “Add”. 5. Course-Scheduling will validate all information, if it’s correct, the Rooms be added to database. 6. Course-Scheduling redirects to Room page with a message “*Import List of Rooms successful.*“. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **13.01. Building Not Found Exception.**  Course-Scheduling redirects to “*Rooms Management*” page with error message *“Building … are not found! Please Insert It First!”* | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **B5, B6, B7** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-14. Edit Room

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-14. Edit Room | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to update information of a room. | | |
| Description: | Actor uses this function to update information of a room. | | |
| Pre-conditions: | **PRE – 14.01.** Actor is logged to Course-Scheduling Website.  **PRE – 14.02.** Actor is set as the admin or staff of the Website.  **PRE – 14.03.** Actor accessed to “*Rooms Management*” page. | | |
| Post-conditions: | **POST – 14.01.** The information was edited will be saved to database. | | |
| Normal Flow: | **UC – 14. Edit Room**   1. Actor clicks to “Edit” of a staff in list of rooms. 2. Course-Scheduling shows a dialog with information of this room for actor to edit. “Capacity” and “Course”. 3. Actor edit information of this room. 4. Actor click “Save”. 5. Course-Scheduling closes dialog and saves information. | | |
| Alternative Flows: | **AF – 14.01. At step 4, Actor chooses “cancel”.**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-15. Remove Room

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-15. Remove Room | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a room. | | |
| Description: | Actor uses this function to remove a room. | | |
| Pre-conditions: | **PRE – 15.01.** Actor is logged to Course-Scheduling Website.  **PRE – 15.02.** Actor is set as the staff of the Website.  **PRE – 15.03.** Actor accessed to “*Rooms Management*” page. | | |
| Post-conditions: | **POST – 15.01.** The room was removed must not be shown in list of all rooms of “*Rooms Management*” page. | | |
| Normal Flow: | **UC – 15. Remove Room**   1. Actor clicks to “Delete” in the row of a room in list of rooms. 2. Course-Scheduling asks for confirmation to remove this room. 3. Actor chooses “Accept” 4. Course-Scheduling removes this room in database. 5. Redirect to Room page with message: “*Remove Room successful!*”. | | |
| Alternative Flows: | **AF – 15.01. At step 3, Actor chooses “Decline”**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-16. Search Rooms

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-16. Search Room | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific room. | | |
| Description: | Actor uses this function to search for a specific room. | | |
| Pre-conditions: | **PRE – 16.01.** Actor is logged to Course-Scheduling Website.  **PRE – 16.02.** Actor is set as the admin or staff of the Website.  **PRE – 16.03** Actor accessed to “*Rooms Management*” page. | | |
| Post-conditions: | **POST – 16.01.** The rooms have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 16. Search** **Room**   1. Actor enters information about the room that they want to find in the search box. 2. Course-Scheduling shows list of rooms that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-17. View All Semesters

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-17. View All Semesters | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the Semesters. Actor can access to “*Semesters Management*” page in many ways:  . In “*Main*” page, click to menu “*Semesters Management*”.  . In “*General Management*” pages (*“Staffs Management”, “Rooms Management”, “Specialized Management”, “Semesters Management”, “Departments Management”, “Programs Management”, “Buildings Management”)* click to tab “*Semesters*” in sidebar. | | |
| Description: | Actor uses this function to see all the Semesters. | | |
| Pre-conditions: | **PRE – 17.01.** Actor is logged to Course-Scheduling Website.  **PRE – 17.02.** Actor is set as the admin of the Website. | | |
| Post-conditions: | **POST – 17.01.** Course-Scheduling shows list of all Semesters. | | |
| Normal Flow: | **UC – 17. View All Semesters**   1. Actor clicks to menu “*Semesters Management*”. 2. Course-Scheduling shows list of all Semesters. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-18. Add Semester

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-18. Add Semester | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new Semester. | | |
| Description: | Actor uses this function to add new Semester. | | |
| Pre-conditions: | **PRE – 18.01.** Actor is logged to Course-Scheduling Website.  **PRE – 18.02.** Actor is set as the staff of the Website.  **PRE – 18.03.** Actor accessed to “*Semesters Management*” page. | | |
| Post-conditions: | **POST – 18.01.** The staff was added will be shown in list of all Semesters of “*Semesters Management*” page. | | |
| Normal Flow: | **UC – 18. Add Semester**   1. In “*Semesters Management*” page, Actor clicks to button "Add Semester”. 2. The actor enters information about new Semester: “Code”, “Name”, “Start Date”, “End Date”. 3. Actor clicks “Save” button. 4. Course-Scheduling will validate all information, if it’s correct, the Semester will be added to database. 5. Course-Scheduling redirects to “*Semesters Management*” page with message “*Add Semester Successful*!” | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **18.01 Semester Existed Exception (Duplicate code).**  **.** Course-Scheduling redirects to “*Semesters Management”* page with error message “*This Semester’s code existed! Please Try Again!*”.  **18.02 Semester’s code input is empty.**  . Course-Scheduling shows message “*Your Code can not be empty!!!*”.  **18.03 Semester’s name input is empty.**  **.** Course-Scheduling shows message “*Your Name can not be empty!!!*”.  **18.04 Semester’s start date input is empty.**  . Course-Scheduling shows message “*Your Start Date can not be empty!!!*”.  **18.05 Semester’s end date input is empty.**  . Course-Scheduling shows message “*Your End Date can not be empty!!!*”.  **18.06 Semester’s end date is a previous date of Semester’s start date.**  . Course-Scheduling shows message “*Your End Date must be greater than Start Date!!!*”. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **B8, B9, B10** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-19. Remove Semester

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-16. Remove Semester | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a semester. | | |
| Description: | Actor uses this function to remove a semester. | | |
| Pre-conditions: | **PRE – 19.01.** Actor is logged to Course-Scheduling Website.  **PRE – 19.02.** Actor is set as the staff of the Website.  **PRE – 19.03.** Actor accessed to “*Semesters Management*” page. | | |
| Post-conditions: | **POST – 19.01.** The semester was removed must not be shown in list of all semesters of “*Semesters Management*” page. | | |
| Normal Flow: | **UC – 19. Remove Semester**   1. Actor clicks to “Delete” in the row of a semester in list of Semesters. 2. Course-Scheduling asks for confirmation to remove this Semester. 3. Actor chooses “Accept” 4. Course-Scheduling removes this Semester in database. 5. Redirect to “*Semesters Management*” page with message: “*Remove Semester successful!*”. | | |
| Alternative Flows: | **AF – 19.01. At step 3, Actor chooses “Decline”**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-20. Search Semester

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-20. Search Semester | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific Semester. | | |
| Description: | Actor uses this function to search for a specific Semester in the website. | | |
| Pre-conditions: | **PRE – 20.01.** Actor is logged to Course-Scheduling Website.  **PRE – 20.02.** Actor is set as the admin or staff of the Website.  **PRE – 20.03.** Actor accessed to “*Semesters Management*” page. | | |
| Post-conditions: | **POST – 20.01.** The Semesters have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 20. Search** **Semester**   1. Actor enters information about the Semester that they want to find in search box. 2. Course-Scheduling shows list of Semesters that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-21. View All Departments

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-17. View All Departments | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the Semesters. Actor can access to “*Departments Management*” page in many ways:  . In “*Main*” page, click to menu “*Departments Management*”.  . In “*General Management*” pages (*“Staffs Management”, “Rooms Management”, “Specialized Management”, “Semesters Management”, “Departments Management”, “Programs Management”, “Buildings Management”)* click to tab “*Semesters*” in sidebar. | | |
| Description: | Actor uses this function to see all the Departments. | | |
| Pre-conditions: | **PRE – 17.01.** Actor is logged to Course-Scheduling Website.  **PRE – 17.02.** Actor is set as the admin of the Website. | | |
| Post-conditions: | **POST – 17.01.** Course-Scheduling shows list of all Departments. | | |
| Normal Flow: | **UC – 17. View All Semesters**   1. Actor clicks to menu “*Departments Management*”. 2. Course-Scheduling shows list of all Departments. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-22. Add Department

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-22. Add Department | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new Department. | | |
| Description: | Actor uses this function to add new Department. | | |
| Pre-conditions: | **PRE – 22.01.** Actor is logged to Course-Scheduling Website.  **PRE – 22.02.** Actor is set as the staff of the Website.  **PRE – 22.03.** Actor accessed to “*Departments Management*” page. | | |
| Post-conditions: | **POST – 22.01.** The staff was added will be shown in list of all Semesters of “*Departments Management*” page. | | |
| Normal Flow: | **UC – 22. Add Semester**   1. In “*Departments Management*” page, Actor clicks to button "Add Department”. 2. The actor enters information about new Department: “Code”, “Name”. 3. Actor clicks “Save” button. 4. Course-Scheduling will validate all information, if it’s correct, the Semester will be added to database. 5. Course-Scheduling redirects to “*Departments Management*” page with message “*Add Department Successful*!” | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **22.01 Department Existed Exception (Duplicate code).**  **.** Course-Scheduling redirects to “*Departments Management”* page with error message “*This Department’s code existed! Please Try Again!*”.  **22.02 Department’s code input is empty.**  . Course-Scheduling shows message “*Your Code can not be empty!!!*”.  **22.03 Department’s name input is empty.**  **.** Course-Scheduling shows message “*Your Name can not be empty!!!*”. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-23. Remove Department

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-23. Remove Department | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a department. | | |
| Description: | Actor uses this function to remove a department. | | |
| Pre-conditions: | **PRE – 23.01.** Actor is logged to Course-Scheduling Website.  **PRE – 23.02.** Actor is set as the staff of the Website.  **PRE – 23.03.** Actor accessed to “*Departments Management*” page. | | |
| Post-conditions: | **POST – 23.01.** The semester was removed must not be shown in list of all semesters of “*Departments Management*” page. | | |
| Normal Flow: | **UC – 23. Remove Department**   1. Actor clicks to “Delete” in the row of a semester in list of Departments. 2. Course-Scheduling asks for confirmation to remove this Department. 3. Actor chooses “Accept” 4. Course-Scheduling removes this Semester in database. 5. Redirect to “*Departments Management*” page with message: “*Remove Department successful!*”. | | |
| Alternative Flows: | **AF – 23.01. At step 3, Actor chooses “Decline”**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-24. Search Department

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-24. Search Department | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific Department. | | |
| Description: | Actor uses this function to search for a specific Department in the website. | | |
| Pre-conditions: | **PRE – 24.01.** Actor is logged to Course-Scheduling Website.  **PRE – 24.02.** Actor is set as the admin or staff of the Website.  **PRE – 24.03.** Actor accessed to “*Departments Management*” page. | | |
| Post-conditions: | **POST – 24.01.** The Departments have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 24. Search** **Department**   1. Actor enters information about the Semester that they want to find in search box. 2. Course-Scheduling shows list of Semesters that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-25. View All Teachers

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-25. View All Teachers | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the Teachers. | | |
| Description: | Actor uses this function to see all the Teachers. | | |
| Pre-conditions: | **PRE – 25.01.** Actor is logged to Course-Scheduling Website.  **PRE – 25.02.** Actor is set as the admin or staff of the Website.  **PRE – 25.03.** Actor accessed to “*Teachers Management*” page. | | |
| Post-conditions: | **POST – 25.01.** Course-Scheduling shows list of all Teachers. | | |
| Normal Flow: | **UC – 25. View All Teachers**   1. Actor choose Semester in left panel. 2. Course-Scheduling shows list of all Teachers in the semester was chosen. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-26. Add Teacher

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-26. Add Teacher | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new teacher. | | |
| Description: | Actor uses this function to add new room. | | |
| Pre-conditions: | **PRE – 26.01.** Actor is logged to Course-Scheduling Website.  **PRE – 26.02.** Actor is set as the staff of the Website.  **PRE – 26.03.** Actor accessed to “*Teacher Management*” page. | | |
| Post-conditions: | **POST – 26.01.** The teacher was added will be shown in list of all teachers of Teacher page. | | |
| Normal Flow: | **UC – 26. Add Teacher**   1. In Teacher page, Actor clicks to button "Add Teacher”. 2. The actor enters information about new room: “Code”, “Capacity” and some specific “course” will be learnt in this room. 3. Actor click “Save” button. 4. Course-Scheduling will validate all information, if it’s correct, the Room will be added to database. 5. Course-Scheduling redirects to Room page with message “*Add Teacher Successful.*”. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **B1, B2, B3, B4** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-27. Add List of Teachers

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-27. Add List of Teachers | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add List of new teachers. | | |
| Description: | Actor uses this function to add List of new teachers. | | |
| Pre-conditions: | **PRE – 27.01.** Actor is logged to Course-Scheduling Website.  **PRE – 27.02.** Actor is set as the staff of the Website.  **PRE – 27.03.** Actor accessed to “*Teacher Management*” page. | | |
| Post-conditions: | **POST – 27.01.** The teachers were added will be shown in list of all staffs of Staff page. | | |
| Normal Flow: | **UC – 27. Add List of Rooms**   1. In “*Teacher Management*” page, Actor clicks to button "Add List of Teachers”. 2. Course-Scheduling shows a dialog with an input to enter the link of excel file contains all information about rooms. 3. Actor selects “Add Teachers” option. 4. Actor enters the link of file. 5. Actor choose “Add”. 6. Course-Scheduling will validate all information, if it’s correct, the Rooms be added to database. 7. Course-Scheduling redirects to Teacher page with a message “*Import List of Teachers successful*!“. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **B5, B6, B7** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-28. Add Courses For Teachers

