MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

Data Structures and Algorithms Visualizer

Group 9	
Group members	Hà Lê Duy Khang – Team Leader – SE61886 Nguyễn Lương Triều Vỹ – Team Member – SE61811 (dropped out) Đặng Xuân Huy – Team Member – SE61318 (dropped out) Dư Đức Long - Team Member - SE62137 (dropped out)
Supervisor	Mr. Kiều Trọng Khánh
Ext. Supervisor	N/A
Capstone Project code	DSAV

- Ho Chi Minh City, 01/2018 –

This page is intentionally left blank

A. Introduction

1. Project Information

- Project name: Data Structures and Algorithms Visualizer
- Project Code: DSAV
- Product Type: Web app & Mobile app
- Start Date: 08/01/2018
- End Date: 28/04/2018

2. Introduction

We introduce to an application that helps learners get clearly understanding and verify their knowledges about data structures and algorithms. In current situation, many learners do not understand deeply in data structures and algorithms.

We build a system, which helps users solve current problems. In the process of analysis, users can input their data based on forms and create new visualization models, so they can see the operating process of algorithms. We believe with visual graph and animations, users are capable to memorize and visualize data structures and algorithms more clearly. Generally, the system will visualize data structures and algorithms step by step by animations.

3. Current Situation

- When learning data structures and algorithms, learners or developers only have text documents, plain source codes about definitions or examples. With complicated algorithms, those algorithms cannot be demonstrated clearly and easily understanding by text definition. Learners often deal with the exercises by looking for source code on the internet rather than coding them.
- Currently, most of learners are studying by lecturers' instruction on class, written on board.
- Some websites support animated algorithms such as visualgo.net, it contains variety of structures and algorithms and it helps users to watch how the algorithm works step by step through animation.

4. Problem Definition

Advantages of current situation:

- Learners can easily understand the algorithm and structure through the animations.
- That application has many common algorithms and structures for learners to refer.
- When the algorithm and structures is implemented, the application can display pseudo code.

Disadvantages of current situation:

- Learners are only aware of basic data structures and algorithms.
- Cannot compare performance between algorithms having the same complexity in different cases.
- Learners lack of a platform to test their understanding and verify their knowledges.

Advantages of common algorithm visualizer websites:

- Implemented diverse of data structures and algorithms.
- Have multiple functions in each data structures.

- Have many algorithms and data structures for users to select.
- Clarify algorithms and data structures through animation.
- Show pseudo code when algorithms start running.

Disadvantages of common algorithm visualizer websites:

- Cannot tracks or visualize their plain source code.
- User cannot suggest or post their new algorithms by plain source code.

5. Proposed Solution

Our proposed solution is build a Web and Mobile Application which can show users visualization model and transforms during process from beginning to the end through animations.

5.1. Feature functions

- The system can model some data structures and algorithms (at least 10 theories).
- The system can visualize the step by step of these theories to users using example.
- The system allows the user to make input into formatted form that the application can visual with specifying theory.

5.2. Values and challenges

Values:

- Provides source code display function.
- Highlighting source code every steps of the function.
- The system analyzes the old models to support the best modeling for general structures and algorithms. If users can describe their data structures or algorithms based on the giving template, the system can be able to visualize their ideas.

Challenges:

- The system cannot provide all existing algorithms and data structures.
- System only understand one existing pseudo code format.

6. Functional Requirements

- User components:
  - Users can find out which models are in the system and select which model to use.
  - Users can get knowledge about detail definitions of data structures and algorithms.
  - Users can know how each data structure or algorithm works step by step through animations.
  - Users can make more examples to understand more about how each data structure or algorithm works by entering new data input.
  - Users can make new data structure or algorithm to enhance the diversity.
- Staff components:
  - Web Application for supporting staff to import or maintain the new data structures or algorithms.
  -

7. Role and Responsibility

No	Full name	Role	Position	Contact
1	Kiều Trọng Khánh	Project Manager	Supervisor	khanhkt@fpt.edu.vn
2	Hà Lê Duy Khang	Developer	Leader	khanghldse61886@fpt.edu.vn
3	Nguyễn Lương Triều Vỹ	Developer	Member	vynltse61811@fpt.edu.vn
4	Đặng Xuân Huy	Developer	Member	huydxse61318@fpt.edu.vn
5	Dư Đức Long	Developer	Member	longddse62137@fpt.edu.vn

Table 1: Roles and Responsibilities

B. Software Project Management Plan

1. Problem Definition

1.1. Name of this Capstone Project

Data Structures and Algorithms Visualizer (DSAV).

1.2. Problem Abstract

A large body of research indicates that visual cues help us to better retrieve and remember information. The research outcomes on visual learning make complete sense when you consider that our brain is mainly an image processor (much of our sensory cortex is devoted to vision), not a word processor. In fact, the part of the brain used to process words is quite small in comparison to the part that processes visual images.

Data Structures and Algorithms is a difficult subject to imagine and understand clearly. We think that visualization may help us to deal with this subject. We want to build a reliable software about data structures and algorithms visualizer to help learners to better retrieve and remember knowledge about this subject.

1.3. Project Overview

1.3.1. Current Situation and Disadvantages

Project team are facing many challenges on this project:

- Learning new technologies (canvas, HTML, JavaScript, JQuery and d3js framework, ReactJS).
- Implementation (convert data structure into drawings, display method by animation, display algorithm's code progress).
- Design format for user to create their own algorithms.

1.3.2. The Proposed System

We have to study new technologies for some these main reasons:

- How to visualize a specify data structure into pictures, drawings.
- How to create animation when the system runs an algorithm.
- How to display algorithm's code to users.
- How to make form with format for user to input their new algorithms.

1.3.2.1. Web Application

- Staffs can import or maintain a data structure or algorithm.
- Users can browse all algorithms and data structures that our system support and select a specify one.
- Users can search for the data structure or algorithm that they want to learn.
- Users can view detail definitions of data structures and algorithms.
- Users can view and control the visualizer of data structure or algorithm.
- Users can import new data structure or algorithm to the system through flow chart.
- The system allows users to upgrade new algorithms to enhance the diversity.

1.3.3. Boundaries of the System

- The system should do:
    - Allow user to learn data structure that our system support.
    - Allow user to edit data structure's elements.
    - Allow user to watch the system runs algorithms step by step.

- Allow user to add new algorithms to some data structures that the system support.

- Our system will not:
  - Implement all existing data structures and algorithms.

1.3.4. Development Environment

1.3.4.1. Hardware requirements

For server

	Minimum requirements	**Recommended**
Internet Connection	Cable, Wi-Fi (8 Mbps)	Cable, Wi-Fi (20 Mbps)
Operating System	Window Server 2008	Window Server 2008
Computer Processor	Intel® Xeon ® 1.4GHz	Intel® Xeon ® Quad Core (12M Cache, 2.50 GHz)
Computer Memory		

1.3.4.2. Software requirements

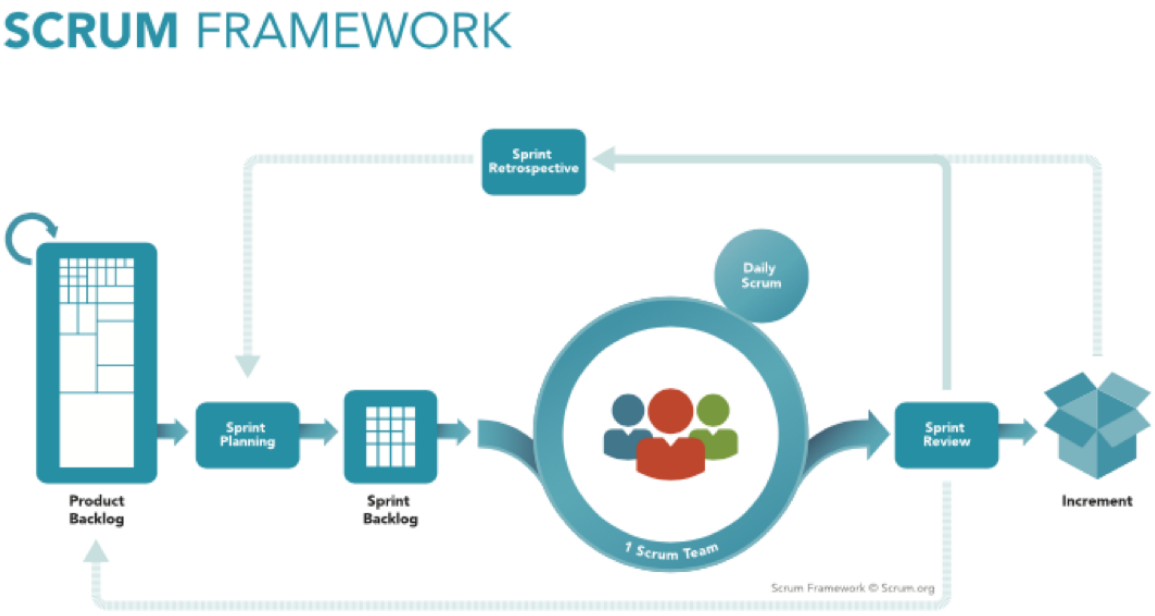
- SQL Server 2008 Enterprise R2: used to create and manage the database for system.
- Netbeans, Webstorm, Visual Studio Code: used to implement web application, web service and Android application.
- StarUML: used to create models and diagrams.
- Ninjamock: used to create UI mockup.

2. Project Organization

2.1. Software Process Model

This project is developed by using Scrum model – part of an agile framework for Software development project. Our team choose Scrum model because of the following reasons:

- Prototypes are delivered frequently for evaluation, usually weekly, rather than months due to our time is only 14 weeks.
- Take fewer risks when there is a change in requirement since we need to improve project more than the existing one.
- All members must work together in order to avoid misunderstanding or miscommunication.
- Able to study new skills or knowledge at the same time as developing.



https://www.scrum.org/

Artifact	Feature
Product backlog	The product backlog comprises an ordered list of requirements that a scrum team maintains for a product. It consists of features, bug fixes, non-functional requirements … - whatever must be done to successfully deliver a viable product.
Sprint backlog	The sprint backlog is the list of work the development team must address during the next sprint. The list is derived by the scrum team progressively selecting product backlog items in priority order from the top of the product backlog until they feel they have enough work to fill the sprint. The development team should keep in mind its past performance assessing its capacity for the new sprint, and use this as a guide line of how much 'effort' they can complete.
Product increment	The increment (or potentially shippable increment, PSI) is the sum of all the product backlog items completed during a sprint, integrated with the work of all previous sprints. At the end of a sprint, the increment must be complete, according to the scrum team's definition of done (DoD), fully functioning, and in a usable condition regardless of whether the product owner decides to actually release it.

2.2. Roles and responsibilities

No	Full name	Role	Responsibilities
1	Kiều Trọng Khánh	Project owner	- Specify scope and user requirement. - Supervise the development progress. - Provide professional techniques and business analysis support.
2	Hà Lê Duy Khang	Scrum master	- Managing process - Designing database - Clarifying requirements - Prepare documents - GUI design - Create test plan - Coding - Testing
3	Nguyễn Lương Triều Vỹ	Member	- Designing database - Clarifying requirements - Prepare documents - GUI design - Create test plan - Coding - Testing
4	Đặng Xuân Huy	Member	- Designing database - Clarifying requirements - Prepare documents - GUI Design - Create test plan - Coding - Testing
5	Dư Đức Long	Member	- Designing database - Clarifying requirements - Prepare documents - GUI Design - Create test plan - Coding - Testing

2.3. Tools and Techniques

	Tools	**Techniques**
Front-end	Visual Studio Code	- HTML5 - CSS3 - JavaScript - jQuery - ReactJS
Back-end	Visual Studio Code	- NodeJS
Database management system	MySQL	- SQL

3. Project Management Plan

3.1. Product Backlog

Story ID	Features	Task ID	Task description	Sprint
1	Create Product Backlog	1	Create Product Backlog	1
2	Write Introduction document	2	Write Introduction document	1
3	Write Project Management Plan	3.1	Problem definition	1
		3.2	Project organization	1
		3.3	Project management plan	1
		3.4	Coding convention	1
		3.5	Review document	1
4	Build System Structure	4.1	Backend Structure	2
		4.2	Web Structure	2
5	Write Software Requirements	5.1	User Requirement Specification	2
		5.2	External Interface Requirement	2
		5.3	Use case diagram	2
		5.4	Software System Attributes	2
6	Write Software Design Description	6.1	Design Overview	3
		6.2	System Architectural Design	3
		6.3	Component Diagram	3
		6.4	Detailed Description of Components	3
		6.5	Sequence Diagram	3
		6.6	User Interface Diagram	3
		6.7	Database Design	3
		6.8	Entity Diagram	3
		6.9	Class Diagram	3
7	Implementation	7		4
8	Create Software Test Documentation	8.1	Test Plan	5
		8.2	Test Cases	5
		8.3	Check lists	5
9	Quality Assurance	9.1	Quality Assurance for Backend	5

Story ID	Features	Task ID	Task description	Sprint
1	Create Product Backlog	1	Create Product Backlog	1
2	Write Introduction document	2	Write Introduction document	1
3	Write Project Management Plan	3.1	Problem definition	1
		3.2	Project organization	1
		9.2	Quality Assurance for Web	5
		9.3	Quality Assurance for Mobile	5
10	Software User's Manual	10.1	Installation Guide	6
		10.2	User's Guide	6
11	Mobile prototype	11	Mobile prototype	7

Product backlog

C. Software Requirement Specification

1. User Requirement Specification

1.1. Administrator Requirement

Administrator is a person who access to the system as Administrator role. Administrator can use following functions:

- Change user's role.

1.2. Staff Requirement

Staff is a person who access to the system as Staff role. Staff can use following functions:

- Manage course: create, update, remove course.
- Manage topic: create, update, remove topic.
- Manage lesson: create, update, remove lesson.
- Manage algorithm: create, update, remove algorithm.

1.3. Authenticated User Requirement

Authenticated User is a person who access to the system as Authenticated User role. Authenticated User can use following functions:

- Log out.
- View profile.
- Edit profile.
- Search course.
- View learned courses.
- View course.
- Enroll course.
- Resume last lesson.
- Learn lesson.
- Interact with the Visualizer: Create data, Control the Visualizer.

1.4. Unauthenticated User Requirement

Unauthenticated User is a person who does not access to the system. Unauthenticated User can use following functions:

- Sign up.
- Log in.

2. Software Requirement Specification

2.1. External Interface Requirement

2.1.1. User Interface

- The user interface uses English as main language for users and for Staff, Manager and Admin on Web application.
- The user interface displays best on 1024x768 and above screen size.

