**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Examination Tools applying Block Chain Technology**

|  |  |
| --- | --- |
| Group 4 | |
| Group members | Nguyễn Minh Hưng – SE61794  Cao Trung Hiếu – SE61801  Trương Tấn Sang – SE61926 |
| Supervisor | Kiều Trọng Khánh |
| Ext. Supervisor | N/A |
| Capstone Project code | ExamTool |

- Ho Chi Minh City, **19 November 2018** -

Table of Contents

[Table of Contents 2](#_Toc532244960)

[List Of Tables 6](#_Toc532244961)

[List of Figures 8](#_Toc532244962)

[Definitions, Acronyms, and Abbreviations 9](#_Toc532244963)

[1. Project Information 13](#_Toc532244964)

[2. Introduction 13](#_Toc532244965)

[3. Current Situation 13](#_Toc532244966)

[4. Problem Definition 14](#_Toc532244967)

[4.1. Proposed Solution 14](#_Toc532244968)

[4.2. Feature Functions 14](#_Toc532244969)

[5. Values and Challenges Value 14](#_Toc532244970)

[6. Functional Requirement 14](#_Toc532244971)

[7. Role and Responsibility 15](#_Toc532244972)

[8. Software Process Model 15](#_Toc532244973)

[9. Conceptual Diagram 16](#_Toc532244974)

[10. Use-case Diagram 18](#_Toc532244975)

[11. Use-case Specification 22](#_Toc532244976)

[11.1. <IT> Add Must Be Killed Process to List 22](#_Toc532244977)

[11.2. <IT> Add to List Exception Process 23](#_Toc532244978)

[11.3. <IT> Kill Process 25](#_Toc532244979)

[11.4. <IT> Get Student Screen 26](#_Toc532244980)

[11.5. <IT> Submit Student Test Exam 28](#_Toc532244981)

[11.6. <IT> Scan student by process 29](#_Toc532244982)

[11.7. <Training Department Staff> Add new Semester 31](#_Toc532244983)

[11.8. <Teacher> Add Question by Form 33](#_Toc532244984)

[11.9. <Teacher> Add Question by Import File 35](#_Toc532244985)

[11.10. <Teacher> Edit Questions 36](#_Toc532244986)

[11.11. <Teacher> Approve Test Exam 39](#_Toc532244987)

[11.12. <Training Department Staff> Generate Test Exam 41](#_Toc532244988)

[11.13. <Training Department Staff> Assign Approver 43](#_Toc532244989)

[12. System Architecture Design 44](#_Toc532244990)

[12.1. Web Server Architecture Design 45](#_Toc532244991)

[12.2. Client Architecture Design 45](#_Toc532244992)

[13. Component Diagram 47](#_Toc532244993)

[14. Detailed Description 48](#_Toc532244994)

[14.1. Class Diagram 49](#_Toc532244995)

[14.2. Class Diagram Explanation 51](#_Toc532244996)

[14.2.1 Semester 51](#_Toc532244997)

[14.2.2 Exam 51](#_Toc532244998)

[14.2.3 ExamStudent 51](#_Toc532244999)

[14.2.4 Student 52](#_Toc532245000)

[14.2.5 BaseTest 52](#_Toc532245001)

[14.2.6 BaseTestContent 53](#_Toc532245002)

[14.2.7 Test 53](#_Toc532245003)

[14.2.8 TestContent 53](#_Toc532245004)

[14.2.9 Question 54](#_Toc532245005)

[14.2.10 Option 54](#_Toc532245006)

[14.2.11 Level 55](#_Toc532245007)

[14.2.12 QuestionType 55](#_Toc532245008)

[14.2.13 QuestionGroup 55](#_Toc532245009)

[14.2.14 QuestionObjective 55](#_Toc532245010)

[14.2.15 Chapter 56](#_Toc532245011)

[14.2.16 Course 56](#_Toc532245012)

[14.2.17 MainObjective 56](#_Toc532245013)

[14.2.18 LearningOutcome 56](#_Toc532245014)

[14.2.19 TeacherTest 57](#_Toc532245015)

[14.2.20 TeacherTestContent 57](#_Toc532245016)

[15. Database Design 58](#_Toc532245017)

[15.1. Entity Relationship Diagram 58](#_Toc532245018)

[15.2. Data Dictionary 59](#_Toc532245019)

[15.3. ERD Explanation 59](#_Toc532245020)

[15.3.1 Semester 59](#_Toc532245021)

[15.3.2 Exam 60](#_Toc532245022)

[15.3.3 Student 60](#_Toc532245023)

[15.3.4 BaseTest 60](#_Toc532245024)

[15.3.5 BaseTestContent 61](#_Toc532245025)

[15.3.6 Test 61](#_Toc532245026)

[15.3.7 TestContent 62](#_Toc532245027)

[15.3.8 Question 62](#_Toc532245028)

[15.3.9 Option 62](#_Toc532245029)

[15.3.10 Level 62](#_Toc532245030)

[15.3.11 QuestionType 63](#_Toc532245031)

[15.3.12 QuestionGroup 63](#_Toc532245032)

[15.3.13 QuestionObjective 63](#_Toc532245033)

[15.3.14 Chapter 63](#_Toc532245034)

[15.3.15 Course 64](#_Toc532245035)

[15.3.16 MainObjective 64](#_Toc532245036)

[15.3.17 LearningOutcome 64](#_Toc532245037)

[15.3.18 Course Department 64](#_Toc532245038)

[15.3.19 TeacherTest 64](#_Toc532245039)

[15.3.20 TeacherTestContent 65](#_Toc532245040)

[16. Interaction Diagram 65](#_Toc532245041)

[16.1.1 Activity Diagram 65](#_Toc532245042)

[16.1.1.1 Teacher Import Question 65](#_Toc532245043)

[16.1.1.2 Generate Test Exam 67](#_Toc532245044)

[16.1.1.3 Teacher Approve Test Exam 67](#_Toc532245045)

[16.1.1.4 Student Take a Test Exam 68](#_Toc532245046)

[17. Database Relationship Diagram 69](#_Toc532245047)

[17.1. Physical Diagram: 69](#_Toc532245048)

[17.2. Data Dictionary: 72](#_Toc532245049)

[17.2.1 Semester 72](#_Toc532245050)

[17.2.2 Exam 72](#_Toc532245051)

[17.2.3 Student 72](#_Toc532245052)

[17.2.4 BaseTest 72](#_Toc532245053)

[17.2.5 BaseTestContent 73](#_Toc532245054)

[17.2.6 Test 73](#_Toc532245055)

[17.2.7 TestContent 74](#_Toc532245056)

[17.2.8 Question 74](#_Toc532245057)

[17.2.9 Option 75](#_Toc532245058)

[17.2.10 Level 75](#_Toc532245059)

[17.2.11 QuestionType 75](#_Toc532245060)

[17.2.12 QuestionGroup 76](#_Toc532245061)

[17.2.13 QuestionObjective 76](#_Toc532245062)

[17.2.14 Chapter 76](#_Toc532245063)

[17.2.15 Course 76](#_Toc532245064)

[17.2.16 MainObjective 77](#_Toc532245065)

[17.2.17 LearningOutcome 77](#_Toc532245066)

[17.2.18 Course Department 77](#_Toc532245067)

[17.2.19 TeacherTest 77](#_Toc532245068)

[17.2.20 TeacherTestContent 78](#_Toc532245069)

[17.2.21 AuthenticationApp 78](#_Toc532245070)

[17.2.22 Import Result 79](#_Toc532245071)

[17.2.23 DefaultProcess 79](#_Toc532245072)

[18. System Implement 79](#_Toc532245073)

[18.1. Web Server Application 79](#_Toc532245074)

[18.2. Web Client Application 80](#_Toc532245075)

[19. Algorithms 81](#_Toc532245076)

[19.1. Calculating Percentage Similarity of 2 Question 81](#_Toc532245077)

[19.1.1 Problem 81](#_Toc532245078)

[19.1.2 Solution 82](#_Toc532245079)

[19.1.3 Complexity 84](#_Toc532245080)

[20. Future Plans 84](#_Toc532245081)

List Of Tables

[Table 1: Definitions, Acronyms, and Abbreviations 9](#_Toc532244874)

[Table 2: Role and Responsibilities 15](#_Toc532244875)

[Table 4 CONCEPTUAL DIAGRAM EXPLAINATION 18](#_Toc532244876)

[Table 4 Add Must Be Killed Process to List 23](#_Toc532244877)

[Table 6 ADD TO LIST EXCEPTION PROCESS 25](#_Toc532244878)

[Table 7 Kill Process 26](#_Toc532244879)

[Table 9 <IT> GET STUDENT SCREEN 27](#_Toc532244880)

[Table 10 <IT> Submit Student Test Exam 29](#_Toc532244881)

[Table 23 <IT> Scan student by process 31](#_Toc532244882)

[Table 11 <TRAINING DEPARTMENT STAFF> ADD NEW SEMESTER 32](#_Toc532244883)

[Table 15 <TEACHER> ADD QUESTION BY FORM 34](#_Toc532244884)

[Table 16 <Teacher> Add Question by Import File 36](#_Toc532244885)

[Table 17 <Teacher> Edit Questions 38](#_Toc532244886)

[Table 18 <Teacher> Approve Test Exam 40](#_Toc532244887)

[Table 19 <Training department staff> Generate Test Exam 42](#_Toc532244888)

[Table 20 <Training department staff> Assign Approver 44](#_Toc532244889)

[Table 21 ARCHITECTURE OVERVIEW DESCRIPTION 45](#_Toc532244890)

[Table 22Component diagram dictionary 48](#_Toc532244891)

[Table 23 Class diagram Description 51](#_Toc532244892)

[Table 24 Class Semester Explanation 51](#_Toc532244893)

[Table 26 Class exam Explanation 51](#_Toc532244894)

[Table 28 Class ExamStudent Explanation 52](#_Toc532244895)

[Table 30 CLASS STUDENT EXPLANATION 52](#_Toc532244896)

[Table 32 CLASS BASETEST EXPLANATION 53](#_Toc532244897)

[Table 34 CLASS BASETESTCONTENT EXPLANATION 53](#_Toc532244898)

[Table 36 CLASS TEST EXPLANATION 53](#_Toc532244899)

[Table 38 CLASS TESTCONTENT EXPLANATION 54](#_Toc532244900)

[Table 40 CLASS QUESTION EXPLANATION 54](#_Toc532244901)

[Table 42 CLASS OPTION EXPLANATION 55](#_Toc532244902)

[Table 44 CLASS level EXPLANATION 55](#_Toc532244903)

[Table 46 CLASS QUESTIONTYPE EXPLANATION 55](#_Toc532244904)

[Table 48 CLASS QUESTIONGROUP EXPLANATION 55](#_Toc532244905)

[Table 50 CLASS QUESTIONOBJECTIVE EXPLANATION 56](#_Toc532244906)

[Table 52 CLASS CHAPTER EXPLANATION 56](#_Toc532244907)

[Table 54 CLASS COURSE EXPLANATION 56](#_Toc532244908)

[Table 56 CLASS MAINOBJECTIVE EXPLANATION 56](#_Toc532244909)

[Table 58 CLASS LEARNINGOUTCOME EXPLANATION 57](#_Toc532244910)

[Table 60 CLASS TeacherTest EXPLANATION 57](#_Toc532244911)

[Table 63 Class teachertestcontent explanation 57](#_Toc532244912)

[Table 64 ER Dictionary 59](#_Toc532244913)

[Table 65 Semester table Description 60](#_Toc532244914)

[Table 66 Exam table Description 60](#_Toc532244915)

[Table 67 Student table Description 60](#_Toc532244916)

[Table 68 BaseTest table Description 61](#_Toc532244917)

[Table 69 BaseTestContent table Description 61](#_Toc532244918)

[Table 70 Test table Description 61](#_Toc532244919)

[Table 71 TestContent table Description 62](#_Toc532244920)

[Table 72 Question table Description 62](#_Toc532244921)

[Table 73 Option table Description 62](#_Toc532244922)

[Table 74 Level table Description 63](#_Toc532244923)

[Table 75 QuestionType table Description 63](#_Toc532244924)

[Table 76 QuestionGroup table Description 63](#_Toc532244925)

[Table 77 QuestionObjective table Description 63](#_Toc532244926)

[Table 78 Chapter table Description 63](#_Toc532244927)

[Table 79 Course table Description 64](#_Toc532244928)

[Table 80 MainObjective table Description 64](#_Toc532244929)

[Table 81 LearningOutcome table Description 64](#_Toc532244930)

[Table 82 CourseDepartment table Description 64](#_Toc532244931)

[Table 83 TeacherTest table Description 65](#_Toc532244932)

[Table 84 TeacherTestContent table Description 65](#_Toc532244933)

[Table 85 Physical Diagram Description 71](#_Toc532244934)

[Table 86 SEMESTER TABLE DESCRIPTION 72](#_Toc532244935)

[Table 87 Exam table Description 72](#_Toc532244936)

[Table 88 Student table Description 72](#_Toc532244937)

[Table 89 BaseTest table Description 73](#_Toc532244938)

[Table 90 BaseTestContent table Description 73](#_Toc532244939)

[Table 91 Test table Description 74](#_Toc532244940)

[Table 92 TestContent table Description 74](#_Toc532244941)

[Table 93 Question table Description 75](#_Toc532244942)

[Table 94 Option table Description 75](#_Toc532244943)

[Table 95 Level table Description 75](#_Toc532244944)

[Table 96 QuestionType table Description 76](#_Toc532244945)

[Table 97 QuestionGroup table Description 76](#_Toc532244946)

[Table 98 QuestionObjective table Description 76](#_Toc532244947)

[Table 99 Chapter table Description 76](#_Toc532244948)

[Table 100 Course table Description 76](#_Toc532244949)

[Table 101 MainObjective table Description 77](#_Toc532244950)

[Table 102 LearningOutcome table Description 77](#_Toc532244951)

[Table 103 CourseDepartment table Description 77](#_Toc532244952)

[Table 104 TeacherTest table Description 78](#_Toc532244953)

[Table 105 TeacherTestContent table Description 78](#_Toc532244954)

[Table 106 AuthenticationApp table Description 78](#_Toc532244955)

[Table 107 ImportResult table Description 79](#_Toc532244956)

[Table 108 DefaultProcess table Description 79](#_Toc532244957)

List of Figures

[Figure 1 The Scrum Process 17](#_Toc532244838)

[Figure2 Conceptual Diagram 18](file:////Volumes/Document/Capston/document/tomtat.docx#_Toc532244839)

[Figure 3: Use case overview Examination System 19](#_Toc532244840)

[Figure 4: Teacher use case Detail 20](#_Toc532244841)

[Figure 5: Leader use case detail 20](#_Toc532244842)

[Figure 6: IT use case detail 21](#_Toc532244843)

[Figure 7: Manager use case detail 21](#_Toc532244844)

[Figure 8: Student use case detail 21](#_Toc532244845)

[Figure 9: Training Department Staff use case detail 22](#_Toc532244846)

[Figure 10 Add Must Be Killed Process to List 23](#_Toc532244847)

[Figure 11 Add to List Exception Process 24](#_Toc532244848)

[Figure 12 Kill Process 26](#_Toc532244849)

[Figure 14 <IT> Get Student Screen 27](#_Toc532244850)

[Figure 15 <IT> Submit Student Test Exam 29](#_Toc532244851)

[Figure 13 <IT> Scan student by process 30](#_Toc532244852)

[Figure 16 <Training Department Staff> Add new Semester 32](#_Toc532244853)

[Figure 20 <Teacher> Add Question by Form 34](#_Toc532244854)

[Figure 21 <Teacher> Add Question by Import File 36](#_Toc532244855)

[Figure 22 <Teacher> Edit Questions 37](#_Toc532244856)

[Figure 23 <Teacher> Approve Test Exam 40](#_Toc532244857)

[Figure 24 <Teacher> generate Test Exam 42](#_Toc532244858)

[Figure 25 < Training Department Staff > Approve Test Exam 44](#_Toc532244859)

[Figure 26 System Architecture Design 45](#_Toc532244860)

[Figure 27 Architecture Overview 46](#_Toc532244861)

[Figure 28 Web Admin MVC Architecture 47](#_Toc532244862)

[Figure 29 Component Diagram 48](file:////Volumes/Document/Capston/document/tomtat.docx#_Toc532244863)

[Figure 32 ER DIAGRAM 59](#_Toc532244864)

[Figure 33 Activity diagram Teacher Import Question 68](#_Toc532244865)

[Figure 34 Activity diagram Training department staff generate Test Exam 68](#_Toc532244866)

[Figure 35 Activity diagram Approve Test Exam 69](#_Toc532244867)

[Figure 36 Activity diagram Student take a test exam 70](#_Toc532244868)

[Figure 37 Physical diagram 71](file:////Volumes/Document/Capston/document/tomtat.docx#_Toc532244869)

[Figure 38 Onion Architecture 81](#_Toc532244870)

[Figure 39 Event-driven architecture 82](#_Toc532244871)

[Figure 40 Angular Component Example 82](#_Toc532244872)

[Figure 41 Levenshtein distance example result 85](#_Toc532244873)

Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Name** | **Definition** |
| L.O | Learning Outcome |
| MVC | Model View Controller |
| API | Application Programming Interface |
| IDE | Integrated development environment |
| GUI | Graphic user interface |
| ERD | Entity relationship diagram |
| I/O | Input/Output |

Table 1: Definitions, Acronyms, and Abbreviations

**

**CAPSTONE PROJECT REGISTER**

Class: Duration time: from / /2018…. To / /2018…..

(\*) Profession: <Software Engineer> Specialty: <ES> <IS>

x

(\*) Kinds of person make registers: Lecturer Students

x

1. Register information for supervisor (if have)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-Mail** | **Title** |
| Supervisor 1 | Kiều Trọng Khánh |  | khanhkt@fpt.edu.vn | Mr. |

2. Register information for students (if have)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Full name** | **Student code** | **Phone** | **E-mail** | **Role in Group** |
| Student 1 |  |  |  |  |  |
| Student 2 |  |  |  |  |  |
| Student 3 |  |  |  |  |  |
| Student 4 |  |  |  |  |  |

3. Register content of Capstone Project

(\*) 3.1. Capstone Project name:

English: Examination Tools applying Block Chain Technology

Vietnamese: Xây dựng hệ thống quản lý đề thi và việc thi cử sử dụng công nghệ Block Chain

Abbreviation: ExamTool

**- Context:**

+ The Question Bank need to manipulate and make it more security.

