**ONLINE EXAM SYSTEM**

**SOFTWARE PROJECT REPORT**

<Hanoi, 15-02-2017>

# RECORD OF CHANGE

\*A – Added M – Modified D – Deleted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Effective Date** | **Changed Items** | **\*A, M, D** | **Change Description** | **New version** |
| 15-02-2017 | The whole document | A | Initial version | 1.0 |
| 17-02-2017 | Table of contents | A |  | 1.1 |
| 20-03-2017 | Complete document | M | Final version | 1.2 |
|  |  |  |  |  |

**TABLE CONTENT**

[**RECORD OF CHANGE** 2](#_Toc477791479)

[I. TEAM INTRODUCTION 6](#_Toc477791480)

[II. SOFTWARE REQUIREMENTS 6](#_Toc477791481)

[*1.* INTRODUCTION 6](#_Toc477791482)

[1.1 Purpose 6](#_Toc477791483)

[1.2 Scope 6](#_Toc477791484)

[1.3 Definition, Acronyms and Abbreviations 6](#_Toc477791485)

[1.4 Reference 7](#_Toc477791486)

[1.5 Overview 7](#_Toc477791487)

[2 OVERALL DESCRIPTION 7](#_Toc477791488)

[2.1 Product Perspective 7](#_Toc477791489)

[2.2 Product Functions 7](#_Toc477791490)

[2.2.1 Client side 8](#_Toc477791491)

[2.2.2 Server side 8](#_Toc477791492)

[*2.3*  User characteristics 9](#_Toc477791493)

[2.4 General Constraints 10](#_Toc477791494)

[3 SPECIFIC REQUIREMENTS 10](#_Toc477791495)

[3.1 Functionality 10](#_Toc477791496)

[3.1.1 Client side 10](#_Toc477791497)

[3.1.2 Server side 15](#_Toc477791498)

[3.2 Non-functional requirements 25](#_Toc477791499)

[3.3 Performance requirements 26](#_Toc477791502)

[3.4 Technical issues 26](#_Toc477791503)

[4 INTERFACE REQUIREMENTS 26](#_Toc477791504)

[4.1 User interface 26](#_Toc477791505)

[4.2 Software requirement 26](#_Toc477791507)

[4.3 Hardware requirement 26](#_Toc477791509)

[III. SOFTWARE DESIGN SPECIFICATION 26](#_Toc477791511)

[1 Discussion of chosen design 26](#_Toc477791512)

[1.1 Overview 27](#_Toc477791513)

[1.2 Server 28](#_Toc477791514)

[1.3. Client 28](#_Toc477791515)

[2 System Layer Architecture (Architecture Diagram) 28](#_Toc477791516)

[3 Context Diagram 30](#_Toc477791517)

[4 Project Structural Design 31](#_Toc477791518)

[4.1 Package model 32](#_Toc477791519)

[4.1.1 Brief Description 32](#_Toc477791520)

[4.1.2 Classes explanation 32](#_Toc477791521)

[4.1.2.1. Account.java 32](#_Toc477791522)

[4.1.2.2 AccountProfile.java 33](#_Toc477791523)

[4.1.2.3 Attempt.java 34](#_Toc477791524)

[4.1.2.4 Choice.java 35](#_Toc477791525)

[4.1.2.5 Course.java 35](#_Toc477791526)

[4.1.2.6 Question.java 36](#_Toc477791527)

[4.1.2.7 Role.java 37](#_Toc477791528)

[4.1.2.8 Test.java 37](#_Toc477791529)

[4.1.2.9 Rate.java 38](#_Toc477791530)

[4.2 Package view 39](#_Toc477791531)

[4.2.1 Brief Description 39](#_Toc477791532)

[4.2.2 Expexted Files: 39](#_Toc477791533)

[4.3 Package dal 40](#_Toc477791534)

[4.3.1 Brief Description 40](#_Toc477791535)

[4.3.2 Expected Classes: 40](#_Toc477791536)

[4.4 Package controller 41](#_Toc477791537)

[4.4.1 Brief Description: 41](#_Toc477791538)

[4.4.2 Expected Classes: 41](#_Toc477791539)

[IV. CONCLUSION AND DISCUSSION 42](#_Toc477791540)

[1 System review 42](#_Toc477791541)

[2 Member reflection 42](#_Toc477791545)

[3 Contribution 42](#_Toc477791546)

1. **TEAM INTRODUCTION**

This is our team project in Java Web Application course (PRJ321) in FPT University. Our team has 6 members:

|  |  |
| --- | --- |
| **Name** | **Student ID** |
| Khổng Đức Cảnh | SE04533 |
| Lê Cao Nguyên | SE04555 |
| Nguyễn Duy Hải | SE04546 |
| Phan Đăng Lâm | SE04797 |
| Nguyễn Quang Minh | SE04781 |
| Công Tôn Minh | SE04659 |

1. **SOFTWARE REQUIREMENTS**

# INTRODUCTION

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, reference, system proposals and an overview of the SRS.