2.1.2. Hardware Interface

Desktop PC

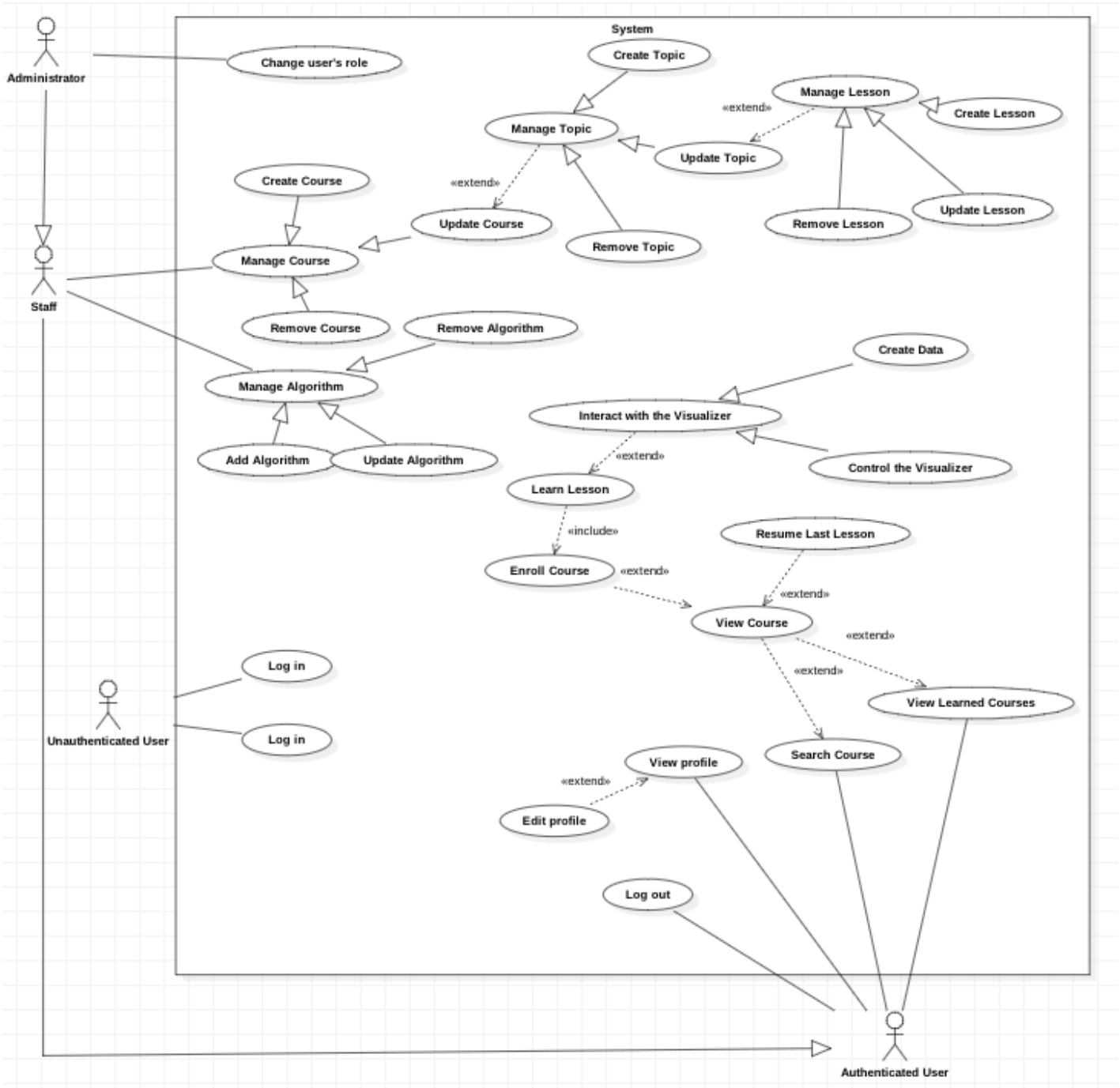
2.1.3. Software Interface

- Web application: work with Firefox (v30 or above), Chromes (v25 or above)

2.1.4. Communication Protocol

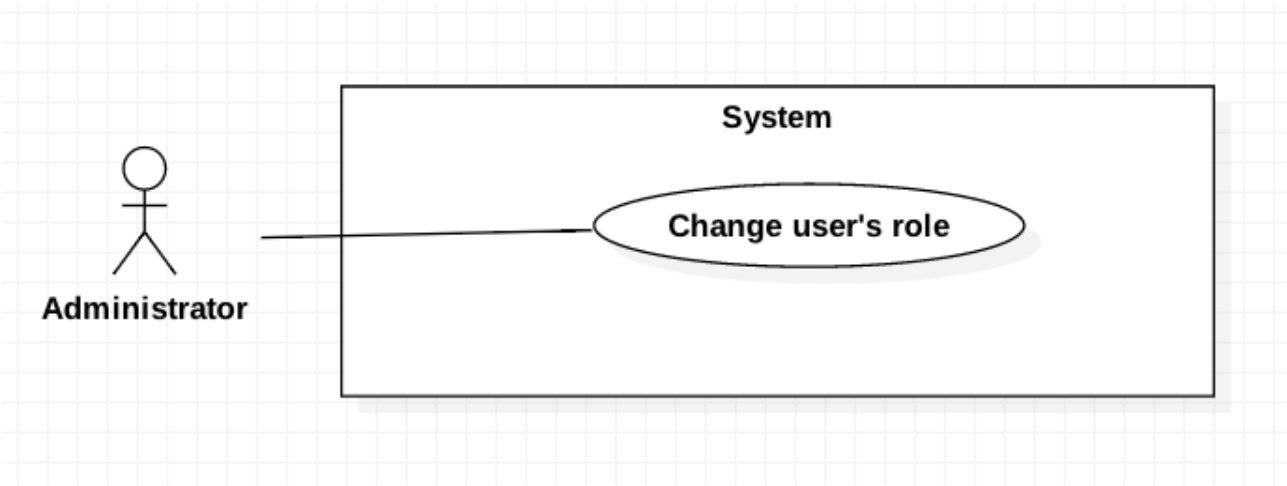
- Use HTTP protocol 1.1 for communication between the web browser and the web server.
- Use HTTP protocol 1.1 for communication between the server and the Microsoft service.

2.2. System Overview Use Case

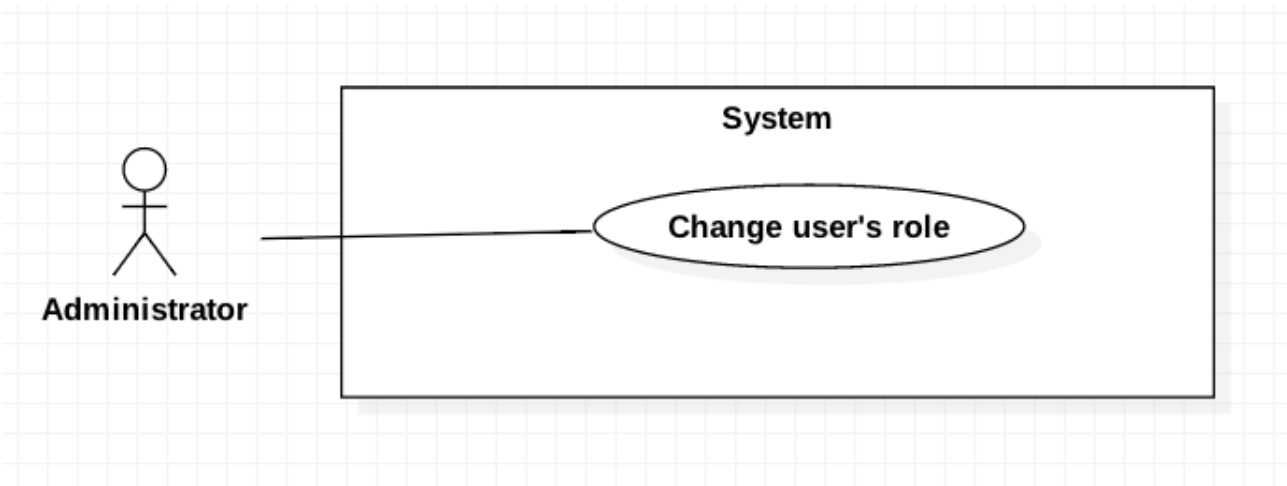


2.3. List of Use Case

2.3.1. <Administrator> Overview Use Case

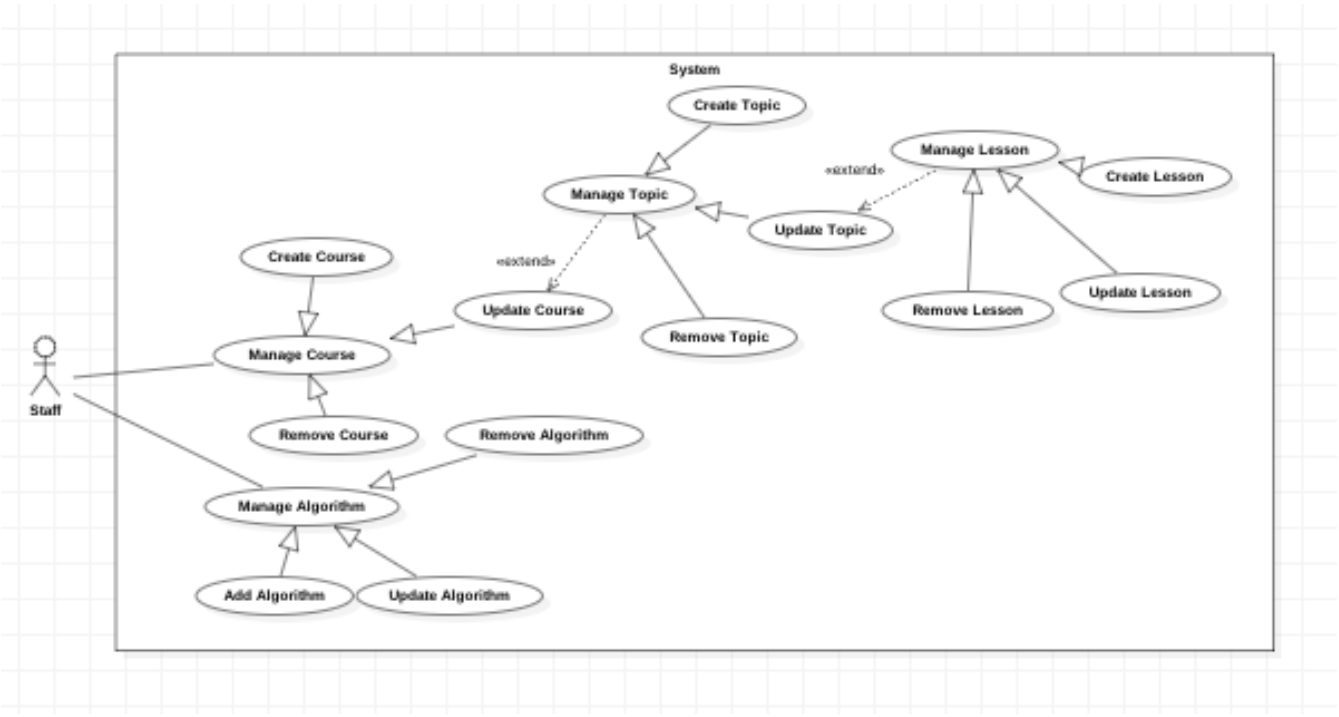


2.3.1.1. <Administrator> Change user's role (UC_AD01)



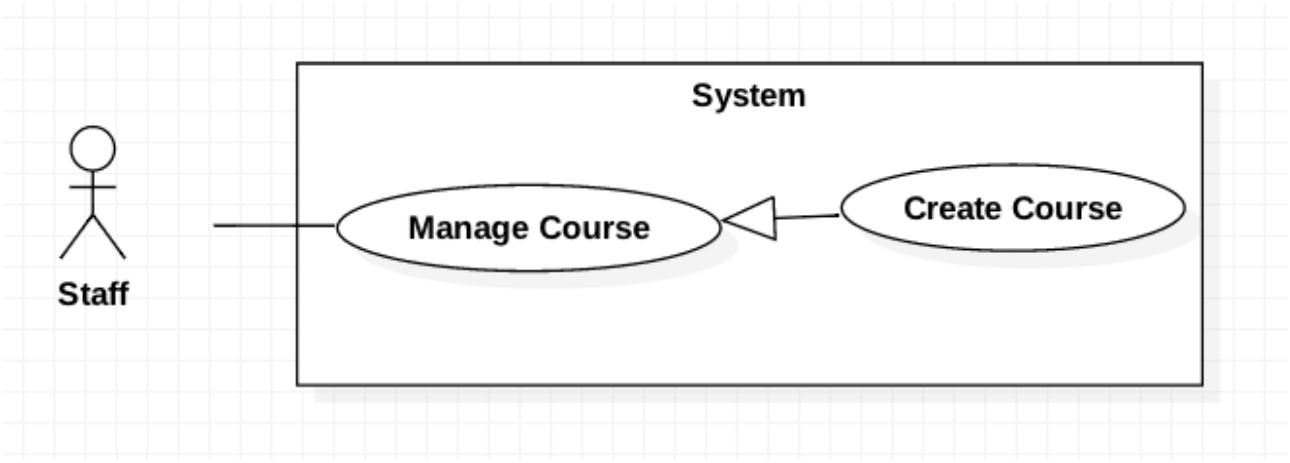
USE CASE – UC_AD01			
Use Case No.	UC_AD01	**Use Case Version**	1.0
Use Case Name	Change user's role		
Author	KhangHLD		

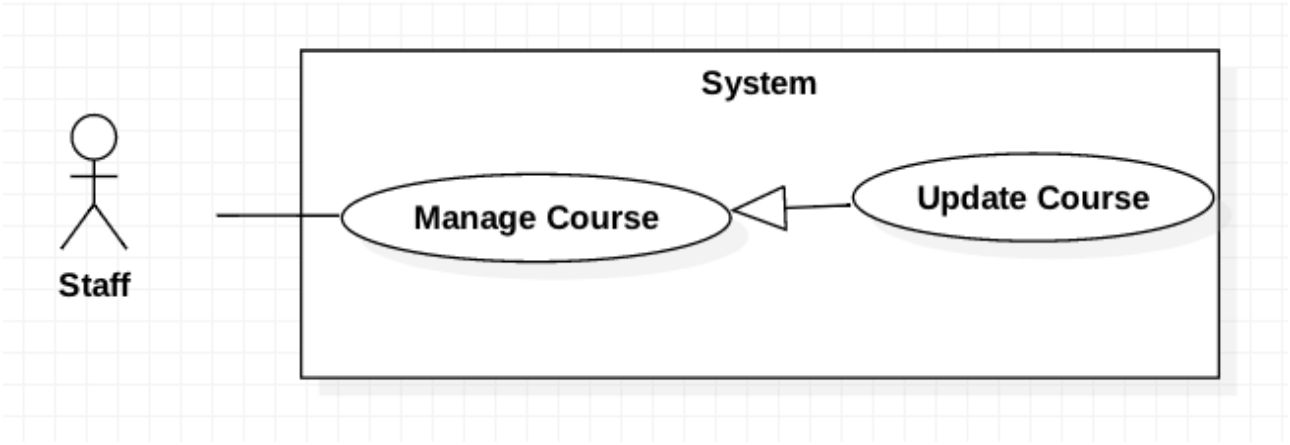
USE CASE – UC_AD01			
Date	14/02/2018	**Priority**	High
Actor	Administrator		
Summary	This use case allows Administrator to change a user role		
Goal	Update a user's role.		
Triggers	Actor sends update role command.		
Pre Conditions	Actor logged in as Administrator role.		
Post Conditions	• Success: Actor successfully updates role of a user to the database. • Fail: System shows error messages		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor visits to "Manage users" page	System redirects to "Manage users" page
	2	Actor changes role of selected user by editing the Role dropdown list.	
	3	Actor clicks 'Save' button of selected user.	System saves updated information and displays them immediately
Exceptions	N/A		
Relationships	N/A		
Business Rules	• User already logged in the system as "Administrator" role. • The available role values for Administrator to change are: Administrator, Staff and User.		



2.3.2. <Staff> Overview Use Case

2.3.2.1. <Staff> Create Course (UC_S01)





USE CASE – UC_S01			
Use Case No.	UC_S01	**Use Case Version**	1.0
Use Case Name	Create Course		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High

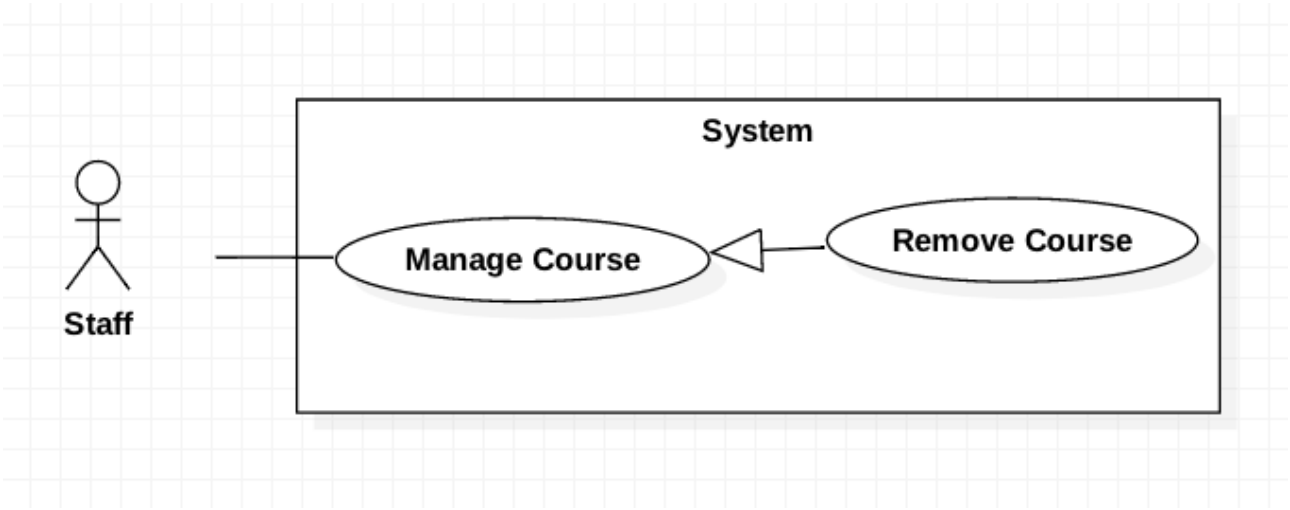
USE CASE – UC_S01			
Actor	Staff		
Summary	This use case allows staff to create course.		
Goal	Staff successfully creates new course.		
Triggers	Actor sends create course command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: New course is created into database and redirects actor to that course's information page. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks "Create Course" button from Manage course page	System displays modal with field: "Enter Course name" requires actor to input course name.
	2	Actor input course name and clicks "Create Course" button. [Exception no.1]	System redirects actor to that course's page including details about that course.
	3	Actor clicks 'Save' button of selected user.	System saves updated information and displays them immediately
Exceptions	No	Cause	System Response
	1	Actor does not input required field or input wrong field's requirement	System notices that "Can not create course. The course name must be 10-50 characters."
Relationships	Abstract use case: Manage course Extended by: Manage topic		
Business Rules	• User already logged in the system as "Staff" role. • The name of course must be in range of 10-50 characters. • When created, course will have default course image. Staff can change it in update course. • List course in "Manage course" page will be updated with new course that just created. • When created, the course will be "Unpublished" as default. When finishing create topics and lessons, actor needs to press 'Publish' button to publish the course.		