+ The examination should be more security and the result should be matched on the server

+ The application must compliance with Testing Department process depending user requirements.

+ The system must be built following the designed and developed DB

**- Building the system provides following services:**

* The system allows to import many question with different types. Then, it stores to storage with security format
* Building as examination tool allows to generate examination test from question banks. The tools support exam with block chain technology to match the result and examinee in progress.
* …

**- Simulator:**

* Make the simulator with import many type of questions into question banks and make the examination test to explain how’d you do exam in progress.

(\*) 3.2. Main proposal content (including result and product)

1. Theory and practice (document):

* Student should apply the software development process and the UML.
* Software artifacts include User Requirement, Software Requirement Specification, Architecture Design, Detail Design, System Implementation and Testing Document, Installation Guide, sources code, and deployable software packages.
* 3 tiers should be applied.
* Server-side technique:
  + Database design, OOA, OOD, OOP, MVC, Java or .Net technology, …
* Client-side technique:
  + HTML5, CSS, JavaScript, JQuery, Ajax, Android, JavaFX, .Net Framework, ...
* Communication technique:
  + Exchange information and transfer data in effective in networks, communicating protocol between devices or technology...
  + ...
* Research
  + Block chain.
  + Encrypt/ Decrypt
  + Question Type Formatting
  + …

1. Program:

* Main functions:
  + Manage question Banks
  + Build the Examination Test Tools
  + …

1. Other products:

* All of management functions of the system must be implemented to support the operating system in best.
* Papers.

4. Other comment (propose all relative thing if have)

+ The system must be built following the designed and developed DB

+ The application must compliance with Testing Department process depending user requirements

|  |  |
| --- | --- |
| **Supervisor (If have)**  *(Sign and full name)* | HCM city, date 30/8/2018  **On behalf of Registers**  *(Sign and full name)* |

Kiều Trọng Khánh

## Project Information

* Project name: **Examination Tools applying Block Chain Technology**
* Project Code: **ExamTool**
* Product Type: **Web Application, Desktop Application**
* Start Date: **September 11, 2018**
* End Date: **December 13, 2018**

## Introduction

In this document, we will introduce a solution for FPT University Examination system. The current system have some problems like it does not guarantee the quality of the result and it may be used by students to exploit vulnerabilities for cheating. Based on our researches and analysis, we proposed a new solution for FPT University Examination system.

We will organize questions, exams by learning outcome and apply block-chain technology in there. L.O are statements that describe the knowledge or skills which the students should acquire by the end of a particular assignment, class, course, or program and help students understand why those knowledge and skills will be useful to them. A test exam created with L.O will ensure the coverage of a course content inside it. Beside the insurance of the test exam content, we also ensure the test exam will be protected by Block-chain technology. Block-chain is a technology which allows data to be transmitted safely and correctly by a very complex encryption. It will be used in the test exam approving phase to ensure that test exam will not be editable.

## Current Situation

Two weeks before the exam start, staff will create test exam for each subject. The current question management system is manage questions by chapters. Depending on the teacher's requirements, the staff will randomly select questions based on the topic or chapter. Such random selection like that can not guarantee the test for being covered with the entire knowledge of the subject. After generated, lecture of the subject will be notified and review the test exam. During review process, lectures can remove or edit any questions in the test but the system could not update those modifications automatically in the question bank. Therefore, the reviewer has to update the question once over again in next time. If nothing is wrong, the test will be kept confidential until the exam. The reviewer has to sign in a document to confirm that the test can be used for the upcoming exam.

Upon exam arrival, students will enter the examination room, open the current examination software to take the exam. Before entering the room, students can turn on fraudulent applications and hide it. The examiners cannot know what students are doing outside the room so if the software is not in the blacklist it will not be destroyed by the exam software and students can use actions such as using shortcut to trigger an event to capture a test page, or to black out a self-contained search text message, or, more importantly, to use TeamViewer to complete the test. Such behaviors will help students pass the exam easily and not evaluate the real ability of students. After the exam finish, the exam software can add those cheating applications to the blacklist, students will find another application and so on, cannot prevent students from cheating during the exam.

## Problem Definition

### Proposed Solution

* The course is divided by topic and chapter, not cover all the content of a course.
* The question updating is repeated and not synchronized. So that, reduce test exam quality by duplicate question and waste of server’s storage.
* Cannot manage student take the exam process during examinations

### Feature Functions

* **Random question base on L.O:** To generate test exam, staff will select L.O first, then all questions will be random base on them.
* **Update question in question bank:** After lecture edit questions in the test exam, that questions in question bank will be updated.
* **Anti-cheating:** We will prevent unnecessary actions and all processes in the list which was not necessary during the test, will be killed.

## Values and Challenges Value

**Values**

* Provide better question bank management.
* Generate exam that can cover all content of the course.
* Kill processes which is not allowed while doing an exam.

**Challenges**

* Many import types is supported.
* Reduce cheating on exam.

## Functional Requirement

Functional requirements of the system are listed as below:

**Training Department Staff**

* Manage exam.
* Support generate examination test from the question banks.

**Teacher**

* Import question from some different type.
* Edit question and warning when duplicate.
* Approve test exam quality.

**Leader**

* Manage course.
* Manage chapter.
* Manage learning outcome.

**Student**

* Support doing an exam.
* Get result of exam when submit test exam process has problem by connection, etc.

**IT**

* IT can manage student process

**System Handler**

* Reduced cheating
* Explicit generate test exam, exam, approve process by applying block chain.
* Using authenticate service from FPT University

## Role and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
|  |  |  |  |  |
| 1 | Kiều Trọng Khánh | Project Owner | Supervisor | khanhkt@fe.edu.vn |
|  |  |  |  |  |
| 2 | Nguyễn Minh Hưng | Scrum team members | Leader | hungnmse61794@fpt.edu.vn |
|  |  |  |  |  |
| 3 | Cao Trung Hiếu | Scrum master | Member | hieuctse61801@fpt.edu.vn |
|  |  |  |  |  |
| 4 | Trương Tấn Sang | Scrum team members | Member | sangttse61926@fpt.edu.vn |
|  |  |  |  |  |

Table 2: Role and Responsibilities

## Software Process Model

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

* Our team only has 3 members, and tasks are assigned vertically, do all steps from design, coding, testing, and implementation. Scrum is the most suitable model for the small and medium project.
* In the project, there are many new technologies that need to be learned. With the Scrum model, the team can learn and develop in parallel to meet the deadline.
* The product owner can change the requirement or extend scope. The team will adapt to change better.

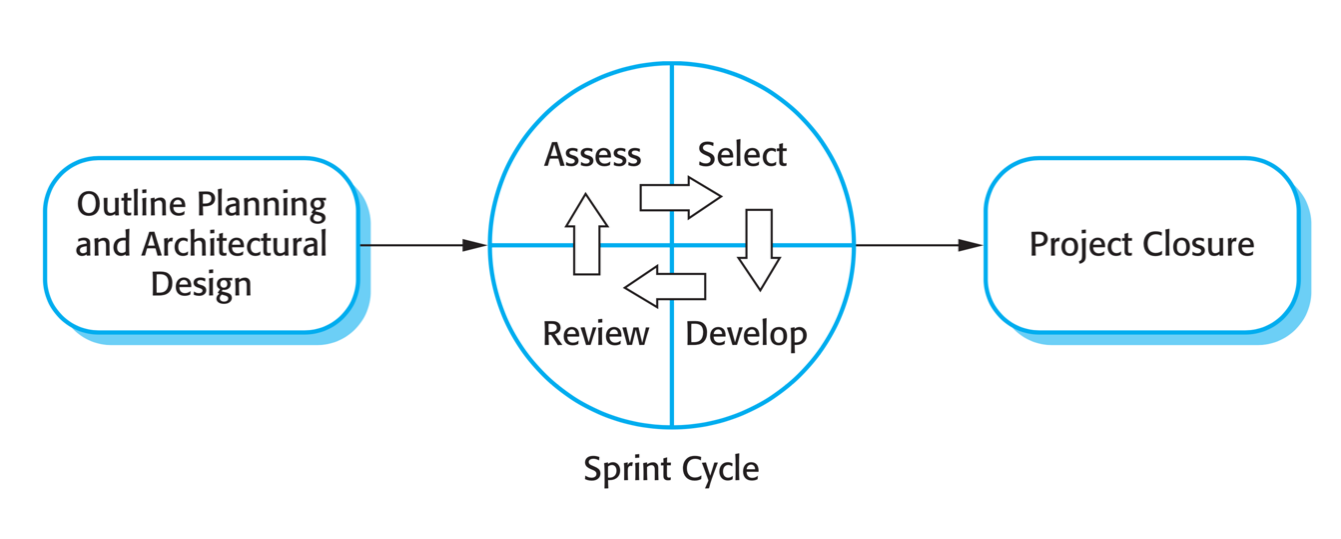


Figure 1 The Scrum Process

*Reference: Software Engineering 9th by Somerville, page 73*

## Conceptual Diagram

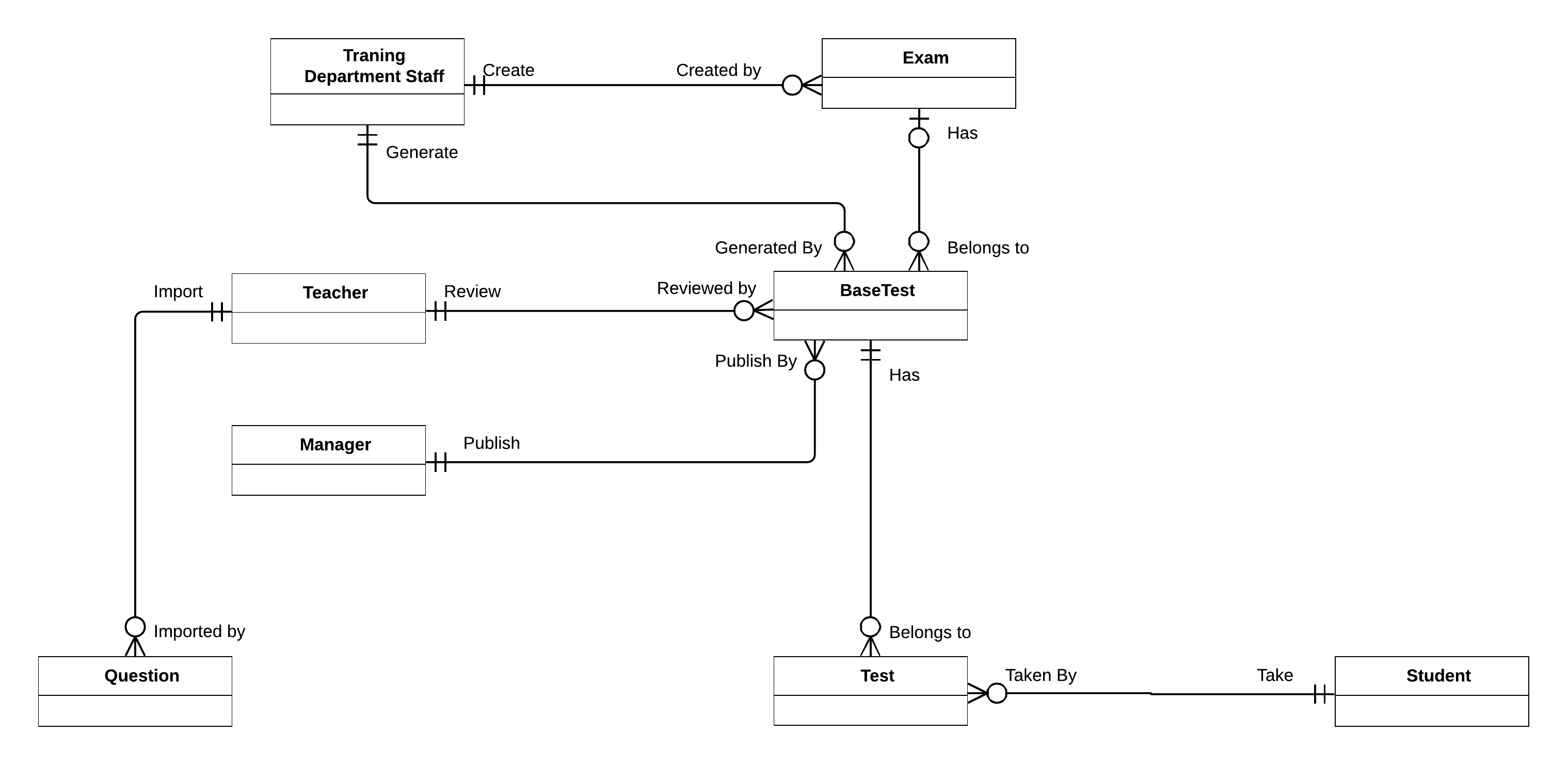


Figure2 Conceptual Diagram

|  |  |
| --- | --- |
| **Entity Name** | **Description** |
| Training Department Staff | The Staff who generate exam, Test exam, semester. |
| Exam | Created by Training department Staff for every end of semester |
| BaseTest | Contain test exam information |
| Teacher | The people who import question and approve test exam |
| Manager | The people who publish test exam |
| Question | Contain questions information |
| Test | Contain student’s test information |
| Student | Contain Student’s information |

