PART 2: REQUIREMENTS SPECIFICATION

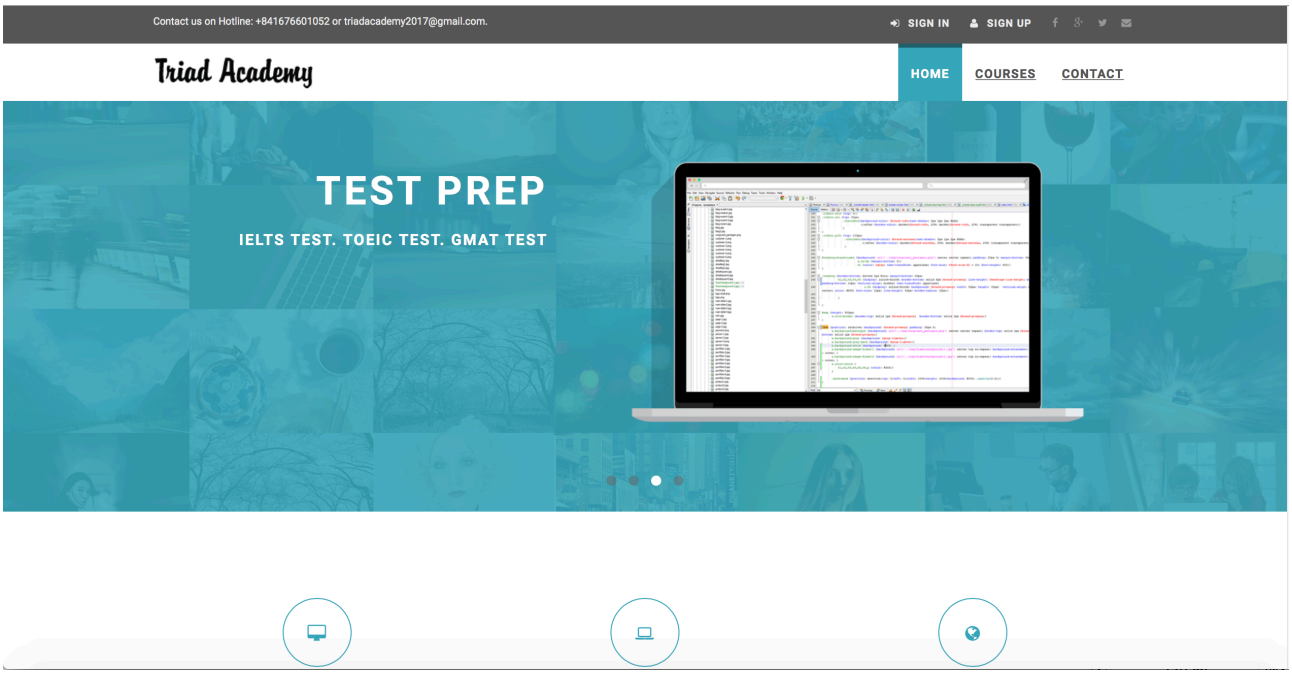
COURSE ACTIVITY MANAGEMENT FOR DISTANCE LEARNING

18.11.2017

COURSE: SOFTWARE ENGINEERING
FALL 2017

CLASS: TUESDAY MORNING (THEORY)
STATURDAY AFTERNOON (LABORATORY)

INSTRUCTOR: MS. NGUYEN THI THANH SANG



MEMBER CONTRIBUTION

NAME:	ID	CONTRIBUTION
Huynh Le Ngoc Han	ITITIU14027	I. Introduction III. Usecase & User stories
Le Truong Trong Nguyen	ITITIU14066	V. Non-Functional Requirements
Ly Thu Thao	ITITUN15008	II. Glossary IV. Functional Requirements