2.3.2.2. <Staff> Update Course (UC_S02)

USE CASE – UC_S02			
Use Case No.	UC_S02	**Use Case Version**	1.0
Use Case Name	Update Course		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to update a course.		
Goal	Staff successfully updates existed course.		
Triggers	Actor sends update course command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: The course with new information will be updated in database. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks selected Course in Manage Course page.	System displays the Course Information page with following fields: • Name: text box • Description: text area • Difficulty: dropdown list • Price: numeric input field • ‘Save’ button
	2	Actor updates these fields.	
	3	Actor clicks ‘Save’ button. [Exception no.1]	System displays popup “Updated successfully”.
Exceptions	No	Cause	System Response
	1	Actor does not input required field or input wrong field’s requirement	System notices that “The course name must be 10-50 characters.”
Relationships	Abstract use case: Manage course Extended by: Manage topic		

USE CASE – UC_S02	
Business Rules	• User already logged in the system as “Staff” role. • If update successfully, the course will be updated in the database immediately. • New information will be updated in Manage course page. • The name of course must be in range of 10-50 characters. • The price will be between 0 (free course) and 100 (maximum). • Only staff who created the course can modify it.

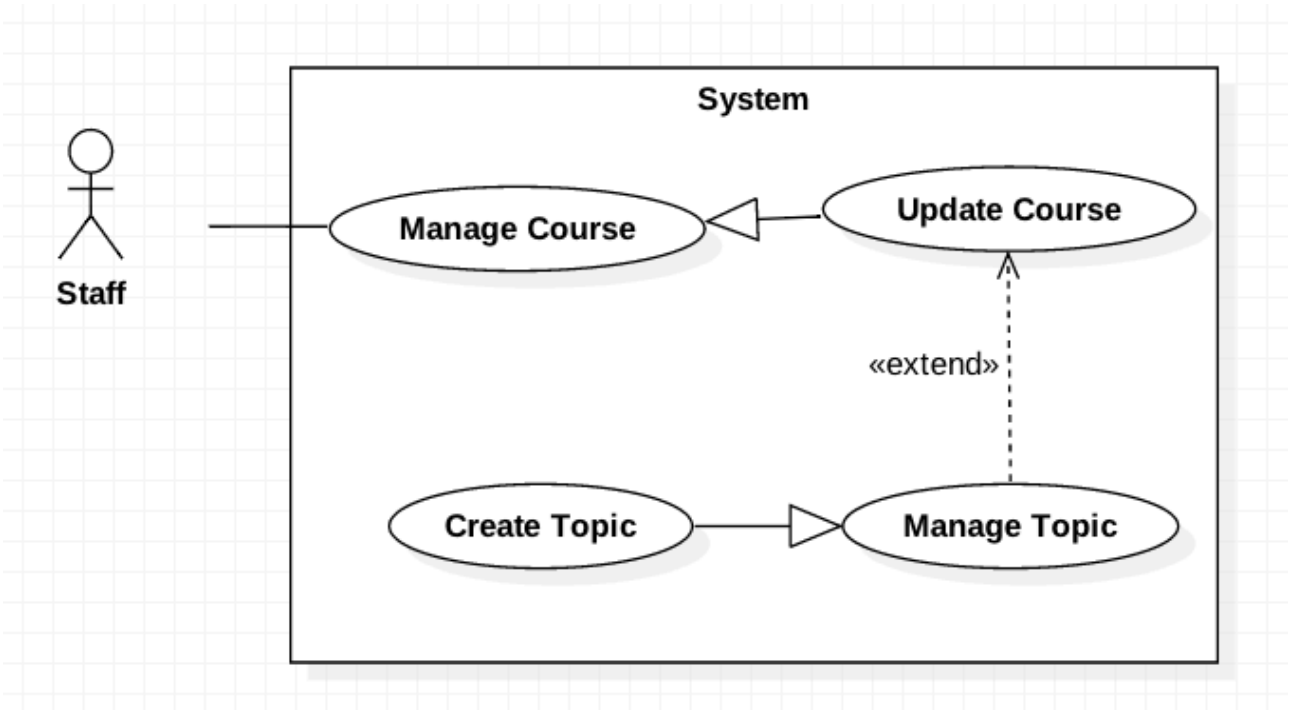
2.3.2.3. <Staff> Remove Course (UC_S03)



USE CASE – UC_S03			
Use Case No.	UC_S03	**Use Case Version**	1.0
Use Case Name	Remove Course		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to remove a course.		
Goal	Staff successfully removes existed course.		
Triggers	Actor sends remove course command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: The course’s status in database will be changed to “-1” and display successful message. • Fail: System displays error messages.		

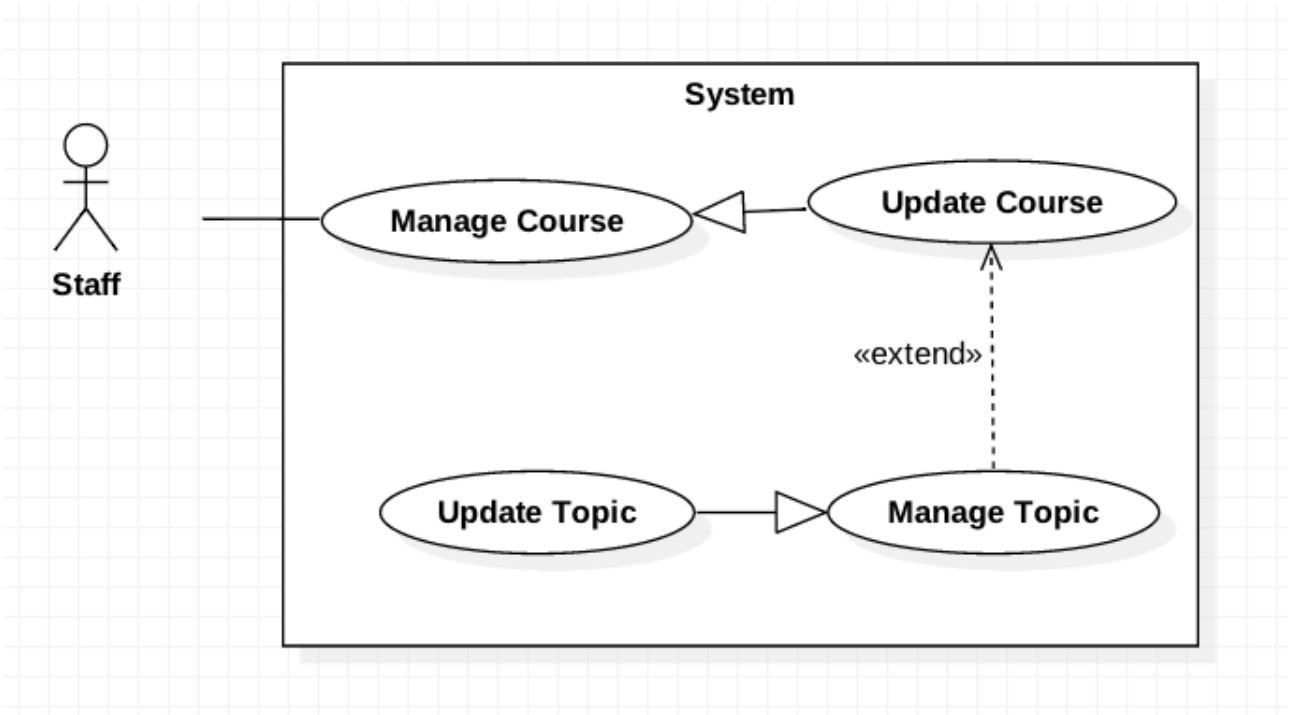
USE CASE – UC_S03			
Main Success Scenario	Step	Actor Action	System Response
	1	Actor changes the 'Status' radio button of selected Course from 'Published' to 'Unpublished' in Manage Course page.	
	2	Actor clicks 'Save button'.	System displays popup "Removed Course successfully".
Exceptions	N/A		
Relationships	Abstract use case: Manage course		
Business Rules	• User already logged in the system as "Staff" role. • The removed course's status will be change to "-1" in database. Therefore, any data such as topics, lessons relating to the course are not completely deleted from database. • The removed course can be changed to 'Published'. • User cannot see or access to the course that has been removed. • Only staff who created the course can remove it.		

2.3.2.4. <Staff> Create Topic (UC_S04)

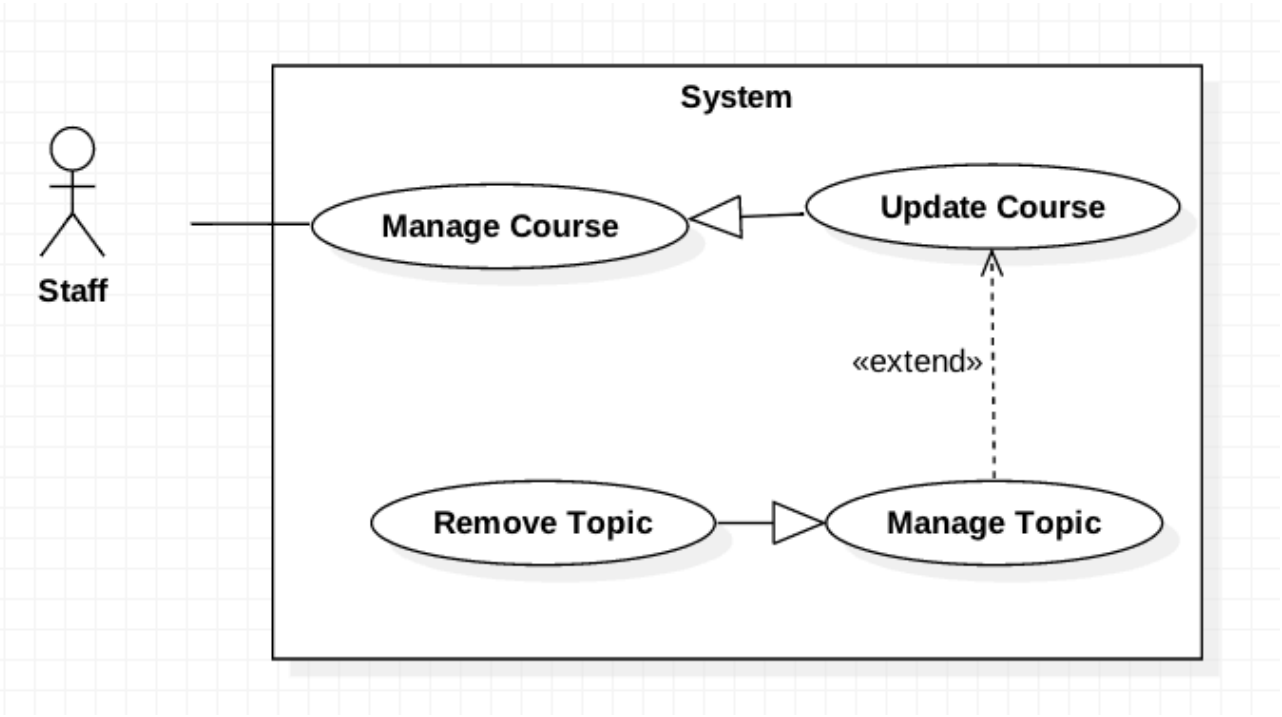


USE CASE – UC_S04			
Use Case No.	UC_S04	**Use Case Version**	1.0
Use Case Name	Create Topic		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to create topic of existed course.		
Goal	Staff successfully creates new topic of existed course.		
Triggers	Actor sends create topic command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: New topic is created into database. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks selected Course in Manage Course page.	System displays the Course Information page.
	2	Actor moves to ‘Course Contents’ field.	
	3	Actor clicks ‘Create new Topic’ button.	System creates new Topic with default title: ‘Topic Title’.
	4	Actor enters topic title in the ‘Topic Title’ filed then press Enter. [Exception no.1]	System displays popup “Create topic successfully”.
Exceptions	No	Cause	System Response
	1	Actor does not input required field or input wrong field’s requirement	System notices that “Can not create topic. The topic title must be 10-50 characters.”
Relationships	Abstract use case: Manage topic Extended by: Manage lesson		
Business Rules	• User already logged in the system as “Staff” role. • The title of topic must be in range of 10-50 characters. • The topic created must be in a specific course. • Staff can only create topic in their own’s course. • Course Contents in Course page will be updated with new topic that just be created.		

2.3.2.5. <Staff> Update Topic (UC_S05)



USE CASE – UC_S05			
Use Case No.	UC_S05	**Use Case Version**	1.0
Use Case Name	Update Topic		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to update a topic.		
Goal	Staff successfully updates existed topic.		
Triggers	Actor sends update topic command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: The topic with new information will be updated in database. • Fail: System displays error messages.		



USE CASE – UC_S05			
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks selected Course in Manage Course page.	System displays the Course Information page.
	2	Actor moves to ‘Course Contents’ field.	
	3	Actor clicks ‘Edit’ button. The only field that Actor can edit is ‘Topic Title’.	
	4	Actor clicks ‘Save’ button. [Exception no.1]	System displays popup “Updated successfully”.
Exceptions	No	Cause	System Response
	1	Actor does not input required field or input wrong field’s requirement	System notices that “The topic title must be 10-50 characters.”