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-28. Add Courses For Teachers | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add List of new teachers. | | |
| Description: | Actor uses this function to add List of new teachers. | | |
| Pre-conditions: | **PRE – 28.01.** Actor is logged to Course-Scheduling Website.  **PRE – 28.02.** Actor is set as the staff of the Website.  **PRE – 28.03.** Actor accessed to “*Teacher Management*” page. | | |
| Post-conditions: | **POST – 28.01.** The teachers were added will be shown in list of all staffs of Staff page. | | |
| Normal Flow: | **UC – 28. Add Courses For Teachers**   1. In “*Teacher Management*” page, Actor clicks to button "Add List of Teachers”. 2. Course-Scheduling shows a dialog with an input to enter the link of excel file contains all information about rooms. 3. Actor selects “Add Courses For Teachers” option. 4. Actor enters the link of file. 5. Actor choose “Add”. 6. Course-Scheduling will validate all information, if it’s correct, the Rooms be added to database. 7. Course-Scheduling redirects to Teacher page with a message “*Import Courses For Teachers successful*!“. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **28.01 Course Not Found Exception.**  Course-Scheduling redirects to Room Management with message *“Course … not found. Please Insert It First.”* | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-29. Edit Teacher

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-29. Edit Teacher | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to update information of a teacher. | | |
| Description: | Actor uses this function to update information of a teacher. | | |
| Pre-conditions: | **PRE – 29.01.** Actor is logged to Course-Scheduling Website.  **PRE – 29.02.** Actor is set as the staff of the Website.  **PRE – 29.03.** Actor accessed to “*Teacher Management*” page. | | |
| Post-conditions: | **POST – 29.01.** The information was edited will be saved to database. | | |
| Normal Flow: | **UC – 29. Edit Teacher**   1. Actor clicks to “Edit” of a staff in list of teachers. 2. Course-Scheduling shows a dialog with information of this teacher for actor to edit. (Edit courses the teacher will teach). 3. Actor edit information of this teacher. 4. Actor click “Save”. 5. Course-Scheduling closes dialog and saves information. 6. Course-Scheduling redirects to “*Teacher Management*” with message “*Edit Teacher’s Information Successful!*”. | | |
| Alternative Flows: | **AF – 29.01. At step 4, Actor chooses “cancel”.**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-30. Remove Teacher

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-30. Remove Teacher | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a teacher. | | |
| Description: | Actor uses this function to remove a teacher. | | |
| Pre-conditions: | **PRE – 30.01.** Actor is logged to Course-Scheduling Website.  **PRE – 30.02.** Actor is set as the staff of the Website.  **PRE – 30.03.** Actor accessed to “*Teacher Management*” page. | | |
| Post-conditions: | **POST – 30.01.** The teacher was removed must not be shown in list of all teachers of Teacher page. | | |
| Normal Flow: | **UC – 30. Remove Room**   1. Actor clicks to “Delete” in the row of a teacher in list of teachers. 2. Course-Scheduling asks for confirmation to remove this teacher. 3. Actor chooses “Accept”. 4. Course-Scheduling removes this teacher in database. 5. Redirect to “*Teacher Management*” page with message: “*Remove Teacher successfully*”. | | |
| Alternative Flows: | **AF – 30.01. At step 3, Actor chooses “Cancel”**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-31. Search Teachers

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-31. Search Teachers | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific teacher. | | |
| Description: | Actor uses this function to search for a specific teacher. | | |
| Pre-conditions: | **PRE – 31.01.** Actor is logged to Course-Scheduling Website.  **PRE – 31.02.** Actor is set as the staff of the Website.  **PRE – 31.03** Actor accessed to “*Teacher Management*” page. | | |
| Post-conditions: | **POST – 31.01.** The teachers have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 31. Search** **Room**   1. Actor enters information about the teacher that they want to find in the search box. 2. Course-Scheduling shows list of teachers that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-32. View All Students

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-32. View All Students | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the students. | | |
| Description: | Actor uses this function to see all the students. | | |
| Pre-conditions: | **PRE – 32.01.** Actor is logged to Course-Scheduling Website.  **PRE – 32.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 32.01.** Course-Scheduling shows list of all students. | | |
| Normal Flow: | **UC – 32. View All Staffs**   1. Actor clicks to menu “*Students Management*”. 2. Course-Scheduling shows list of all Studentss. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-33. Add Student

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-33. Add Student | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new student. | | |
| Description: | Actor uses this function to add new student. | | |
| Pre-conditions: | **PRE – 33.01.** Actor is logged to Course-Scheduling Website.  **PRE – 33.02.** Actor is set as the staff of the Website.  **PRE – 33.03.** Actor accessed to “*Students Management*” page. | | |
| Post-conditions: | **POST – 33.01.** The student was added will be shown in list of all students of Student page. | | |
| Normal Flow: | **UC – 33. Add Student**   1. In “*Students Management*” page, Actor clicks to button "Add Student”. 2. The actor enters information about new student: “Name”, “Specialized”, “Batch”, “Semester”. 3. Actor click “Save” button. 4. Course-Scheduling will validate all information, if it’s correct, the Room will be added to database. 5. Course-Scheduling redirects to “*Students Management*” page with message: “*Add Student Successful*.”. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-34. Add List of Students

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-34. Add List of Students | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add List of new student. | | |
| Description: | Actor uses this function to add List of new student. | | |
| Pre-conditions: | **PRE – 34.01.** Actor is logged to Course-Scheduling Website.  **PRE – 34.02.** Actor is set as the staff of the Website.  **PRE – 34.03.** Actor accessed to “*Students Management*” page. | | |
| Post-conditions: | **POST – 34.01.** The students were added will be shown in list of all students of Student page. | | |
| Normal Flow: | **UC – 34. Add List of Students**   1. In “*Students Management*” page, Actor clicks to button "Add List of Students”. 2. Course-Scheduling shows a dialog with an input to enter the link of excel file contains all information about students. 3. Actor enters the link of file. 4. Actor choose “Add”. 5. Course-Scheduling will validate all information, if it’s correct, Students will be added to database. 6. Course-Scheduling redirects to “*Students Management*” page with a message “*Import List of Students successful!*“. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **34.01. Specialized not found Exception**  Course-Scheduling redirects to “*Students Management*” with message *“Specialized … not found. Please Insert It First.”* | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-35. Edit Student

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-35. Edit Student | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to update information of a student. | | |
| Description: | Actor uses this function to update information of a student. | | |
| Pre-conditions: | **PRE – 35.01.** Actor is logged to Course-Scheduling Website.  **PRE – 35.02.** Actor is set as the admin or staff of the Website.  **PRE – 35.03.** Actor accessed to “*Students Management*” page. | | |
| Post-conditions: | **POST – 35.01.** The information was edited will be saved to database. | | |
| Normal Flow: | **UC – 35. Edit Student**   1. Actor clicks to “Edit” of a staff in list of students. 2. Course-Scheduling shows a dialog with information of this student for actor to edit. 3. Actor edit information of this student. 4. Actor click “Save”. 5. Course-Scheduling closes dialog and saves information. | | |
| Alternative Flows: | **AF – 35.01. At step 4, Actor chooses “cancel”.**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-36. Remove Student

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-36. Remove Student | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a student. | | |
| Description: | Actor uses this function to remove a student. | | |
| Pre-conditions: | **PRE – 36.01.** Actor is logged to Course-Scheduling Website.  **PRE – 36.02.** Actor is set as the staff of the Website.  **PRE – 36.03.** Actor accessed to “*Students Management*” page. | | |
| Post-conditions: | **POST – 36.01.** The student was removed must not be shown in list of all students of Student page. | | |
| Normal Flow: | **UC – 36. Remove Student**   1. Actor clicks to “Delete” in the row of a room in list of students. 2. Course-Scheduling asks for confirmation to remove this students. 3. Actor chooses “Accept” 4. Course-Scheduling removes this student in database. 5. Redirect to “*Students Management*” page with message: “*Remove Student successful!”.* | | |
| Alternative Flows: | **AF – 36.01. At step 3, Actor chooses “Cancel”**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-37. Search Students

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-37. Search Students | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific student. | | |
| Description: | Actor uses this function to search for a specific student. | | |
| Pre-conditions: | **PRE – 37.01.** Actor is logged to Course-Scheduling Website.  **PRE – 37.02.** Actor is set as the admin or staff of the Website.  **PRE – 37.03** Actor accessed to “*Students Management*” page. | | |
| Post-conditions: | **POST – 37.01.** The students have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 37. Search** **Student**   1. Actor enters information about the room that they want to find in the search box. 2. Course-Scheduling shows list of students that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-38. View All Courses

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-38. View All Courses | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the courses. | | |
| Description: | Actor uses this function to see all the courses. | | |
| Pre-conditions: | **PRE – 38.01.** Actor is logged to Course-Scheduling Website.  **PRE – 38.02.** Actor is set as the staff of the Website.  **PRE – 38.03.** Actor accessed to “*Courses Management*” page. | | |
| Post-conditions: | **POST – 38.01.** Course-Scheduling shows list of all courses. | | |
| Normal Flow: | **UC – 38. View All Staffs**   1. Actor clicks to menu “*Courses Management*”. 2. Actor choose Semester in left panel. 3. Course-Scheduling shows list of all Courses. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-39. Add Course

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-39. Add Course | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new course. | | |
| Description: | Actor uses this function to add new course. | | |
| Pre-conditions: | **PRE – 39.01.** Actor is logged to Course-Scheduling Website.  **PRE – 39.02.** Actor is set as the admin, staff of the Website.  **PRE – 39.03.** Actor accessed to “*Courses Management*” page. | | |
| Post-conditions: | **POST – 39.01.** The course was added will be shown in list of all courses of Course page. | | |
| Normal Flow: | **UC – 39. Add Course**   1. In “*Courses Management*” page, Actor clicks to button "Add Course”. 2. The actor enters information about new room: “Code”, “Name”, “Slots”, “Semester”, “Department” and “Course Condition” if needed. 3. Actor clicks “Save” button. 4. Course-Scheduling will validate all information, if it’s correct, the Course will be added to database. 5. Course-Scheduling redirects to Room page with message “Add Course Successful.”. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **39.01. Can not connect to server.**  Browser shows error message. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: |  | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-40. Add List of Courses

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-40 Add List of Courses | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add List of new courses. | | |
| Description: | Actor uses this function to add List of new courses. | | |
| Pre-conditions: | **PRE – 40.01.** Actor is logged to Course-Scheduling Website.  **PRE – 40.02.** Actor is set as the staff of the Website.  **PRE – 40.03.** Actor accessed to “*Courses Management*” page. | | |
| Post-conditions: | **POST – 40.01.** The courses were added will be shown in list of all courses of Course page. | | |
| Normal Flow: | **UC – 40. Add List of Courses**   1. In Courses page, Actor clicks to button "Add List of Courses”. 2. Course-Scheduling shows a dialog with an input to enter the link of excel file contains all information about courses. 3. Actor enters the link of file. 4. Actor choose “Add”. 5. Course-Scheduling will validate all information, if it’s correct, the Courses be added to database. 6. Course-Scheduling redirects to “*Course Management*” page with a message “*Import List of Courses successful!*“. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **B10** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-41. Edit Course

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-41. Edit Course | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to update information of a course. | | |
| Description: | Actor uses this function to update information of a course. | | |
| Pre-conditions: | **PRE – 41.01.** Actor is logged to Course-Scheduling Website.  **PRE – 41.02.** Actor is set as the admin or staff of the Website.  **PRE – 41.03.** Actor accessed to “*Courses Management*” page. | | |
| Post-conditions: | **POST – 41.01.** The information was edited will be saved to database. | | |
| Normal Flow: | **UC – 41. Edit Course**   1. Actor clicks to “Edit” of a staff in list of courses. 2. Course-Scheduling shows a dialog with information of this courses for actor to edit. 3. Actor edit information of this course. 4. Actor click “Save”. 5. Course-Scheduling closes dialog and saves information. 6. Course-Scheduling redirects to “*Courses Management*” Page with message “*Edit Course’s information successful!*” | | |
| Alternative Flows: | **AF – 41.01. At step 4, Actor chooses “cancel”.**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-42. Remove Course

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-42. Remove Course | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a course. | | |
| Description: | Actor uses this function to remove a course. | | |
| Pre-conditions: | **PRE – 42.01.** Actor is logged to Course-Scheduling Website.  **PRE – 42.02.** Actor is set as the staff of the Website.  **PRE – 42.03.** Actor accessed to “*Courses Management*” page. | | |
| Post-conditions: | **POST – 42.01.** The room was removed must not be shown in list of all courses of Course page. | | |
| Normal Flow: | **UC – 42. Remove Room**   1. Actor clicks to “Delete” in the row of a room in list of rooms. 2. Course-Scheduling asks for confirmation to remove this course. 3. Actor chooses “Accept” 4. Course-Scheduling removes this course in database. 5. Redirect to “*Courses Management*” page with message: “*Remove Course successfully*”. | | |
| Alternative Flows: | **AF – 42.01. At step 3, Actor chooses “Cancel”**   1. Dialog is closed. 2. Nothing will be saved. | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-43. Search Courses

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-43. Search Courses | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific course. | | |
| Description: | Actor uses this function to search for a specific course. | | |
| Pre-conditions: | **PRE – 43.01.** Actor is logged to Course-Scheduling Website.  **PRE – 43.02.** Actor is set as the admin or staff of the Website.  **PRE – 43.03** Actor accessed to “*Courses Management*” page. | | |
| Post-conditions: | **POST – 43.01.** The courses have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 43. Search** **Courses**   1. Actor enters information about the course that they want to find in the search box. 2. Course-Scheduling shows list of courses that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-44. View All Classes

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-44. View All Classes | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the classes. | | |
| Description: | Actor uses this function to see all the classes. | | |
| Pre-conditions: | **PRE – 44.01.** Actor is logged to Course-Scheduling Website.  **PRE – 44.02.** Actor is set as the staff of the Website.  **PRE – 44.03.** Actor accessed to “*Classes Management*” page. | | |
| Post-conditions: | **POST – 44.01.** Course-Scheduling shows list of all classes. | | |
| Normal Flow: | **UC – 44. View All Staffs**   1. Actor clicks to menu “*Classes Management*”. 2. Actor choose Semester in left panel. 3. Course-Scheduling shows list of all Classes in the semester selected. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-45. Add Class