Table 4 CONCEPTUAL DIAGRAM EXPLAINATION

## Use-case Diagram

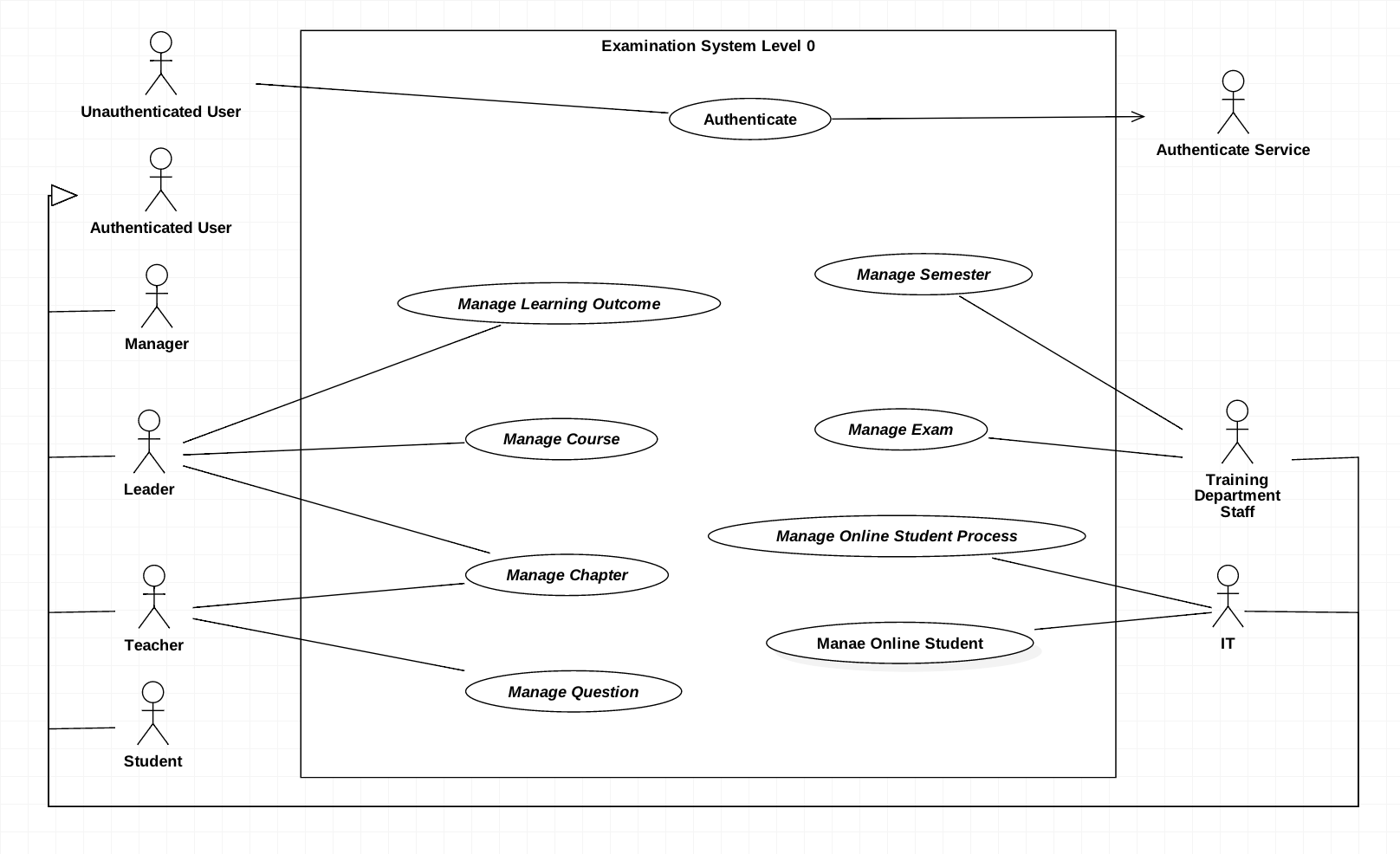
****

Figure 3: Use case overview Examination System

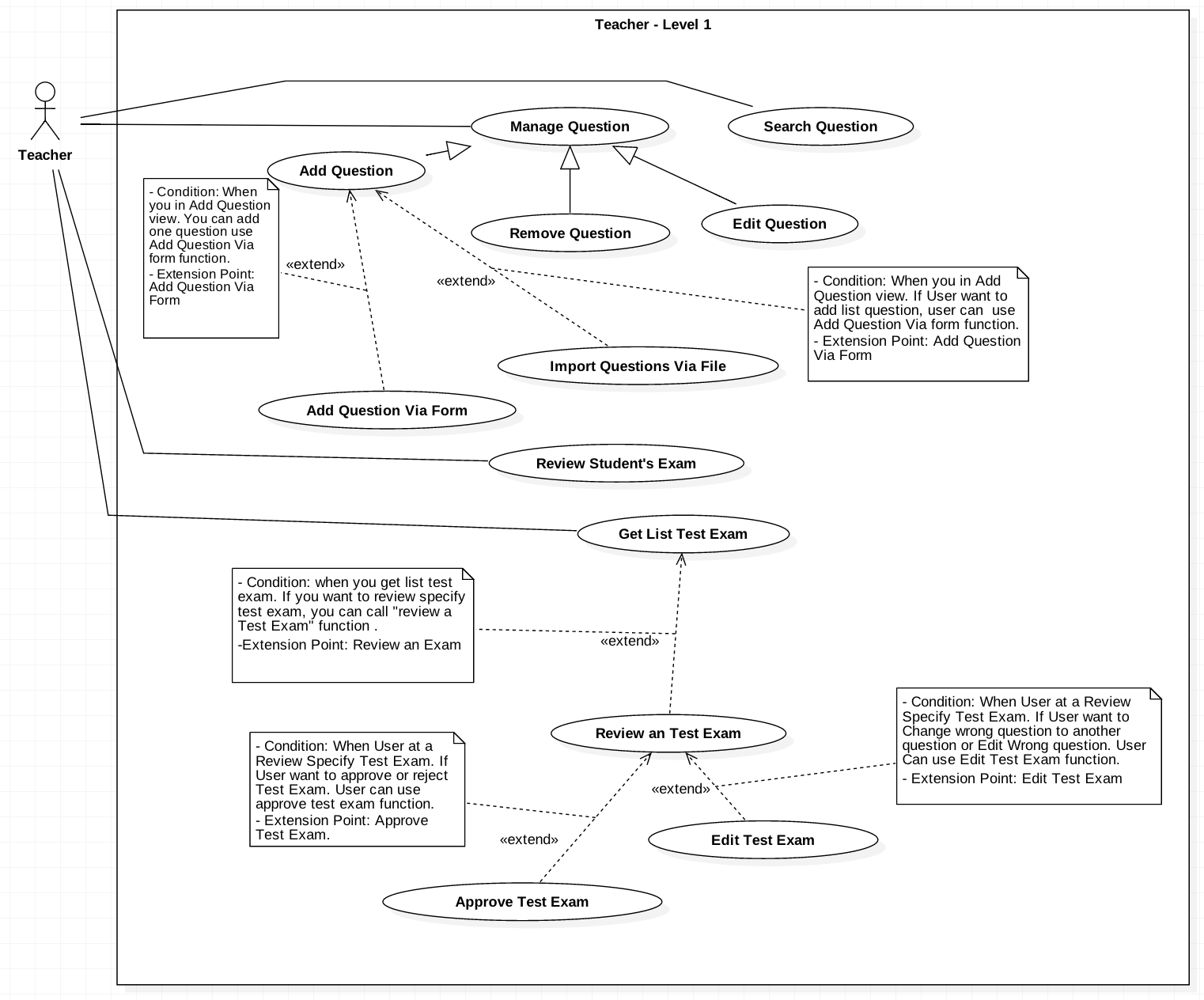


Figure 4: Teacher use case Detail

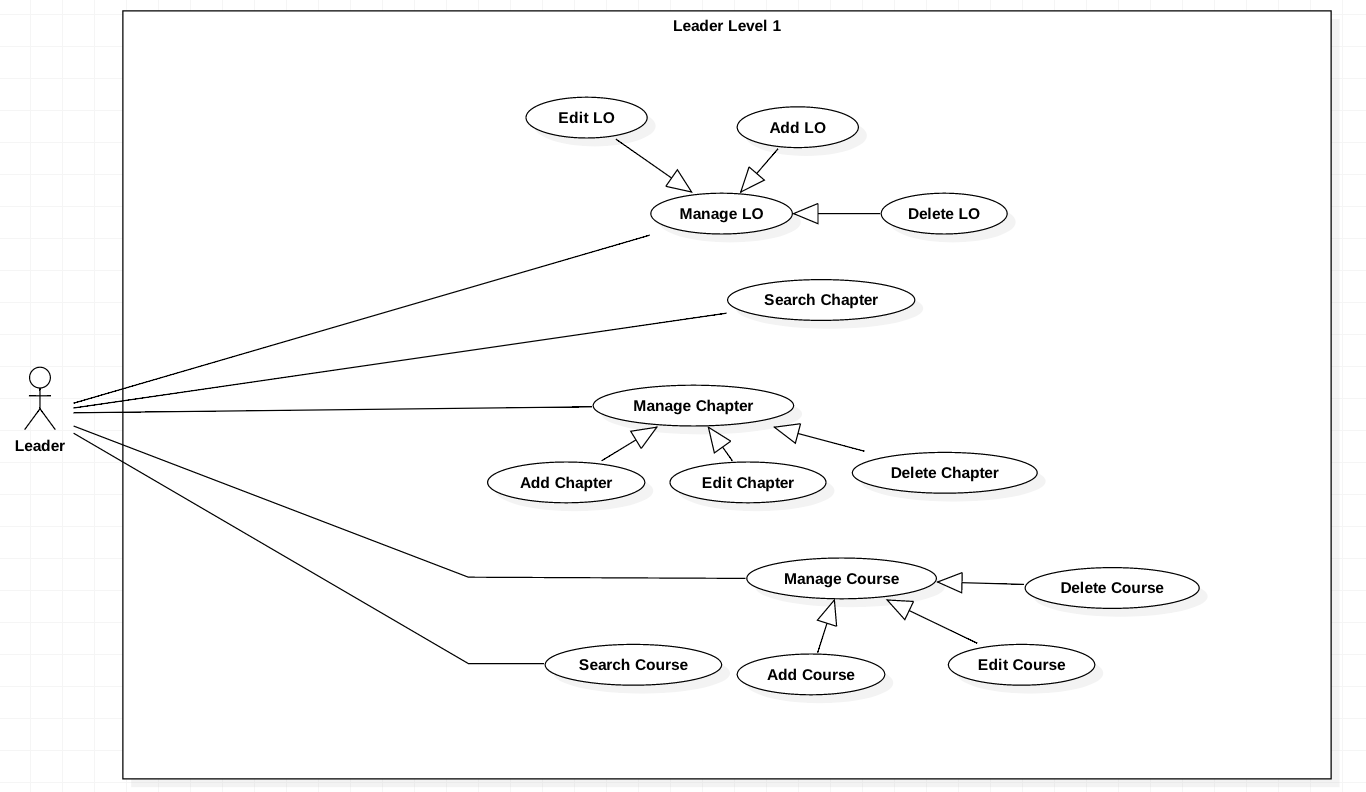
******

Figure 5: Leader use case detail

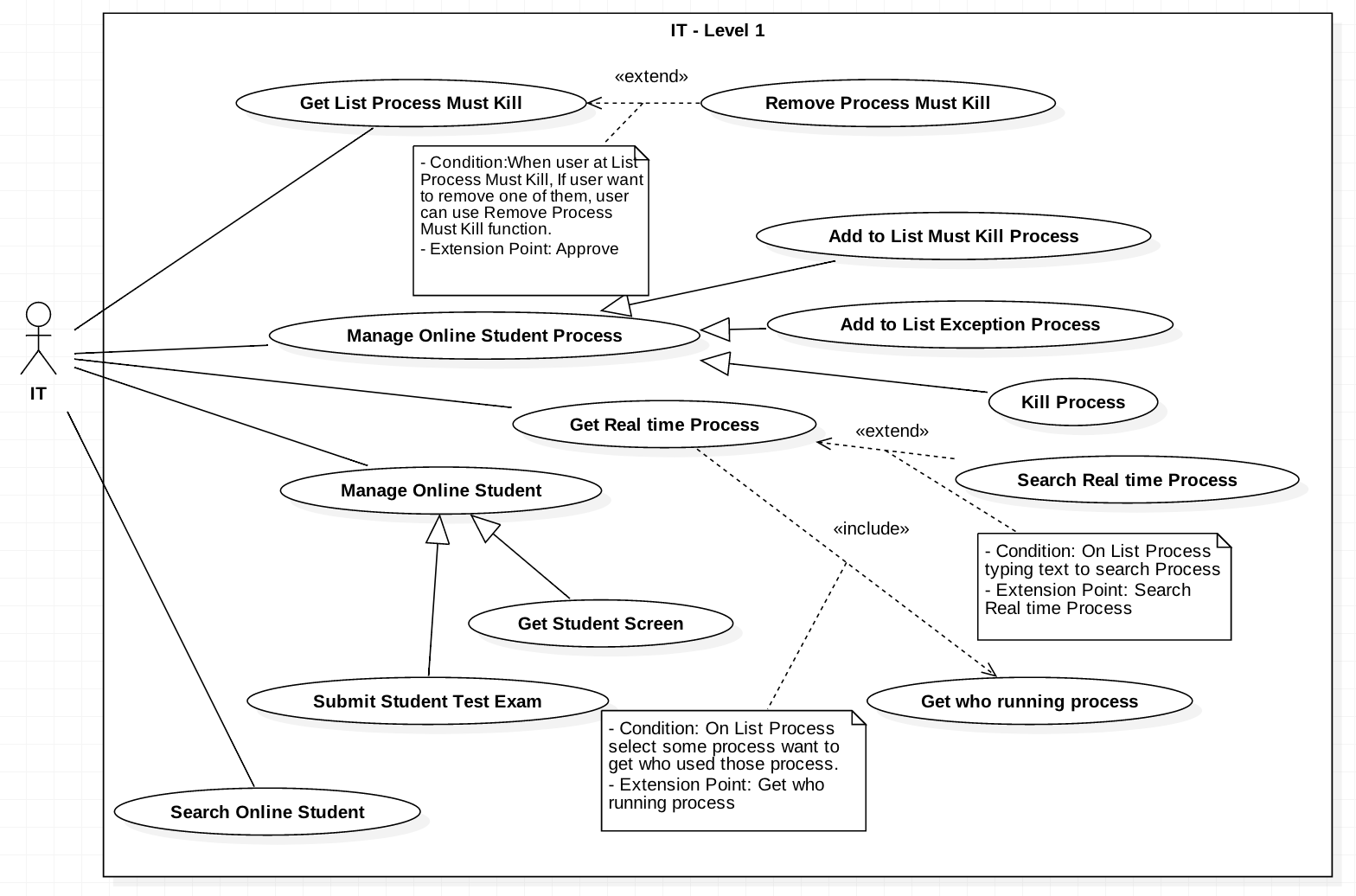
******

Figure 6: IT use case detail

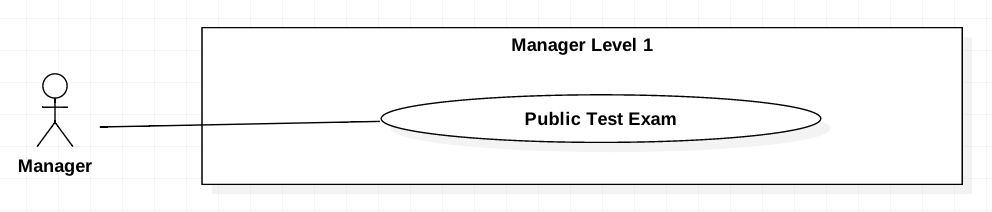


Figure 7: Manager use case detail

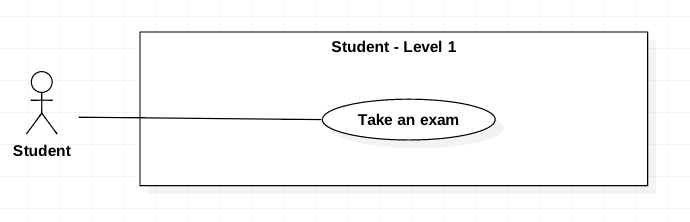
******

Figure 8: Student use case detail

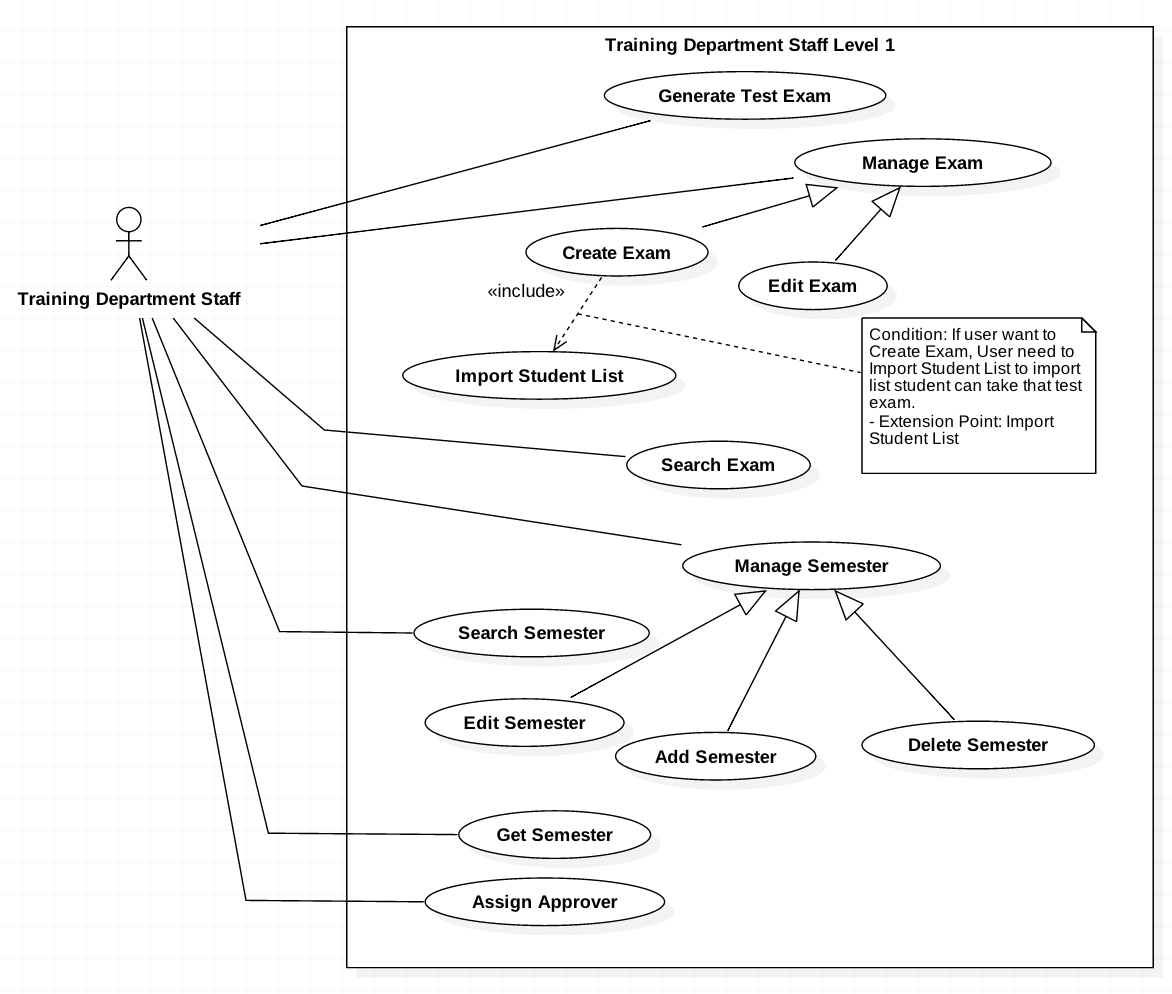


Figure 9: Training Department Staff use case detail

## Use-case Specification

### <IT> Add Must Be Killed Process to List

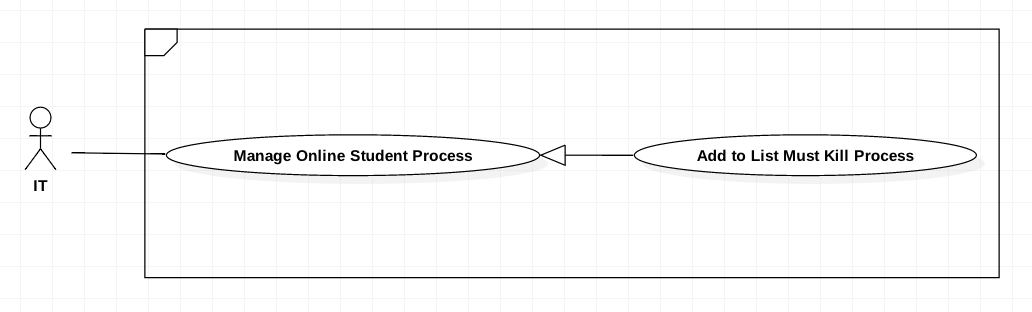


Figure 10 Add Must Be Killed Process to List

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_IT.03** | | | |
| **Use Case No.** | UC\_IT03 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add to list must be killed process | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * *IT*   **Summary:**   * *This use case allows IT can add new process to process must killed list.*   **Goal:**   * *Add a new process to list process must be killed.*   **Trigger:**   * *IT sends create command*   **Preconditions:**   * *IT scan student computers‘s running processes.*   **Post Conditions:**   * *Success: added new records in list process must be killed* * *Fail: Some missing process information*   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *IT goes to list process real time which show student computer process real time.* | *The system will be displayed list process must kill with information:*   * *Process name* * *Application name* * *Process path* | | *2* | *Select the process row which IT want to Add* | *Parse process data to process information format* | | *3* | *IT send add to process must kill command* |  | | *4* |  | *Add new process to processes must killed. System check if process information is missing*  *[Exception]* | | *5* |  | *The system will send Kill Process command to student’s application which have Process Name equal added Process Name* |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **Step** | **Cause** | **System Response** | | *1* | *Process information is missing* | *Show message to IT client* |   **Relationships: N/A**  **Business Rules:**   * *When student start the Exam Client up, IT Managers can get the real time processes of that student (A).* * *Base on A, they can add any of those processes to the must kill list by select process from A (B) and call add to process must kill command.* * *Push B’s information to list process must kill.* * *B’s information contains:*   + *Process Name*   + *Application Name (Application run that process)*   + *Process Path (Path of application running that process)* * *After add processes to list process must kill. Kill all process in A which contain Process Name in B.* | | | |

Table 4 Add Must Be Killed Process to List

### <IT> Kill Process

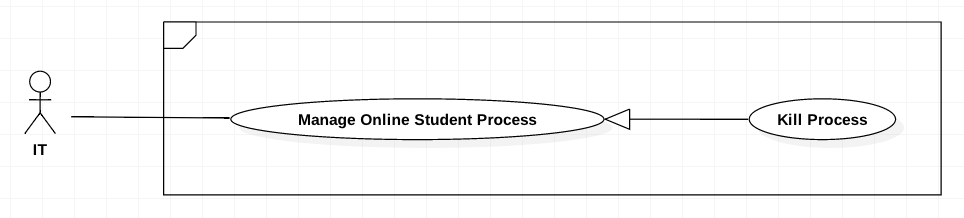


Figure 12 Kill Process

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_IT.05** | | | |
| **Use Case No.** | UC\_IT05 | **Use Case Version** | 2.0 |
| **Use Case Name** | Kill Process | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * *IT*   **Summary:**   * *When the IT want to remove process on student computer, IT can click X button to remove processes on student computer.*   **Goal:**   * *IT can remove process which must be killed they scanned for preventing cheating.*   **Triggers:**   * *IT call command to server.*   **Preconditions:**   * *IT must scan student computer‘s processes.* * *IT must record processes which they defined cheating processes.*   **Post Conditions:**   * *Success: Process is killed on student computer.* * *Fail: Notify an error to IT*   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *IT goes to list process real time which show student computer process real time.* | *The system will be displayed list process must kill with information:*   * *Process Name* * *Application Name* * *Process Path* | | *2* | *Select the process row which IT want to Kill* | *Parse process data to process information format* | | *3* | *IT send Kill Process command* |  | | *4* |  | *Process selected will kill on all student computer if it existed. System check if process information is missing*  *[Exception]* |   **Relationships: N/A**  **Business Rules:**   * *When student start the Exam Client up, IT Managers get the real time processes of that student (A).* * *Base on A, Get all process’s name of selected process by user (B).* * *Kill process in A which have Process Name contain in B.* | | | |