OUTLINE

I. INTRODUCTION

II. GLOSSARY

III. USECASE AND USER STORIES

IV. FUNCTIONAL REQUIREMENTS

V. NON – FUNCTIONAL REQUIREMENTS

I. INTRODUCTION

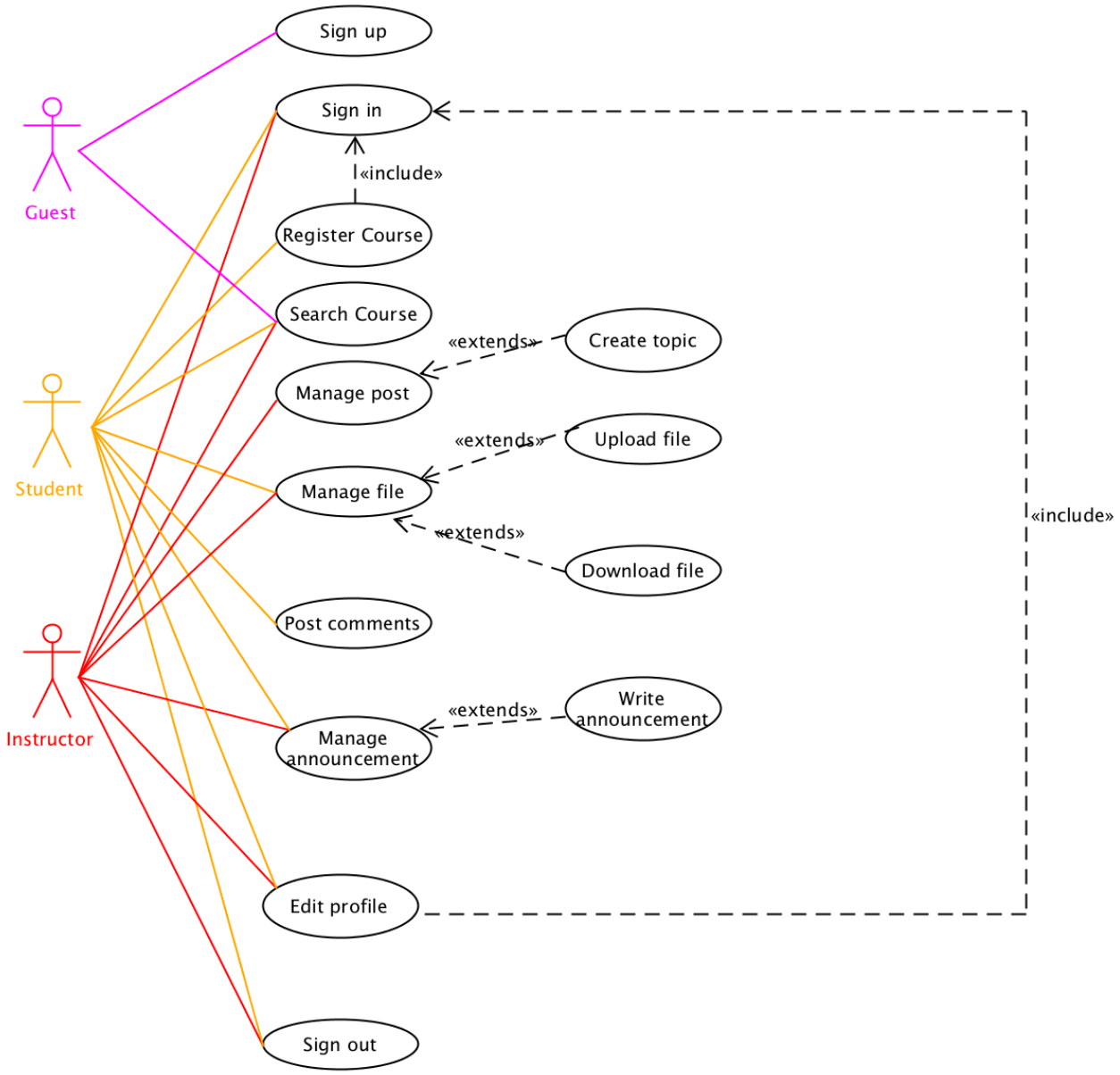
The Requirement Specification focuses on the introduction of the project including the purpose, scope as well as the details of the system requirements.
Based on this requirements specification, implementation of each function including all of the conditions as well as functional and non-functional requirements, supplied by the customers, will be made.
This document helps the reader to understand the Course Activity Management for Distance Learning web application by providing details into the product features.

II. GLOSSARY

Use Case	Identify the actors involved in an interaction and names the type of interaction. The additional information may be a textual description or one or more graphical models such as UML sequence or state charts.
Functional Requirements	Description of the system features to show how the software operates and processes the data from the user's input.
Non-functional Requirements	Standards and constraints on the functionalities of the system to meet customer's expectation

III. USECASE AND USER STORIES

3.1 Use Case Diagram



3.2 Use Case Decription

Use case description for user story 1: As a guest, I want to register an account so that I can have all member's accessibilities.