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-45. Add Class | | |
| Created By: | **DucHM** | Date Created: | **2015-06-18** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new Class. | | |
| Description: | Actor uses this function to add new Class. | | |
| Pre-conditions: | **PRE – 45.01.** Actor is logged to Course-Scheduling Website.  **PRE – 45.02.** Actor is set as the staff of the Website.  **PRE – 45.03.** Actor accessed to “*Classes Management*” page. | | |
| Post-conditions: | **POST – 45.01.** The class was added will be shown in list of all Classes of Class page. | | |
| Normal Flow: | **UC – 45. Add Class**   1. In “*Classes Management*” page, Actor clicks to button "Add Class”. 2. The actor enters information about new Class: “Type”, “Batch”, “Specialized/Detail Specialized or Course”. 3. The actor clicks to button “Save”. 4. Course-Scheduling will validate all information and generate class name, if it’s correct, the Class will be added to database. 5. Course-Scheduling redirect to “*Classes Management*” page with message: “*Add Class Successful!*” | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **45.01. Unexpected exception.**  **1.** Course-Scheduling shows error message.  **2.** Rollback transaction. | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-46. Add List of Classes

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-46. Add List of Classes | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add List of new classes. | | |
| Description: | Actor uses this function to add List of new classes. | | |
| Pre-conditions: | **PRE – 46.01.** Actor is logged to Course-Scheduling Website.  **PRE – 46.02.** Actor is set as the staff of the Website.  **PRE – 46.03.** Actor accessed to “*Classes Management*” page. | | |
| Post-conditions: | **POST – 46.01.** The students were added will be shown in list of all students of Student page. | | |
| Normal Flow: | **UC – 46. Add List of Classes**   1. In “*Classes Management*” page, Actor clicks to button "Add List of Classes”. 2. Course-Scheduling shows a dialog with an input to enter the link of excel file contains all information about students. 3. Actor enters the link of file. 4. Actor choose “Add”. 5. Course-Scheduling will validate all information, if it’s correct, Students will be added to database. 6. Course-Scheduling redirects to “*Classes Management*” page with a message “*Import List of Classes successful!*“. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-47. Remove Class

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-47. Remove Class | | |
| Created By: | **DucHM** | Date Created: | **2015-06-18** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a Class. | | |
| Description: | Actor uses this function to remove a Class. | | |
| Pre-conditions: | **PRE – 47.01.** Actor is logged to Course-Scheduling Website.  **PRE – 47.02.** Actor is set as the staff of the Website.  **PRE – 47.03.** Actor accessed to Manage Classes page. | | |
| Post-conditions: | **POST – 47.01.** The class was removed will not be shown in list of all Classes of Class page. | | |
| Normal Flow: | **UC – 47. Remove Class**   1. The actor clicks to button “Delete” corresponding to a class 2. The actor clicks to button “Yes” in confirm popup. 3. Course-Scheduling will validate all information, if it’s correct, the Class will be remove from database. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **47.01. Unexpected exception.**  **1.** Course-Scheduling shows error message.  **2.** Rollback transaction. | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-48. Search Classes

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-48. Search Classes | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific class. | | |
| Description: | Actor uses this function to search for a specific class. | | |
| Pre-conditions: | **PRE – 48.01.** Actor is logged to Course-Scheduling Website.  **PRE – 48.02.** Actor is set as the admin or staff of the Website.  **PRE – 48.03** Actor accessed to “*Classes Management*” page. | | |
| Post-conditions: | **POST – 48.01.** The classes have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 48. Search** **Classes**   1. Actor enters information about the class that they want to find in the search box. 2. Course-Scheduling shows list of classes that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-49. View All Programs

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-49. View All Programs | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the classes. | | |
| Description: | Actor uses this function to see all the classes. | | |
| Pre-conditions: | **PRE – 44.01.** Actor is logged to Course-Scheduling Website.  **PRE – 44.02.** Actor is set as the staff of the Website.  **PRE – 44.03.** Actor accessed to “*Programs Management*” page. | | |
| Post-conditions: | **POST – 44.01.** Course-Scheduling shows list of all classes. | | |
| Normal Flow: | **UC – 44. View All Programs**   1. Actor clicks to menu “*Programs Management*”. 2. Actor choose Semester in left panel. 3. Course-Scheduling shows list of all Programs in the semester selected. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-50. Add List of Programs

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-50. Add List of Classes | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add List of new programs. | | |
| Description: | Actor uses this function to add List of new programs. | | |
| Pre-conditions: | **PRE – 50.01.** Actor is logged to Course-Scheduling Website.  **PRE – 50.02.** Actor is set as the staff of the Website.  **PRE – 50.03.** Actor accessed to “*Programs Management*” page. | | |
| Post-conditions: | **POST – 50.01.** The students were added will be shown in list of all programs of Program page. | | |
| Normal Flow: | **UC – 50. Add List of Programs**   1. In “*Programs Management*” page, Actor clicks to button "Add List of Programs”. 2. Course-Scheduling shows a dialog with an input to enter the link of excel file contains all information about Programs. 3. Actor enters the link of file. 4. Actor choose “Add”. 5. Course-Scheduling will validate all information, if it’s correct, Students will be added to database. 6. Course-Scheduling redirects to “*Programs Management*” page with a message “*Import List of Programs successful!*“. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-51. Search Programs

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-51. Search Programs | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific program. | | |
| Description: | Actor uses this function to search for a specific program. | | |
| Pre-conditions: | **PRE – 51.01.** Actor is logged to Course-Scheduling Website.  **PRE – 51.02.** Actor is set as the admin or staff of the Website.  **PRE – 51.03** Actor accessed to “*Programs Management*” page. | | |
| Post-conditions: | **POST – 51.01.** The programs have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 51. Search** **Programs**   1. Actor enters information about the class that they want to find in the search box. 2. Course-Scheduling shows list of programs that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-52. View All Merge-Classes

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-52. View All Merge-Classes | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to see all the merge-classes. | | |
| Description: | Actor uses this function to see all the merge-classes. | | |
| Pre-conditions: | **PRE – 52.01.** Actor is logged to Course-Scheduling Website.  **PRE – 52.02.** Actor is set as the staff of the Website.  **PRE – 52.03.** Actor accessed to “*Merge-Classes Management*” page. | | |
| Post-conditions: | **POST – 52.01.** Course-Scheduling shows list of all classes. | | |
| Normal Flow: | **UC – 52. View All Merge-Classes**   1. Actor clicks to menu “*Merge-Classes Management*”. 2. Actor choose Semester in left panel. 3. Course-Scheduling shows list of all Merge-Classes in the semester selected. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-53. Add Merge-Class

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-53. Add Merge-Class | | |
| Created By: | **DucHM** | Date Created: | **2015-06-18** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to add new Merge-Class. | | |
| Description: | Actor uses this function to add new Merge-Class. | | |
| Pre-conditions: | **PRE – 53.01.** Actor is logged to Course-Scheduling Website.  **PRE – 53.02.** Actor is set as the staff of the Website.  **PRE – 53.03.** Actor accessed to “*Merge-Classes Management*” page. | | |
| Post-conditions: | **POST – 53.01.** The Merge-Class was added will be shown. | | |
| Normal Flow: | **UC – 53. Add Merge-Class**   1. In “*Merge-Classes Management*” page, Actor clicks to button "Add Class”. 2. The actor enters information about new Merge-Class: “Course”, “Class 1”, “Class 2”. 3. The actor clicks to button “Save”. 4. Course-Scheduling will validate all information if it’s correct, the Merge-Class will be added to database. 5. Course-Scheduling redirect to “*Merge-Classes Management*” page with message: “*Add Merge-Classes Successful!*” | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **53.01. Unexpected exception.**  **1.** Course-Scheduling shows error message.  **2.** Rollback transaction. | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-54. Remove Merge-Class

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-54. Remove Merge-Class | | |
| Created By: | **DucHM** | Date Created: | **2015-06-18** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a Merge-Class. | | |
| Description: | Actor uses this function to remove a Merge-Class. | | |
| Pre-conditions: | **PRE – 54.01.** Actor is logged to Course-Scheduling Website.  **PRE – 54.02.** Actor is set as the staff of the Website.  **PRE – 54.03.** Actor accessed to Manage Merge-Classes page. | | |
| Post-conditions: | **POST – 54.01.** The merge-class was removed will not be shown in list. | | |
| Normal Flow: | **UC – 54. Remove Merge-Class**   1. The actor clicks to button “Delete” corresponding to a Merge-Class 2. The actor clicks to button “Yes” in confirm popup. 3. Course-Scheduling will validate all information, if it’s correct, the Merge-Class will be remove from database. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **54.01. Unexpected exception.**  **1.** Course-Scheduling shows error message.  **2.** Rollback transaction. | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Low** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-55. Search Merge-Classes

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-55. Search Merge-Classes | | |
| Created By: | **HaNS** | Date Created: | **2015-06-01** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to search for a specific class. | | |
| Description: | Actor uses this function to search for a specific class. | | |
| Pre-conditions: | **PRE – 55.01.** Actor is logged to Course-Scheduling Website.  **PRE – 55.02.** Actor is set as the admin or staff of the Website.  **PRE – 55.03** Actor accessed to “*Merge-Classes Management*” page. | | |
| Post-conditions: | **POST – 55.01.** The Merge-Classes have information similar with the information that actor entered will be shown. | | |
| Normal Flow: | **UC – 55. Search** **Merge-Classes**   1. Actor enters information about the merge-class that they want to find in the search box. 2. Course-Scheduling shows list of merge-classes that have this information. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **N/A** | | |
| Priority: | **Medium** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-56. Auto put students into class

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-56. Auto put students into class | | |
| Created By: | **DucHM** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to automatically put students into classes. | | |
| Description: | Actor uses this function to automatically put students into classes. | | |
| Pre-conditions: | **PRE – 56.01.** Actor is logged to Course-Scheduling Website.  **PRE – 56.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 56.01.** Students were put into classes. | | |
| Normal Flow: | **UC – 56. Auto put students into class**   1. Actor clicks to “Add Students to Class” link. 2. Actor select semester 3. Actor select class 4. Actor clicks to “Set Students For Class” 5. Course-Scheduling auto puts all suitable students into class. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **56.01. Unexpected exception.**  **1.** Course-Scheduling shows error message.  **2.** Rollback transaction. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-57. Clear students from class

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-57. Clear students from class | | |
| Created By: | **DucHM** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to automatically put students into classes. | | |
| Description: | Actor uses this function to automatically put students into classes. | | |
| Pre-conditions: | **PRE – 57.01.** Actor is logged to Course-Scheduling Website.  **PRE – 57.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 57.01.** Students were cleared from class. | | |
| Normal Flow: | **UC – 57. Auto put students into class**   1. Actor clicks to “Add Students to Class” link. 2. Actor select semester 3. Actor select class 4. Actor clicks to “Clear Students Of Class” 5. Course-Scheduling auto clear all students from class. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **57.01. Unexpected exception.**  **1.** Course-Scheduling shows error message.  **2.** Rollback transaction. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-58. Auto put students into classes

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-58. Auto put students into classes | | |
| Created By: | **DucHM** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to automatically put students into classes. | | |
| Description: | Actor uses this function to automatically put students into classes. | | |
| Pre-conditions: | **PRE – 58.01.** Actor is logged to Course-Scheduling Website.  **PRE – 58.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 58.01.** Students were put into classes. | | |
| Normal Flow: | **UC – 58. Auto put students into class(es)**   1. Actor clicks to “Add Students to Class” link. 2. Actor select semester 3. Actor clicks to “Set Students For Semester” 4. Course-Scheduling auto puts all suitable students into classes. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **58.01. Unexpected exception.**  **1.** Course-Scheduling shows error message.  **2.** Rollback transaction. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-59. Clear students from classes

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-59. Auto put students into classes | | |
| Created By: | **DucHM** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to automatically put students into classes. | | |
| Description: | Actor uses this function to automatically put students into classes. | | |
| Pre-conditions: | **PRE – 59.01.** Actor is logged to Course-Scheduling Website.  **PRE – 59.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 59.01.** Students were cleared from classes. | | |
| Normal Flow: | **UC – 59. Auto put students into class(es)**   1. Actor clicks to “Add Students to Class” link. 2. Actor select semester 3. Actor clicks to “Clear Students Of Semester” 4. Course-Scheduling auto puts all suitable students into classes. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **59.01. Unexpected exception.**  **1.** Course-Scheduling shows error message.  **2.** Rollback transaction. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-60. Manual put student into class

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-60. Manual put student into class | | |
| Created By: | **DucHM** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to put a student into a class. | | |
| Description: | Actor uses this function to put a student into a class. | | |
| Pre-conditions: | **PRE – 60.01.** Actor is logged to Course-Scheduling Website.  **PRE – 60.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 60.01.** Student was put into class. | | |
| Normal Flow: | **UC – 60. Manual put student into class**   1. Actor clicks to “Add Students To Class” link. 2. Actor select semester 3. Actor select class 4. Actor select one course or all courses 5. Actor select a student from suitable students list to put into class and clicks “Add”. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **60.01. Max Student Number Exception.**  **1.** Course-Scheduling shows error message: “Student number of this class exceed 30”.  **2.** Rollback transaction. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **B11** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-61. Manual remove student from class

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-61. Manual remove student from class | | |
| Created By: | **DucHM** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor wants to remove a student from a class. | | |
| Description: | Actor uses this function to remove a student from a class. | | |
| Pre-conditions: | **PRE – 61.01.** Actor is logged to Course-Scheduling Website.  **PRE – 61.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 61.01.** Student was removed from class. | | |
| Normal Flow: | **UC – 61. Manual put student into class**   1. Actor clicks to “Add Students To Class” link. 2. Actor select semester 3. Actor select class 4. Actor select one course or all courses 5. Actor select a student from students list to remove from class and clicks “Remove”. | | |
| Alternative Flows: | N/A | | |
| Exceptions: | **N/A** | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **N/A** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-62. Manual Create Timetable