Table 7 Kill Process

### <IT> Get Student Screen

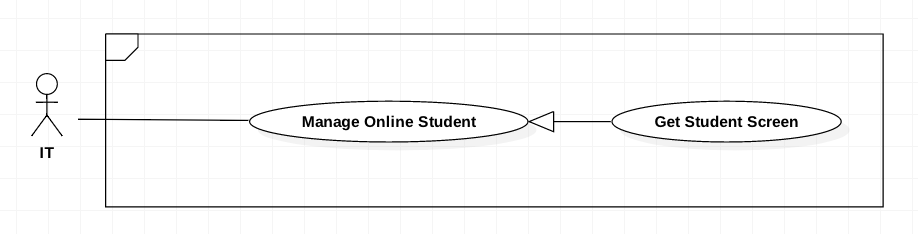


Figure 14 <IT> Get Student Screen

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_IT.08** | | | |
| **Use Case No.** | UC\_IT08 | **Use Case Version** | 2.0 |
| **Use Case Name** | Get student screen | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * IT   **Summary:**   * *This use case allows IT can view student screen.*   **Goal:**   * *IT can prevent cheating when trace back images recorded student screens.*   **Preconditions:**   * *ExamTool must be running in student computer.*   **Post Conditions:**   * *Success: Show student screen on view* * *Fail: show error message*   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *IT goes to list student online and login to ExamTool* | *The system run get all student online with ExamTool command.* | | *2* |  | *Show list student online with ExamTool with information:*   * *Student User* * *Student Name* * *Connection ID* | | *3* | *IT select student want to get screen.* | *The system run get Student screen command with connection Id parameter.* | | *4* |  | *The system request to ExamTool have this connection Id and response continuous capture image.* | | *5* |  | *The system show continuous capture image to screen.* |   **Business Rules:**   * *When student running the ExamTool, IT Managers can get all processes running on that computer (A). Process information contains:*   + *Process Name*   + *Application Name (Application run that process)*   + *Process Path (Path of application running that process)* * Searching value inputted by user (B) * Filter A which Process Name, Application Name, Process Path contain B | | | |

Table 9 <IT> GET STUDENT SCREEN

### <IT> Scan student by process

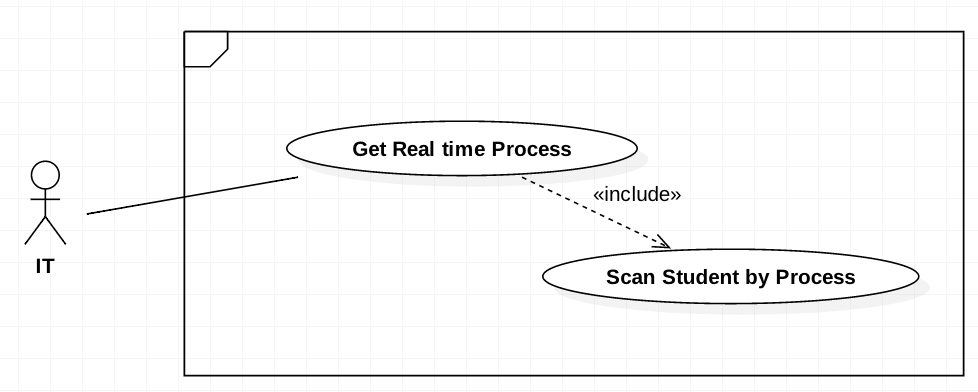


Figure 13 <IT> Scan student by process

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_IT.06** | | | |
| **Use Case No.** | UC\_IT06 | **Use Case Version** | 2.0 |
| **Use Case Name** | Scan student by process | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * *IT*   **Summary:**   * *This use case allows user to get who running selected process*   **Goal:**   * *IT can scan who running selected process for preventing cheating.*   **Triggers:**   * *IT send Scan student by process command.*   **Preconditions:**   * *ExamTool must be running in student computer.*   **Post Conditions:**   * *Success: Get list student use selected process.* * *Fail: Show error message*   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *IT goes to list process real time which show student computer process real time.* | *The system will be displayed list process real time with information:*   * *Process Name* * *Application Name* * *Process Path* * *Student Connection Id* | | *2* |  | *The system will matching list Student in list online student with list process information. Display list student with information*   * *Student Code* * *Student Connection Id* |   **Business Rules:**   * *When student running the ExamTool, IT Managers can get all processes running on that computer (A). Process information contains:*   + *Process Name*   + *Application Name (Application run that process)*   + *Process Path (Path of application running that process)*   + *Student Connection Id of this process* * *Get list selected process by user (B)* * *Get all Student Connection Id and deduplicate B by Student Connection Id (C).* * *Get All Student who using ExamTool (D)* * *Filter D which D’s Student Connection Id contain in C.* | | | |

Table 23 <IT> Scan student by process

### <Training Department Staff> Add new Semester

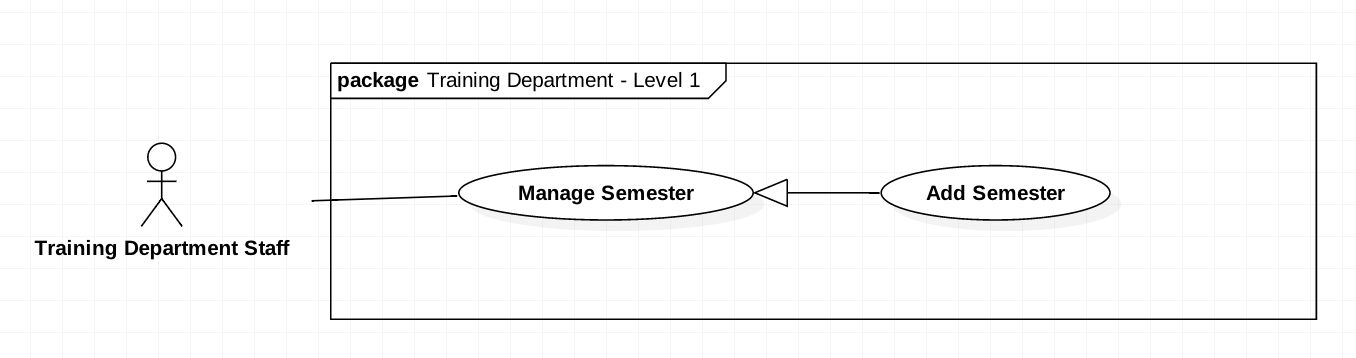


Figure 16 <Training Department Staff> Add new Semester

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_ TrainingDepartmentStaff.11** | | | |
| **Use Case No.** | UC\_ Training Department Staff 11 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add new Semester | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| 1. **Actor:**    * + *Training Department Staff.* 2. **Summary:**    * + *This use case allows Training Department Staff to add a new semester in order to create exams.* 3. **Goal:**    * + *Admin* *can create a new semester with start date and end date so that exams can be created in that semester.* 4. **Triggers:**    * + *Training Department Staff sends a create command to create a new semester.* 5. **Preconditions:** 6. **Post Conditions:**    * + *Success: The new semester will be saved.*      + *Fail: Show error message* 7. **Main Success Scenario:**  |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *Actor sends a create command to create a new semester.* | *The system requires information of the semester:*   * + *Semester code* | | *2* | *Actor input information* |  | | *3* | *Actor sends a command to send* |  |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *The actor does not send a command to add medicine.* | *The system will not display semester information.* | | *2* | *The actor sends a command to cancel.* | *The system will close the create semester view.* |   **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | *1* | *Semester record has existed* | *Show message to notify the actor that the semester has existed in the system.* | | *2* | *The actor doesn’t fill the semester code* | *Show message to notify the actor that the semester code is required.* |   **Relationships: Manage Semester**  **Business Rules:**   * + *Semester name cannot be duplicated*   + *End Date cannot before Start Date*   + *Training Department Staff log into the system, then Staff can create a new Semester with Code, Start Date and End Date.*   + *Training Department Staff need to create a new Semester to create Exam and test exam in this Semester.* | | | |

Table 11 <TRAINING DEPARTMENT STAFF> ADD NEW SEMESTER

### <Teacher> Add Question by Form



Figure 20 <Teacher> Add Question by Form

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_Teacher.20** | | | |
| **Use Case No.** | UC\_Teacher 20 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Question by Form | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * 1. Teacher   **Summary:**   * 1. This use case allows teacher to add questions by form.   **Goal:**   * 1. *Teacher can add question by form.*   **Triggers:**   * 1. *Teacher send an add question by form command.*   **Preconditions:**   * 1. *Login the system with teacher role.*   **Post Conditions:**   * 1. *Success: Questions will be added.*   2. *Fail: Questions will not be added.*  1. **Main Success Scenario:**  |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *Teacher fill the form* | *The system requires teacher to input:*   * + *Chapter*   + *Level*   + *Mark*   + *Question Content*   + *Options* | | *2* | *Teacher send a save command to save the questions locally* |  | | *3* | *Teacher send a send command add question.* | *The system will run command and question and comeback to question list.*  *[Exception]* |   **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | *1* | *Teacher not fill full question info* | *Show message to fill full form.* | | *2* | *Teacher not choose write answer for question.* | *Show message require answer* |  1. **Alternative Scenario:**  |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *Teacher add question with content matching >80% question already in database* | *Show warning may question already in database and confirm continuous to add or cancel action.* |   **Relationships: Add Question**  **Business Rules:**   * + - *Teacher fill question form (A).*     - *A has format:*  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Content | Options | Mark | Level | Chapters | Learning outcomes |  * + - *Add A to question list.* | | | |

Table 15 <TEACHER> ADD QUESTION BY FORM

### <Teacher> Add Question by Import File

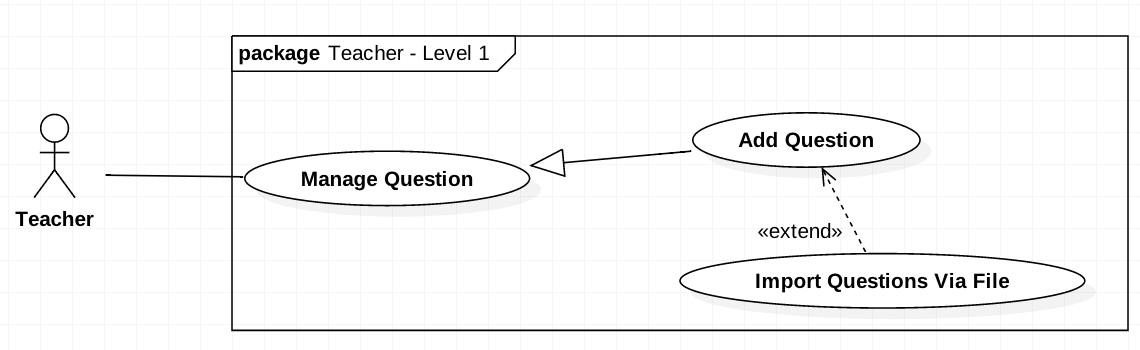


Figure 21 <Teacher> Add Question by Import File

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_Teacher.21** | | | |
| **Use Case No.** | UC\_Teacher 21 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Question by Import File | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * *Teacher*   **Summary:**   * *This use case allows teacher to add questions by uploading file.*   **Goal:**   * *Upload multiple question list exported by Moodle (old system)*   **Triggers:**   * *Teacher send an add questions command.*   **Preconditions:**   * *Login the system with teacher role.*   **Post Conditions:**   * *Success: Questions will be added.* * *Fail: Questions will not be added.*   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | |  |  |  | | *1* | *Teacher upload the file* | *The system requires teacher to upload a file with GIFT format or file extensions: .doc,.xml* | |  |  | *System run command to validate and check duplicate question.* | | *2* | *Teacher review the uploaded questions* | *Show question format status:*   * + *Success*   + *Warning*   + *Error* | | *3* | *Teacher send a submit command to save the questions to the database* | *System run add questions to add question list* |   **Alternative Scenario: N/A**  **Exceptions: N/A**  **Relationships: Add Question**  **Business Rules:**   * + - *Uploaded a file (A).*     - *Validate A by checking file type (.xml, .docx, .txt) and file format (GIFT format)(B)*     - *If B valid, parse B to question list.*     - *Question information has format:*  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Content | Options | Mark | Level | Chapters | Learning outcomes | | | | |

Table 16 <Teacher> Add Question by Import File

### <Teacher> Edit Questions

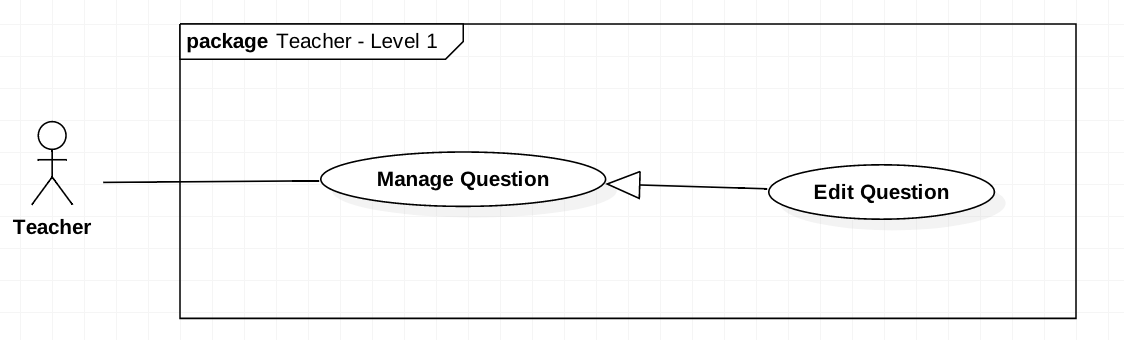


Figure 22 <Teacher> Edit Questions

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_Teacher.23** | | | |
| **Use Case No.** | UC\_Teacher 23 | **Use Case Version** | 2.0 |
| **Use Case Name** | Edit Questions | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * *Teacher*   **Summary:**   * *This use case allows teacher to edit questions.*   **Goal:**  *Edit question content, answer, percent when student check wrong option, level... of question*  **Triggers:**   * *Teacher send an edit question command.*   **Preconditions:**   * *Login the system with teacher role.*   **Post Conditions:**   * *Success: Questions will be edited.* * *Fail: Questions will not be edit and show message error*  1. **Main Success Scenario:**  |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *Teacher goes to list question view* | *The system will display list question information:*   * + *Course*   + *Mark*   *Level* | | *2* | *Teacher choose to view detail question* | *The system will display detail information of this question:*   * + *Question code*   + *Question content*   + *Option*   + *Learning Outcome*   + *Chapter*   + *Mark*   + *Percent*   + *Level*   + *Right answer* | | *3* | *Teacher edit question information* | *The system require fill full info except learning outcome.* | | *4* | *Teacher send a submit command to save the questions.* | *The system run save question command to save and comeback to list question.* |   **Alternative Scenario: N/A**  **Exceptions: N/A**  **Relationships: Manage Question**  **Business Rules:**  *- Get the question base on question’s code (A)*  *- Update A by user input as format :*   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Content | Options | Mark | Learning outcomes | Chapters | Level |   *- Save A to system.* | | | |

Table 17 <Teacher> Edit Questions

### <Teacher> Approve Test Exam

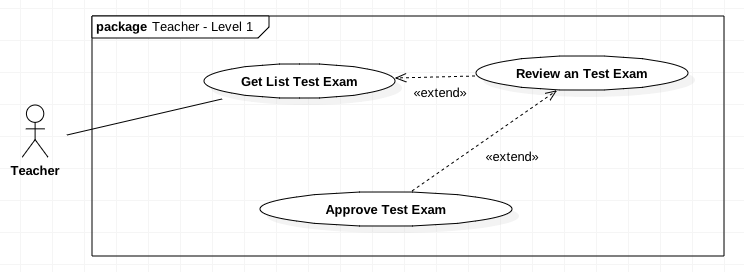


Figure 23 <Teacher> Approve Test Exam

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_Teacher.25** | | | |
| **Use Case No.** | UC\_Teacher 25 | **Use Case Version** | 2.0 |
| **Use Case Name** | Approve Test Exam | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * *Teacher*   **Summary:**   * *This use case allows teacher to approve an exam assigned to that teacher.*   **Goal:**   * *Approve test exam quality to storage in test exam bank*   **Triggers:**   * *Teacher send an approve exam command.*   **Preconditions:**   * *Login the system with teacher role.*   **Post Conditions:**   * *Success: Exam status will be changed to “Approved”.* * *Fail: Exam will still be “Waiting”.*  1. **Main Success Scenario:**  |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *Teacher goes to Approve Test Exam View* | *The system will display list test exam information assigned to that teacher:*   * + *Course*   + *Number of questions*   + *Duration* | |  | *Teacher send command to view detail Test Exam* | *The system will display list question and require teacher take that exam.* | | *3* | *Teacher take an exam* |  | | *4* | *Teacher submit test exam* | *The system will calculate mark and change Test Exam status to taken and display Test Exam result to interface with information:*   * + *Mark*   + *Test Exam Result* | | *5* | *Teacher send command approve/ reject test exam.*  *[Alternative Scenario 1]* | *The system will change Test Exam to approved or rejected and come back display list exam assigned to approve.* |  1. **Alternative Scenario:**  |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *Teacher send command edit questions* | *The system will require teacher to retake exam* |   **Exceptions: N/A**  **Relationships: Extend “Review an test exam”**  **Business Rules:**   * *Get test exam base on code (A).*   + *If A’s status is “Pending”, teacher take an exam. After A finished, A’s status change to “Taken”.*   + *If A’s status is “Taken:*     - *If teacher update A, A’s status change to “Edited”.*     - *If teacher approve. A’s status change to “Approved”.*     - *If teacher reject, A’s status change to “Rejected”.*   + *If A’s status is “Edited”, teacher available to retake and A’s status change to “Pending”.* | | | |