## Purpose

The aim of this document is to gather, analyse and provide a consistent and complete description of the requirements for the software: Online Exam System. The Online Exam System (OES) provides facility to online examination inside and outside FPT University. Administrator and Test masters can create, modify, and delete tests and paticular questions. User can register, login, authenticize, take the test and review also. The system can eliminate the use of items such as papers, pen, etc.

* 1. Scope

The system is designed to access into 2 scopes:

- Public: Every user can publicly access available tests and review on OBS.

- Private: Only allowed users can access to a paticular test. Test master and Administrator can provide access to users.

## 1.3 Definition, Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| **ID** | **Acronym** | **Definition** |
| 1 | SRS | Software Requirements Specification |
| 2 | OES | Online Exam System |
|  |  |  |

* 1. Reference
* Software Engineering - Ninth Edition (by Ian Sommerville)
  1. Overview

We intend to develop the application with client-server architecture:

* Client side: This part includes features to help user use the core function of website such as: login with gmail, register, view course, join test, view test reports, feedback and review.
* Server side: This part contains all features to manage the content of the system such as: add/update/remove test masters, view/delete/add users(for administrators), manage tests and questions ,add list users who are allowed to join test, etc.

1. **OVERALL DESCRIPTION**
   1. Product Perspective

* OES is an online examination system. It is accessible 24/7. The objective of the system is to connect candidate and examiner in a long distance and provide virtual educational environment. The user flow is as follow:
* Login with gmail

User will log into the system with gmail authentication. Especially, the domain will be limited to FPT University domain with private test for FPT University students

* System Overview:

Logged in users can view available courses with corresponding tests, review taken test (if allowed).

* Test:

Users can join public tests or private test if they are allowed to.

* Testing report:

After the test, users can view results and reviewing the test. User can only view his/her own test.

* 1. Product Functions

### 2.2.1 Client side

|  |  |  |
| --- | --- | --- |
| **Class of use cases** | **Use case** | **Description of use cases** |
| User account | Login | User can log into the website via created account or with FPT gmail |
| Test | Search for available test | Search for test using test name, time, etc. |
| Join test | Join public test or private test if allowed |
| Submit test | User submits test to send answers to examiners |
| Review test | User can view result and review after particular test if allowed by test masters |
| Send feedback | Send feedback to examiner if there is any complaints. |

### 2.2.2 Server side

|  |  |  |
| --- | --- | --- |
| **Class of use cases** | **Use case** | **Description of use cases** |
| Admin and Test master account | Login | Login using admin account |
| Change password | Change admin account password |
| Test | Add test | Test master can add test with questions and allowed users |
| Remove test | Test master can remove his/her own tests. |
| Update test | Test master can update question, time, user list in his/her own test. |
| View own tests | Test master can view his/her own tests. |
| View feedback | View feedback made by users |
| Question | Add question | Test master can add question to questionList |
| View all questions | Test master can view all questions in questionList |
| Remove own question | Test master can remove his/her own question |
| Test master | Add test master | Admin can change user from student to test master |
| View test master | Admin can view test master list |
| Remove test master | Admin can change user from test master to student |

## 2.3 User characteristics

The website doesn’t require any special characteristics of user.

* 1. General Constraints
* The website requires Internet connection and browser to assess (client side).

1. **SPECIFIC REQUIREMENTS**
   1. Functionality

|  |  |
| --- | --- |
| **Primary Actor** | **Use Cases** |
| User/Student | 1. Login 2. Search for tests 3. Join test 4. Review test 5. Send feedback |
| Administrator | 1. Login 2. Change password 3. View test 4. View userlist 5. Add test master 6. Remove test master |
| Test Master | 1. Login 2. Change password 3. Add questions 4. Add Test 5. Update own test 6. Remove own test 7. View Test report |