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-62. Manual Create Timetable | | |
| Created By: | **HaNS** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor will create timetable for classes. | | |
| Description: | Actor uses this function to create timetable for classes. | | |
| Pre-conditions: | **PRE – 62.01.** Actor is logged to Course-Scheduling Website.  **PRE – 62.02.** Actor is set as the staff of the Website.  **PRE – 62.03.** All classes was set courses successful. | | |
| Post-conditions: | **POST – 62.01.** Class was created timetable successful | | |
| Normal Flow: | **UC – 62. Manual Create Timetable**   1. Actor clicks to activity “Schedule”. 2. Course-Schedule redirects to “Schedule Info” page. 3. Actor choose semester to create timetable. 4. Actor clicks to button “Manual”. 5. Course-Schedule redirects to “Schedule” page. 6. Actor chooses a class of this semester. 7. Course-Schedule shows timetable of the first week in semester. 8. Actor chooses a week to create or update timetable. 9. Actor clicks to a cell (represent for a day and the slot of this day). Then choose Course will be learnt. 10. Actor clicks button Save, Course-Scheduling will validate information and if it’s acceptable then save to database. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **62.01. Number of classes learn the course in a day-slot is greater than number of teachers teach the course.**  . Course-Scheduling shows error message “Don’t have enough teacher to set”.  **62.02. Number of class learn in a day-slot is greater than number of rooms.**  **.** Course-Scheduling shows error message “Don’t have enough room to set”.  **62.03. Conflict timetable between merge-class.**  **.** Course-Scheduling shows error message “Conflict with timetable of merge-class”. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **B11, B12, B13** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-63. Manual Generate Timetable From Previous Week

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-63. Manual Generate Timetable From Previous Week | | |
| Created By: | **HaNS** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor want to generate timetable from previous week. | | |
| Description: | Actor uses this function to generate timetable from previous week. | | |
| Pre-conditions: | **PRE – 63.01.** Actor is logged to Course-Scheduling Website.  **PRE – 63.02.** Actor is set as the staff of the Website.  **PRE – 63.03.** All classes was set courses successful. | | |
| Post-conditions: | **POST – 63.01.** Class was created timetable successful | | |
| Normal Flow: | **UC – 63. Manual Create Timetable**   1. Actor clicks to activity “Schedule”. 2. Course-Schedule redirects to “Schedule Info” page. 3. Actor choose semester to create timetable. 4. Actor clicks to button “Manual”. 5. Course-Schedule redirects to “Schedule” page. 6. Actor chooses a class of this semester. 7. Course-Schedule shows timetable of the first week in semester. 8. Actor chooses a week to and then click “Generate from previouse week”. 9. Course-Scheduling will remove all data of current week and change them to data of previous week. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **63.01 Week 1.**  **.** Course-Scheduling redirect to “Schedule” page with message “Can not generate for week 1.” | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **B11, B12, B13** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-64. Manual Room Arrangement.

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-64. Manual Room Arrangement | | |
| Created By: | **HaNS** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor will create timetable for classes. | | |
| Description: | Actor uses this function to create timetable for classes. | | |
| Pre-conditions: | **PRE – 64.01.** Actor is logged to Course-Scheduling Website.  **PRE – 64.02.** Actor is set as the staff of the Website.  **PRE – 64.03.** All classes was set timetable successful. | | |
| Post-conditions: | **POST – 64.01.** Class was set room successful | | |
| Normal Flow: | **UC – 64. Manual Room Arrangement**   1. Actor clicks to activity “Schedule”. 2. Course-Schedule redirects to “Schedule Info” page. 3. Actor choose semester to arrange room. 4. Actor clicks to choose action 2. 5. Actor clicks to button “Manual”. 6. Course-Schedule redirects to “Room Arrangement” page. 7. Actor chooses a class of this semester. 8. Course-Schedule shows table with name of courses and room selection of each course. 9. Actor chooses a room for all course of this class or choose room for each course. 10. Actor clicks button Save, Course-Scheduling will validate information and if it’s acceptable then save to database. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **64.01. Room was set for the course is not a correct room.**  . Course-Scheduling shows error message: “ ’Course’ need a specific room! Refer to Special rooms for more information”.  **64.02. Room** **was used for another class in this day-slot.**  . Course-Scheduling shows error message: “Timetable of Room … has … slots conflict with timetable of course”. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **B14, B16, B17** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-65. Manual Teacher Arrangement

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-65. Manual Teacher Arrangement | | |
| Created By: | **HaNS** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor will create timetable for classes. | | |
| Description: | Actor uses this function to create timetable for classes. | | |
| Pre-conditions: | **PRE – 65.01.** Actor is logged to Course-Scheduling Website.  **PRE – 65.02.** Actor is set as the staff of the Website.  **PRE – 65.03.** All classes was set timetable successful. | | |
| Post-conditions: | **POST – 65.01.** Class was set teacher successful | | |
| Normal Flow: | **UC – 65. Manual Room Arrangement**   1. Actor clicks to activity “Schedule”. 2. Course-Schedule redirects to “Schedule Info” page. 3. Actor choose semester to arrange teacher. 4. Actor clicks to choose action 3. 5. Actor clicks to button “Manual”. 6. Course-Schedule redirects to “Teacher Arrangement” page. 7. Actor chooses semester, department and course of the department. 8. Course-Schedule shows table with name of classes and teacher selection of each classes. 9. Actor chooses a room for a class. 10. Actor clicks button Save, Course-Scheduling will validate information and if it’s acceptable then save to database. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **65.01. Teacher** **was set to teach another class in ththe same day-slot.**  . Course-Scheduling shows error message: “Teacher was arranged to class…! Can’t be arranged to class…”. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **B15, B18** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-66. View All Timetable of Class by Staff

|  |  |  |  |
| --- | --- | --- | --- |
| UC ID and Name: | UC-66. View All Timetable of Class by Staff | | |
| Created By: | **HaNS** | Date Created: | **2015-07-16** |
| Primary Actor: | **Staff** | Secondary Actors: | N/A |
| Trigger: | The actor want to view all timetable of a class | | |
| Description: | Actor uses this function to create timetable for classes. | | |
| Pre-conditions: | **PRE – 66.01.** Actor is logged to Course-Scheduling Website.  **PRE – 66.02.** Actor is set as the staff of the Website. | | |
| Post-conditions: | **POST – 66.01.** Timetable of the class will be shown. | | |
| Normal Flow: | **UC – 66. View All Timetable of Class by Staff**   1. Actor clicks to activity “View Timetable” in main page. 2. Course-Schedule redirects to “Schedule Info” page. 3. Actor choose semester to arrange teacher. 4. Actor clicks to choose action 3. 5. Actor clicks to button “Manual”. 6. Course-Schedule redirects to “Teacher Arrangement” page. 7. Actor chooses semester, department and course of the department. 8. Course-Schedule shows table with name of classes and teacher selection of each classes. 9. Actor chooses a room for a class. 10. Actor clicks button Save, Course-Scheduling will validate information and if it’s acceptable then save to database. | | |
| Alternative Flows: | **N/A** | | |
| Exceptions: | **66.01. Teacher** **was set to teach another class in ththe same day-slot.**  . Course-Scheduling shows error message: “Teacher was arranged to class…! Can’t be arranged to class…”. | | |
| Priority: | **High** | | |
| Frequency of Use: | **Medium** | | |
| Business Rules: | **B15, B18** | | |
| Other Information: | **N/A** | | |
| Assumptions: | **N/A** | | |

##### UC-67. View All Timetable By Teacher

##### UC-68. View All Timetable By Student

## Non-functional Requirements

### Security

### Maintainability & Extensibility

### Performance

## Entity Relationship Diagram

# CHAPTER 4: SOFTWARE DESIGN

## Purpose

## Architecture Overview

### System Architecture

### System Architecture Explanation

## Design of CourseScheduling system

### Architecture Layers Design

### Database Design

#### Database Diagram

### Common Design

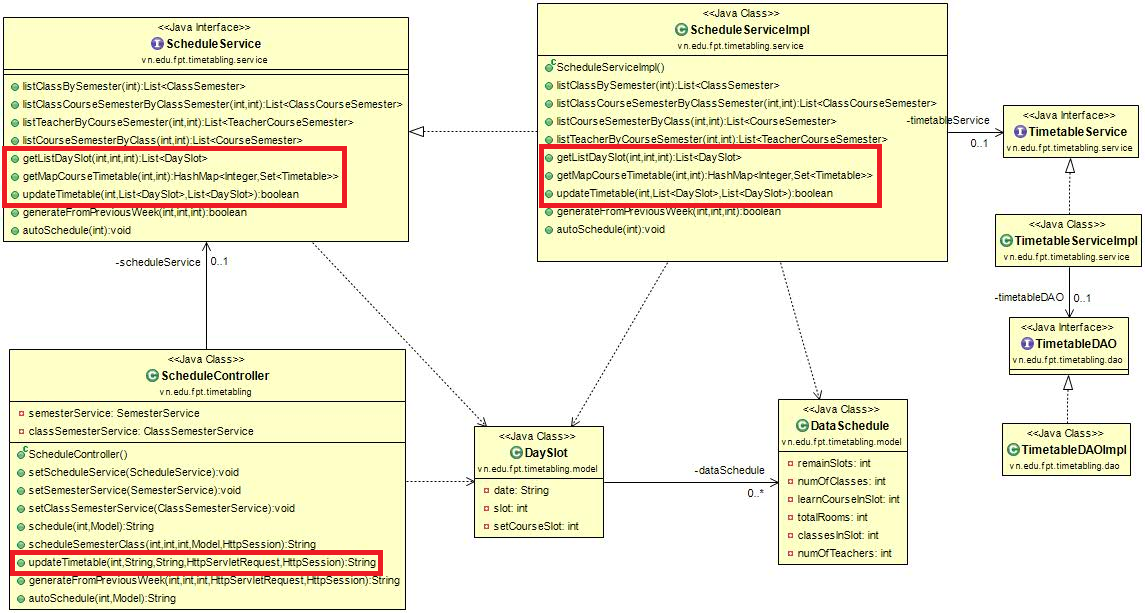
### Detail Design

#### Manual Create Timetable

##### Description

* Actor creates timetable for class.

##### Class Diagram



##### Class Specification

###### CLASS: DaySlot

**Physical Address:** ..\vn\edu\fpt\timetabling\model\DaySlot.java

**Attribute:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Description |
| 1 | date | String | String was converted from data Date with format:  yyyy-MM-dd. |
| 2 | slot | Int | Index of slot |
| 3 | setCourseSlot | Int | Id of classCourseSemester was set in this DaySlot.  (-1 represents for not set) |
| 4 | dataSchedule | HashMap<String, DataSchedule> | Map between course’s Codes of class and dataSchedule. |

###### CLASS: DataSchedule

**Physical Address:** ..\vn\edu\fpt\timetabling\model\DataSchedule.java

**Description:** This class is used to contain data of a course in a DaySlot. (A DaySlot can have one or more courses can be set and each course will be map with a data schedule object).

**Attribute:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Description |
| 1 | remainSlot | Int | Number of slots was not set of the course in semester. |
| 2 | numOfClasses | Int | Number of classes learn this course in semester. |
| 3 | learnCourseInSlot | Int | Number of classes learn this course in this slot. |
| 4 | totalRooms | Int | Total rooms, |
| 5 | classesInSlot | Int | Number of classes learn in this slot. |
| 6 | numOfTeachers | Int | Number of Teachers teach this course. |

###### CLASS: ScheduleController

**Physical Address:** ..\vn\edu\fpt\timetabling\ScheduleController.java

**Method: updateTimetable**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Receive data from client, convert to data and call updateTimetable in ScheduleService. | | | |
| Return | **Type** | | | **Description** |
| String | | | Redirect to previous page |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of Semester. |
| 2 | JSONToSubmit | String | String of JSON data after updated by actor. |
| 3 | JSONPrev | String | String of JSON data before updated by actor. |

###### CLASS: ScheduleService

**Physical Address:** ..\vn\edu\fpt\timetabling\service\ScheduleService.java

**Attribute:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Description |
|  |  |  |  |

**Method: getListDaySlot**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Get List of DaySlot in a week of a class in semester. | | | |
| Return | **Type** | | | **Description** |
| List<DaySlot> | | | List of DaySlot in a week of a class in semester. |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of Semester. |
| 2 | classId | Int | Id of Class. |
| 3 | week | Int | Index of week is counted from the beginning of semester. (week = 1 represent for the first week of semester) |

**Method: getMapCourseTimetable**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Get lists of all timetable of Merge-Class that learn together with main class. | | | |
| Return | **Type** | | | **Description** |
| HashMap<String, List<Timetable>> | | | String: key of group merge-classes.  List<Timetable>: list timetable of merge-classes. |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of Semester. |
| 2 | classId | List<DaySlot> | List of data was updated. |

**Method: updateTimetable**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Receive updated data and decide to add new, update or delete timetable of a class and Merge-Class also. | | | |
| Return | **Type** | | | **Description** |
| boolean | | | Action success or fail. |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of Semester. |
| 2 | daySlots | List<DaySlot> | List of data was updated. |
| 3 | prevDaySlots | List<DaySlot> | List of data before updated. |

##### Implementation

. When user accesses “Schedule” page all data include semester, class of the semester, courses of class, teachers for the courses, merge-classes and data about timetable for week will be load as a string of JSON data. All action add, delete will be validate in javascript.

. When user clicks save button, the JSON data contains information about timetable will be converted to string and push back to server.

. In Server, for each timetable has changed, based on the different of setCourseSlot value to decide create new, update existed or delete and the action for merge-class also.

**Pseudo code:**

**updateTimetable(int semesterId, List<DaySlot> daySlots, List<DaySlot> prevDaySlots)**

// Create Map Action to handle with merge class.

// Put to Map 2 key “add” and “delete” for 2 action add and delete timetable.

1. **Create** **HashMap<String, Set<Timetable>> mActionTimetable.**

// loop through List daySlots and prevDaySlots with index based on daySlots.

1. **For (int I = 0, I < daySlots.size(), i++)**

// 2 setCourseSlots are different mean the course in day slot was updated.