- **Name:** Account Registration
- **Identifier:** UC1
- **Input:**
  + Name
  + Password
  + Email
- **Output:**
  + If success: The welcome home page with user information.
  + If fail: The registration page with error message.
- **Pre-conditions:**
  + Email must be unique
  + Name should not include character or symbol.
- **Basic Course:**

User: Guest	System
1. Open Home Page	1.1 Display Home Page
2. Click on "Sign Up" button in Home Page.	2.1 Direct & display Registration Page
3. Enter user's information (include name, password, email) in the required box.	
4. Click on "Register" button	4.1 Check the user's input 4.1.1 Check user's name availability 4.1.2 Check user's email existed or not 4.1.2 Check password's validation 4.2 Redirect user to welcome home page, with user's account automatically logged in.

- **Alternative Path:**
  + Condition 1: Name, Email and password are invalid

In step 4, if name, password and email inputted by guest are invalid, the system will display an error message "Invalid name/password/email", and prompt guest to choose a different name / password or re-enter email to register again.

- Post-condition:

+ New user's account will be added into database.

Use case description for user story 2: As a user (instructor/student), I want to login/logout my account.

Use case Description 2.1: Login

- **Name:** Log in
- **Identifier:** UC2.1
- **Input:**
  + Email
  + Password
- **Output:**
  + Home page after successfully logged in.
- **Pre-conditions:**
  + Email and password must be valid.
- **Basic Course:**

User: Students/Instructors	System
1. Open Home Page	1.1 Display Home Page
2. Click on "Sign in" button	2.1 Direct & display Login Page
3. Enter email & password	
4. Click on "Login" button	4.1 Validate the input 4.2 Retrieve matching user from database 4.3 Record the result user object to session 4.4 Redirect user to the homepage

- **Alternative Path:**
  + Condition 1: Users enter invalid email or password.

  In step 4, after validating user's input, system displays warning message "Invalid email/password" and ask users to enter again.
- **Post-condition:**
  + Users successfully log in.

Use case Description 2.2: Log out

- **Name:** Log out
- **Identifier:** UC2.2
- **Input:** None
- **Output:**
    + Return home page after successfully logged out.
- **Pre-conditions:**
    + User has already logged in.
- **Basic Course:**

User: Students/Instructors	System
1. Click on “Log out” button	1.1 Retrieve user object from session 1.2 Erase current user object from session 1.3 Redirect user to the home page

- **Alternative Path**: None
- **Post-condition:**
    + Users successfully log out.

Use case description for user story 3: As a user (instructor/student), I want to search for available courses.

- **Name:** Search course
- **Identifier**: UC3
- **Input:**
    + Course name
- **Output:**
    + All courses related to keywords are listed.
- **Pre-conditions:**
    + User is in the Course page.
- **Basic Course:**

User: Students/Instructors	System
1. Click on “Search” button	
2. Input keyword	

3. Press “Enter”	3.1 Find all courses related to keyword in database 3.2 Displays all course found

- Alternative Path:

+ Condition 1: In step 2, if users input nothing and then press “Enter”, the system will return all of course.

- Post-condition:

+ Courses related to keyword are listed out.

Use case description for user story 4: As a student, I want to register a course

- Name: Course Registration

- Identifier: UC4

- Input: None

- Output:

+ “Registered” status is displayed and number of slots is decreased by 1.

- Pre-conditions:

+ Users had to log in into the system with their authorized student account.

- Basic Course:

User: Students	System
1. Click on “Available Courses”	1.1 Displays Course page which include all courses available.
2. Click on “Register”	2.1 Save registration into database 2.2 Displays “Registered” status and decrease number of slots available.

- Alternative Path: None

- Post-condition:

+ Student’s information and chosen course are saved into database.

+ List of students in each course is generated.

Use case description for user story 5: As an instructor, I want to post a thread (including setting deadline and contents).

- **Name**: Create Post (Thread)

- Identifier: UC5

- Input:

+ Subject
+ Content
+ Deadline

- Output:

+ Posts are publically added in Course homepage.

- Pre-conditions:

+ User must log in as instructor
+ User must be in the Course page (where all courses he/she teach are displayed)

- Basic Course:

User: Instructors	System
1. Choose the course which instructor want to create post by click on "View"	1.1 Display "All Discussion page
2. Choose "All discussion" session	2.1 Display all discussions created before
3. Click on "Create"	3.1 Display "Create Discussion" box
4. Enter discussion name, content and set deadline	
5. Click "Create" button	5.1 Create new post with the inputted information 5.2. Add this post to the current Discussion board and store it into database

- Alternative Path:

+ Condition 1: The users miss some information boxes

In step 4, after submitting the description, the system displays warning messages, asks users fill in all required boxes.

- Post-condition:

+ A new post is created and stored in the database. It must also be added to the current Discussion board.

Use case description for user story 6: As an instructor/student, I want to upload/download (documents, images, etc.)