Table 18 <Teacher> Approve Test Exam

### <Training Department Staff> Generate Test Exam

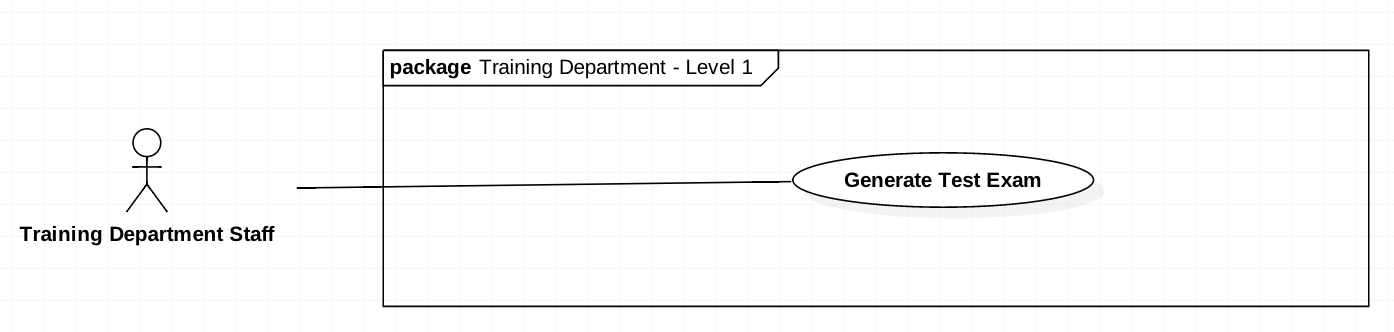


Figure 24 <Teacher> generate Test Exam

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_Training Department Staff.25** | | | |
| **Use Case No.** | UC\_ Training Department Staff 25 | **Use Case Version** | 2.0 |
| **Use Case Name** | Generate Test Exam | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * + *Training department staff*   **Summary:**   * + *This use case allows training department staff to generate test exam*   **Goal:**   * + *Generate new Test Exam to Test Exam bank.*   **Triggers:**   * + *Training Department Staff call Generate Test Exam command*   **Preconditions:**   * + *Login the system with “Training department staff” role.*   **Post Conditions:**   * + *Success: Generate new Test Exam*   + *Fail: Test exam not be generated and show message error*   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *Manager click the Approve/Reject Exam button in the menu* |  | | *2* | *Manager click the Approve/Reject Exam tab* |  | | *3* | *Teacher select the semester* |  | | *4* | *Teacher send a command to get list of exam* | *Show list of exam with:*  *- Course*  *- Start Date*  *- End Date*  *- Duration*  *- Status*  *- Actions* | | *5* | *Teacher click “More Detail” button of an exam in the Actions column* |  | | *6* | *Teacher click the “Submit” button* |  | | *7* | *Teacher click the “Approve” button* |  | | *8* | *Teacher send an approve exam command* |  |   **Alternative Scenario: N/A**  **Exceptions: N/A.**  **Relationships: N/A.**  **Business Rules:**   * 1. *Get user’s input as format (A):* * *CourseCode* * *Chapters* * *Questions* * *LearningOutcomes* * *NumberOfTest* * *NumberOfQuestions* * *Duration* * *Type*   1. *If A’s type is “Manufacturing”, system create test exam base on A’s questions.*   2. *If A’s type is “LearningOutcomes”, system generate test exam base on A’s LearningOutcomes.*   3. *If A’s type is “Chapters”, system generate test exam base on A’s Chapters.* | | | |

Table 19 <Training department staff> Generate Test Exam

### <Training Department Staff> Assign Approver

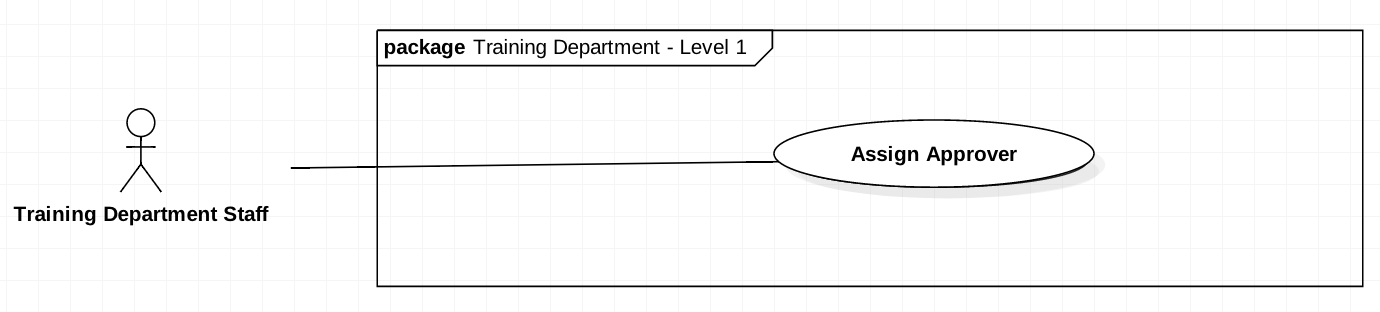


Figure 25 < Training Department Staff > Approve Test Exam

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE-UC\_Training Department Staff.25** | | | |
| **Use Case No.** | UC\_ Training Department Staff 25 | **Use Case Version** | 2.0 |
| **Use Case Name** | Assign Approver | | |
| **Author** | HieuCT | | |
| **Date** | 19/11/2018 | **Priority** | Normal |
| **Actor:**   * + *Training department staff*   **Summary:**   * + *This use case allows training department staff to assign the teacher to approve test exam.*   **Goal:**   * + *Assign Test Exam to teacher who approve that test exam*   **Triggers:**   * + *Training Department Staff call Set approver command*   **Preconditions:**   * + *Login the system with “Training department staff” role.*   **Post Conditions:**   * + *Success: Assign test exam to approver*   + *Fail: Can’t assign test exam to approve*   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | *1* | *Training Department Staff goes to Assign Approver view* | *System require complete information:*   * + *Course*   + *Test Exam*   + *Start Date to approve*   + *End Date to take Exam*   + *Approver* | | *2* | *Training Department Staff full fill information* |  | | *3* | *Teacher run Assign Approver command* | *The system run Assign Approver command*  *[Exception]* |   **Exception:**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | *1* | *Staff not full fill form* | *Show error message* |   **Alternative Scenario: N/A**  **Relationships: N/A.**  **Business Rules:**   * 1. *Get test exam base on code (A).*   2. *Update A by user’s input as format:*  |  |  |  | | --- | --- | --- | | Approver | StartDate | EndDate |  * 1. *System save A.* | | | |

Table 20 <Training department staff> Assign Approver

## System Architecture Design

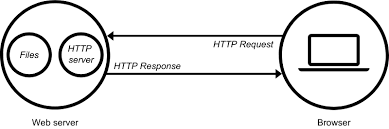


Figure 26 System Architecture Design

### Web Server Architecture Design

Overall architecture of Examination Tool System with functional layers and the collaboration between the system and the external systems is shown in figure 3 below.

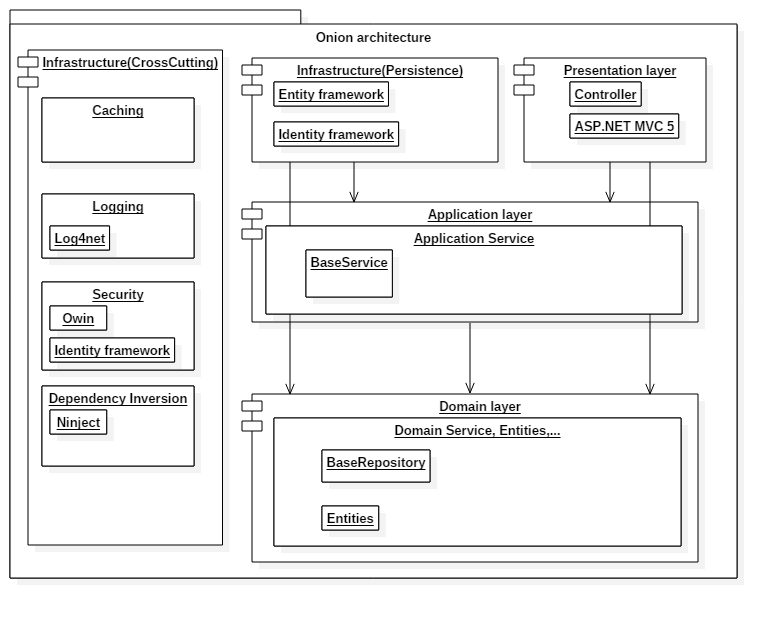


Figure 27 Architecture Overview

|  |  |  |
| --- | --- | --- |
| **No.** | **Layer Name** | **Description** |
| 1 | Presentation layer | Contains UI, handle request |
| 2 | Application layer | Provides services for application |
| 3 | Domain layer | Provides full services, stores entities |
| 4 | Infrastructure layer | Communicate with other framework, work with database |

Table 21 ARCHITECTURE OVERVIEW DESCRIPTION

### Client Architecture Design

In Web Application, the system is developed under MVC architecture. We choose this architecture because of the following advantage:

• The Model-View-Controller pattern highly supports the separation of concerns. This advantage not only increases the testability of the code but it also makes it easier to extend, allowing a fairly easy implementation of new features.

• If the Views respect the single responsibility principle then their role is just to update the Controller for every user event and just display data from the Model, without implementing any business logic. In this case, UI tests should be enough to cover the functionalities of the View.

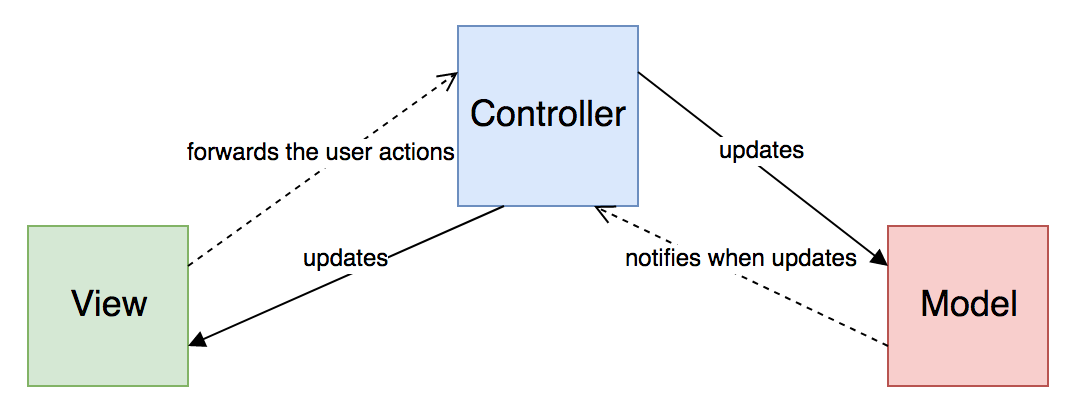


Figure 28 Web Admin MVC Architecture

(Reference: https://medium.com/swlh/ios-design-patterns-a9bd07818129)

* ***Model*:** which represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface.
* ***View*:** which is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth)
* ***Controller*:** which represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.

## Component Diagram

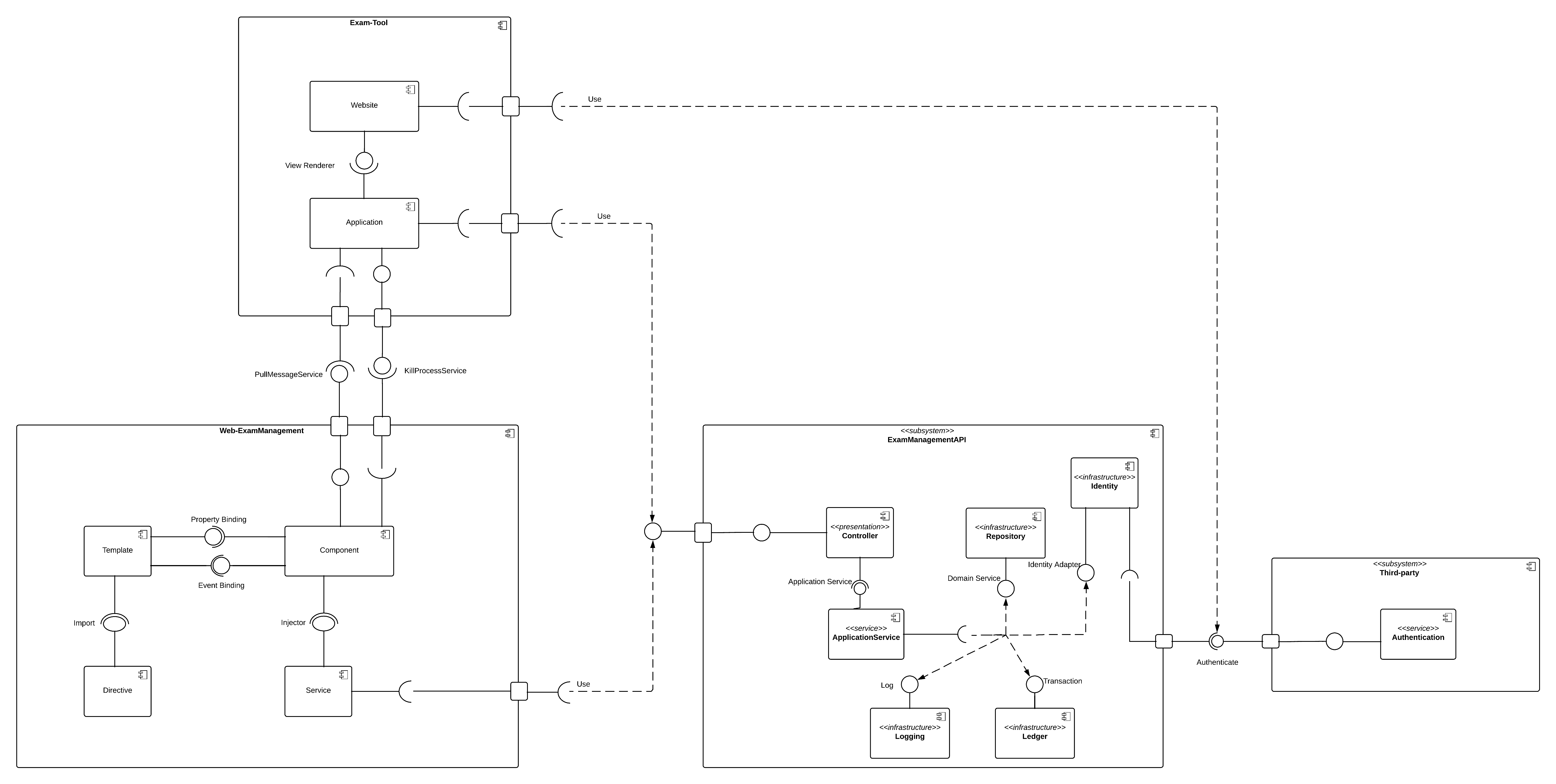


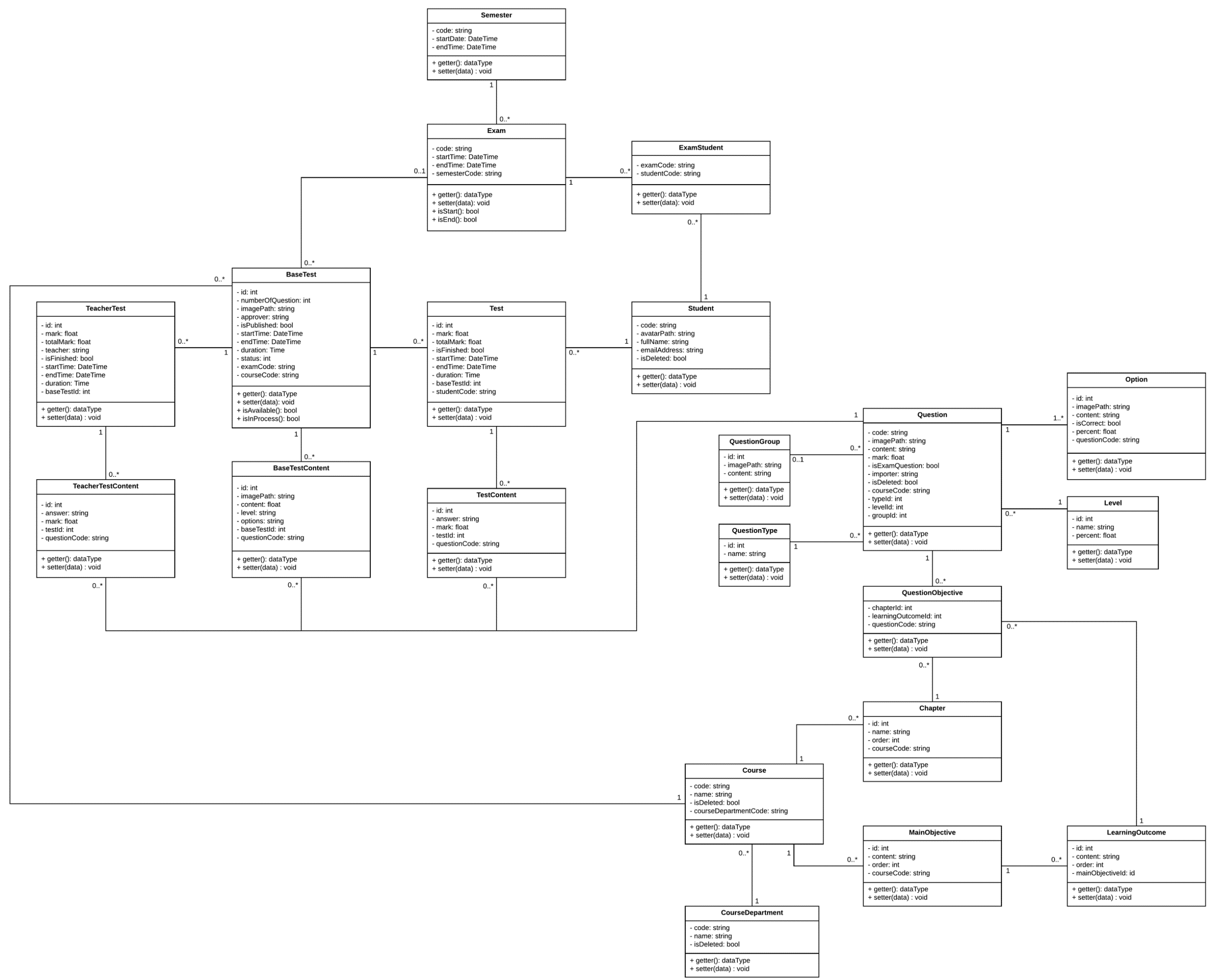
Figure 29 Component Diagram

|  |  |
| --- | --- |
| **Entity Name** | **Description** |
| Exam-tool | This subsystem provide tool help Student can take a test exam and through this tool IT can manage process on student computer. This subsystem include 2 main component:   * 1. Website take a test exam   2. Application |
| Website | Provide interface for student take a test exam |
| Application | Provide tool for IT can collect process on student’s computer and send to IT management website. It can screenshot student screen. |
| Web-Exam Management | This web build for manage and build by angular with angular structure. This website use API provide by ExamManagementAPI subsystem |
| Authenticate service | This service provide by FPT University. (Login, Get User info…) |
| ExamManagementAPI | Provide API for System. |