1. **If(daySlot[i].setCourseSlot != prevDaySlots[i].setCourseSlot)**

// If they are different, check value of them to decide action.

// check don’t set course before.

1. **If(prevDaySlot[i].setCourseSlot == -1)**

// If don’t set course before 🡪 add new timetable.

1. **addTimetable**

// If it’s merge-class 🡪 add to List add of mActionTimetable.

1. **add Timetable to mActionTimetable key “add”.**

// If set course before 🡪 Check value of prevDaySlot[i].setCourseSlot.

// If prevDaySlot[i].setCourseSlot is merge-class.

1. **add Timetable to mActionTimetable key “delete”.**

// check if current setCourseSlot is = -1 🡪 delete course in this DaySlot.

// if not 🡪 update

1. **If(daySlot[i].setCourseSlot == -1) 🡪 delete Timetable**
2. **If(daySlot[i].setCourseSlot != -1) 🡪 update Timetable**

// if current setCourseSlot != -1 and is mergeClass

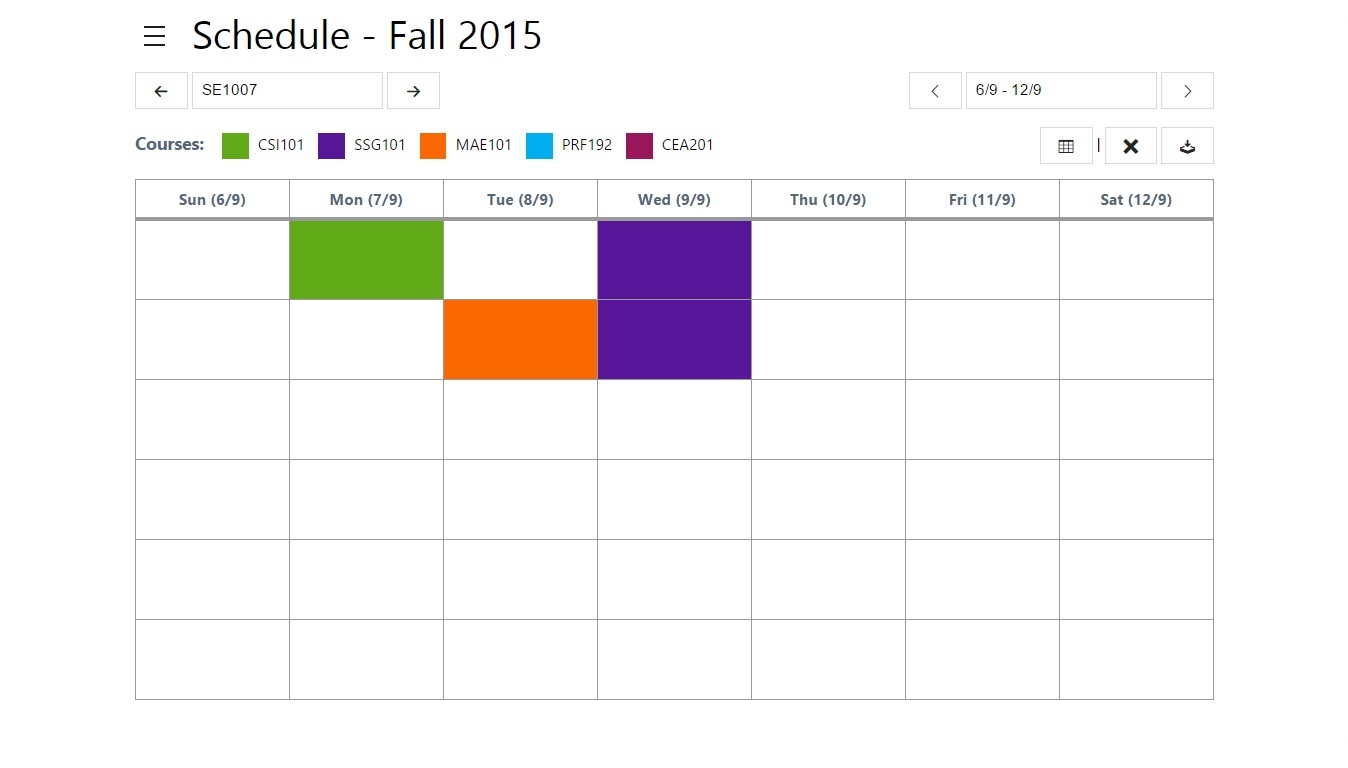
1. **add Timetable to mActionTimetable key “add”.**

// end loop of daySlots then Loop through mActionTimetable and do the action as key of map.

1. **For(String key : mActionTimetable.keySet())**
2. **Do action as key of map for mergeClass with the current course in each timetable object.**

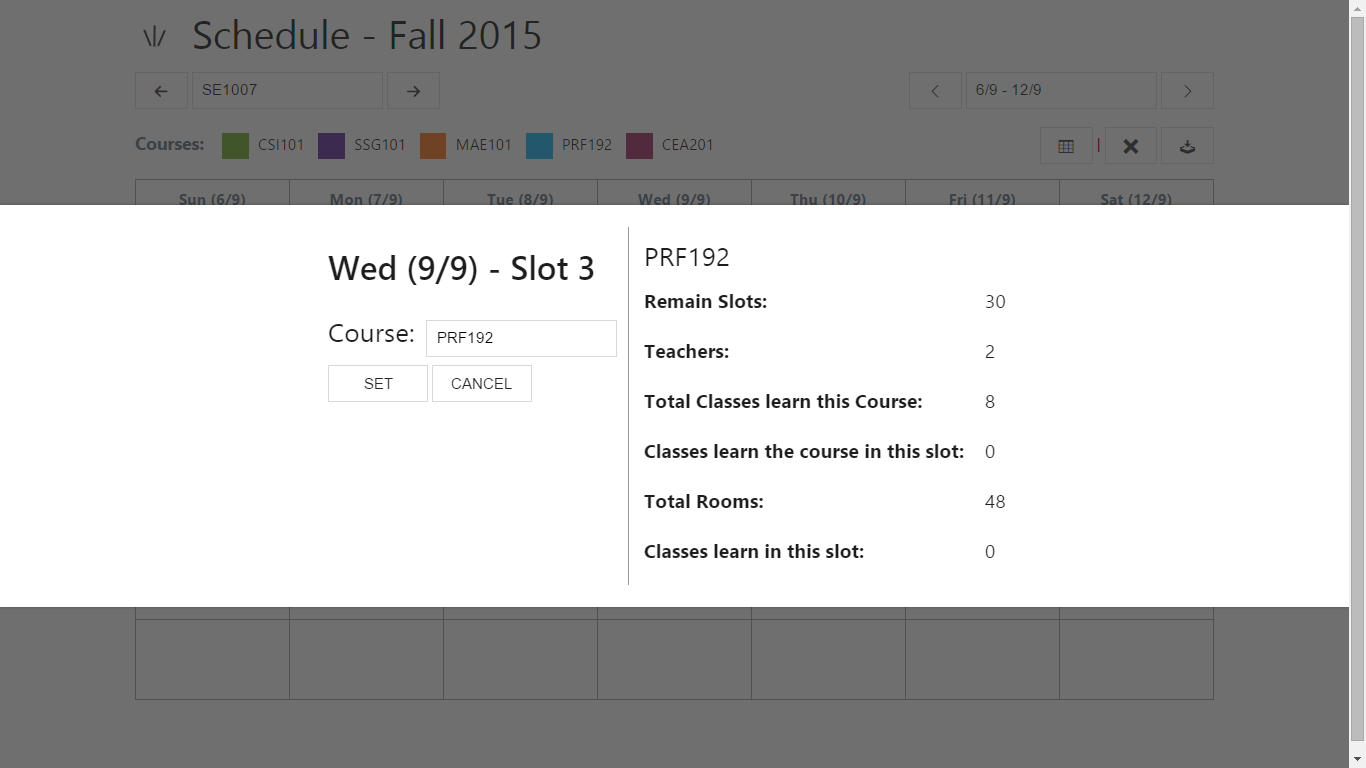
##### Sequence Diagram

##### Screen Design



**Figure 4.1 Schedule Page.**

|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| **Class Selection** | Select | List of class in semester and used to choose class will be scheduled. |
| **Next Class** | Button | Next Class In List. |
| **Previous Class** | Button | Previous Class In List. |
| **Week Selection** | Select | List of week and display by date of Sunday – date of Saturday. |
| **Next Week** | Button | Next week In List. |
| **Previous** | Button | Previous week In List. |
| **Save** | Button | Save to database. |
| **Clear** | Button | Clear all data in current week. |
| **Timetable** | Table | Used to set, update, remove timetable of a class in a week. |



**Figure 4.2 Add a course for a DaySlot**

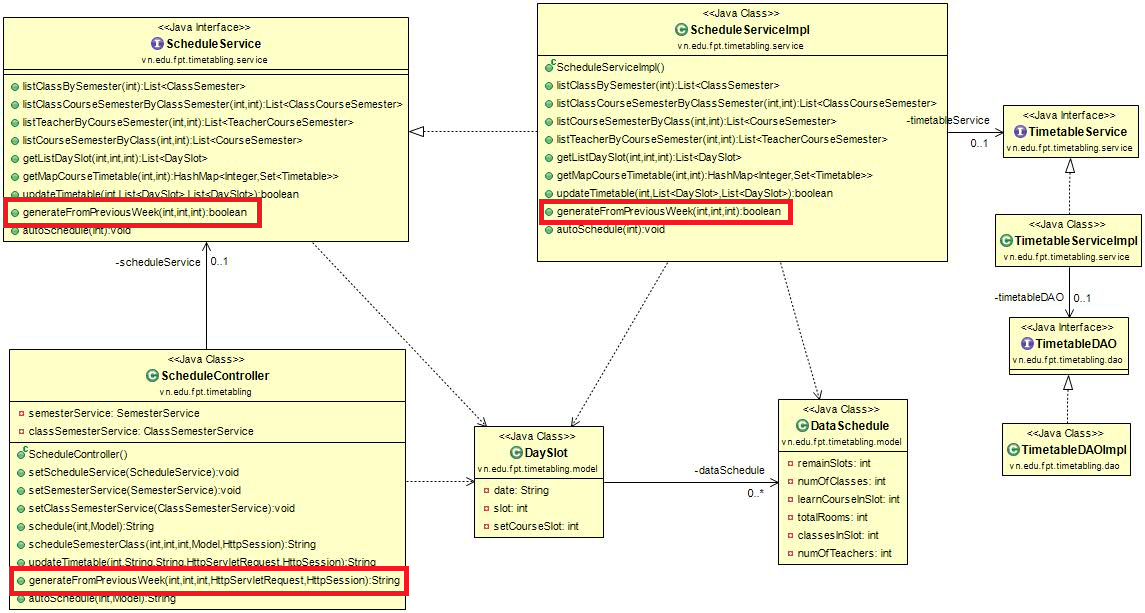
|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| Course Selection | Select | Select a course for DaySlot |
| Set | Button | Set course for a DaySlot |
| Cancel | Button | Close Dialog |

#### Generate From Previous Week.

##### Description

* Actor generates timetable from previous week.

##### Class Diagram



##### Class Specification

###### CLASS: ScheduleController

**Physical Address:** ..\vn\edu\fpt\timetabling\ScheduleController.java

**Method: generateFromPreviouseWeek**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Receive data from client, convert to data and call generateFromPreviousWeek in ScheduleService. | | | |
| Return | **Type** | | | **Description** |
| String | | | Redirect to previous page |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of Semester. |
| 2 | classId | classId | Id of Class |
| 3 | week | int | Index of week is counted from the beginning of semester. (week = 1 represent for the first week of semester) |

###### CLASS: ScheduleService

**Method: generateFromPreviouseWeek**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Generate timetable from previous week. | | | |
| Return | **Type** | | | **Description** |
| boolean | | | Action success or false. |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of Semester. |
| 2 | classId | classId | Id of Class |
| 3 | week | int | Index of week is counted from the beginning of semester. (week = 1 represent for the first week of semester) |

##### Implementation

. When user click to “Generate From Previous Week”, check current of week is 1 or not. If week is 1 then call redirect to previous page with message error.

. If week is not 1, then clear all timetable of current week and copy all timetable from previous week to current week.

**Pseudo code:**

**generateFromPreviousWeek(int semesterId, int classId, int week)**

// Delete all timetable of class in current week.

1. **Delete:** deleteTimetableByCCSInWeek(int classSemesterId, Date startWeek, Date endWeek)

// Get all timetable of previous week.