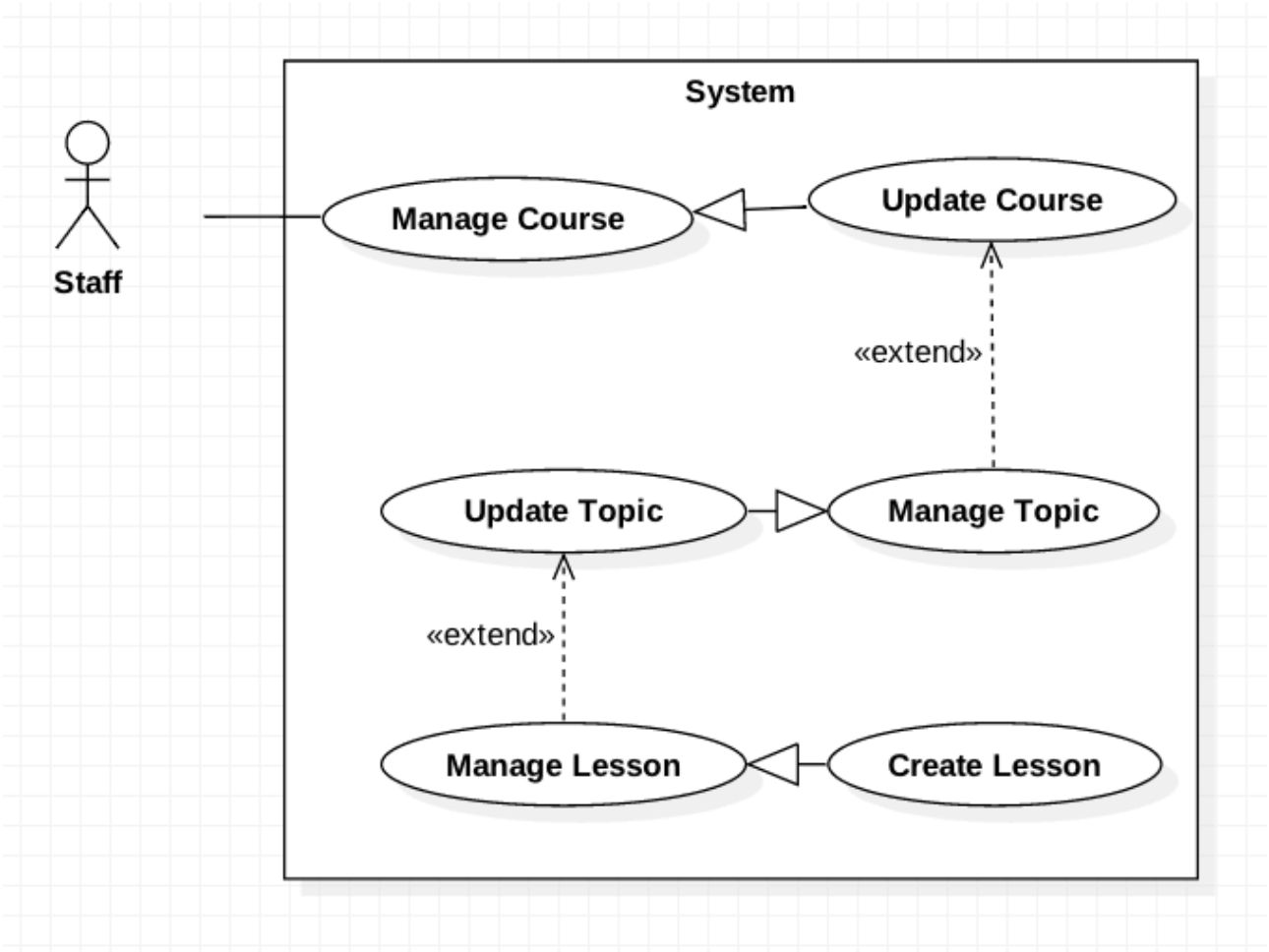
USE CASE – UC_S05	
Relationships	Abstract use case: Manage topic Extended by: Manage lesson
Business Rules	• User already logged in the system as “Staff” role. • If update successfully, the topic will be updated in the database immediately. • New information of the topic will be updated in Course page. • The topic title must be in range of 10-50 characters. • Only staff who created the course can modify topic in that course.

2.3.2.6. <Staff> Remove Topic (UC_S06)

USE CASE – UC_S06			
Use Case No.	UC_S06	**Use Case Version**	1.0
Use Case Name	Remove Topic		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to remove a topic.		
Goal	Staff successfully removes existed topic.		
Triggers	Actor sends remove topic command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: The topic will be removed from database. • Fail: System displays error messages.		
Main Success	Step	Actor Action	System Response

USE CASE – UC_S06			
Scenario	1	Actor clicks selected Course in Manage Course page.	System displays the Course Information page.
	2	Actor moves to ‘Course Contents’ field.	
	3	Actor clicks ‘Delete’ button of selected Topic.	System displays a popup to confirm.
	4	Actor clicks ‘Yes’ button. [Exception no.1]	System displays popup “Removed successfully”.
Exceptions	N/A		
Relationships	Abstract use case: Manage topic		
Business Rules	• User already logged in the system as “Staff” role. • The topic will be completely removed from database including its lessons. • User cannot see or access to the topic that has been removed. • Only staff who created the course can remove topic in that course.		

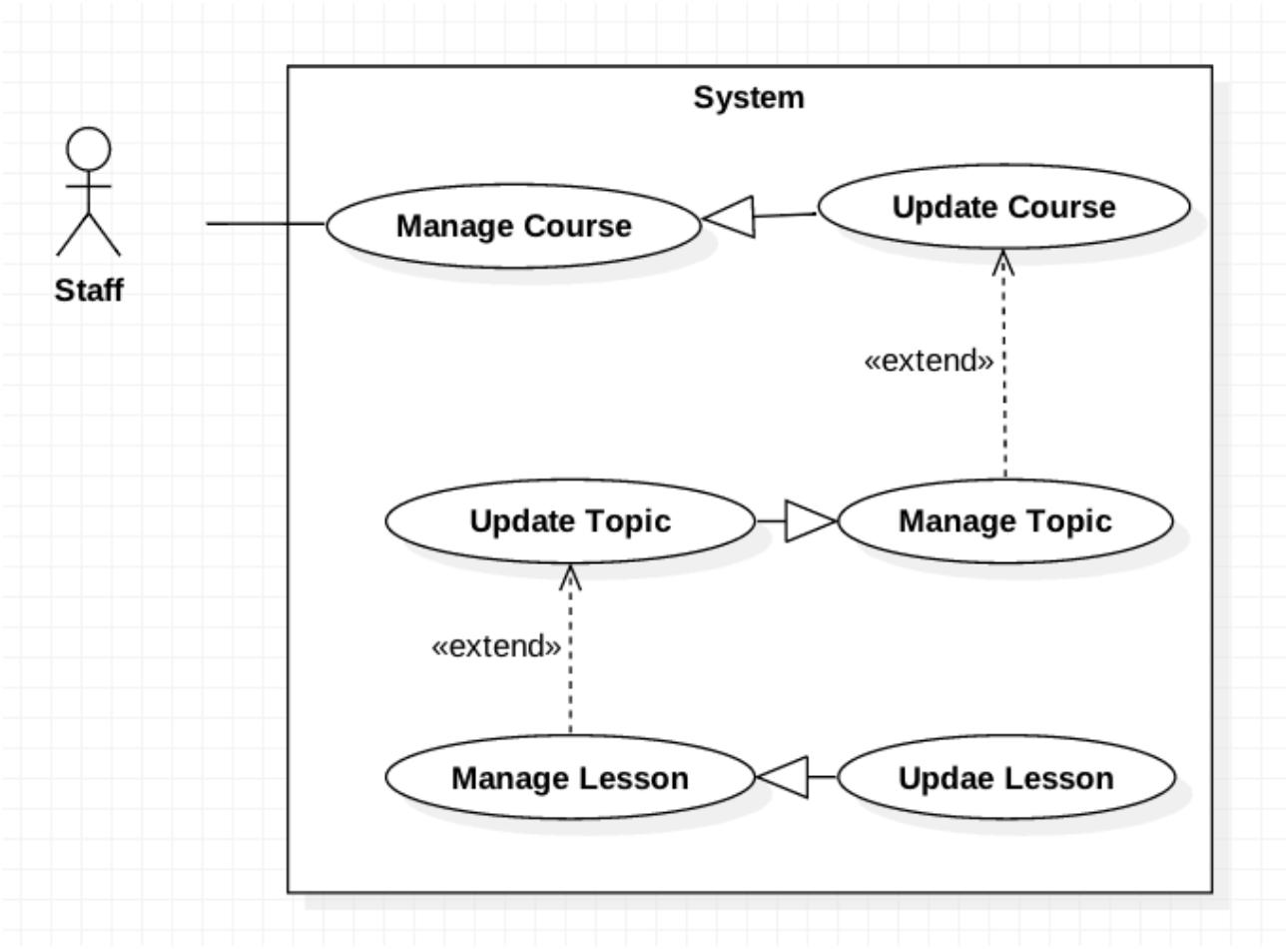
2.3.2.7. <Staff> Create Lesson (UC_S07)



USE CASE – UC_S07			
Use Case No.	UC_S07	**Use Case Version**	1.0
Use Case Name	Create Lesson		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to create lesson of existed topic.		
Goal	Staff successfully creates new lesson of existed topic.		
Triggers	Actor sends create lesson command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: New lesson is created into database. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks selected Course in Manage Course page.	System displays the Course Information page.
	2	Actor moves to ‘Course Contents’ field.	
	3	Actor selects Topic to create Lesson.	System displays the content lessons and ‘Create new Lesson’ button.
	4	Actor clicks ‘Create Lesson’ button.	System displays popup “Create topic successfully”.
	5	Actor enters ‘Lesson name’, chooses ‘Lesson type’ (text or animation).	System displays the ‘Lesson contents’ field if actor chose Text type. System displays the algorithms dropdown list if actor chose Animation type.
	6	Actor enters ‘Lesson contents’ (if actor chose the Text type), or chooses the Algorithm in dropdown list (if actor chose the Animation type). Actor clicks ‘Finish’ button. [Exception no.1]	System displays popup “Create new Lesson successfully”.
Exceptions	No	Cause	System Response

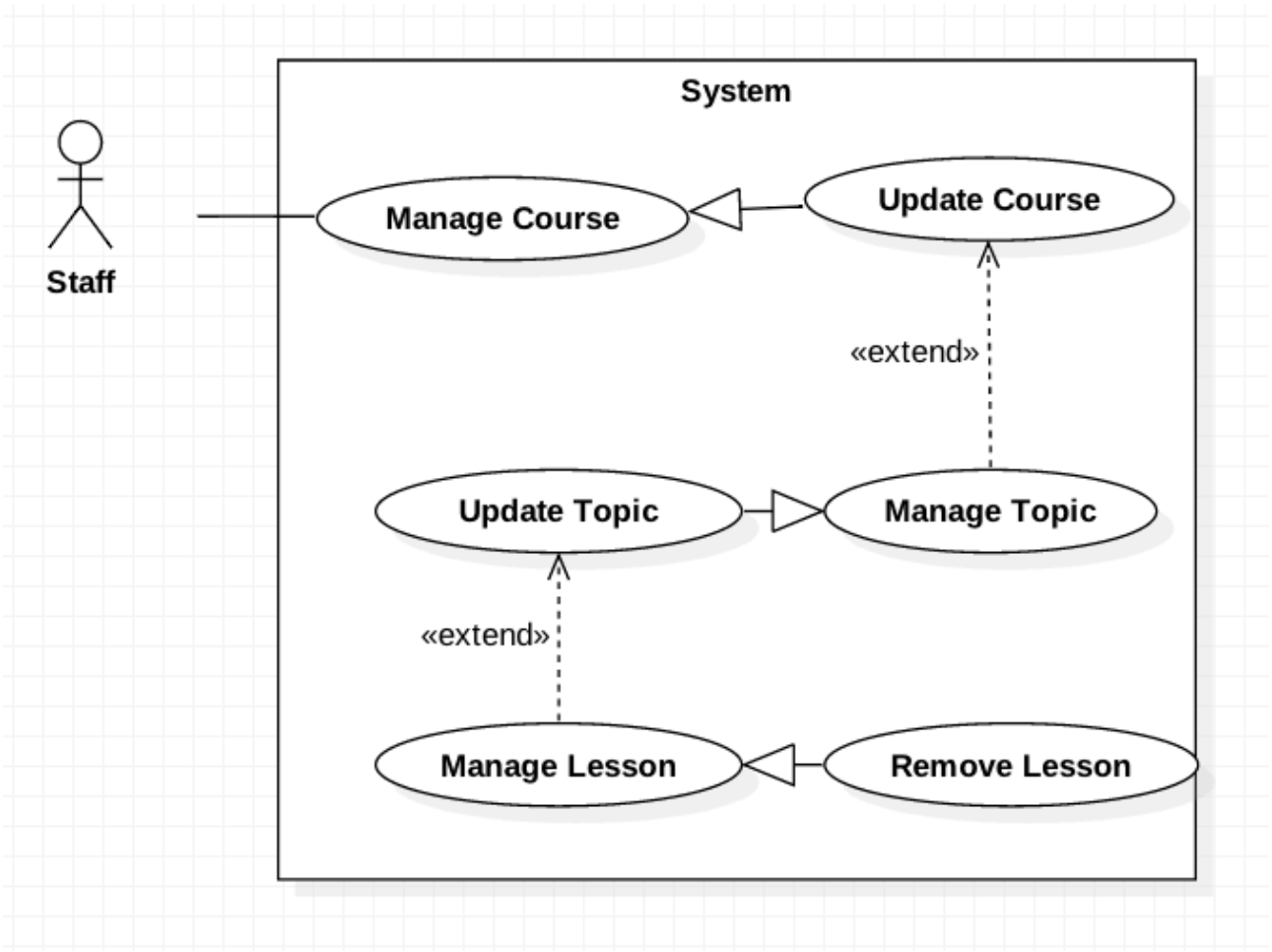
USE CASE – UC_S07			
	1	Actor does not input required field or input wrong field's requirement	System notices that "Can not create lesson. The lesson name must be 10-50 characters."
Relationships	Abstract use case: Manage lesson		
Business Rules	• User already logged in the system as "Staff" role. • The name of lesson must be in range of 10-50 characters. • The lesson created must be in a specific course. • Staff can only create lesson in their own's course. • Course Contents in Course page will be updated with new lesson that just be created.		

2.3.2.8. <Staff> Update Lesson (UC_S08)



USE CASE – UC_S08			
Use Case No.	UC_S08	**Use Case Version**	1.0
Use Case Name	Update Lesson		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to update lesson of existed topic.		
Goal	Staff successfully updates new lesson of existed topic.		
Triggers	Actor sends update lesson command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: New lesson is created into database. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks selected Course in Manage Course page.	System displays the Course Information page.
	2	Actor moves to ‘Course Contents’ field.	
	3	Actor selects Lesson to edit, then press ‘Edit’ button.	System displays the lesson.
	4	Actor edits lesson with specific fields (like Create Lesson use case).	
	5	Actor clicks ‘Save’ button. [Exception no.1]	System displays popup “Update Lesson successfully”.
Exceptions	No	Cause	System Response
	1	Actor does not input required field or input wrong field’s requirement	System notices that “Can not update lesson. The lesson name must be 10-50 characters.”
Relationships	Abstract use case: Manage lesson		
Business Rules	• User already logged in the system as “Staff” role. • The name of lesson must be in range of 10-50 characters. • The lesson updated must be in a specific course. • Staff can only update lesson in their own’s course. • Course Contents in Course page will be updated when finishing updated lesson.		

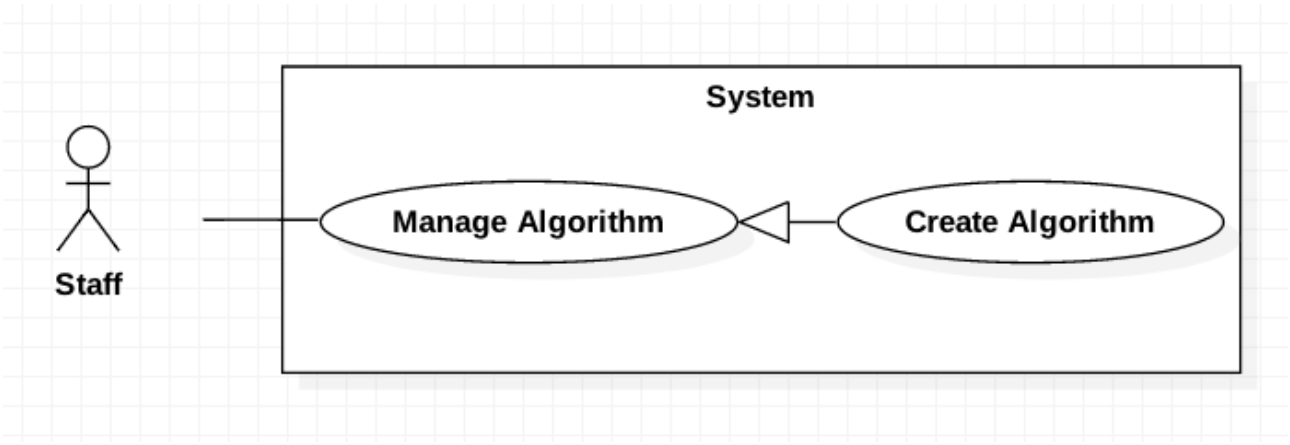
2.3.2.9. <Staff> Remove Lesson (UC_S09)



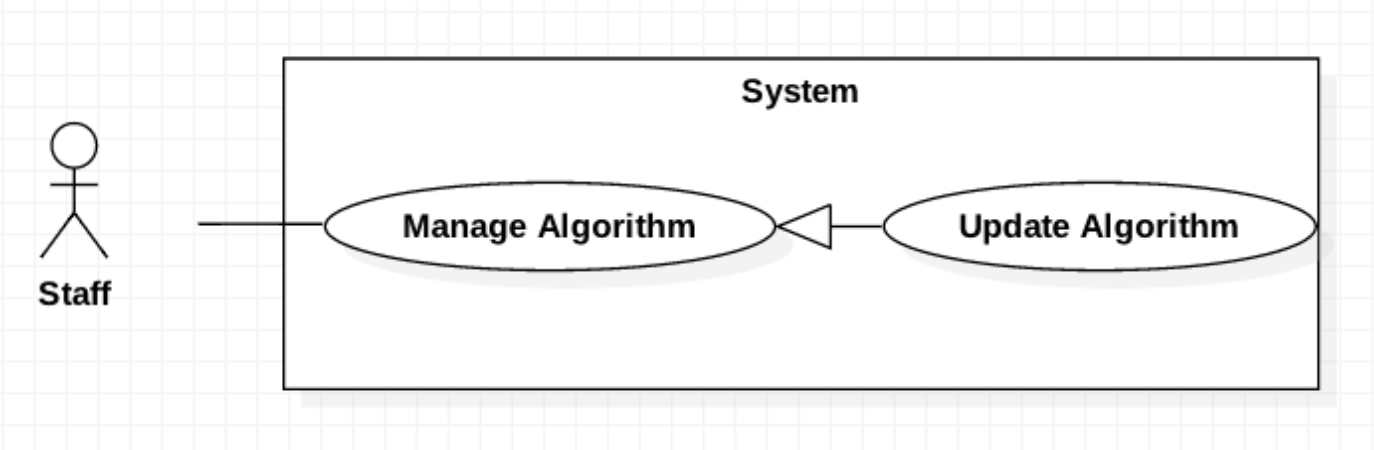
USE CASE – UC_S09			
Use Case No.	UC_S09	**Use Case Version**	1.0
Use Case Name	Remove Lesson		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to remove a lesson.		
Goal	Staff successfully removes existed lesson.		
Triggers	Actor sends remove lesson command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: The lesson will be removed from database. • Fail: System displays error messages.		