Table 22Component diagram dictionary

## Detailed Description

### Class Diagram



|  |  |  |
| --- | --- | --- |
| **Entity Data dictionary: describe all content of all entities** | | |
| **Class Name** | **Mapping column with conceptual diagram** | **Description** |
| BaseTest | BaseTest | Contain the test exam information |
| Semester | N/A | Contain the Semester information |
| Exam | Exam | Contain the exam information |
| ExamStudent | N/A | Contain relationship between Student and Exam, this describe the accessible to the test exam of student |
| Student | Student | Contain the Student’s information |
| Test | Test | Contain student’s Test exam information |
| TeacherTest | N/A | Contain teacher’s test exam information (because teacher need to take exam to approve test exam) |
| TeacherTestContent | N/A | Contain answer of each teacher’s question |
| BaseTestContent | N/A | Contain test exam’s question information |
| TestContent | N/A | Contain Student’s answer information |
| Question | Question | Contain question information (question bank) |
| QuestionGroup | N/A | Contain some group question like reading question which contain some question in reading question |
| Option | N/A | Contain option can choice of each question |
| QuestionType | N/A | Type of question information (ex: single choice, multiple choice) |
| Level | N/A | Contain some level of question (hard, easy …) |
| QuestionObjective | N/A | Question corresponding with LO |
| Chapter | N/A | Contain chapter information |
| Course | N/A | Contain Course information |
| MainObjective | N/A | Contain Purpose of course information |
| LearningOutcome | N/A | Decompose of big Main Objective |
| CourseDepartment | N/A | Contain Course Department information |

Table 23 Class diagram Description

### Class Diagram Explanation

#### Semester

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Code | string | private | Code of Semester (ex: Fall 2018) |
| StartDate | DateTime | private | Start Date of semester |
| EndDate | DateTime | private | End Date of semester |

Table 24 Class Semester Explanation

#### Exam

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Code | string | private | Code of Exam (ex: FinalX) |
| StartDate | DateTime | private | Start Date of Exam |
| EndDate | DateTime | private | End Date of Exam |
| SemesterCode | string | private | Semester Of that Exam |
| CourseCode | string | private | Course of that Exam |

Table 26 Class exam Explanation

#### ExamStudent

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ExamCode | string | private | Code of Exam (ex FinalX) |
| StudentCode | string | private | Code of a Student (ex SE61801) |

Table 28 Class ExamStudent Explanation

#### Student

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Code | string | private | Code of a Student (ex SE61801) |
| AvatarPath | string | private | Relative image path of Student Avatar |
| FullName | string | private | Student’s Full Name |
| EmailAddress | string | private | Student’s Email |
| isDeleted | boolean | private | Flag to know is this student deleted (true = is deleted) |

Table 30 CLASS STUDENT EXPLANATION

#### BaseTest

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Base Test Id |
| NumberOfQuestion | int | private | Number question of this Test Exam |
| ImagePath | string | private | Relative folder storage image of base test. |
| ExamCode | string | private | Exam of that base test |
| CourseCode | string | private | Course of that base test |
| StartTime | DateTime | private | Start time for student take test exam |
| EndTime | DateTime | private | End time for student take test exam |
| Status | int | private | Status of that base test (approved, rejected, Edited…) |
| Approver | string | private | The people who approve quality of that base test |
| IsPublished | boolean | private | Is that base test published for student |

Table 32 CLASS BASETEST EXPLANATION

#### BaseTestContent

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Question Id of base test |
| ImagePath | string | private | Relative folder storage image of base test content. |
| Content | string | private | Content of that question |
| Level | string | private | Level of question (describe how question hard) |
| BaseTestId | int | private | Base test of that question |
| QuestionCode | string | private | Code of question (ex PRX-L01) |
| Options | string | private | Json string list option can choice in that question |

Table 34 CLASS BASETESTCONTENT EXPLANATION

#### Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Id Student’s Test Exam |
| Mark | double | private | Mark’s Student Test Exam |
| TotalMark | double | private | Total mark of that test exam |
| isFinished | boolean | private | Is student done this test exam |
| StudentCode | string | private | Student who take this test exam |
| StartTime | DateTime | private | Time when student take test exam |
| EndTime | DateTime | private | End time when student take test exam |
| Duration | Time | private | Time of student take that test exam |
| BaseTestId | int | private | Base test of that test exam |

Table 36 CLASS TEST EXPLANATION

#### TestContent

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Id Student’s answer |
| TestId | int | private | Test id of this Student’s answer |
| QuestionCode | double | private | Question code of this Student’s answer |
| Answers | string | private | Json string list student answer of this question |
| Mark | double | private | Mark of this answer in this question |

Table 38 CLASS TESTCONTENT EXPLANATION

#### Question

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Code | string | private | Question Code |
| ImagePath | string | private | Relative path storage image of question |
| Content | string | private | Question content |
| LevelId | int | private | Level of question (hard, easy…) |
| GroupId | int | private | Mark of this answer in this question |
| IsExamQuestion | boolean | private | Is question already in test exam |
| CourseCode | string | private | Course of this question |
| Mark | double | Private | Mark of this question |
| IsDeleted | boolean | Private | Is this question deleted |
| TypeId | int | Private | Type of question (ex: single choice, multiple choice, matching…) |
| importer | string | Private | The teacher who import this question |

Table 40 CLASS QUESTION EXPLANATION

#### Option

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Id of this option |
| ImagePath | string | private | Relative path storage image of option |
| Content | string | private | Option content |
| IsCorrect | boolean | private | Is this option correct |
| QuestionCode | string | private | Question Code |
| Percent | double | Private | Percent will Minus if choice wrong |

Table 42 CLASS OPTION EXPLANATION

#### Level

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Level Id |
| Name | string | private | Level name (ex: hard, easy) |
| Percent | Float | private | How important the question is |

Table 44 CLASS level EXPLANATION

#### QuestionType

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Type Id |
| Name | string | private | Type name (ex: single choice, multiple choice) |

Table 46 CLASS QUESTIONTYPE EXPLANATION

#### QuestionGroup

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Question Group Id |
| ImagePath | string | private | Relative Path of this question group |
| Content | String | private | Content of this question group |

Table 48 CLASS QUESTIONGROUP EXPLANATION

#### QuestionObjective

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ChapterId | int | private | Chapter Id |
| LearningOutcomeId | int | private | Learning Outcome Id |
| QuestionCode | string | private | Question |

Table 50 CLASS QUESTIONOBJECTIVE EXPLANATION

#### Chapter

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Chapter Id |
| Name | string | private | Name of chapter (ex: chapter 1) |
| CourseCode | string | private | Code of course |
| Order | int | private | Order of chapter (1, 2…) |

Table 52 CLASS CHAPTER EXPLANATION

#### Course

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Code | string | private | Course Code |
| Name | string | private | Course Name |
| CourseDepartmentCode | string | private | Department of course |
| IsDeleted | boolean | private | Is this course deleted |

Table 54 CLASS COURSE EXPLANATION

#### MainObjective

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Main Objective Id |
| Content | string | private | Main objective information |
| CourseCode | string | private | Course of this main objective |
| Order | int | private | Order of this main objective |

Table 56 CLASS MAINOBJECTIVE EXPLANATION

#### LearningOutcome

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Learning Outcome Id |
| Content | string | private | Learning outcome information |
| CourseCode | string | private | Big main objective |
| Order | int | private | Order of this Learning outcome |

Table 58 CLASS LEARNINGOUTCOME EXPLANATION

#### TeacherTest

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Teacher Test Id |
| Mark | float | private | Mark of this test exam |
| TotalMark | Float | private | Total mark of this test exam |
| Teacher | string | private | Get from FPT Authenticate Service |
| IsFinished | boolean | private | Describe: is this test exam finished |
| StartTime | DateTime | private | Time teacher start to take test exam |
| EndTime | DateTime | private | Time teacher End take test exam |
| Duration | Time | private | Duration teacher take that test exam |
| BaseTestId | int | Private | Base test id |

Table 60 CLASS TeacherTest EXPLANATION

#### TeacherTestContent

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | int | private | Teacher Test Content Id |
| Answer | string | private | Answer of question |
| Mark | Float | private | Mark of this answer |
| TestId | int | private | Test Exam of this answer |
| QuestionCode | boolean | private | Question of this answer |

Table 63 Class teachertestcontent explanation

## Database Design

### Entity Relationship Diagram

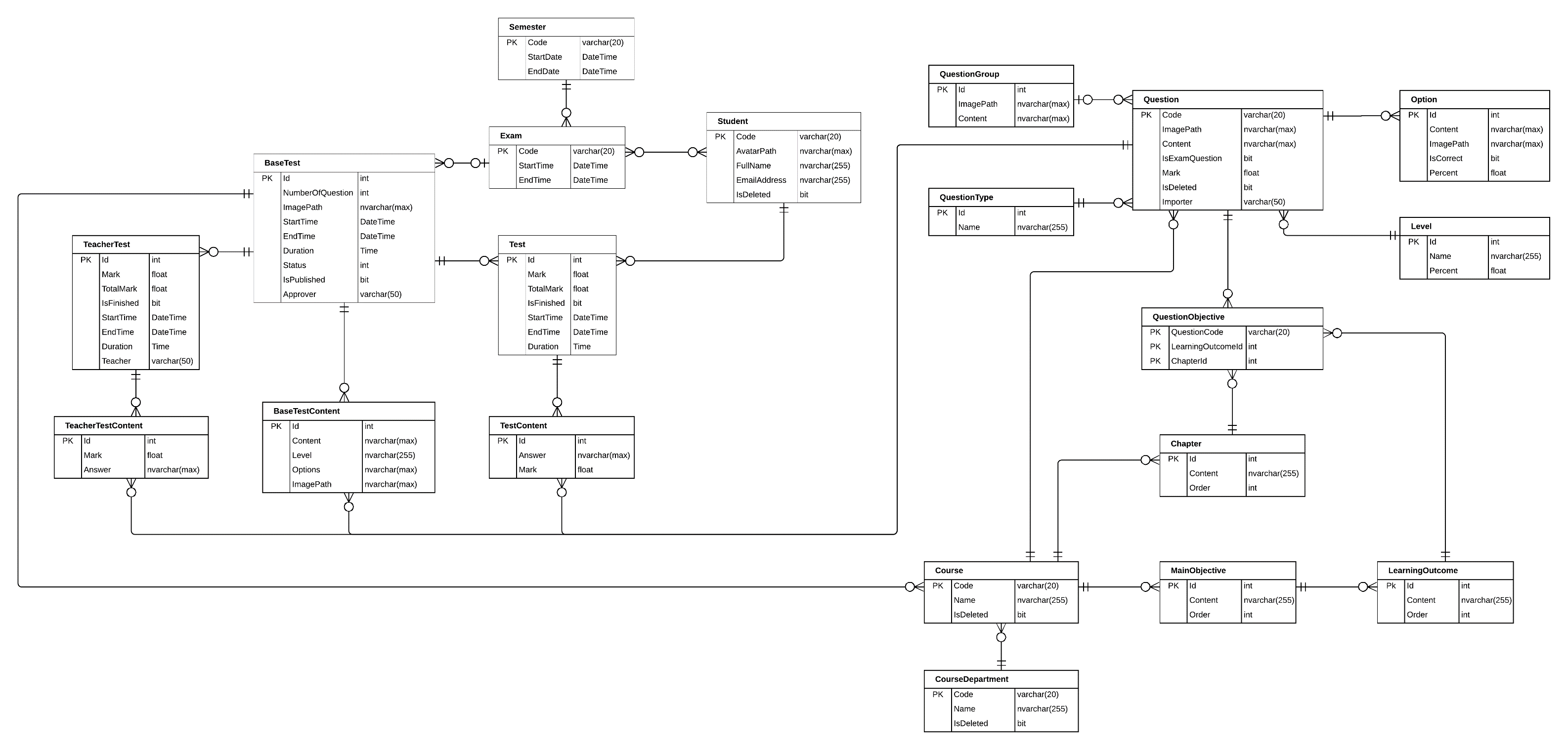


Figure 32 ER DIAGRAM

### Data Dictionary

|  |  |
| --- | --- |
| **Entity Name** | **Description** |
| BaseTest | Contain the test exam information |
| Semester | Contain the Semester information |
| Exam | Contain the exam information |
| ExamStudent | Contain between Student and Exam, it describe the accessible to the test exam of student |
| Student | Contain the Student information |
| Test | Contain student Test exam information |
| TeacherTest | Contain teacher test exam information (because teacher need to take exam to approve test exam) |
| TeacherTestContent | Contain answer of each teacher’s question |
| BaseTestContent | Contain test exam’s question information |
| TestContent | Contain Student’s answer information |
| Question | Contain question information (question bank) |
| QuestionGroup | Contain some group question like reading question which contain some question in reading question |
| Option | Contain option can choice of each question |
| QuestionType | Type of question information (ex: single choice, multiple choice) |
| Level | Contain some level of question (hard, easy…) |
| QuestionObjective | Question corresponding with LO |
| Chapter | Contain chapter information |
| Course | Contain Course information |
| MainObjective | Contain Purpose of course information |
| LearningOutcome | Decompose of big Main Objective |
| CourseDepartment | Contain Course Department information |

Table 64 ER Dictionary

### ERD Explanation

#### Semester

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | varchar(20) | Code of Semester (ex: Fall 2018) |
| StartDate | Datetime | Start Date of semester |
| EndDate | Datetime | End Date of semester |

Table 65 Semester table Description

#### Exam

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | varchar(20) | Code of Exam (ex: FinalX) |
| StartDate | Datetime | Start Date of Exam |
| EndDate | Datetime | End Date of Exam |

Table 66 Exam table Description

#### Student

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | varchar(20) | Code of a Student (ex SE61801) |
| AvatarPath | nvarchar(Max) | Relative image path of Student Avatar |
| FullName | nvarchar(255) | Student’s Full Name |
| EmailAddress | nvarchar(255) | Student’s Email |
| IsDeleted | bit | Flag to know is this student deleted (true = is deleted) |

Table 67 Student table Description

#### BaseTest

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Base Test Id |
| NumberOfQuestion | int | Number question of this Test Exam |
| ImagePath | nvarchar(max) | Relative folder storage image of base test. |
| StartTime | DateTime | Start time for student take test exam |
| EndTime | DateTime | End time for student take test exam |
| Duration | Time | Duration of test exam |
| Status | int | Status of that base test (approved, rejected, Edited…) |
| Approver | varchar(50) | The people who approve quality of that base test |
| IsPublished | bit | Is that base test published for student |

Table 68 BaseTest table Description

#### BaseTestContent

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Question Id of base test |
| ImagePath | nvarchar(Max) | Relative folder storage image of base test content. |
| Content | nvarchar(max) | Content of that question |
| Level | nvarchar(255) | Level of question (describe how question hard) |
| Options | nvarchar(max) | Json string list option can choice in that question |

Table 69 BaseTestContent table Description

#### Test

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Id Student’s Test Exam |
| Mark | float | Mark’s Student Test Exam |
| TotalMark | float | Total mark of that test exam |
| IsFinished | bit | Is student done this test exam |
| StartTime | DateTime | Time when student take test exam |
| EndTime | DateTime | End time when student take test exam |
| Duration | Time | Time of student take that test exam |

Table 70 Test table Description

#### TestContent

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Id Student’s answer |
| Answer | nvarchar(max) | Json string list student answer of this question |
| Mark | Float | Mark of this answer in this question |

Table 71 TestContent table Description

#### Question

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | nvarchar(20) | Question Code |
| ImagePath | nvarchar(max) | Relative path storage image of question |
| Content | nvarchar(max) | Question content |
| IsExamQuestion | bit | Is question already in test exam |
| Mark | float | Mark of this question |
| IsDeleted | bit | Is this question deleted |
| Importer | varchar(50) | The people who import this question |

Table 72 Question table Description

#### Option

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Id of this option |
| ImagePath | nvarchar(max) | Relative path storage image of option |
| Content | nvarchar(max) | Option content |
| IsCorrect | bit | Is this option correct |
| Percent | float | Percent will Minus if choice wrong |

Table 73 Option table Description

#### Level

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Level Id |
| Name | nvarchar(255) | Level name (ex: hard, easy) |
| Percent | float | How important it is |

Table 74 Level table Description

#### QuestionType

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Type Id |
| Name | nvarchar(255) | Type name (ex: single choice, multiple choice) |

Table 75 QuestionType table Description

#### QuestionGroup

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | QuestionGroup Id |
| ImagePath | nvarchar(max) | Relative Path of this question group |
| Content | nvarchar(max) | Content of this question group |

Table 76 QuestionGroup table Description

#### QuestionObjective

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| ChapterId | int | Chapter Id |
| LearningOutcomeId | int | Learning Outcome Id |
| QuestionCode | varchar(20) | Question |

Table 77 QuestionObjective table Description

#### Chapter

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Chapter Id |
| Name | nvarchar(255) | Name of chapter (ex: chapter 1) |
| Order | int | Order of chaper (1, 2…) |

Table 78 Chapter table Description

#### Course

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | Int | Course Code |
| Name | nvarchar(255) | Course Name |
| IsDeleted | bit | Is this course deleted |

Table 79 Course table Description

#### MainObjective

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | MainObjective Id |
| Content | nvarchar(max) | Main objective information |
| Order | int | Order of this main objective |

Table 80 MainObjective table Description

#### LearningOutcome

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Learning Outcome Id |
| Content | nvarchar(max) | Learning outcome information |
| Order | int | Order of this Learning outcome |