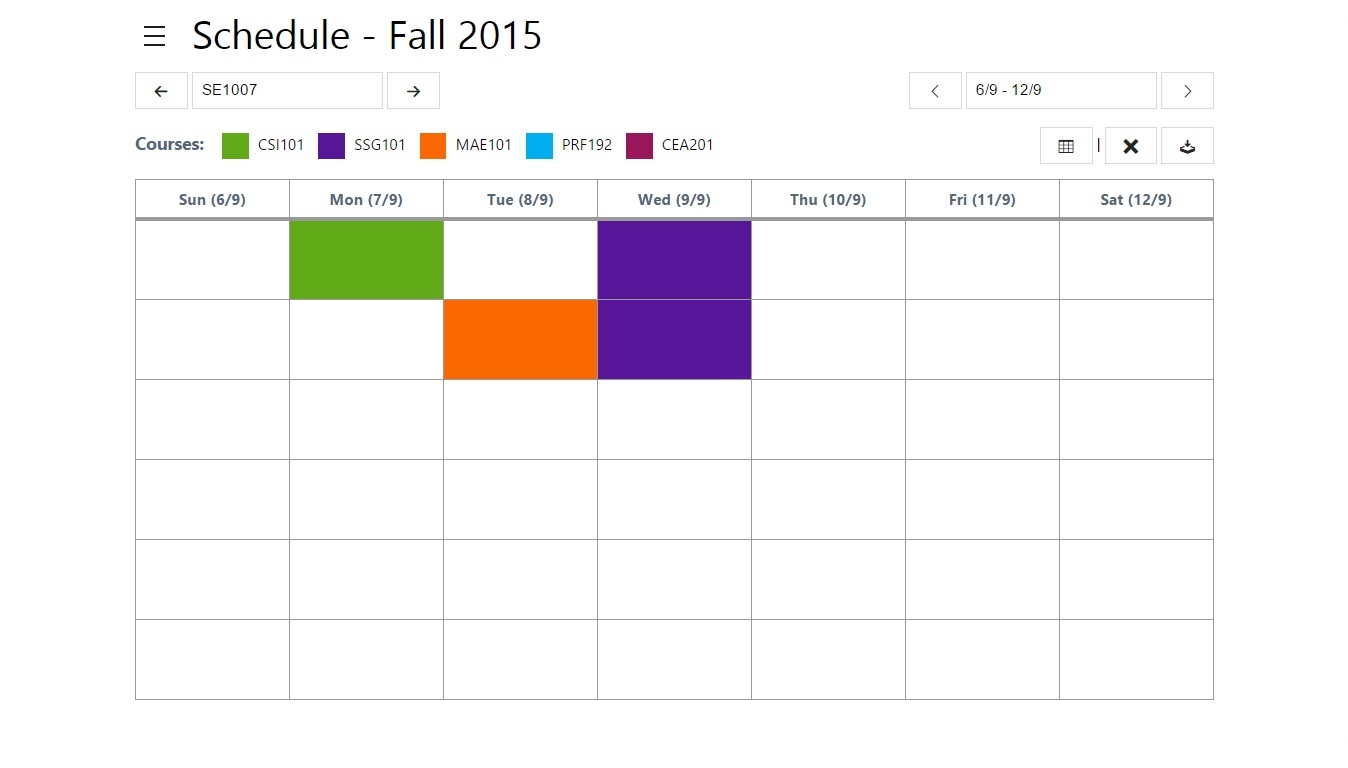
1. **Get:** listTimetableByCCSInWeek(int classSemesterId, Date startWeek, Date endWeek)

// Add Timetable for current week of class and Merge-Class.

1. **Add Timetable.**

##### Sequence Diagram

##### Screen Design



**Figure 4.3 Schedule Page.**

|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| **Generate From Previous Week** | Button | Generate from previous week. |

#### Manual Room Arrangement

##### Description

. After creating timetable for all class, actor arranges room for classes.

##### Class DiagramC:\Users\melody\AppData\Local\Microsoft\Windows\INetCache\Content.Word\roomArrangement.jpg

##### Class Specification

###### CLASS: Room

**Physical Address:** ..\vn\edu\fpt\timetabling\model\Room.java

**Attribute:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Description |
| 1 | roomId | Int | Id of Room |
| 2 | code | String | Code of Room |
| 3 | courses | String | A string contains all code of courses will be taught in this room. |
| 4 | capacity | Int | Capacity of room |
| 5 | building | Building | Building that room belong to. |
| 6 | timetables | Set<Timetable> | Timetable of this room (or DaySlot this room is used). |

###### CLASS: DataRoomArrangement

**Physical Address:** ..\vn\edu\fpt\timetabling\model\DataRoomArrangement.java

**Description:** Was used to check classSemester was set room successful or not.

**Attribute:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Description |
| 1 | classSemester | ClassSemester | Class in semester |
| 2 | setRoomSuccessful | Boolean | True or false. |
| 3 | numberOfSlots | Int | Number of Slots of this class. |
| 4 | numberOfSlots\_WereSetSuccessful | Int | Number of Slots of this class were set room successful. |
| 5 | note | String | Note about status of this class. |

###### CLASS: RoomArrangementController

**Physical Address:** ..\vn\edu\fpt\timetabling\TeacherArrangementController.java

**Method: updateTimetableRoom**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Receive data was updated from client side and convert to data then call updateTimetable() from RoomArrangementService. | | | |
| Return | **Type** | | | **Description** |
| String | | | Redirect to previous page. |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | dataToSet | String | String from JSON was sent from client side. |

###### CLASS: RoomArrangementService

**Physical Address:** ..\vn\edu\fpt\timetabling\service\RoomArrangementService.java

**Method: updateTimetable**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Update Timetable by data received. | | | |
| Return | **Type** | | | **Description** |
| Boolean | | | Success or not |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | data | List<ClassSemester> | Data will be used to update timetable. |

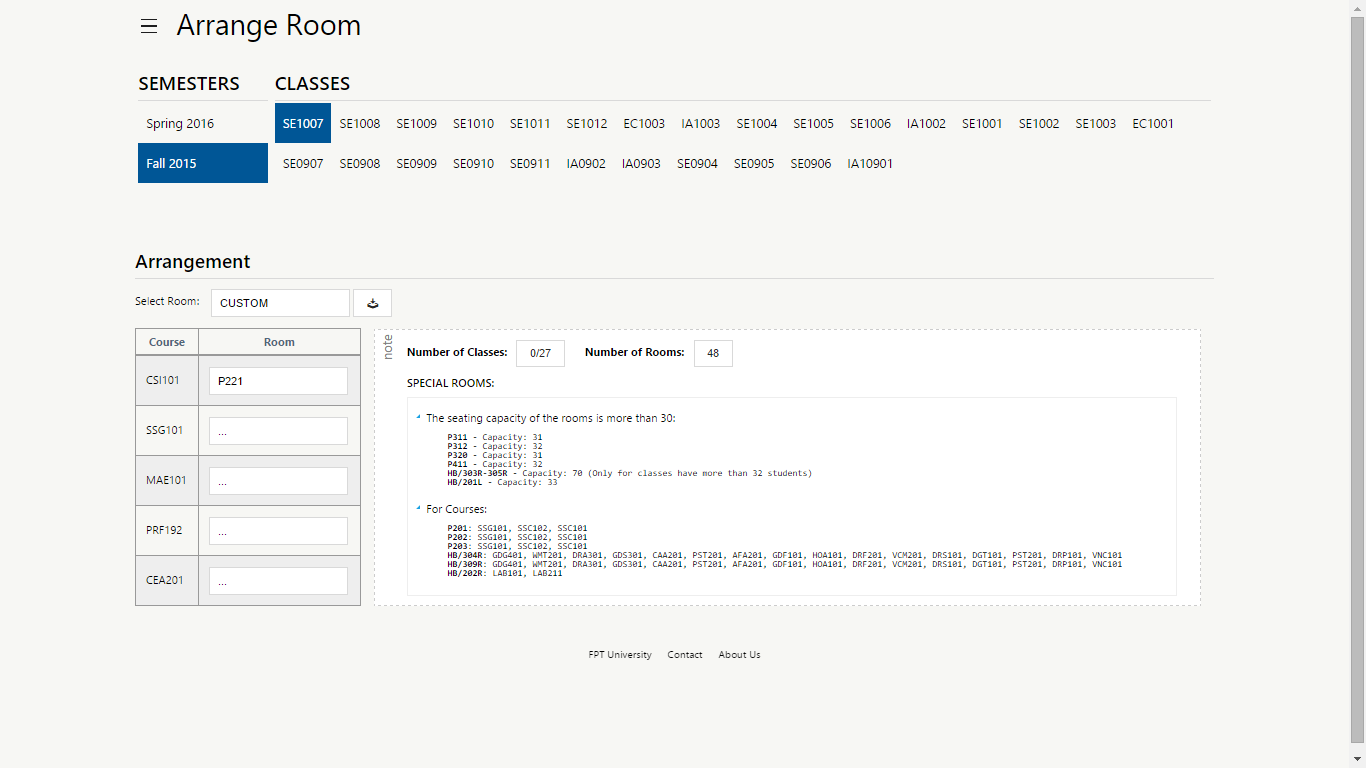
##### Implementation

. When actor clicks save, data json will be converted to text and send to server.

. Server get all timetable of the class and update room of them with new data.

##### Sequence Diagram

##### Screen Design

**Figure 4.4 Room Arrangement Page**

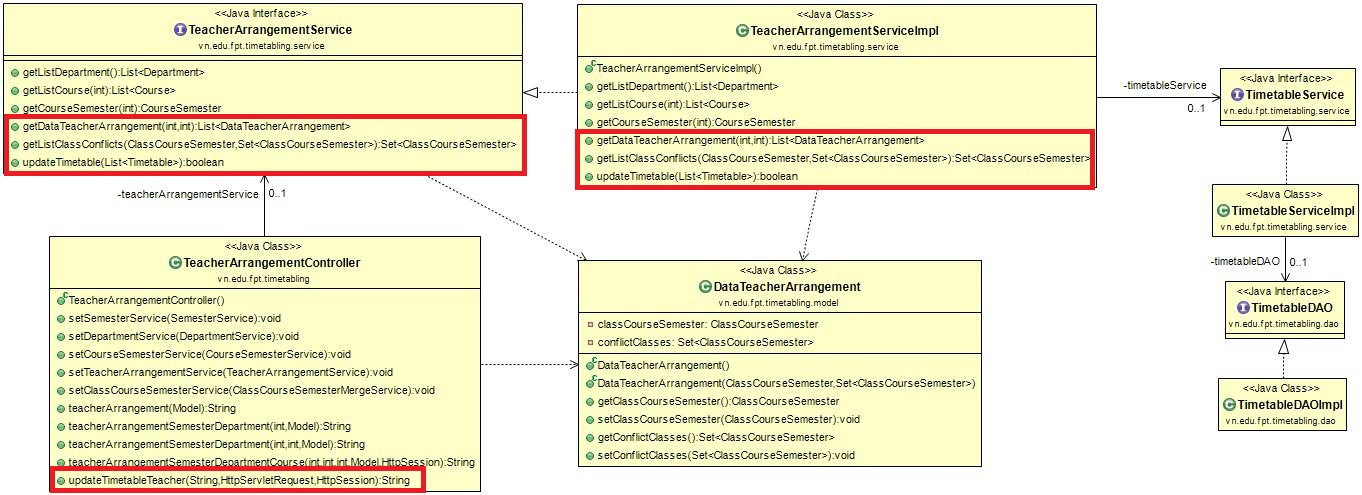
|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| SEMESTERS | a | Select semester and redirect with semesterId params. |
| CLASSES | a | Select class and redirect with semesterId and classId params. |
| Room Selections | Select | Select room for class and select room for courses. |
| Save | Button | Button save |

#### Manual Teacher Arrangement

##### Description

. After creating timetable for all class, actor arranges teacher for classes.

##### Class Diagram



##### Class Specification

###### CLASS: DataTeacherArrangement

**Physical Address:** ..\vn\edu\fpt\timetabling\model\DataTeacherArrangement.java

**Description:** Was used to get classCourseSemster and List of conflict classCourseSemesters.

**Attribute:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Type | Description |
| 1 | classCourseSemester | ClassCoursSemester | classCourseSemester represents for course of a class in a semester. |
| 2 | conflictClasses | List<ClassCourseSemester> | List of classCourseSemesters that have timetable conflicts with main classCourseSemester. |

###### CLASS: TeacherArrangementController

**Physical Address:** ..\vn\edu\fpt\timetabling\model\TeacherArrangementController.java

**Method: updateTimetableTeacher**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Receive data was updated from client side and convert to data then call updateTimetable() from TeacherArrangementService. | | | |
| Return | **Type** | | | **Description** |
| String | | | Redirect to previous page. |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | dataToSet | String | String from JSON was sent from client side. |

###### CLASS: TeacherArrangementService

**Physical Address:** ..\vn\edu\fpt\timetabling\model\service\TeacherArrangementService.java

**Method: getDataTeacherArrangement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Get list of DataTeacherArrangement for a course | | | |
| Return | **Type** | | | **Description** |
| List<DataTeacherArrangement | | | List of DataTeacherArrangement |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of semester. |
| 2 | courseId | Int | Id of course. |

**Method: getListClassConflits**

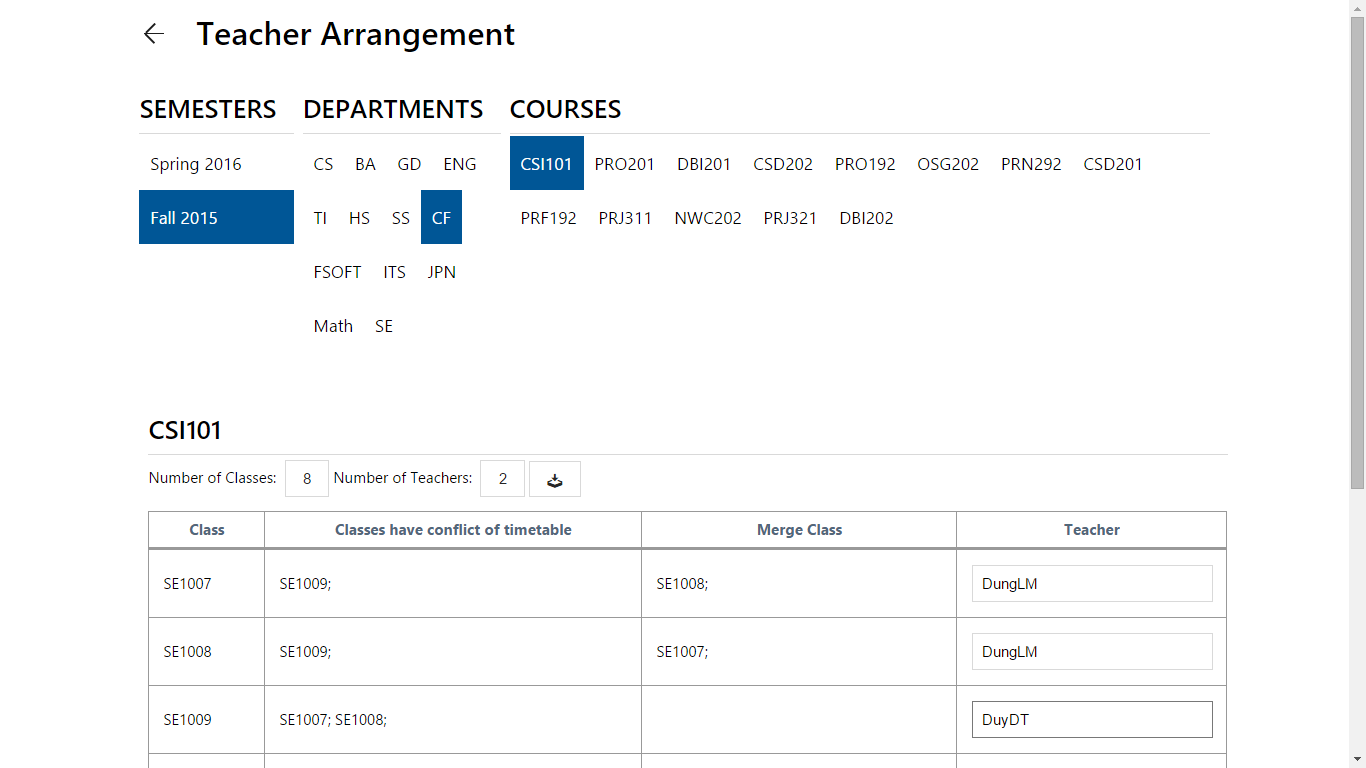
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Get list of conflict class for DataTeacherArrangement | | | |
| Return | **Type** | | | **Description** |
| Set<ClassCourseSemester> | | | List of conflict classCourseSemesters |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classCourseSemester | ClassCourseSemester | Main classCourseSemester |
| 2 | classCourseSemesters | Set<ClassCourseSemester> | Set of class learn the same course. |