USE CASE – UC_S09			
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks selected Course in Manage Course page.	System displays the Course Information page.
	2	Actor moves to ‘Course Contents’ field.	
	3	Actor clicks ‘Delete’ button of selected Lesson.	System displays a popup to confirm.
	4	Actor clicks ‘Yes’ button.	System displays popup “Removed Lesson successfully”.
Exceptions	N/A		
Relationships	Abstract use case: Manage lesson		
Business Rules	• User already logged in the system as “Staff” role. • The lesson will be completely removed from database. • User cannot see or access to the lesson that has been removed. • Only staff who created the course can remove lesson in that course.		

2.3.2.10. <Staff> Create Algorithm (UC_S10)



USE CASE – UC_S10			
Use Case No.	UC_S10	**Use Case Version**	1.0
Use Case Name	Create Algorithm		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		

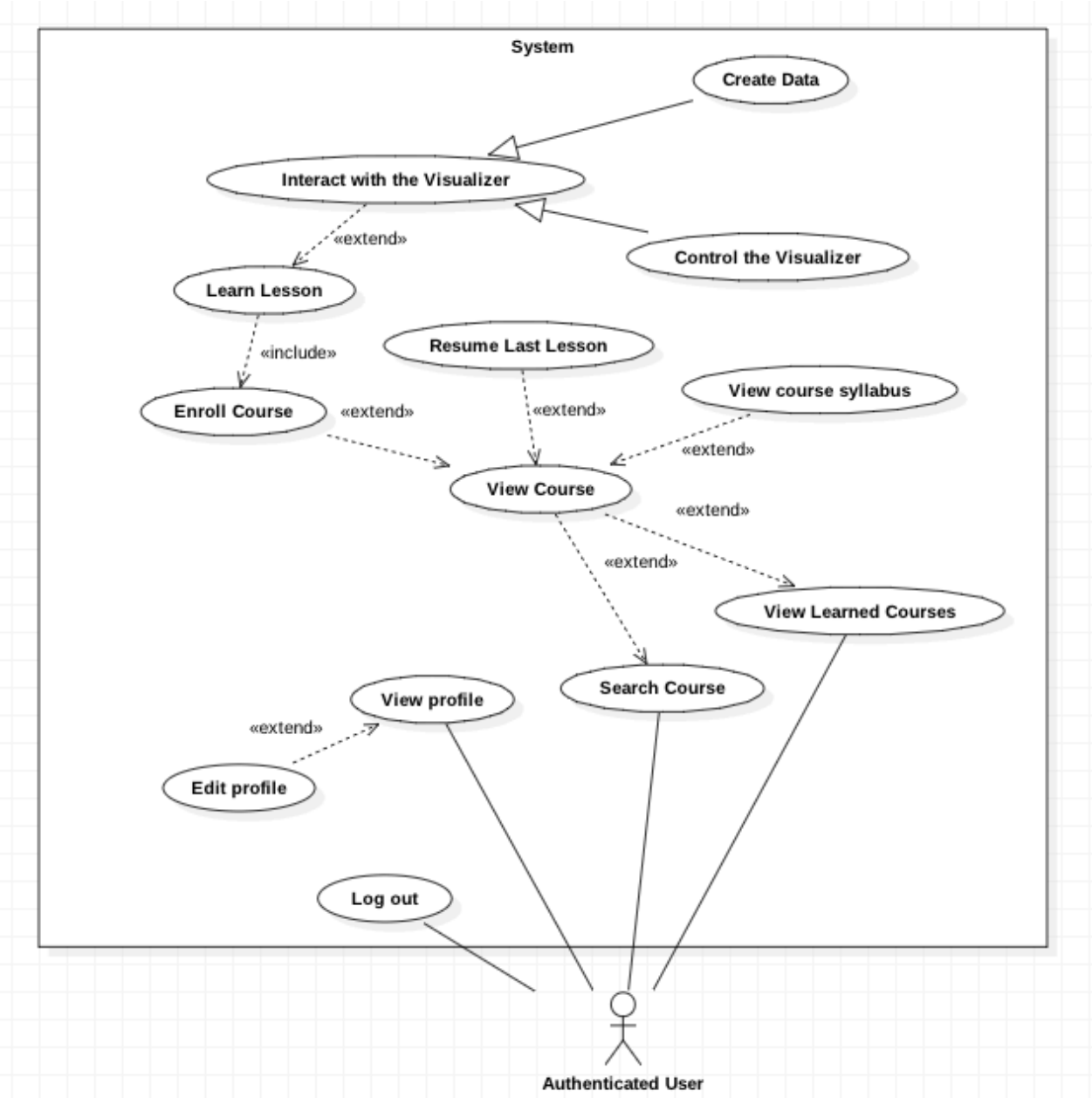


USE CASE – UC_S10			
Summary	This use case allows staff to create specify algorithm that has been coded by Dev.		
Goal	Staff successfully creates new algorithm.		
Triggers	Actor sends create algorithm command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: New algorithm is created into database. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor goes to Manage Algorithms page.	System displays the Manage Algorithms page.
	2	Actor clicks ‘Create new Algorithm’.	System display a new line with 2 fields: Name and URL.
	3	Actor enters Name and URL of the algorithm (those are given by Dev).	
	4	Actor clicks ‘Save’ button. [Exception no.1]	System displays popup “Create algorithm successfully”.
Exceptions	No	Cause	System Response
	1	Actor does not input required field or input wrong field’s requirement	System notices that “Can not create algorithm. The name and the URL must be 10-50 characters.”
Relationships	Abstract use case: Manage algorithm		

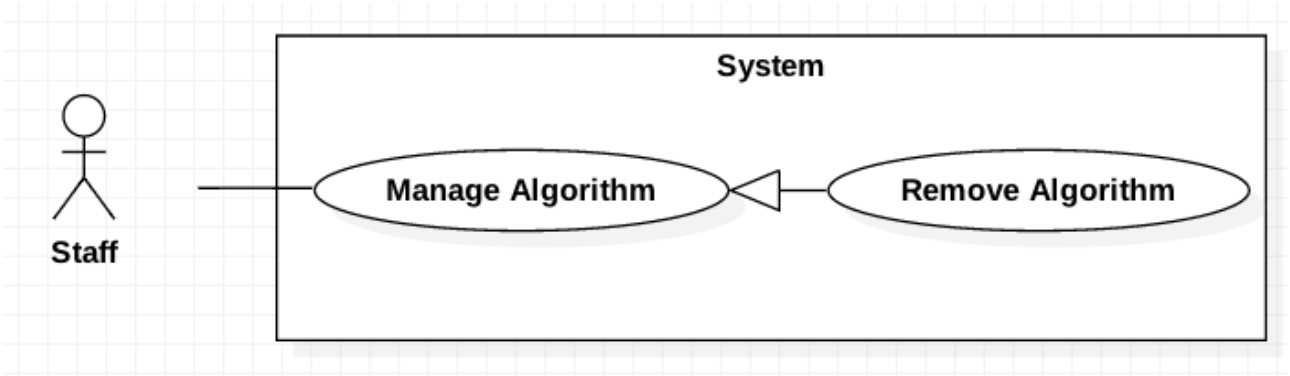
USE CASE – UC_S10	
Business Rules	• User already logged in the system as “Staff” role. • The algorithm’s name and URL must be in range of 10-50 characters. • New algorithm will be added into database. • New algorithm will be existed when staff chooses algorithm for lesson.

2.3.2.11. <Staff> Update Algorithm (UC_S11)

USE CASE – UC_S11			
Use Case No.	UC_S11	**Use Case Version**	1.0
Use Case Name	Update Algorithm		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to update specify algorithm that has been coded by Dev.		
Goal	Staff successfully update the existed algorithm.		
Triggers	Actor sends update algorithm command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: Update algorithm in database. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor goes to Manage Algorithms page.	System displays the Manage Algorithms page.
	2	Actor clicks ‘Edit’ button of the selected Algorithm.	System display information of the selected Algorithm: name and URL.
	3	Actor enters Name and URL of the algorithm (those are given by Dev).	
	4	Actor clicks ‘Save’ button. [Exception no.1]	System displays popup “Update algorithm successfully”.
Exceptions	No	Cause	System Response
	1	Actor does not input required field or input wrong field’s requirement	System notices that “Can not update algorithm. The name and the URL must be 10-50 characters.”
Relationships	Abstract use case: Manage algorithm		
Business Rules	• User already logged in the system as “Staff” role. • The algorithm’s name and URL must be in range of 10-50 characters. • Update edited algorithm to the database. • Updated algorithm will be existed when staff chooses algorithm for lesson.		



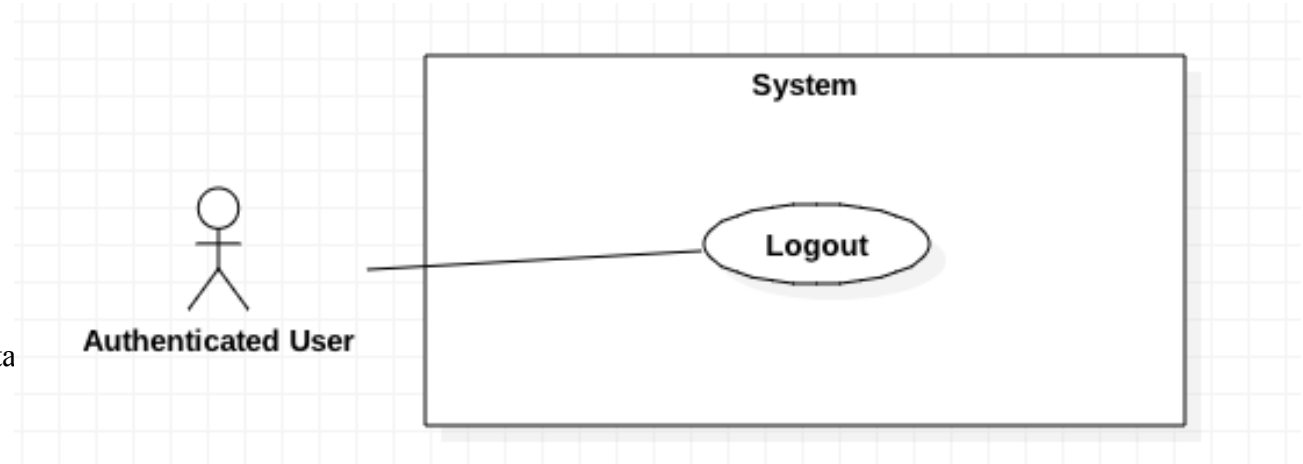
2.3.2.12. <Staff> Remove Algorithm (UC_S12)



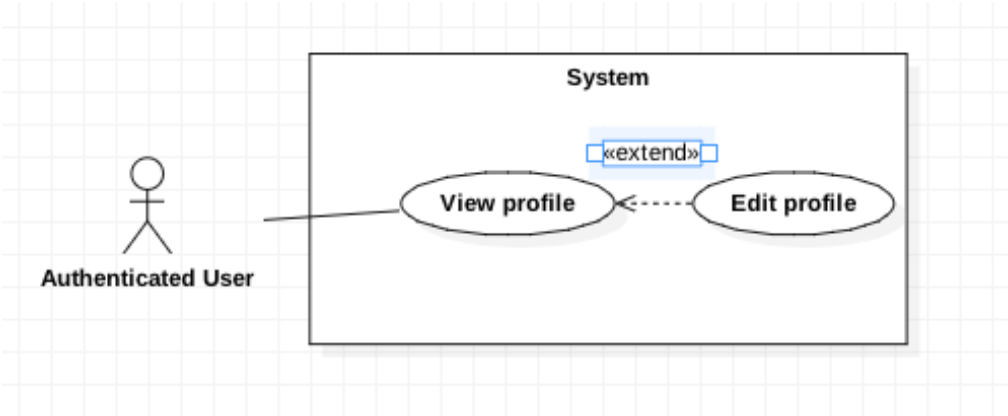
USE CASE – UC_S12			
Use Case No.	UC_S12	**Use Case Version**	1.0
Use Case Name	Remove Algorithm		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Staff		
Summary	This use case allows staff to remove algorithm.		
Goal	Staff successfully remove the existed algorithm.		
Triggers	Actor sends remove algorithm command		
Pre Conditions	Actor logged in as Staff role.		
Post Conditions	• Success: Remove algorithm in database. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor goes to Manage Algorithms page.	System displays the Manage Algorithms page.
	2	Actor clicks ‘Remove’ button of the selected Algorithm.	System displays a popup to confirm.
	3	Actor clicks ‘Yes’ button. [Exception no.1]	System displays popup “Remove algorithm successfully”.
Exceptions	No	Cause	System Response
	1	The removed Algorithm existed in other lesson.	System notices that “Can not remove this algorithm.”
Relationships	Abstract use case: Manage algorithm		
Business Rules	• User already logged in the system as “Staff” role. • Remove selected algorithm if succeeds. • Removed algorithm won’t be existed when staff chooses algorithm for lesson.		