Table 81 LearningOutcome table Description

#### Course Department

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | Varchar(20) | Learning Outcome Id |
| Name | nvarchar(255) | Learning outcome information |
| IsDeleted | bit | Order of this Learning outcome |

Table 82 CourseDepartment table Description

#### TeacherTest

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Teacher Test Id |
| Mark | float | Mark of this test exam of teacher |
| TotalMark | float | Total mark of test test exam |
| IsFinished | bit | Is this teacher test finished |
| StartTime | DateTime | Time teacher start to take the test exam |
| EndTime | DateTime | Time teacher end the test exam |
| Duration | Time | Duration teacher done this test exam |
| Teacher | Varchar(255) | Teacher code get from FPT authentication service |

Table 83 TeacherTest table Description

#### TeacherTestContent

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Teacher Test Id |
| Mark | float | Mark of this test exam of teacher |
| Answer | nvarchar(max) | Answer of teacher test exam |

Table 84 TeacherTestContent table Description

## Interaction Diagram

### Activity Diagram

#### Teacher Import Question

**Summary:** This diagram shows how Teacher import question

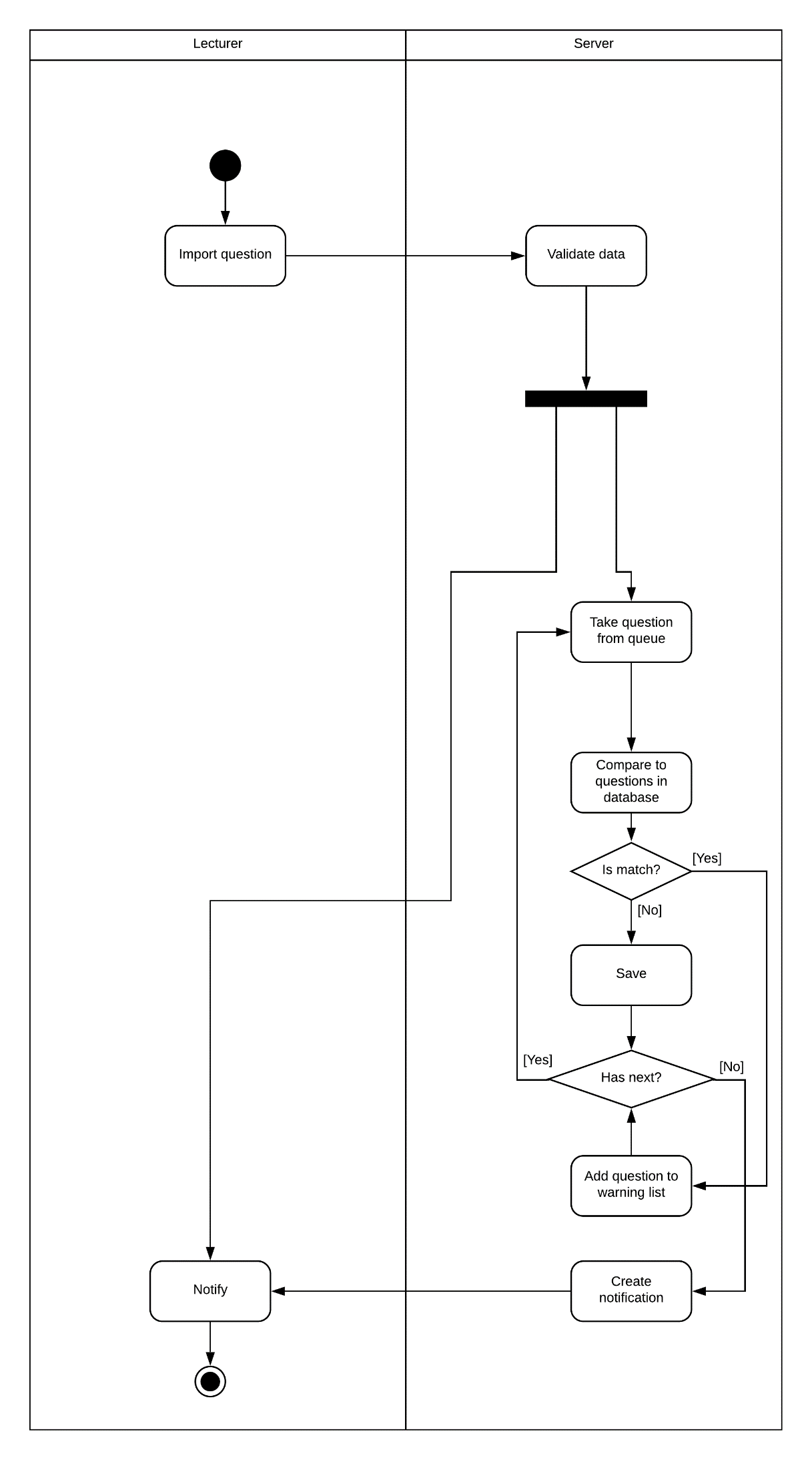


Figure 33 Activity diagram Teacher Import Question

#### Generate Test Exam

**Summary:** This diagram shows how Staff Create Test Exam

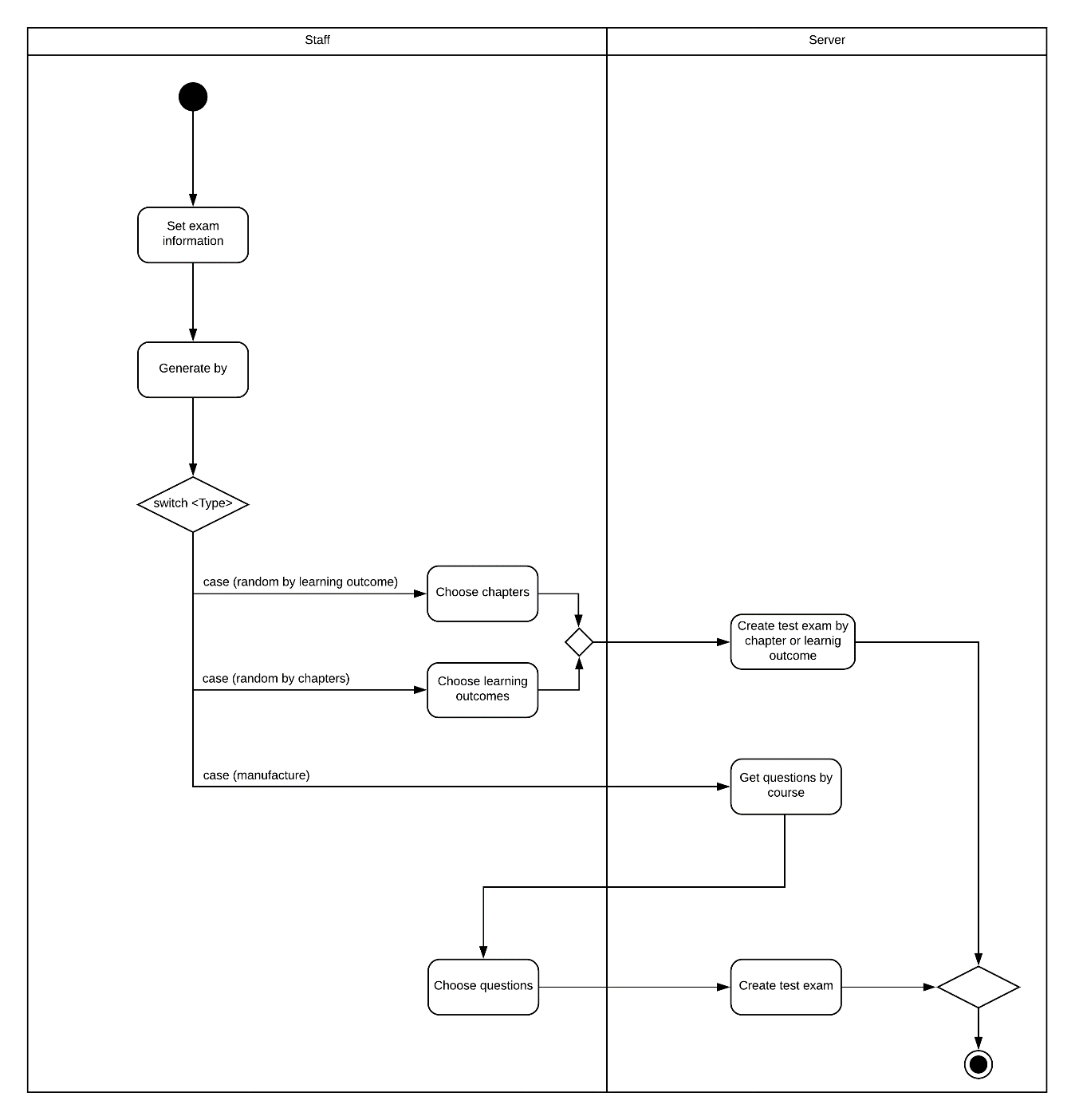


Figure 34 Activity diagram Training department staff generate Test Exam

#### Teacher Approve Test Exam

**Summary**: This diagram shows how Teacher approve test exam

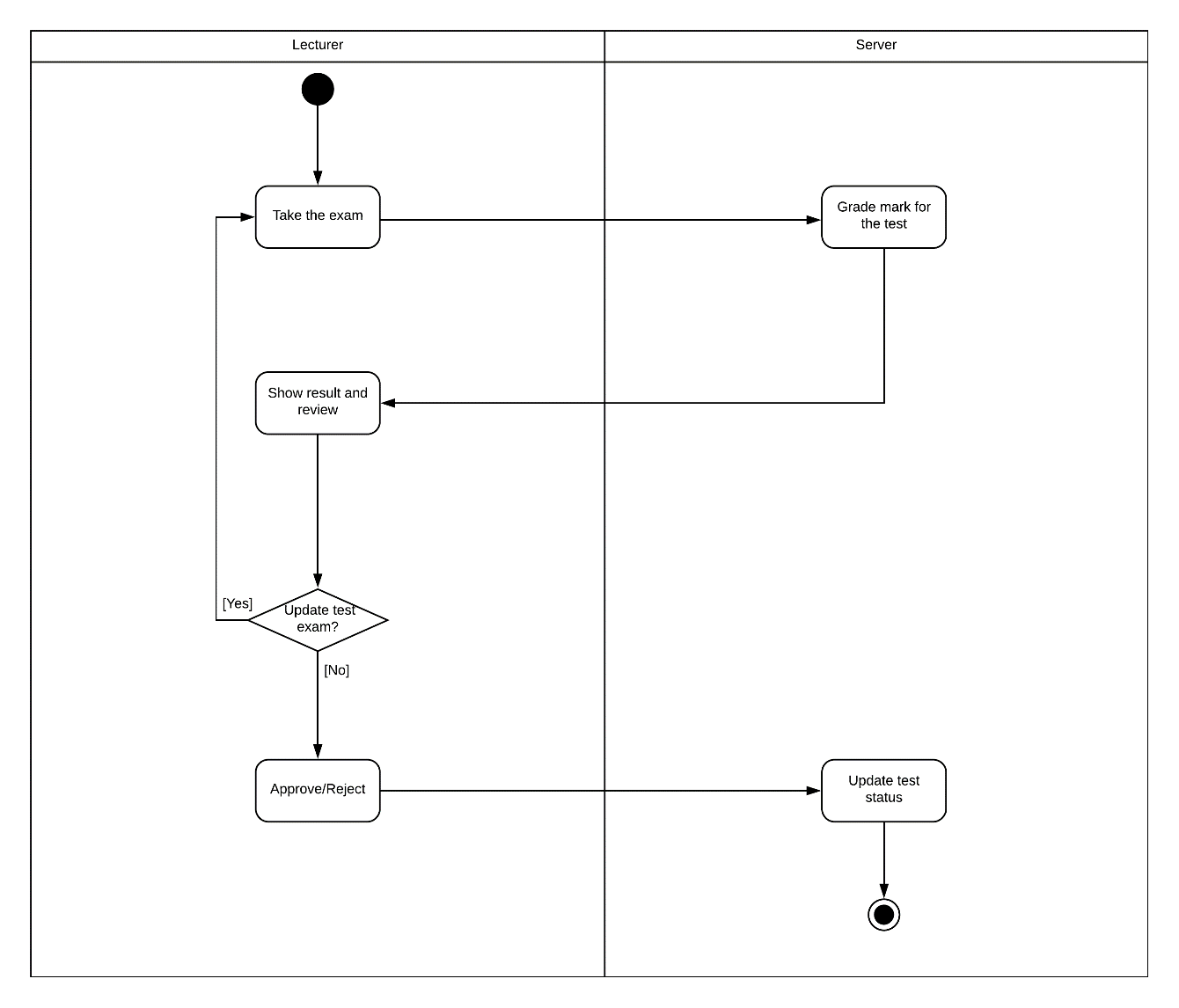


Figure 35 Activity diagram Approve Test Exam

#### Student Take a Test Exam

**Summary**: This diagram shows how Student Take a Test Exam

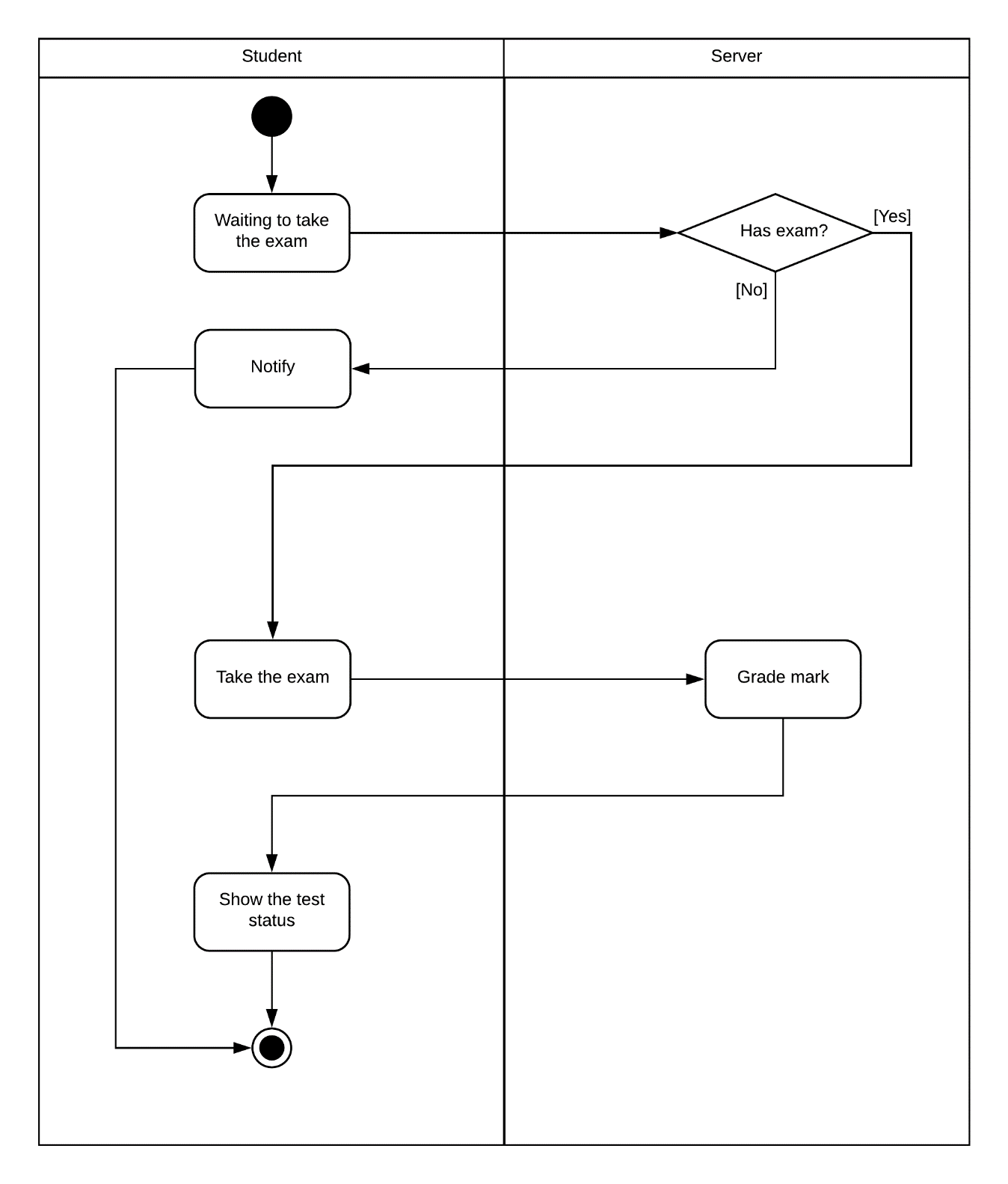


Figure 36 Activity diagram Student take a test exam

## Database Relationship Diagram

### Physical Diagram:

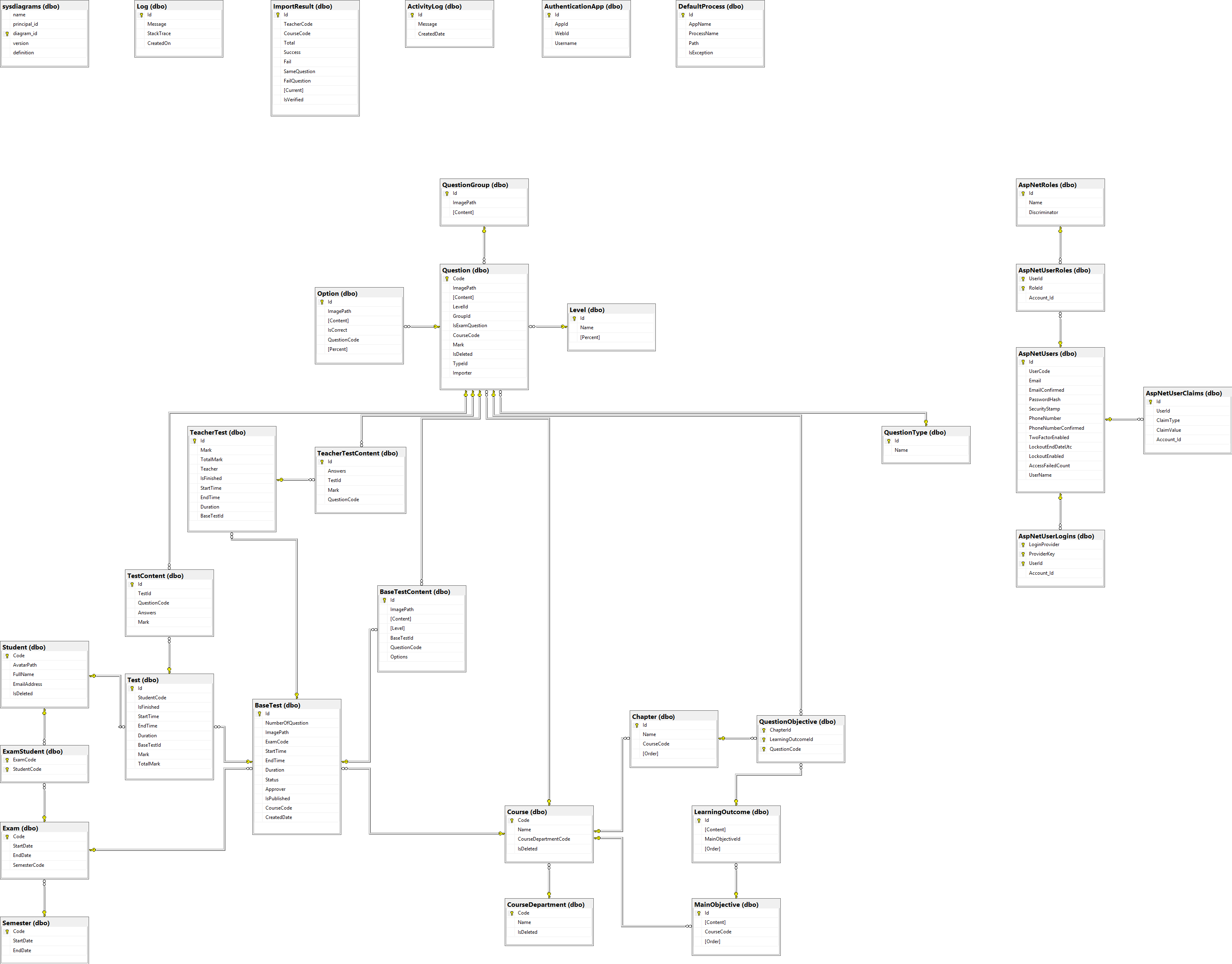
****

Figure 37 Physical diagram

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| BaseTest | Contain the test exam information |
| Semester | Contain the Semester information |
| Exam | Contain the exam information |
| ExamStudent | Contain between Student and Exam, this describe the accessible to the test exam of student |
| Student | Contain the Student information |
| Test | Contain student Test exam information |
| TeacherTest | Contain teacher test exam information (because teacher need to take exam to approve test exam) |
| TeacherTestContent | Contain answer of each teacher’s question |
| BaseTestContent | Contain test exam’s question information |
| TestContent | Contain Student’s answer information |
| Question | Contain question information (question bank) |
| QuestionGroup | Contain some group question like reading question which contain some question in reading question |
| Option | Contain option can choice of each question |
| QuestionType | Type of question information (ex: single choice, multiple choice) |
| Level | Contain some level of question (hard, easy …) |
| QuestionObjective | Question corresponding with LO |
| Chapter | Contain chapter information |
| Course | Contain Course information |
| MainObjective | Contain Purpose of course information |
| LearningOutcome | Decompose of big Main Objective |
| CourseDepartment | Contain Course Department information |
| ImportResult | Result of matching process |
| AuthenticationApp | Store info to Identify Student when using app |
| DefaultProcess | Storage Process must kill |

Table 85 Physical Diagram Description

### Data Dictionary:

#### Semester

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | varchar(20) | Code of Semester (ex: Fall 2018) |
| StartDate | Datetime | Start Date of semester |
| EndDate | Datetime | End Date of semester |

Table 86 SEMESTER TABLE DESCRIPTION

#### Exam

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | varchar(20) | Code of Exam (ex: FinalX) |
| StartDate | Datetime | Start Date of Exam |
| EndDate | Datetime | End Date of Exam |
| SemesterCode | varchar(20) | Code of Semester (ex: Fall 2018) |

Table 87 Exam table Description

#### Student

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | varchar(20) | Code of a Student (ex SE61801) |
| AvatarPath | nvarchar(Max) | Relative image path of Student Avatar |
| FullName | nvarchar(255) | Student’s Full Name |
| EmailAddress | nvarchar(255) | Student’s Email |
| isDeleted | bit | Flag to know is this student deleted (true = is deleted) |

Table 88 Student table Description

#### BaseTest

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Base Test Id |
| NumberOfQuestion | int | Number question of this Test Exam |
| ImagePath | nvarchar(max) | Relative folder storage image of base test. |
| StartTime | DateTime | Start time for student take test exam |
| EndTime | DateTime | End time for student take test exam |
| Duration | Time | Duration of test exam |
| Status | int | Status of that base test (approved, rejected, Edited…) |
| Approver | varchar(50) | The people who approve quality of that base test |
| IsPublished | bit | Is that base test published for student |
| ExamCode | varchar(20) | Code of Semester (ex: Fall 2018) |
| CourseCode | Int | Course Code |

Table 89 BaseTest table Description

#### BaseTestContent

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Question Id of base test |
| ImagePath | nvarchar(Max) | Relative folder storage image of base test content. |
| Content | nvarchar(max) | Content of that question |
| Level | nvarchar(255) | Level of question (describe how question hard) |
| Options | nvarchar(max) | Json string list option can choice in that question |
| baseTestId | int | Base Test Id |
| questionCode | nvarchar(20) | Question of this test exam |

Table 90 BaseTestContent table Description

#### Test

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Id Student’s Test Exam |
| Mark | float | Mark’s Student Test Exam |
| TotalMark | float | Total mark of that test exam |
| isFinished | bit | Is student done this test exam |
| StartTime | DateTime | Time when student take test exam |
| EndTime | DateTime | End time when student take test exam |
| Duration | Time | Time of student take that test exam |
| baseTestId | int | Base Test Id |
| StudentCode | varchar(20) | Code of a Student (ex SE61801) |

Table 91 Test table Description

#### TestContent

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Id Student’s answer |
| Answer | nvarchar(max) | Json string list student answer of this question |
| Mark | Float | Mark of this answer in this question |
| QuestionCode | nvarchar(20) | Question Code |
| TestId | int | Id Student’s Test Exam |

Table 92 TestContent table Description

#### Question

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | nvarchar(20) | Question Code |
| ImagePath | nvarchar(max) | Relative path storage image of question |
| Content | nvarchar(max) | Question content |
| IsExamQuestion | bit | Is question already in test exam |
| Mark | float | Mark of this question |
| isDeleted | bit | Is this question deleted |
| importer | varchar(50) | The people who import this question |
| LevelId | Int | Level of this question |
| TypeId | Int | Type of this question (single choice, multiple choice) |
| GroupId | int | Group question in reading question |

Table 93 Question table Description

#### Option

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Id of this option |
| ImagePath | nvarchar(max) | Relative path storage image of option |
| Content | nvarchar(max) | Option content |
| IsCorrect | bit | Is this option correct |
| Persent | float | Persent will Minus if choice wrong |
| QuestionCode | nvarchar(20) | Question of this option |

Table 94 Option table Description

#### Level

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Level Id |
| Name | nvarchar(255) | Level name (ex: hard, easy) |
| Percent | float | How important it is |

Table 95 Level table Description

#### QuestionType

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Type Id |
| Name | nvarchar(255) | Type name (ex: single choice, multiple choice) |

Table 96 QuestionType table Description

#### QuestionGroup

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | QuestionGroup Id |
| ImagePath | nvarchar(max) | Relative Path of this question group |
| Content | nvarchar(max) | Content of this question group |