Use case Description 6.1: Upload file

- Name: Upload File

- Identifier: UC6.1

- Input:

+ File Name

- Output:

+ File is uploaded successfully.

- Pre-conditions:

+ User must log in as instructor

+ User must be in the Course page (where all courses he/she teach are displayed).

- Basic Course:

User: Instructors	System
1. Choose the course which instructor want to upload file by click on "View"	1.1 Display "All contents" board.
2. Click "Upload"	2.1 Display a box prompt for choosing file
3. Click to choose a file and click "OK"	3.1 Create a store file in local host 3.2 All file's information is saved in database. 3.3 Add file into current contents board.

- Alternative Path: None

- Post-condition:

+ New file is uploaded successfully and added into current content board.

+ File's information is added into database.

Use case Description 6.2: Download file

- Name: Download File

- Identifier: UC6.2

- Input: None

- Output:

+ File is downloaded successfully.

- Pre-conditions:

+ User must log in as students

+ User must be in the Course page (where all courses students joined are displayed).

- Basic Course:

User: Students	System
1. Choose “My course”	1.1 Displays all courses students have joined
2. Click “View”	2.1 Display “All contents” board
3. Choose file that students want to download and click “Download”	

- Alternative Path: None

- Post-condition:

+ File is downloaded successfully.

Use case description for user story 7: As a student, I want to post comments/messages to contribute to topic.

- Name: Post Comments/Message

- Identifier: UC7

- Input:

+ Content of the comment (for normal discussion topic)

+ File Name (for special topic – final examination)

- Output:

+ Current posts with new comments added (for normal discussion topic).

+ Exam paper is upload successfully (for special topic – final examination).

- Pre-conditions:

+ User must log in as student

+ The Discussion topic students need to add comment to contribute must be existed in database (for normal discussion topic).

+ The Final examination students need to submit their paper exam must be existed in database (for special topic – final examination).

+ User must be in the Course page (where all courses students joined are displayed).

- Basic Course (for normal discussion topic):

User: Students	System
1. Choose “My course”	1.1 Displays all courses students have joined
2. Click “View”	2.1 Display “All contents” board

3. Click “All discussion” button	3.1 Display “All discussion” board
4. Choose discussion topic that students want to comment by click on “View”	4.1 Display discussion page
5. Enter the contents of comments	
6. Click “Post comment”	6.1 Create a new comment with the given inputs 6.2 Add the comment into the current discussion board, and store the comment in the database. 6.3 Display comment’s content in a block.

- Basic Course (for special topic – final examination):

User: Students	System
1. Choose “My course”	1.1 Displays all courses students have joined
2. Click “View”	2.1 Display “All contents” board
3. Click “All discussion” button	3.1 Display “All discussion” board
4. Choose Special discussion topic (Final Examination) that students want to upload their exam paper by click on “View”	4.1 Display discussion page
5. Choose “Choose file” button to select file that students want to upload	
6. Click “Upload”	6.1 Create a store file in local host 6.2 All file’s information is saved in database. 6.3 Add file into current contents board.

- Alternative Path: None

- Post-condition:

+ The comment is added into the respective post stored in the database successfully (for normal discussion topic).

+ New file is uploaded successfully and added into current content board (for special topic – final examination).

+ File's information is added into database (for special topic – final examination).

Use case description for user story 8: As an instructor, I want to set announcements to all students.

- **Name**: Set Announcement
- **Identifier:** UC8
- **Input:**
  + Course selection
  + Subject
  + Contents
- **Output:**
  + New announcement is posted in Announcement page.
  + New announcement is sent for all students in this class.
- **Pre-conditions:**
  + User must log in as instructor.
  + User must be in the Announcement page (where all announcements are displayed).
- **Basic Course:**

User: Instructors	System
1. Enter announcement's information (select course, subject, content)	
2. Click "Create Content"	2.1 Announcement's information is saved into database. 2.2 Student account get announcement by getting from database.

- **Alternative Path:** None
- **Post-condition:**
  + Announcement is added into database.

Use case description for user story 9: As a user (students/instructors), I want to edit my profile.

- **Name:** Edit Profile
- **Identifier**: UC9

- Input:

+ First Name
+ Last Name

- Output:

+ Profile with uploaded content according to user's input.

- Pre-conditions:

+ User must be the profile owner.
+ User must have already logged in.
+ User must be in his / her Account Page to perform Edit profile function.

- Basic Course:

User: Students/Instructor	System
1. Click "Account" button	1.1 Display Account page 1.2 Prompt for inputs
2. 2. Enter new content of attributes that user wants to change.	2.1 Record the inputs 2.2 Retrieve user ID from session 2.3 Perform modifications for the corresponding user based on user ID 2.4 New contents are saved in database

- Alternative Path: None

- Post-condition:

+ New contents are added into database.