2.3.3. <Authenticated User> Overview Use Case

2.3.3.1. <Authenticated User> Log out (UC_AU01)

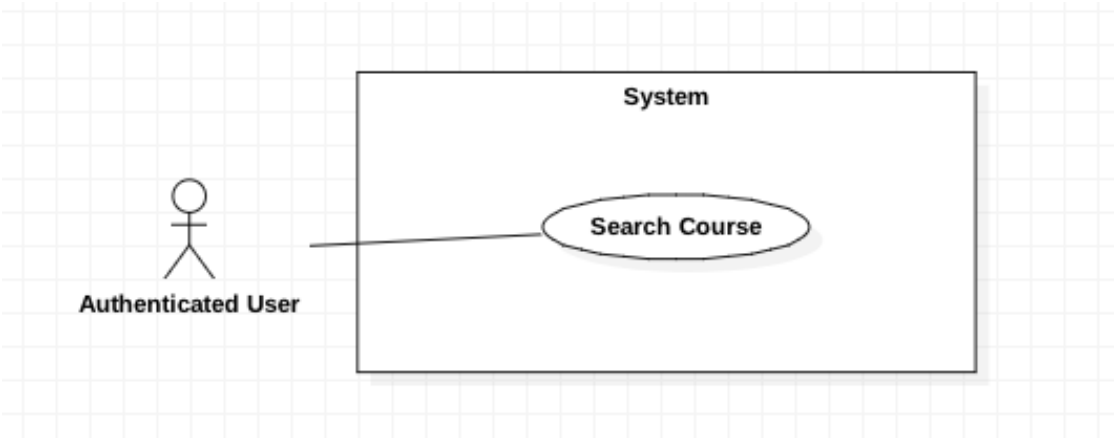


Data



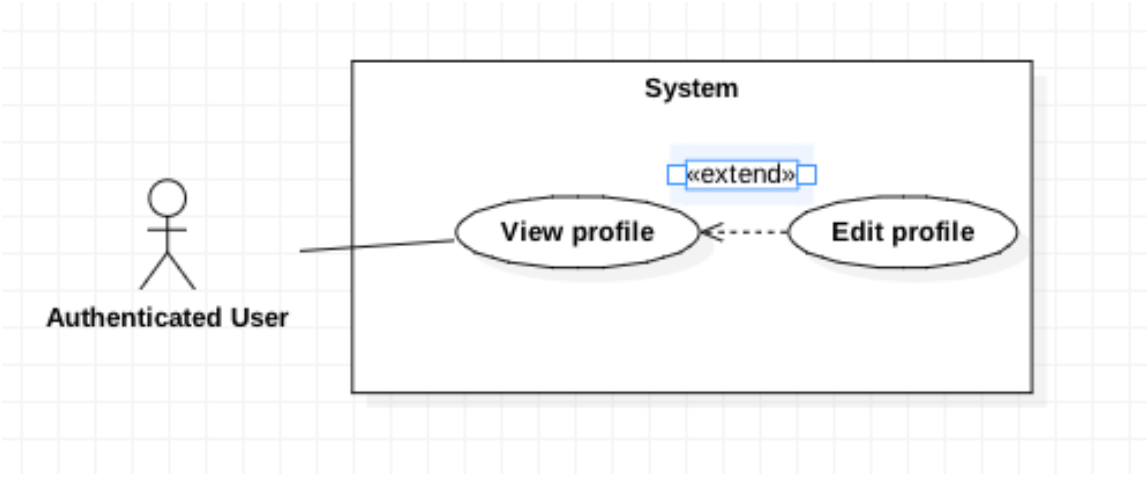
USE CASE – UC_AU01			
Use Case No.	UC_AU01	**Use Case Version**	1.0
Use Case Name	Log out		
Author	KhangHLD		
Date	14/02/2018	**Priority**	Normal
Actor	Authenticated User		
Summary	This use case allows Authenticated User to log out the system.		
Goal	To log out the system.		
Triggers	Actor sends logout command.		
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: User session is removed, and system displays home page. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks “Log out” button	User session is removed, and system displays home page.
Exceptions	N/A		
Relationships	N/A		
Business Rules	• Actor has already accessed to the system. • After signing out, Authenticated user will become “Unauthenticated user”.		

2.3.3.2. <Authenticated User> View profile (UC_AU02)



USE CASE – UC_AU02			
Use Case No.	UC_AU02	**Use Case Version**	1.0
Use Case Name	View profile		
Author	KhangHLD		
Date	14/02/2018	**Priority**	Normal
Actor	Authenticated User		
Summary	This use case allows Authenticated User to view his profile.		
Goal	To view profile.		
Triggers	Actor sends view profile command.		
Pre Conditions	Authenticated User logged in the system.		

USE CASE – UC_AU02			
Post Conditions	• Success: Actor go to his profile page. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks “Profile” button	System redirects to actor’s profile page.
Exceptions	N/A		
Relationships	Extended by: Edit profile		
Business Rules	• Actor has already accessed to the system. • Authenticated user can only view his profile page.		



2.3.3.3. <Authenticated User> Edit profile (UC_AU03)

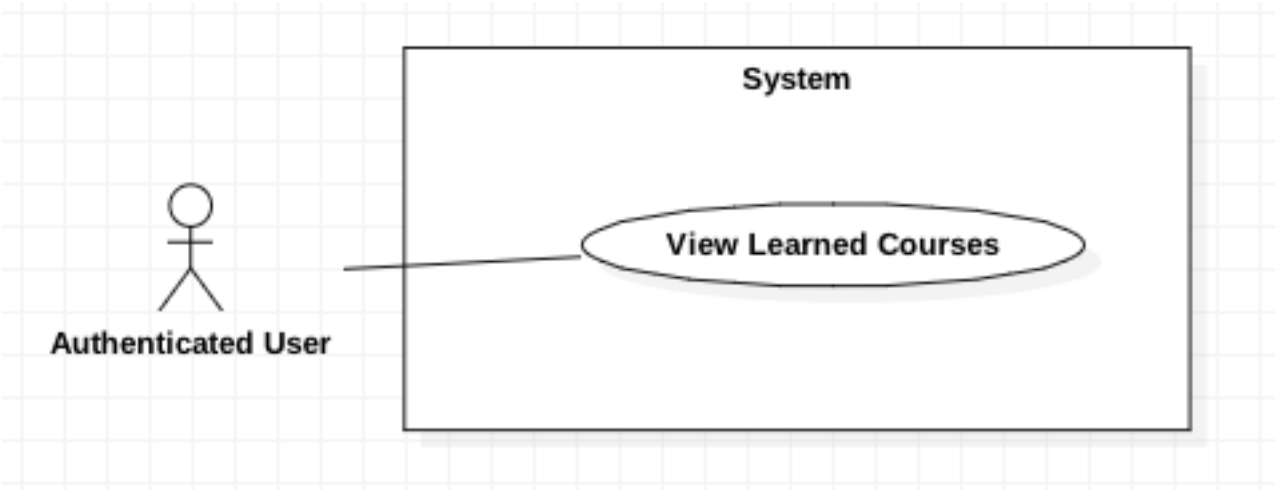
USE CASE – UC_AU03			
Use Case No.	UC_AU03	**Use Case Version**	1.0
Use Case Name	Edit profile		
Author	KhangHLD		

USE CASE – UC_AU03			
Date	14/02/2018	**Priority**	Normal
Actor	Authenticated User		
Summary	This use case allows Authenticated User to edit his profile.		
Goal	To edit profile.		
Triggers	Actor sends edit profile command.		
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: Authenticated User's information has been updated in the database. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks "Profile" button	System redirects to actor's profile page.
	2	Actor clicks 'Edit' button.	System displays fields that can be edited: Display Name, Password, Confirm Password.
	3	Actor edits specific fields then clicks 'Save' button. [Exception no.1, 2]	System displays popup "Updated profile successfully".
Exceptions	No	Cause	System Response
	1	Display Name has less than 5 characters or has more than 50 characters.	System notices that "Display Name must be between 5-50 characters."
	2	Password does not match Confirm Password	System notices that "Password must match Confirm Password"
Relationships	N/A		
Business Rules	• Actor has already accessed to the system. • Authenticated user can only edit his profile information. • Display Name must be between 5-50 characters. • Password must match Confirm Password. • Actor's updated information must be display in the profile page after edited successfully.		

2.3.3.4. <Authenticated User> Search Course (UC_AU04)

USE CASE – UC_AU04			
Use Case No.	UC_AU04	**Use Case Version**	1.0
Use Case Name	Search Course		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Authenticated User		
Summary	This use case allows Authenticated User to search specify course.		
Goal	To search course		
Triggers	Actor sends search course command.		
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: System displays all courses that related to the key word. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor enters key words in the Search box.	
	2	Actor clicks ‘Search’ button.	System displays fields that can be edited: Display Name, Password, Confirm Password.
	3	Actor edits specific fields then clicks ‘Save’ button. [Exception no.1]	System displays all courses that related to the key word.
Exceptions	No	Cause	System Response
	1	No courses relates to the key words.	System notices that “No courses relates to the key words”
Relationships	Extended by: View Course		
Business Rules	• Actor has already accessed to the system. • Authenticated user can only edit his profile information. • Display Name must be between 5-50 characters. • Password must match Confirm Password. • Actor’s updated information must be display in the profile page after edited successfully.		

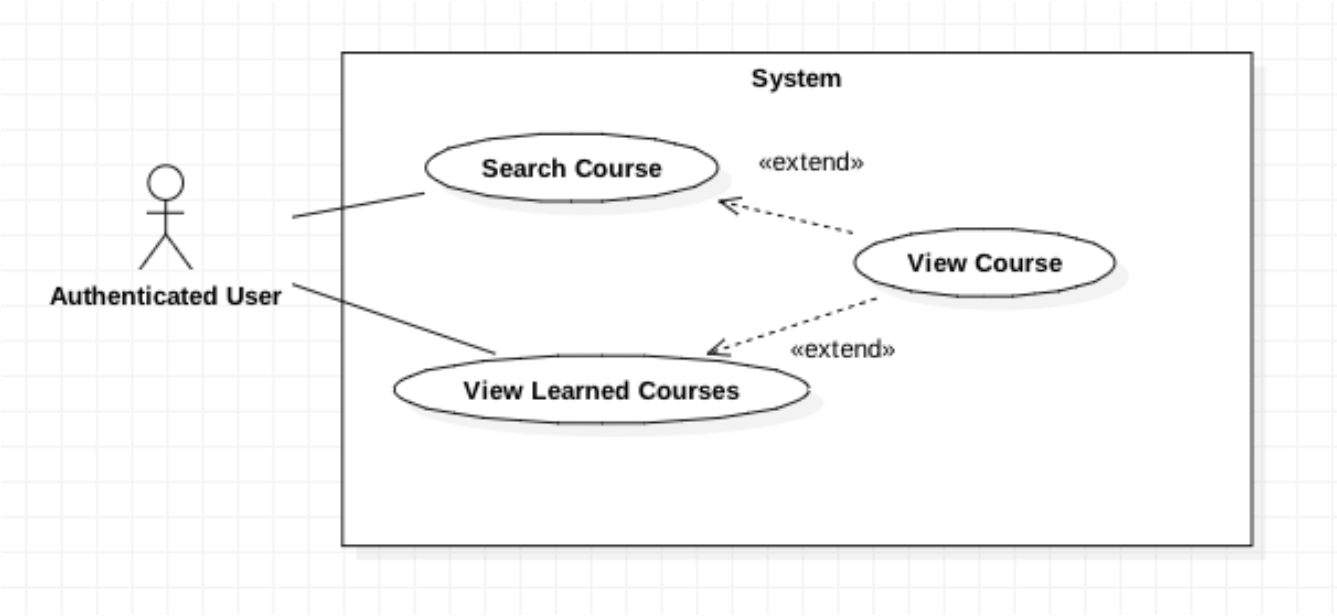
2.3.3.5. <Authenticated User> View Learned Courses (UC_AU05)



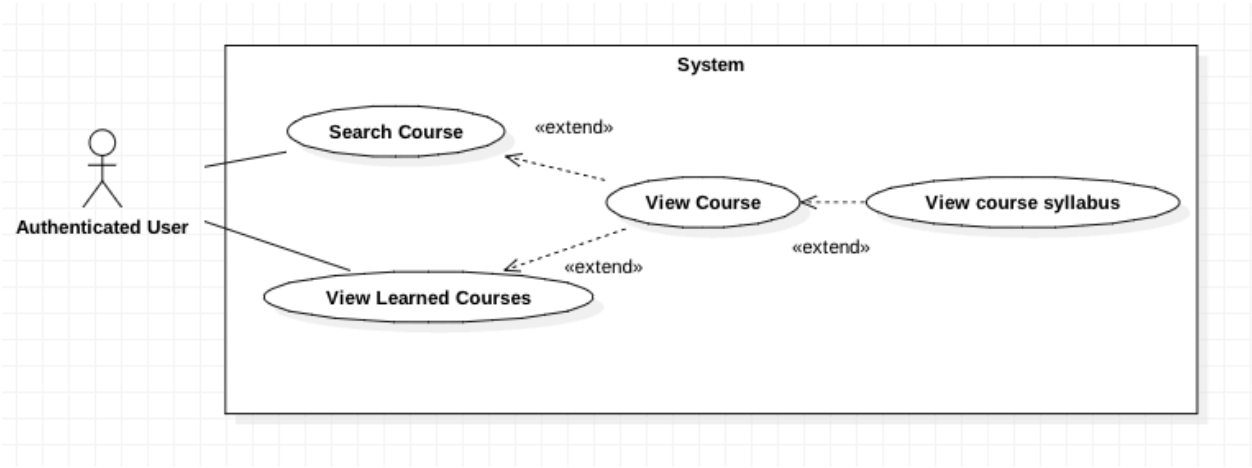
USE CASE – UC_AU05			
Use Case No.	UC_AU05	**Use Case Version**	1.0
Use Case Name	View Learned Courses		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Authenticated User		
Summary	This use case allows Authenticated User to view his learned courses.		
Goal	To view learned courses.		
Triggers	Actor sends view learned courses command.		
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: System displays all actor's learned courses. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor press 'View Learned Courses' button.	System displays all actor's learned courses.
Exceptions	N/A		
Relationships	Extended by: View Course		
Business Rules	• Actor has already accessed to the system. • System displays only the actor's learned courses.		

2.3.3.6. <Authenticated User> View Course (UC_AU06)

USE CASE – UC_AU06			
Use Case No.	UC_AU06	**Use Case Version**	1.0
Use Case Name	View Course		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Authenticated User		
Summary	This use case allows Authenticated User to view selected course.		
Goal	To view course		
Triggers	Actor sends view course command.		
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: System displays the information of the selected course. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks the course that existed in home page, or search page, or learned courses page.	• System displays the information of the selected course. • The course page displays 'Enroll' button if actor has not learned this course before. • The course page display 'Resume' button if actor has enrolled into this course before.
Exceptions	N/A		
Relationships	Extended by: View course syllabus, Resume Last Lesson, Enroll Course		
Business Rules	• Actor has already accessed to the system.		



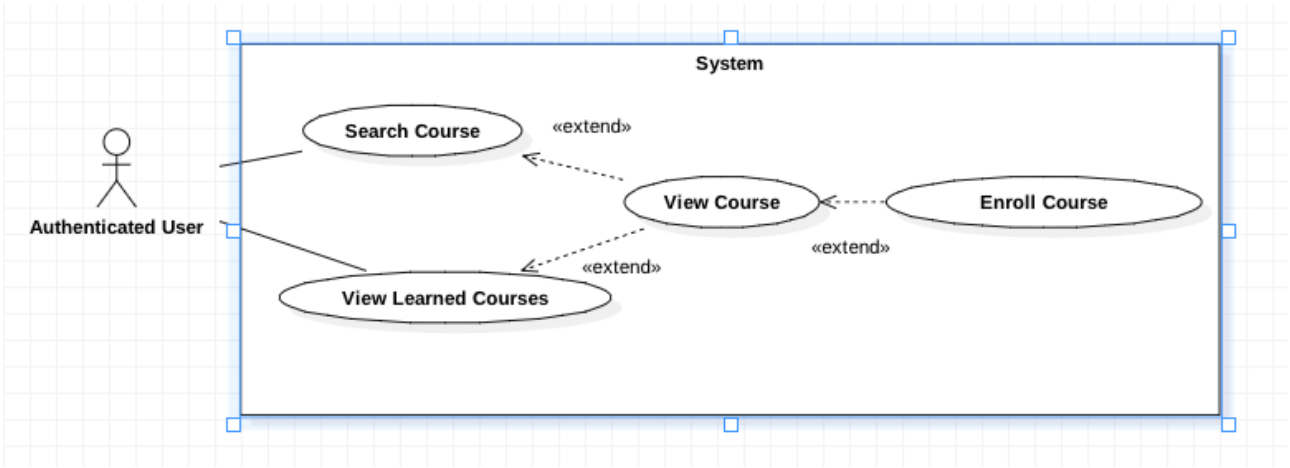
2.3.3.7. <Authenticated User> View course syllabus (UC_AU07)



USE CASE – UC_AU07			
Use Case No.	UC_AU07	**Use Case Version**	1.0
Use Case Name	View course syllabus		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Authenticated User		
Summary	This use case allows Authenticated User to view course syllabus of selected course.		
Goal	To view course syllabus of selected course.		

USE CASE – UC_AU07			
Triggers	Actor sends view syllabus course command.		
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: System displays the syllabus of the selected course. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks the course that existed in home page, or search page, or learned courses page. Then actor clicks the 'Syllabus' tab.	System displays the syllabus of selected course.
Exceptions	N/A		
Relationships	N/A		
Business Rules	• Actor has already accessed to the system.		