Table 97 QuestionGroup table Description

#### QuestionObjective

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| ChapterId | int | Chapter Id |
| LearningOutcomeId | int | Learning Outcome Id |
| QuestionCode | varchar(20) | Question |

Table 98 QuestionObjective table Description

#### Chapter

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Chapter Id |
| Name | nvarchar(255) | Name of chapter (ex: chapter 1) |
| Order | int | Order of chaper (1, 2…) |
| CourseCode | Int | Course Code |

Table 99 Chapter table Description

#### Course

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | Int | Course Code |
| Name | nvarchar(255) | Course Name |
| IsDeleted | bit | Is this course deleted |
| CourseDepartmentCode | nvarchar(20) | Department of this course |

Table 100 Course table Description

#### MainObjective

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | MainObjective Id |
| Content | nvarchar(max) | Main objective information |
| Order | int | Order of this main objective |

Table 101 MainObjective table Description

#### LearningOutcome

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | Int | Learning Outcome Id |
| Content | nvarchar(max) | Learning outcome information |
| Order | int | Order of this Learning outcome |
| MainObjectiveId | Int | MainObjective Id |

Table 102 LearningOutcome table Description

#### Course Department

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Code | varchar(20) | Learning Outcome Id |
| Name | nvarchar(255) | Learning outcome information |
| IsDeleted | bit | Order of this Learning outcome |

Table 103 CourseDepartment table Description

#### TeacherTest

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Teacher Test Id |
| Mark | float | Mark of this test exam of teacher |
| TotalMark | float | Total mark of test test exam |
| isFinished | bit | Is this teacher test finished |
| StartTime | DateTime | Time teacher start to take the test exam |
| EndTime | DateTime | Time teacher end the test exam |
| Duration | Time | Duration teacher done this test exam |
| Teacher | varchar(255) | Teacher code get from FPT authentication service |
| baseTestId | int | Base test of this test exam |

Table 104 TeacherTest table Description

#### TeacherTestContent

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| Id | int | Teacher Test Id |
| Mark | float | Mark of this test exam of teacher |
| Answer | nvarchar(max) | Answer of teacher test exam |
| questionCode | nvarchar(20) | Question of this answer |

Table 105 TeacherTestContent table Description

#### AuthenticationApp

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Id | Authentication app Id |
| AppId | Identify the app by Id, this Id generate by signalR Id |
| WebId | Id of web when load by the app identify by AppId. Generated by signalR |
| Username | Student Username |

Table 106 AuthenticationApp table Description

#### Import Result

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Id | Import Result Id |
| TeacherCode | Teacher who import that section |
| CourseCode | Course import |
| Total | Number of question |
| Success | Number of imported success |
| Fail | Number of imported fail |
| SameQuestion | Number of duplicate question |
| FailQuestion | Number of parse fail question |

Table 107 ImportResult table Description

#### DefaultProcess

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Id | Identify Id |
| AppName | Name of the software run on Student computer |
| ProcessName | Process name |
| Path | Absolute location of software |

Table 108 DefaultProcess table Description

## System Implement

### Web Server Application

Overall architecture of Examination Tool System with functional layers and the collaboration between the system and the external systems is shown in figure 3 below.

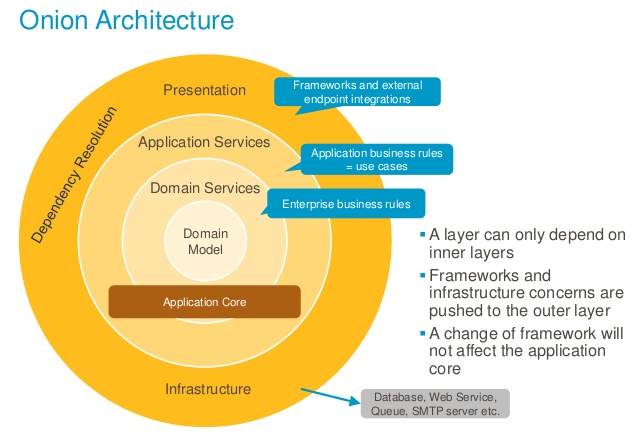


Figure 38 Onion Architecture

### Web Client Application

• Angular separate page to many components, each component is reusable, which make the application easy to debugging, maintenance.

• Component: The component controls the display, control View, so you can imagine the Component as a controller in the MVC model.

• Service classes are capable of performing some commonly used functions. Some common services are: logging service, data service, message bus,...

• Using dependency injection: Allow the creation of class objects with all the additional modules / modules / services.

• The event-driven architecture caters to both the client-side and the server- side that are written in JavaScript and thus the synchronization process is fast and orderly. The event loop through web socket protocol which works on TCP handles the multi-user function and prevents the overhead of HTTP for web development.

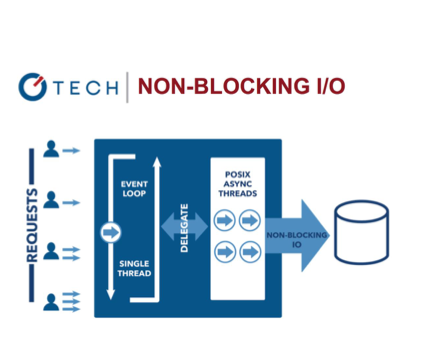


Figure 39 Event-driven architecture

In Web client application, we are using Angular under MVC architecture. We choose this Angular because of the following advantage:

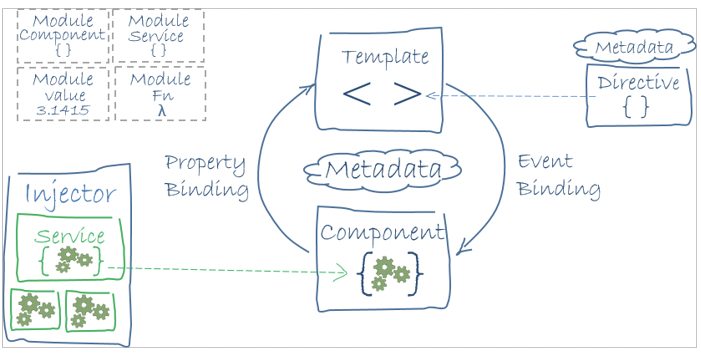


Figure 40 Angular Component Example

Reference: “https://angular.io/guide/architecture”

## Algorithms

### Calculating Percentage Similarity of 2 Question

#### Problem

When a teacher imports a question list into the question bank, some questions may be duplicated. This can lead to a waste of server’s storage.

#### Solution

In order to solve the mentioned problem, we decide to apply the Levenshtein distance algorithm.

Levenshtein distance algorithm is used to identify the differences between the ranges of 2 sequences. The range between these sequences is the minimum steps to make one sequence become the other one. This algorithm includes 3 changing functions:

- Remove a character

- Add a new character

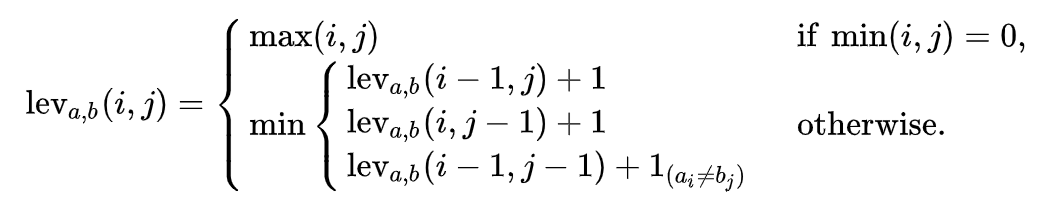
- Replace with another character

Example: To change “kitten” to “sitting”, we have to use at least 3 changing times as following:

1. kitten -> sitten (replace "k" with "s")
2. sitten -> sittin (replace "e" with "i")
3. sittin -> sitting (add "g")

Therefore, the range between "kitten" and sitting" is 3.

The nature of Levenshtein distance algorithm is based on Dynamic Programming. Mathematically, the Levenshtein distance between two strings {\displaystyle a,b}a, b (of length {\displaystyle |a|}|a| and {\displaystyle |b|}|b| respectively) is given by ,{\displaystyle \operatorname {lev} \_{a,b}(|a|,|b|)} where:



With

* {\displaystyle 1\_{(a\_{i}\neq b\_{j})}}  equal to 0 when {\displaystyle a\_{i}=b\_{j}} and equal to 1 otherwise
* {\displaystyle \operatorname {lev} \_{a,b}(i,j)}  is the distance between the first {\displaystyle i} characters of {\displaystyle a} and the first {\displaystyle j} characters of {\displaystyle b}

Note that the first element in the minimum corresponds to deletion (from {\displaystyle a} to {\displaystyle b}), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

Here is a straightforward pseudo code for a function called LevenshteinDistance that takes two strings:

* *s* of length *m*
* *t* of length *n*

And returns the Levenshtein distance between them:

function LevenshteinDistance(char s[1..m], char t[1..n]):

// for all i and j, d[i,j] will hold the Levenshtein distance between

// the first i characters of s and the first j characters of t

// note that d has (m+1)\*(n+1) values

declare int d[0..m, 0..n]

set each element in d to zero

// source prefixes can be transformed into empty string by

// dropping all characters

for i from 1 to m:

d[i, 0] := i

// target prefixes can be reached from empty source prefix

// by inserting every character

for j from 1 to n:

d[0, j] := j

for j from 1 to n:

for i from 1 to m:

if s[i] = t[j]:

substitutionCost := 0

else:

substitutionCost := 1

d[i, j] := minimum(d[i-1, j] + 1, // deletion

d[i, j-1] + 1, // insertion

d[i-1, j-1] + substitutionCost) // substitution

return d[m, n]

Two examples of the resulting matrix (hovering over a tagged number reveals the operation performed to get that number):

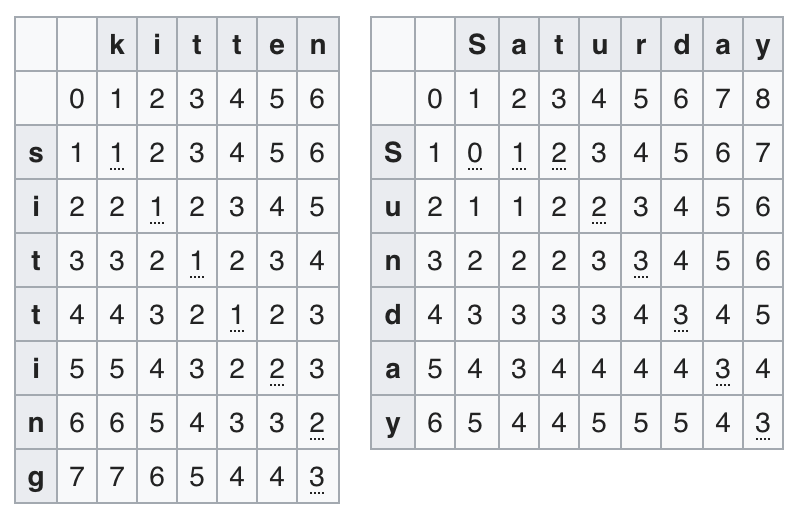


Figure 41 Levenshtein distance example result

#### Complexity

* In totally, the complexity of this algorithm is O (n\*m) with n is length of first question and m is length of second question need to compare

## Future Plans

* Flexible to custom template import question.
* More feature to manage student take exam process.
* Support all question type (reading, listen, writing…)