**Method: updateTimetable**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Read list of timetable were updated in client side then update Timetable in database. | | | |
| Return | **Type** | | | **Description** |
| Boolean | | | Success or not |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | Timetables | List<Timetable> | List of Timetable were updated |

##### Sequence Diagram

##### Screen Design



**Figure 4.5 Teacher Arrangement Page**

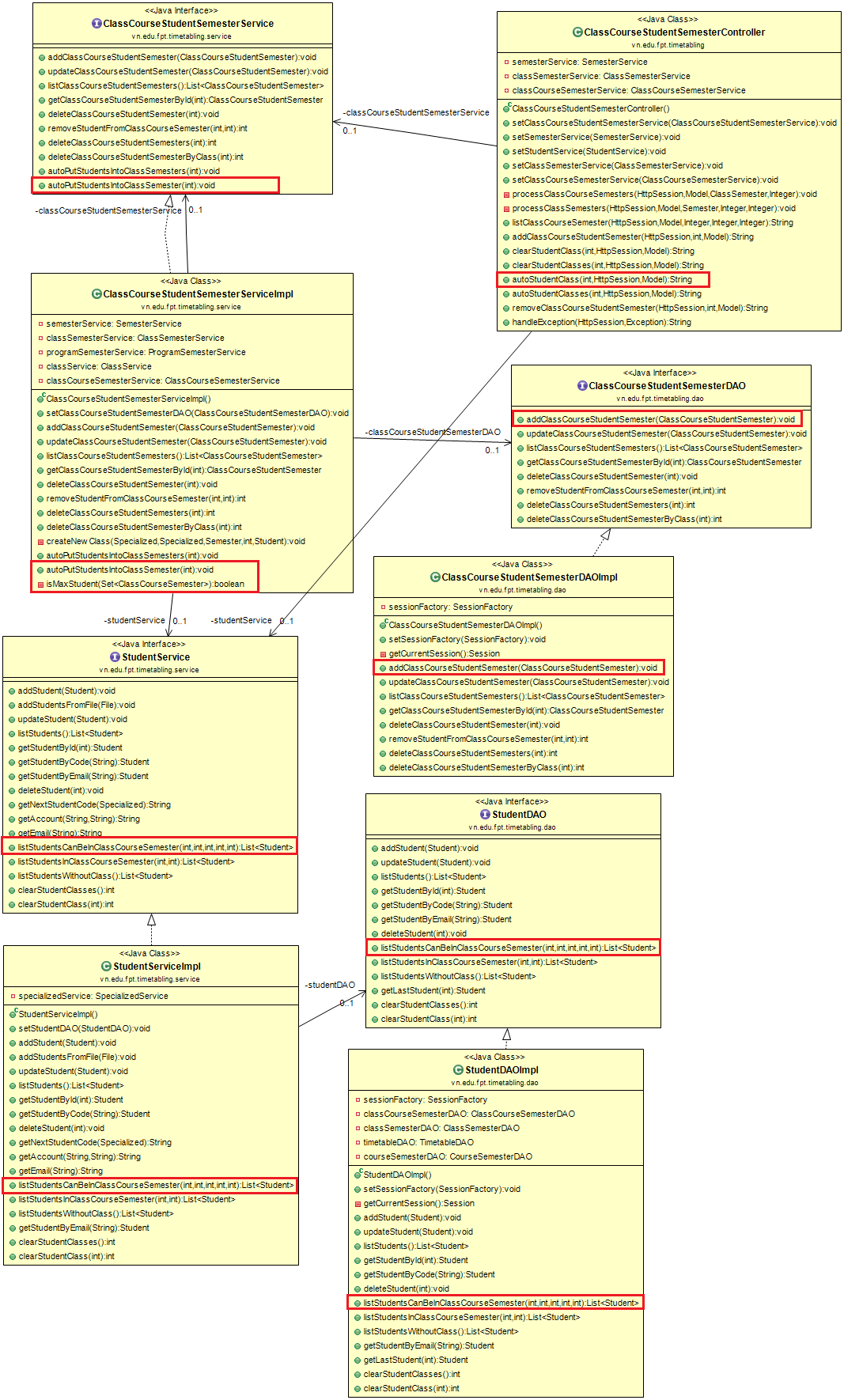
|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| SEMESTERS | a | Select semester and redirect with semesterId params. |
| DEPARTMENTS | a | Select department and redirect with semesterId and departmentId params. |
| COURSES | a | Select course and redirect with semesterId , departmentId and courseId params. |
| Teacher Selections | Select | Select teacher for class. |
| Save | Button | Button save. |

#### Auto put students into class

##### Description

* Auto put suitable students into a class

##### Class Diagram



##### Class Specification

###### CLASS: ClassCourseStudentSemesterController

**Physical Address:** ..\vn\edu\fpt\timetabling\ClassCourseStudentSemesterController.java

**Method: autoStudentClass**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Receive data from client, call autoPutStudentsIntoClassSemester in ClassCourseStudentSemesterService. | | | |
| Return | **Type** | | | **Description** |
| String | | | Redirect to previous page |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classSemesterId | Int | Id of ClassSemester. |

###### CLASS: ClassCourseStudentSemesterServiceImpl

**Physical Address:** ..\vn\edu\fpt\timetabling\service\ClassCourseStudentSemesterServiceImpl.java

**Method: autoPutStudentsIntoClassSemester**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Auto put students into a Class | | | |
| Return | **Type** | | | **Description** |
| void | | |  |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classSemesterId | Int | Id of ClassSemester. |

**Method: isMaxStudent**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Check if any ClassCourseSemester in list have number of student exceed 30 | | | |
| Return | **Type** | | | **Description** |
| boolean | | | Have at least a ClassCourseSemester in list have number of student exceed 30 |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classCourseSemesters | Set<ClassCourseSemester> | List of ClassCourseSemester |

###### CLASS: ClassCourseStudentSemesterDAOImpl

**Physical Address:** ..\vn\edu\fpt\timetabling\dao\ClassCourseStudentSemesterDAOImpl.java

**Method: addClassCourseStudentSemester**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Add a ClassCourseStudentSemester into Database.. | | | |
| Return | **Type** | | | **Description** |
| void | | |  |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classCourseStudentSemester | ClassCourseStudentSemester | ClassCourseStudentSemester to be added. |

###### CLASS: StudentServiceImpl

**Physical Address:** ..\vn\edu\fpt\timetabling\service\StudentServiceImpl.java

**Method: listStudentsCanBeInClassCourseSemester**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Call listStudentsCanBeInClassCourseSemester of StudentDAO | | | |
| Return | **Type** | | | **Description** |
| List<Student> | | | All students can be in a class course semester(s) |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classSemesterId | Int | Id of ClassSemester. |
| 2 | specializedId | Int | Id of Specialized |
| 3 | detailSpecializedId | Int | Id of detail Specialized |
| 4 | semesterNumber | Int | Semester number |
| 5 | classCourseSemesterId | Int | Id of ClassCourseSemester |

###### CLASS: StudentDAOImpl

**Physical Address:** ..\vn\edu\fpt\timetabling\dao\StudentDAOImpl.java

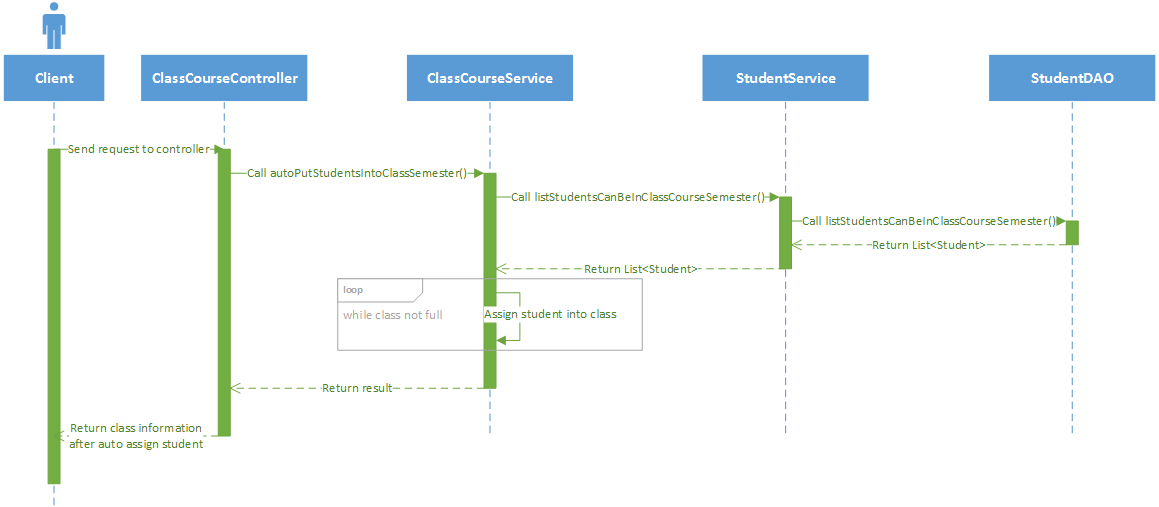
**Method: listStudentsCanBeInClassCourseSemester**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | List all students can be in a class course semester(s) | | | |
| Return | **Type** | | | **Description** |
| List<Student> | | | All students can be in a class course semester(s) |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classSemesterId | Int | Id of ClassSemester. |
| 2 | specializedId | Int | Id of Specialized |
| 3 | detailSpecializedId | Int | Id of detail Specialized |
| 4 | semesterNumber | Int | Semester number |
| 5 | classCourseSemesterId | Int | Id of ClassCourseSemester |

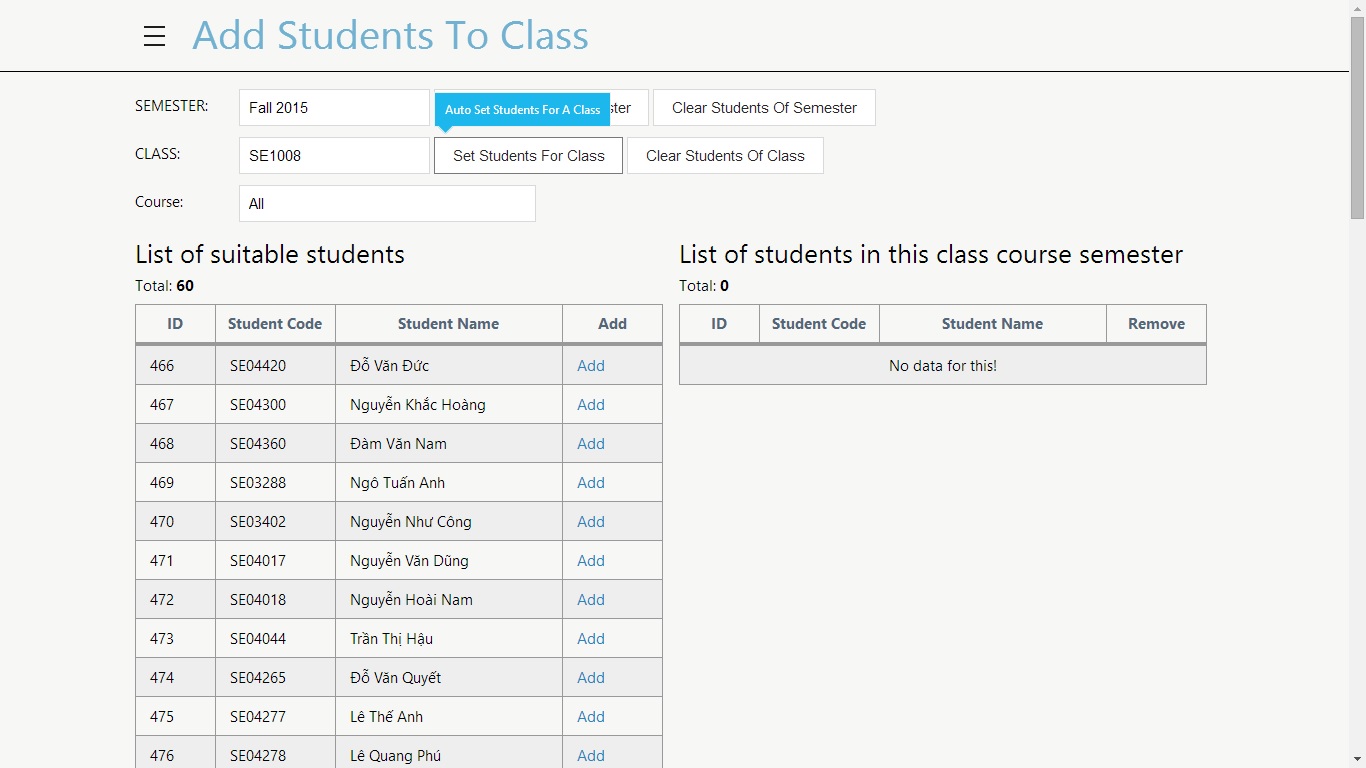
##### Implementation

* Find students with specialized, detail specialized and semester number suitable with this class and not current in any class.
* Add students have been found into class untill number of student exceed 30.