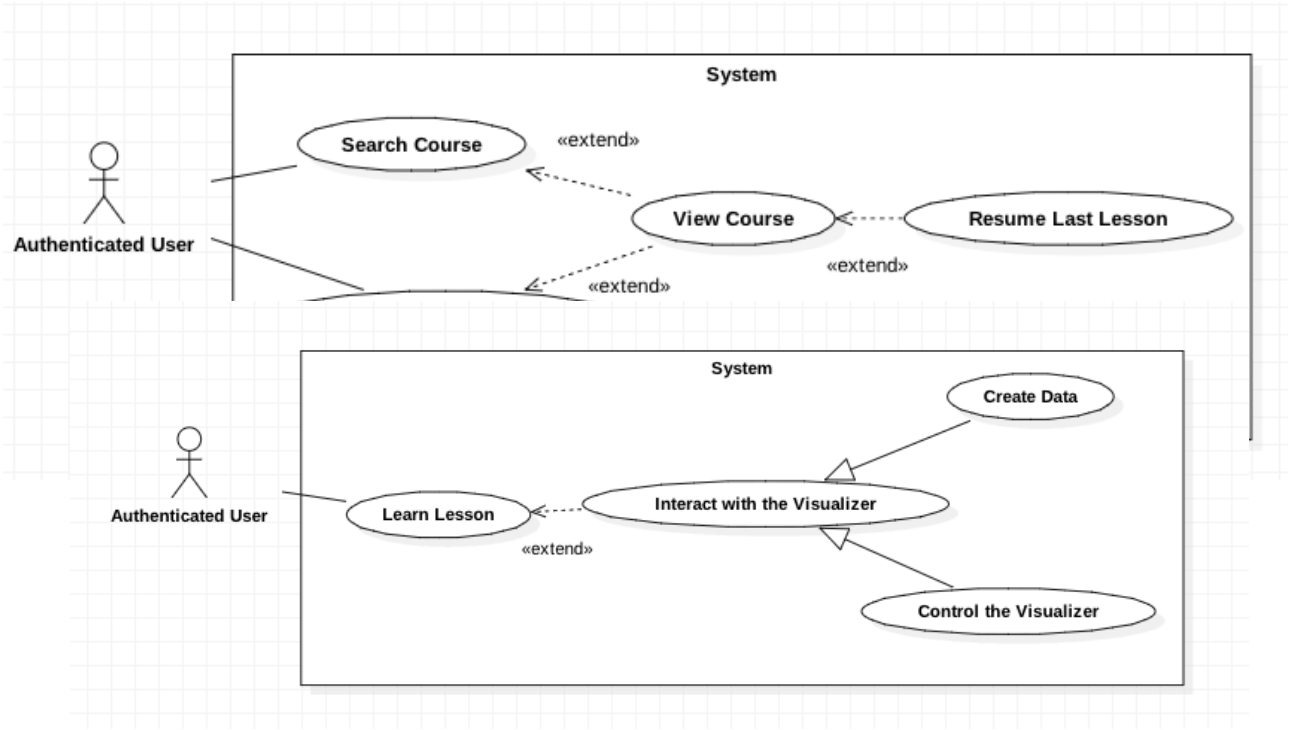
2.3.3.8. <Authenticated User> Enroll Course (UC_AU08)



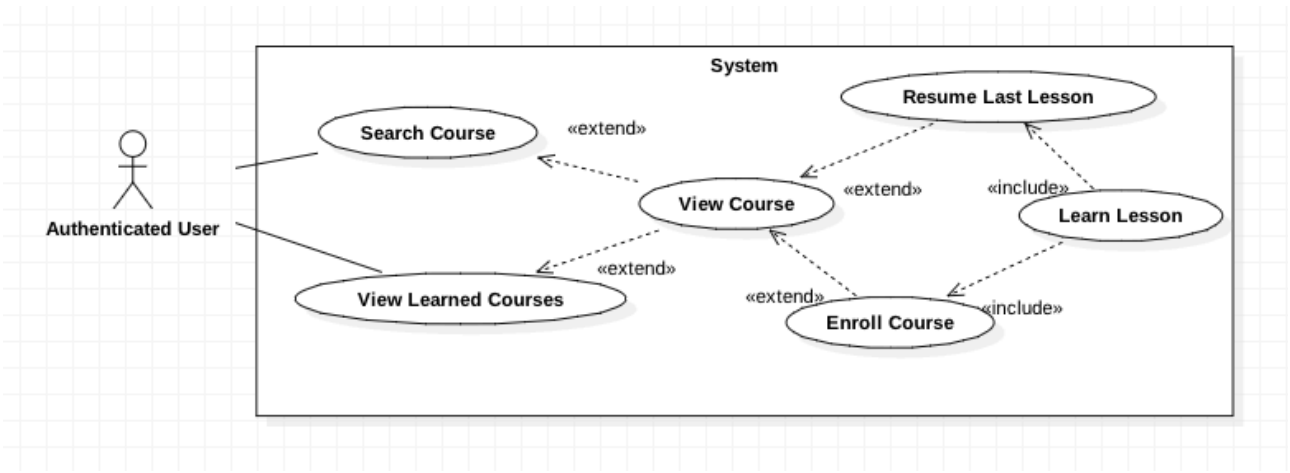
USE CASE – UC_AU08			
Use Case No.	UC_AU08	**Use Case Version**	1.0
Use Case Name	Enroll Course		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Authenticated User		
Summary	This use case allows Authenticated User to enroll in selected course.		
Goal	To enroll into selected course		
Triggers	Actor sends enroll course command.		

USE CASE – UC_AU08			
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: System displays the first lesson page of the enrolled course and the actor's learning process for this course is updated. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor go to the course page.	System displays the course page.
	2	Actor clicks 'Enroll' button.	System displays the first lesson page of the enrolled course and the actor's learning process for this course is updated.
Exceptions	N/A		
Relationships	Included by: Learn Lesson		
Business Rules	• Actor has already accessed to the system. • Actor has not learned the selected course before. • When ever actor enrolls in the course, his learning process for this course is updated.		

2.3.3.9. <Authenticated User> Resume Last Lesson (UC_AU09)



USE CASE – UC_AU09			
Use Case No.	UC_AU09	**Use Case Version**	1.0
Use Case Name	Resume Last Lesson		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Authenticated User		
Summary	This use case allows Authenticated User to resume last lesson of the selected course.		
Goal	To resume last lesson of the selected course.		
Triggers	Actor sends resume last lesson command.		
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: System displays the last lesson page of the selected course. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor go to the course page.	System displays the course page.
	2	Actor clicks ‘Resume’ button.	System displays the last lesson page of the selected course.
Exceptions	N/A		
Relationships	Included by: Learn Lesson		
Business Rules	• Actor has already accessed to the system. • Actor has enrolled in the selected course before.		



2.3.3.10. <Authenticated User> Learn Lesson (UC_AU10)

USE CASE – UC_AU10			
Use Case No.	UC_AU10	**Use Case Version**	1.0
Use Case Name	Learn Lesson		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Authenticated User		
Summary	This use case allows Authenticated User to learn lesson of the selected course.		
Goal	To learn lesson of the selected course.		
Triggers	Actor sends learn lesson command.		
Pre Conditions	Authenticated User logged in the system.		
Post Conditions	• Success: Actor is redirected to learning page of correct lesson type. • Fail: System displays error messages.		
Main Success Scenario	Step	Actor Action	System Response
	1	Actor clicks a specific lesson in a course.	System navigates to: • Learning text page if that lesson is of text type. • Learning animation page if that lesson is of animation type.
	2	Actor clicks ‘Resume’ button.	System displays the last lesson page of the selected course.
Alternative Scenario	Step	Actor Action	System Response
Scenario 1	1	Actor clicks ‘Enroll’ button in Course page.	System navigates to learning page with the first lesson of the course.
Scenario 2	1	Actor clicks ‘Resume’ button in Course page.	System navigates to learning page with the last lesson that actor’s learned of the course.
Exceptions	N/A		
Relationships	Extended by: Interact with the Visualizer		

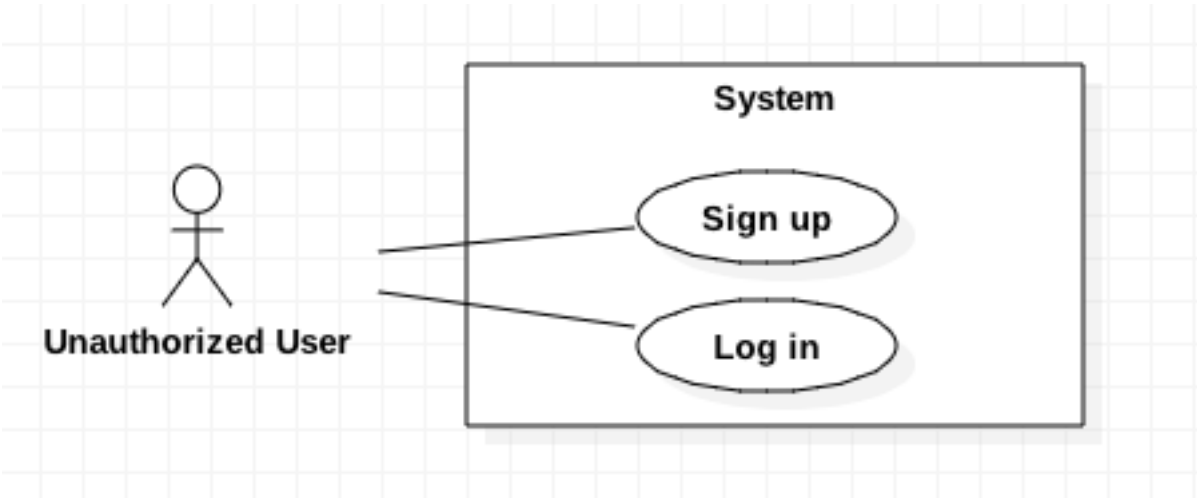
USE CASE – UC_AU10	
Business Rules	• Actor has already accessed to the system. • Actor will be redirected to specific learning page depends on lesson type: Text or Animation. • If the actor clicks the 'Next lesson' button, the learning activity of actor for this course will be updated and the page will redirect to the next lesson in this course. If not, this lesson will be counted as "not finish yet".

2.3.3.11. <Authenticated User> Interact with the Visualizer (UC_AU11)

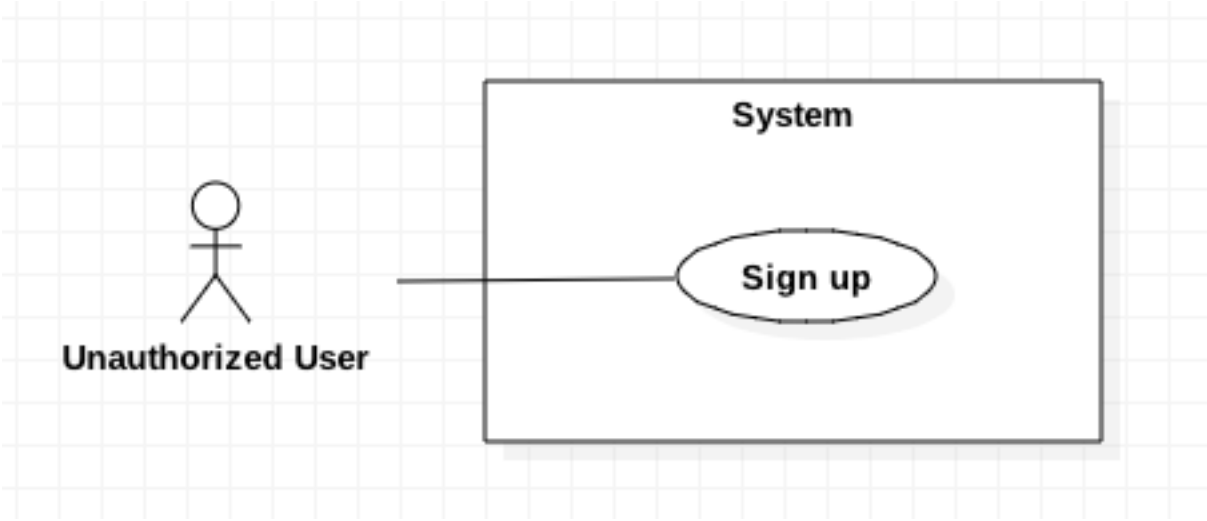
USE CASE – UC_AU11			
Use Case No.	UC_AU11	**Use Case Version**	1.0
Use Case Name	Interact with the Visualizer		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Authenticated User		
Summary	This use case allows Actor to interact with the Visualizer.		
Goal	To interact with the Visualizer.		
Triggers	Actor sends interact command.		
Pre Conditions	Actor is in Animation Lesson page.		
Post Conditions	• Success: Actor can interact with the Visualizer. • Fail: System displays error messages.		
Alternative Scenario	Step	Actor Action	System Response
Scenario 1	1	Actor clicks ‘Create random data’ button in Animation page.	System will create randomly the data for the visualizer.
Scenario 2	1	Actor clicks 'Create data' button.	System displays a text box for collecting data.
	2	Actor enters data in text box.	
	3	Actor clicks ‘Go’ button.	The visualizer displays exactly the data that actor entered for current algorithm.
Scenario 3	1	Actor clicks ‘Run algorithms’ button.	The visualizer displays the animation that show how the current algorithm runs step by step.

USE CASE – UC_AU11			
Scenario 4	1	Actor clicks ‘Control’ button in the visualizer (includes 6 buttons: go to beginning, step backward, play/pause, step forward, go to end).	The visualizer will displays exact the step of current algorithm that actor wants: • Go to beginning: go to the first step of the algorithm. • Step backward: go to the previous step since current step of the algorithm. • Play: continue with current step of the algorithm (default: current step is 1). • Pause: pause at current step of the algorithm. • Step forward: go the the next step since current step of the algorithm. • Go to end: go to the last step of the algorithm.
Exceptions	N/A		
Relationships	Extended by: Interact with the Visualizer		
Business Rules	• Actor has already accessed to the system. • Actor is in the Animation Lesson page.		

2.3.4. <Unauthenticated User> Overview Use Case



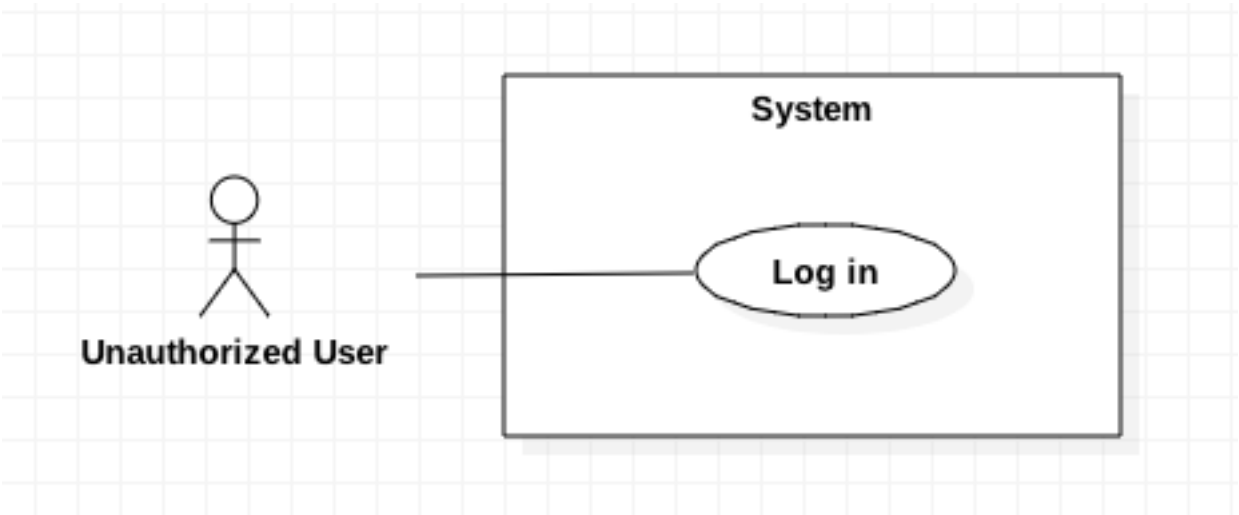
2.3.4.1. <Unauthenticated User> Sign up (UC_UU01)



USE CASE – UC_UU01			
Use Case No.	UC_UU01	**Use Case Version**	1.0
Use Case Name	Sign up		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Unauthenticated User		
Summary	This use case allows Actor to sign up.		
Goal	To sign up.		
Triggers	Actor sends sign up command.		
Pre Conditions	Actor does not log in yet. Actor press ‘Sign up’ button to open Sign up modal.		
Post Conditions	• Success: Actor can sign up an account. • Fail: System displays error messages.		
Alternative Scenario	Step	Actor Action	System Response
Scenario 1	1	Actor clicks ‘Continue with Google’ button.	System displays a login popup via Google account.
	2	Actor logs in gmail account. [Exception no. 1, 2]	System creates automatically an account for that authorized gmail, then makes that account logged in.