IV. FUNCTIONAL REQUIREMENTS

UC1: Account registration

- Shall display a page requiring user's information.
- Shall validate Name, email and password.
- Shall create new account in database after successful registration.
- Shall Check the existed account.

UC2: Login/Logout

- Shall display a box requiring email and password to login.
- Shall validate inputs (email, password) with the database.
- Shall notify user if the login attempt is unsuccessful (username and password do not match, etc).
- Shall redirect user to the home page after logging in successfully.
- Shall display the logout icon on screen only when users have already logged in.
- Shall erase user object from the current session and close all user's activities after user has logged out.
- Shall redirect user back to the home page after user has logged out successfully.
- The user account must be valid for user to be able to login.

UC3: Search Course

- Shall display the 'Search' icon.
- Shall sort the available courses that match from the database.
- Shall display the search results.

UC4: Register Course

- Shall display a page listing the available courses.
- Shall display the 'Register' button on the course.
- Shall display the 'Registed' status after successfully register the course.
- Shall decrease the slot available after successfully register the course.
- Shall delete course from available course if all slots are out.

UC5: Create Post

- Shall display "Create " button in a thread to create discussion topic.
- Shall display all requirements of creating a new discussion topic for members to fill out.

- Shall receive all the information that members having filled out and save into database of that thread.
- Shall store the created topic in the database, and register it with the corresponding thread.
- All topic data should be stored in the database.

UC6: Upload/Download file

- Shall display “Upload ” button in course contents to upload file.
- Shall display “Download ” button in course contents to download file.

UC7: Post comments

- Shall allow the user to type the comment in a text box.
- Shall display the comment in a block.
- Shall store the comment in the database.
- Shall display “Upload ” button in special discussion topic to submit .

UC8: Set Announcements

- Shall allow user (instructor) to create announcement
- Shall allow student to receive announcement form their instructor

UC9: Edit Profile

- Shall display the “Account” button on screen when users have already logged in.
- Shall display all of the current information (first name, last name …) of the user and the confirm button.
- Shall update new information in the account database after users have confirmed their new edited information.
- All user data must be updated and stored in the database accordingly.

V. NON-FUNCTIONAL REQUIREMENTS

N0 #1	**Environment**
1. Summary	The system is built on a reliable environment which must both be stable and upgradable.
2. Rationale	The environment is one of the most important parts when developing an application. If the environment is not stable, it will be dangerous for both user and developer, and may greatly affect the profits made from the application. The environment is also need to be upgradable for future improvement and new functions.
3. Requirements	The system is developed on NetBeans framework. Using HTML Java (Servlet and JSP) language. Database system: MySQL Server 5.6

N0 #2	**Security**
1. Summary	All external communications between the system's data server and clients must be encrypted.
2. Rationale	Only approved users can use the system. The system must be secure enough survive attacks from outsiders.
3. Requirements	Session

N0 #3	Understandability
1. Summary	The webpage UI need to be simple, easy to use. The architecture design of system is clear and code is readable.
2. Rationale	As the growing number of user, the system must be simple and clear enough for many different types of people.
3. Requirements	Bootstrap, HTML5, CSS3, Javascript, Ajax

N0 #4	Usability
1. Summary	Well-structured user manuals include informative error messages Well-formed graphical user interfaces.
2. Rationale	The presentation of information and choices in a clear and concise way, a lack of ambiguity and the placement of important items in appropriate areas. Ensuring that the content works on various devices and browsers.
3. Requirements	Bootstrap, HTML5, CSS3, Javascript, Ajax

N0 #5	Reliability
1. Summary	The system must not return any inaccurate result except when the input information is inaccurate.
2. Rationale	When the user want to search for courses, the result must be accurate and reliable, all information must be authorized.
3. Requirements	Contact information visible on website. Accuracy of the search query result must be ensured.

N0 #6	Stability
1. Summary	The system must be stable when the range of users increases.
2. Rationale	When the webpage get popular, the number and diversity of user will increase, the webpage must be stable, and maintain the loading speed to ensure the user experience.
3. Requirements	Database must be simple, yet sufficient and upgradable.

N0 #7	Supportability
1. Summary	Developed team must reply any questions from the user as soon as possible.
2. Rationale	When the user wants to contact to the administrator or the administrator want to contact the development team, the website must provide a method that is quick and reliable for better communication.
3. Requirements	HTML forms, session, get/post, email.