##### Sequence Diagram



##### Screen Design

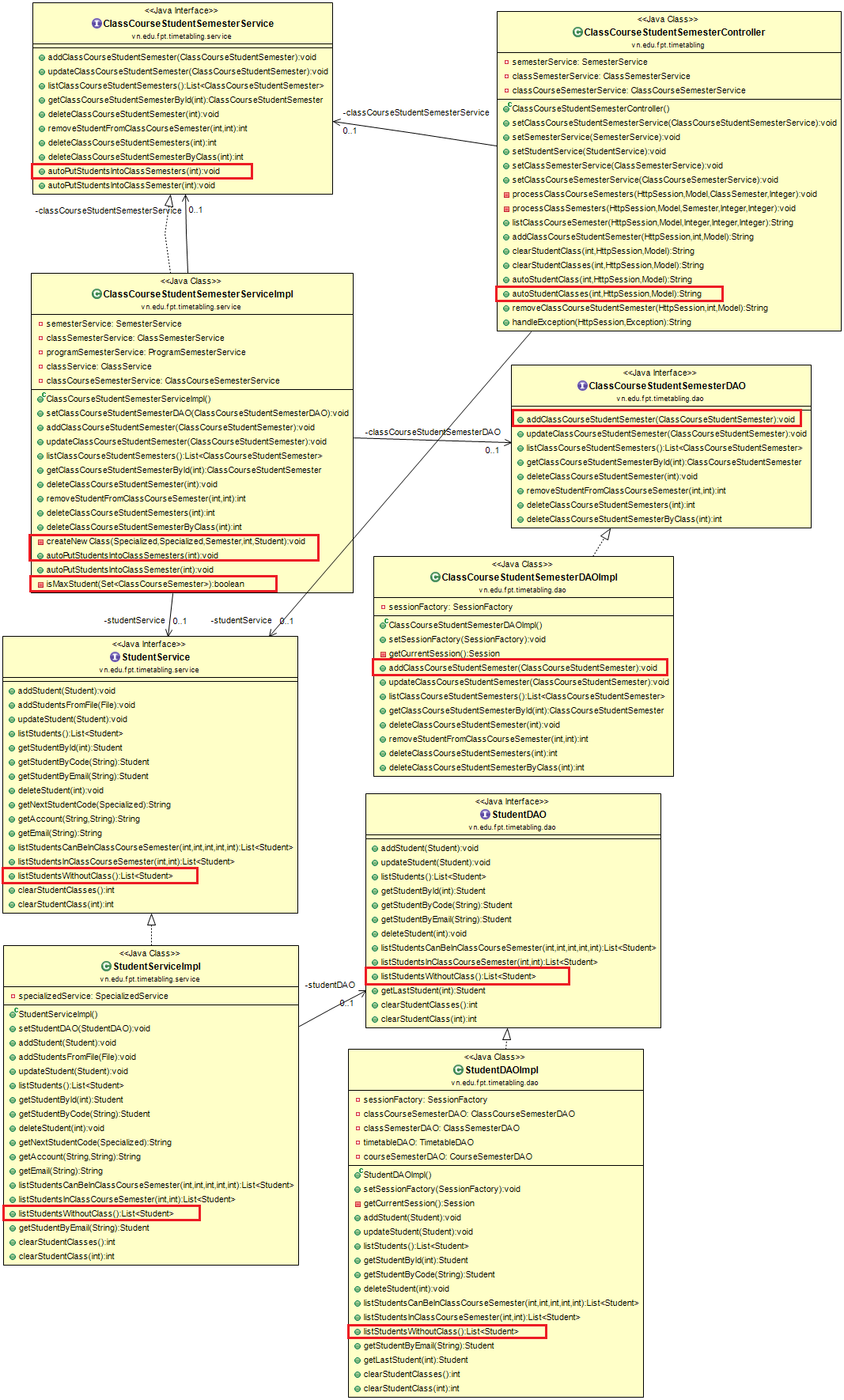


#### Auto put students into classes

##### Description

* Auto put all students without class into classes

##### Class Diagram



##### Class Specification

###### CLASS: ClassCourseStudentSemesterController

**Physical Address:** ..\vn\edu\fpt\timetabling\ClassCourseStudentSemesterController.java

**Method: autoStudentClasses**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Receive data from client, call autoPutStudentsIntoClassSemesters in ClassCourseStudentSemesterService. | | | |
| Return | **Type** | | | **Description** |
| String | | | Redirect to previous page |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of Semester. |

###### CLASS: ClassCourseStudentSemesterServiceImpl

**Physical Address:** ..\vn\edu\fpt\timetabling\service\ClassCourseStudentSemesterServiceImpl.java

**Method: autoPutStudentsIntoClassSemesters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Auto put students into Classes | | | |
| Return | **Type** | | | **Description** |
| void | | |  |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | semesterId | Int | Id of Semester. |

**Method: isMaxStudent**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Check if any ClassCourseSemester in list have number of student exceed 30 | | | |
| Return | **Type** | | | **Description** |
| boolean | | | Have at least a ClassCourseSemester in list have number of student exceed 30 |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classCourseSemesters | Set<ClassCourseSemester> | List of ClassCourseSemester |

**Method: createNewClass**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Create new class and put student into when all suitable classes is full | | | |
| Return | **Type** | | | **Description** |
| void | | |  |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | specialized | Specialized | Specialized of class. |
| 2 | detailSpecialized | Specialized | Detail specialized of class. |
| 3 | semester | Semester | Semester of class. |
| 4 | semesterNumber | int | Semester number of class. |
| 5 | student | Student | Student to put in. |

###### CLASS: ClassCourseStudentSemesterDAOImpl

**Physical Address:** ..\vn\edu\fpt\timetabling\dao\ClassCourseStudentSemesterDAOImpl.java

**Method: addClassCourseStudentSemester**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Add a ClassCourseStudentSemester into Database.. | | | |
| Return | **Type** | | | **Description** |
| void | | |  |
| Parameter | **No.** | **Name** | **Type** | **Description** |
| 1 | classCourseStudentSemester | ClassCourseStudentSemester | ClassCourseStudentSemester to be added. |

###### CLASS: StudentServiceImpl

**Physical Address:** ..\vn\edu\fpt\timetabling\service\StudentServiceImpl.java

**Method: listStudentsWithoutClass**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Call listStudentsWithoutClass of StudentDAO | | | |
| Return | **Type** | | | **Description** |
| List<Student> | | | All students without class. |
| Parameter | **No.** | **Name** | **Type** | **Description** |

###### CLASS: StudentDAOImpl

**Physical Address:** ..\vn\edu\fpt\timetabling\dao\StudentDAOImpl.java

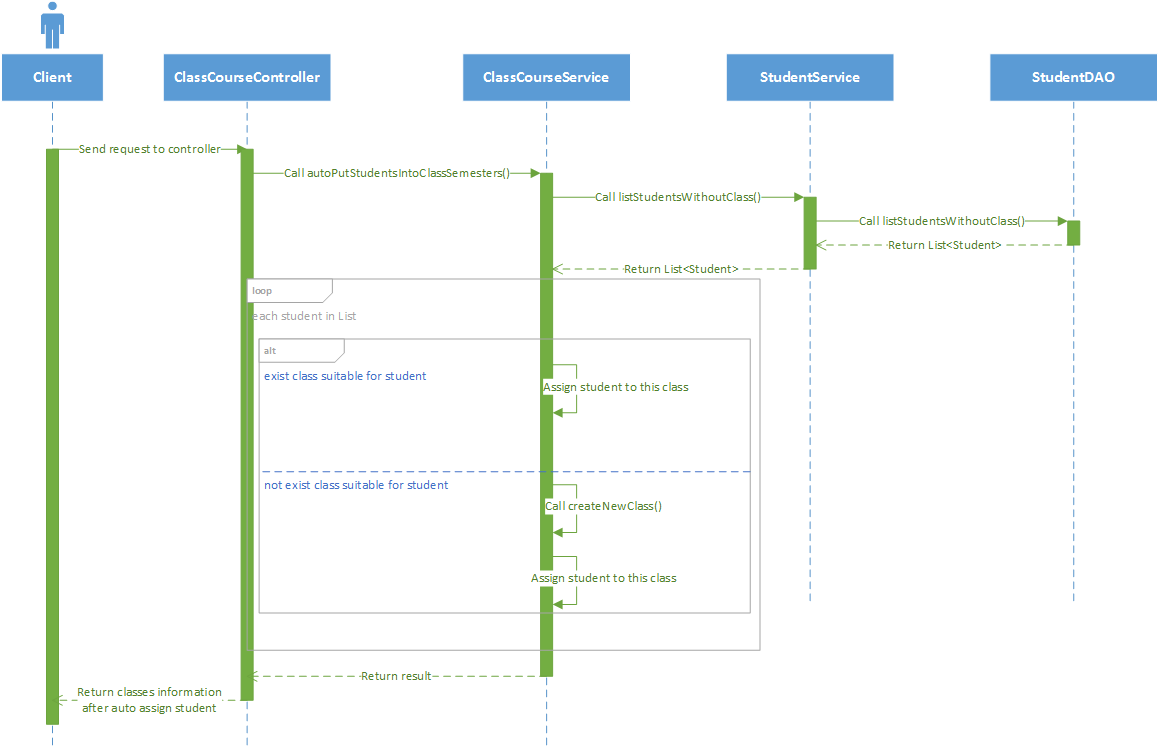
**Method: listStudentsWithoutClass**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Get all student without class. | | | |
| Return | **Type** | | | **Description** |
| List<Student> | | | All students without class. |
| Parameter | **No.** | **Name** | **Type** | **Description** |

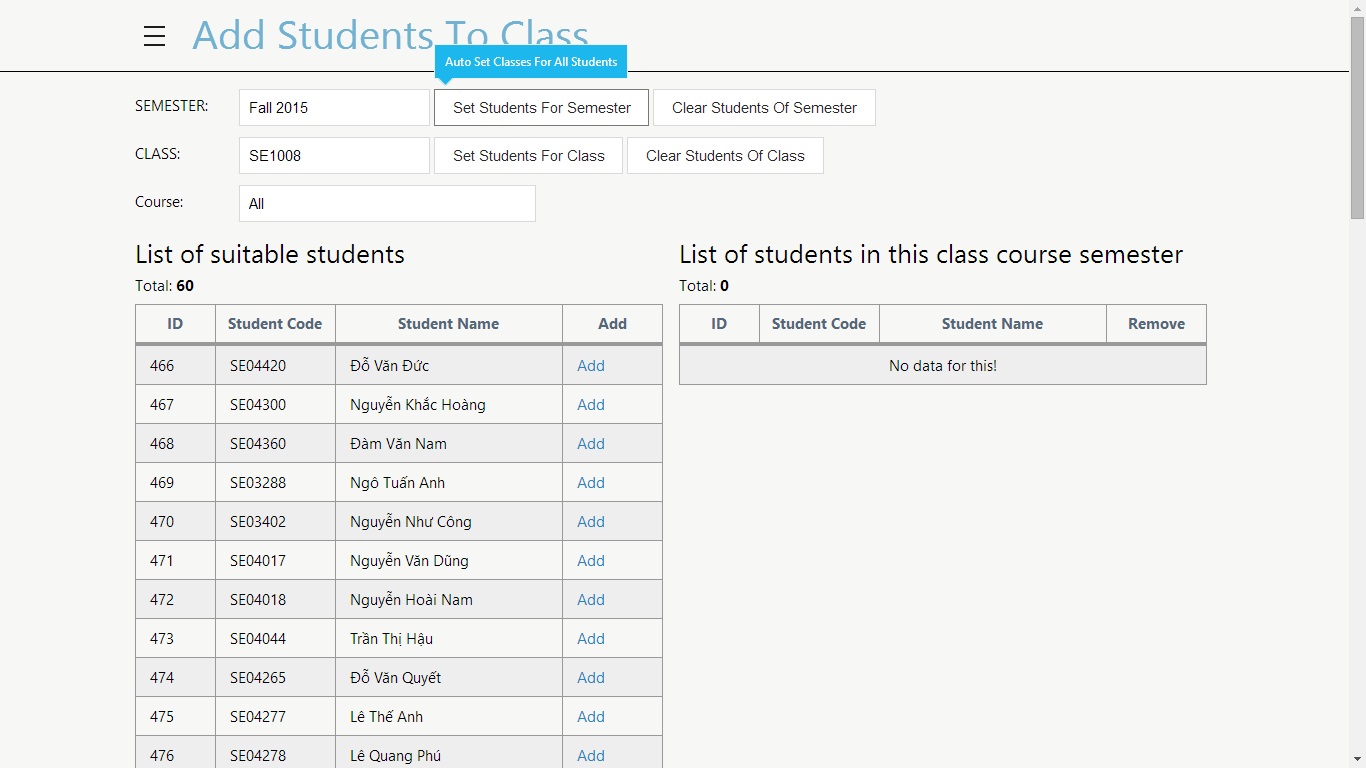
##### Implementation

* Get all student without class.
* For each student find class with suitable specialized, detail specialized, semester number and number of student in class not exceed 30
* If found any suitable class, put student into this class
* It not found ant suitable class, create new class with suitable specialized, detail specialized and semester number and put student into this class

##### Sequence Diagram



##### Screen Design



#### ???

##### Description

##### Class Diagram

##### Class Specification

###### CLASS:

##### Implementation

##### Sequence Diagram

##### Screen Design