USE CASE – UC_UU01			
Scenario 2	1	Actor fills in Email, Password and Confirm Password field.	
	2	Actor clicks ‘Sign up’ button. [Exception no 3, 4]	System creates automatically an account for that email, then makes that account logged in.
Exceptions	No	Cause	System Response
	1	Actor can not authorize the gmail account.	System notices that cannot authorize that gmail account.
	2	The gmail account is existed in the system.	System automatic authenticates that account. Actor logged in successful.
	3	Password does not match Confirm Password	System notices that ‘Password does not match Confirm Password’.
	4	The email is existed in the system.	System notices that ‘That email is existed in the system. Please log in to continue.’
Relationships	N/A		
Business Rules	• The account that has been signed up with gmail account does not have password. That account can be logged in via gmail, or the owner can update password in Edit Profile use case.		

2.3.4.2. <Unauthenticated User> Log in (UC_UU02)



USE CASE – UC_UU01			
Use Case No.	UC_UU02	**Use Case Version**	1.0
Use Case Name	Log in		
Author	KhangHLD		
Date	14/02/2018	**Priority**	High
Actor	Unauthenticated User		
Summary	This use case allows Actor to log in.		
Goal	To log in.		
Triggers	Actor sends log in command.		
Pre Conditions	Actor does not log in yet. Actor press ‘Log in’ button to open Log in modal.		
Post Conditions	• Success: Actor can log in to his account. • Fail: System displays error messages.		
Alternative Scenario	Step	Actor Action	System Response
Scenario 1	1	Actor clicks ‘Continue with Google’ button.	System displays a login popup via Google account.
	2	Actor logs in gmail account. [Exception no. 1, 2]	System redirects to Home page.
Scenario 2	1	Actor fills in Email and Password.	
	2	Actor clicks ‘Log in’ button. [Exception no 3, 4]	System redirects to Home page.
Exceptions	No	Cause	System Response
	1	Actor can not authorize the gmail account.	System notices that cannot authorize that gmail account.
	2	The gmail account is not existed in the system.	System creates automatically an account for that authorized gmail, then makes that account logged in.
	3	The email does not existed in the system.	System notices that ‘This email has not been registered yet.’.
	4	The password does not match with the account.	System notices that ‘Wrong password. Please try again.’
Relationships	N/A		

USE CASE – UC_UU01	
Business Rules	• After signing in to system, actor will be redirected to specific view based on his role on the system: Administrator, Staff, or Authenticated User.

3. Software Requirement Specification

3.1. Usability

- All the texts, labels, alerts and messages will be written in English.
- GUI for the web application is designed based on material design.
- The system usability is easy to use that users generally don't need to look at the document to use including admin and manager.
- Icons that indicate the actions should be easy to understand and users will not meet any troubles to recognize the feature of the page.

3.2. Reliability

- The data should be backed up every day.

3.3.Availability

- The web application must be available 24/7.

3.4. Security

- Each role of user has a specific permission to interact with the system.
- System always checks for authorization and authentication before doing anything.
- Only Administrator can grant permission to other roles.

3.5. Maintainability

- The system is divided into separated modules.
- The code is easy to maintain and upgrade.

3.6. Portability

- The software itself is a web application, therefore it can be used on any platform that has a web browser and can connect to the internet.

3.7. Performance

- System detects and returns results in 4 seconds or less under 4Mbps bandwidth.

4. Conceptual Diagram

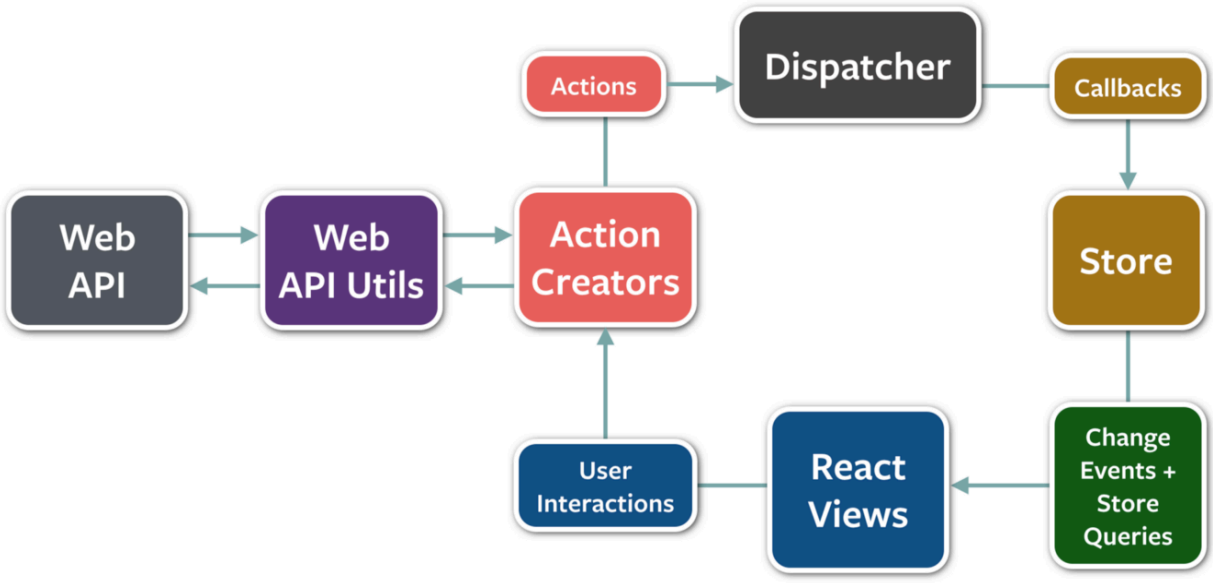
D. Software Design Description

1. Design Overview

- This document describes the technical and user interface design of DSAV Web Application. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.
- The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
- The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.
- The database design describes the relationships between entities and details of each entity.

2. System Architectural Design

2.1. System Architecture Overview



System architecture overview: Flux

Our application is developed mainly on Flux architecture.

Flux is an architecture that Facebook uses internally when working with React. It is *not* a framework or a library. It is simply a new kind of architecture that complements React and the concept of Unidirectional Data Flow.

Flux is probably better explained by explaining its individual components:

- Actions - Helper methods that facilitate passing data to the Dispatcher
- Dispatcher - Receives actions and broadcasts payloads to registered callbacks
- Stores - Containers for application state & logic that have callbacks registered to the dispatcher
- Controller Views - React Components that grab the state from Stores and pass it down via props to child components.
- Web API - Get data from database in server, then send to client via API call.

2.1.1.Dispatcher

The Dispatcher is basically the manager of this entire process. It is the central hub for your application. The dispatcher receives actions and dispatches the actions and data to registered callbacks.

The dispatcher broadcasts the payload to ALL of its registered callbacks, and includes functionality that allows you to invoke the callbacks in a specific order, even waiting for updates before proceeding. There is only ever one dispatcher, and it acts as the central hub within your application.

2.1.2.Stores

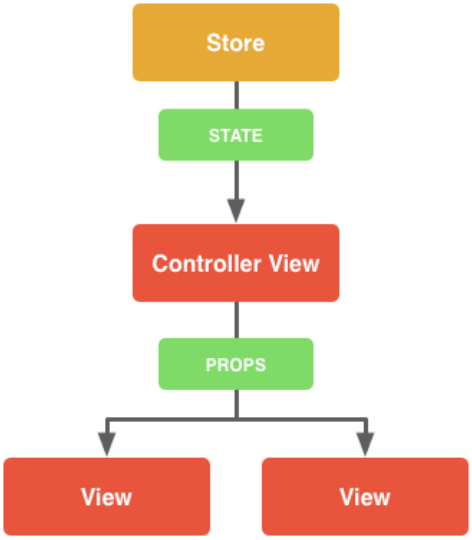
In Flux, Stores manage application state for a particular domain within application. From a high level, this basically means that per app section, stores manage the data, data retrieval methods and dispatcher callbacks.

2.1.3.Action Creators & Actions

Action Creators are collections of methods that are called within views (or anywhere else for that matter) to send actions to the Dispatcher. Actions are the actual payloads that are delivered via the dispatcher.

2.1.4.Controller Views

Controller views are really just React components that listen to change events and retrieve Application state from Stores. They then pass that data down to their child components via props.



2.1.5.Web API

Web API is the component that be called by Actions. It goes to the server to get the data then send the data to Dispatcher.

We choose Flux with React because it's a new technology. And it can handle lots of requests, because all requests are called in clients. Then the requests call web API to get data. Server will be free from handling requests like a controller.

3. Component Diagram

4. Detail Description

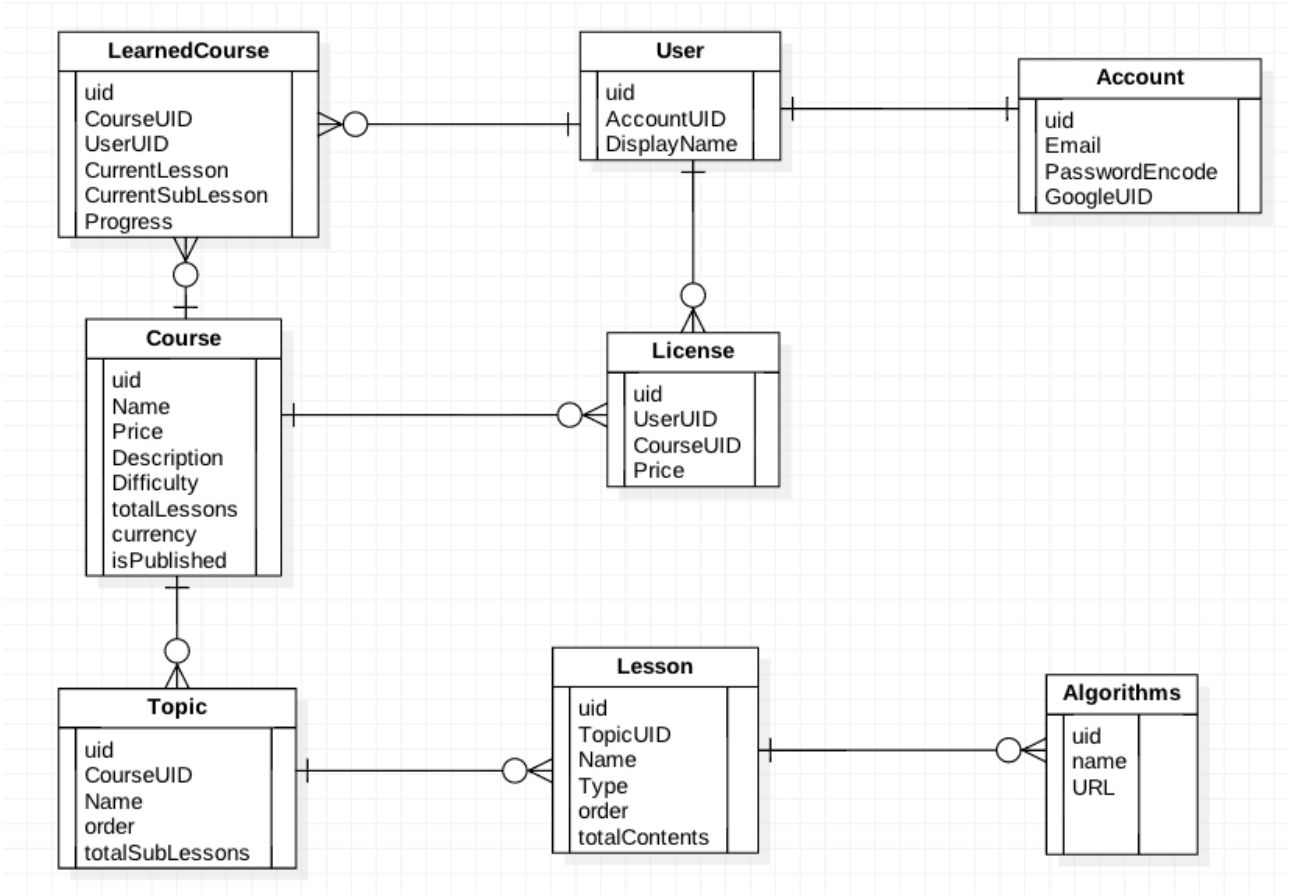
4.1. Class Diagram

4.2. Class Diagram Explanation

4.3. Interactive Diagram

5. Database Design

5.1. Entiry Relationship Diagram (ERD)



E. System Implementation & Test

1. Introduction

1.1. Overview

This section describes the approach and methodologies used by group to plan, organize and manage the testing of DSAV system. It provides in the detail all necessary information about the implementation and testing procedure of the system included test plans, test cases, test result, test environments, pass/fail criteria and risks estimations as well as a checklist to cover all possible cases.

1.2. Test approach

Goal: Test all features in the whole DSAV system based on the core flow.

• Method: black-box testing

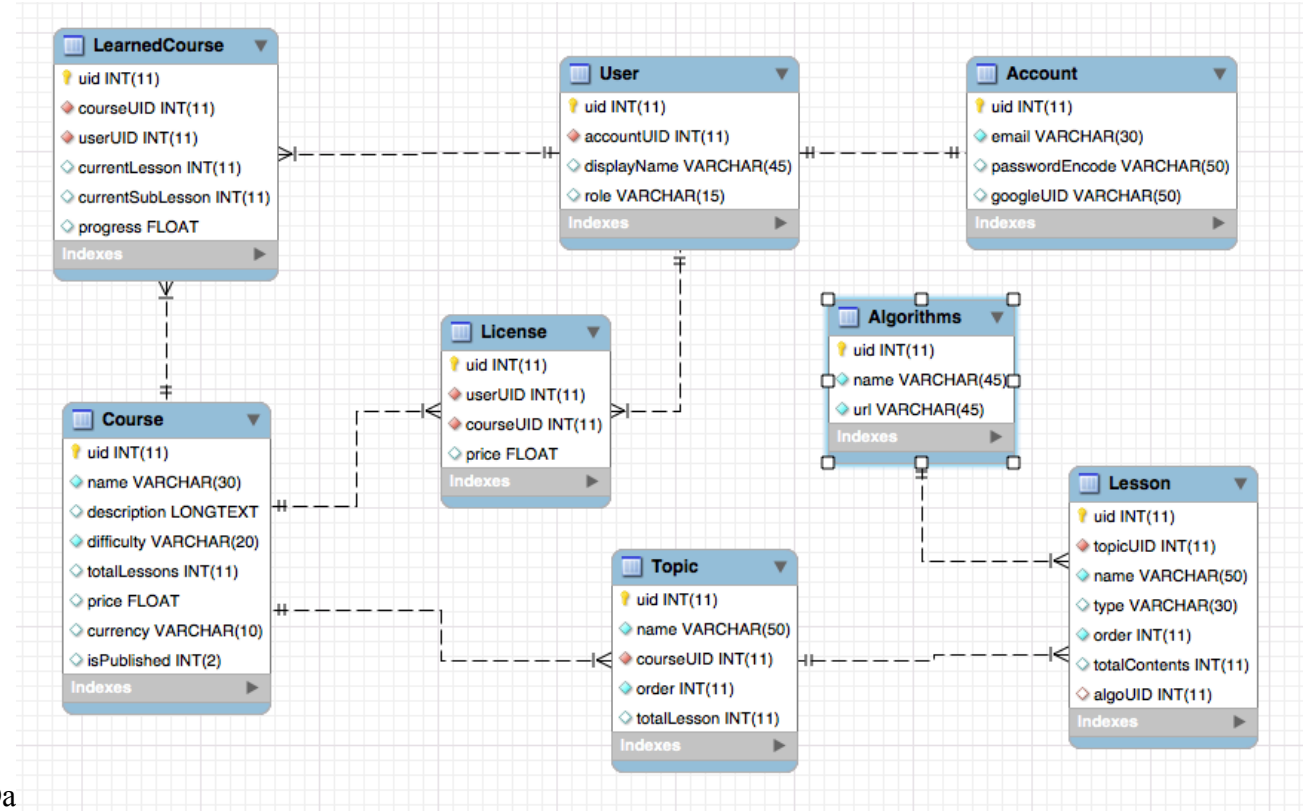
• Technique: check list

The testing for this project will consists of Integration System test level. Testing the program which was integrated and as a complete system to ensure that the software requirements have been met.

• Integration testing would be performed by all member of team and approved by team leader.

System testing is focused on assessing the system's reliability. This process is concerned with finding errors that result from unanticipated interactions between components and component interface problems.

2. Database Relationship Diagram



Da

F. Appendix

1. **MySQL [Online].** Available: https://dev.mysql.com/doc/
2. **ReactJS [Online].** Available: https://reactjs.org/