### Client side

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES01 | | | |
| Use Case Name | Login | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | User | | |
| Description | User can log into the website to be able to view the system overview | | |
| Preconditions | User uses FPT domain gmail | | |
| Postconditions | User is logged in | | |
| Normal Flow | 1. User clicks [Login with FPT mail] 2. Authentication popup appears 3. User click button Allow 4. Server does authentication 5. The main page is displayed | | |
| Alternative Flows | N/A | | |
| Exceptions | * Server authentication failed, prompt the error to user and go back to step 3. | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES02 | | | |
| Use Case Name | Search for tests | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | User | | |
| Description | User can search for tests which can be filtered by test categories such as name, start and end time, etc. | | |
| Preconditions | User entered the website | | |
| Postconditions | A list of related tests are displayed to user | | |
| Normal Flow | 1. User clicks [Search bar] 2. User types in search terms 3. A list of related tests is displayed to the user | | |
| Alternative Flows | N/A | | |
| Exceptions | If no test contains information related to the search terms, display a message indicating there is no test related | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES03 | | | |
| Use Case Name | Join Test | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | User | | |
| Description | User can join public test or private test if allowed | | |
| Preconditions | User logged in  User’s email is included in allowed student list if joining private test | | |
| Postconditions | Test window is displayed to users | | |
| Normal Flow | 1. User clicks on a chosen test 2. User takes the test | | |
| Alternative Flows | N/A | | |
| Exceptions | If user’s email is not included in allowed student list for the test, promp the error to user and redirect to test list. | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES04 | | | |
| Use Case Name | Submit test | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | User | | |
| Description | User can submit test to send answer to examiner | | |
| Preconditions | User are accepted in the test | | |
| Postconditions | User’s answer is added to database | | |
| Normal Flow | 1. User clicks button Submit 2. User click Yes in the confirm popup 3. User’s answer is added to database, redirect to test list. | | |
| Alternative Flows | 2.a User click No in the confirm popup  3.a User is back to test window | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES05 | | | |
| Use Case Name | Review test | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | User | | |
| Description | User can view score, review test | | |
| Preconditions | User submitted test | | |
| Postconditions | Score and review window are displayed to user | | |
| Normal Flow | 1. User clicks Review 2. Score and review window are displayed | | |
| Alternative Flows | N/A | | |
| Exceptions | User is not allowed to review test. | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES06 | | | |
| Use Case Name | Send feedback to examiner | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | User | | |
| Description | User can send feedbacks to examiner in case of complaints | | |
| Preconditions | User logged in | | |
| Postconditions | Feedback is sent to server | | |
| Normal Flow | 1. User clicks on button Contact 2. User completes the feedback form 3. User click button Send 4. Feedback is sent to server | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | All fields must be filled in | | |
| Notes and Issues | N/A | | |

## Server side

* 1. Adminstrator

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES07 | | | |
| Use Case Name | Login | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can log into the system. | | |
| Preconditions | There exists admin account in database | | |
| Postconditions | Admin logged in the system | | |
| Normal Flow | 1. Admin clicks button Normal Login 2. A login form shows up 3. Admin provides necessary information. 4. Server does authentication 5. Dashboard is displayed | | |
| Alternative Flows | 5(a). Authentication fails  5(a).1. Prompt the admin that he/she provided invalid information. | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES08 | | | |
| Use Case Name | Change password | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can change his/her password | | |
| Preconditions | Admin logged into system | | |
| Postconditions | New password is updated | | |
| Normal Flow | 1. Admin clicks Change password 2. A change password form shows up 3. Admin provides and confirms new password 4. Server does authentication 5. Dashboard is displayed | | |
| Alternative Flows | 5(a). Authentication fails  5(a).1. Prompt the admin that he/she provided invalid information | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES09 | | | |
| Use Case Name | Add test master | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can change user from student to test master | | |
| Preconditions | Admin logged into the system | | |
| Postconditions | Chosen user becomes test master | | |
| Normal Flow | 1. Admin click Add test master 2. Admin fills in a user email 3. Chosen user becomes test master | | |
| Alternative Flows | N/A | | |
| Exceptions | Email doesn’t exist. | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES10 | | | |
| Use Case Name | View test master | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can view test master list | | |
| Preconditions | Admin logged in | | |
| Postconditions | Test master list is displayed | | |
| Normal Flow | 1. User clicks button View Test Master 2. Test Master table is displayed | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES11 | | | |
| Use Case Name | Remove test master | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can change user from test master to student | | |
| Preconditions | Admin logged in | | |
| Postconditions | Chosen test master becomes studeny | | |
| Normal Flow | 1. Admin clicks button Delete on a test master record 2. Admin clicks Yes in confirm popup 3. Chosen Test master becomes student | | |
| Alternative Flows | 2.a Admin clicks No in confirm popup  3.a Back to Test master list | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES12 | | | |
| Use Case Name | View all users | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can view all existed users | | |
| Preconditions | Admin logged in  User list is not empty | | |
| Postconditions | User list is displayed | | |
| Normal Flow | 1. Admin clicks View all users 2. User list is displayed | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

* 1. Test master

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES13 | | | |
| Use Case Name | Add question | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can add question to questionList | | |
| Preconditions | Test master is logged in | | |
| Postconditions | Question is updated in questionList | | |
| Normal Flow | 1. Admin clicks Add question 2. Admin fills in question and chooses correct answers 3. Admin click button Add 4. Question is updated in questionList | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES14 | | | |
| Use Case Name | Remove question | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can remove his/her own questions | | |
| Preconditions | Test master is logged in | | |
| Postconditions | Question is removed | | |
| Normal Flow | 1. Admin clicks button Delete in question record 2. Admin chooses Yes in confirm popup 3. Question is removed | | |
| Alternative Flows | 2.a Admin chooses No in confirm popup  3.a Back to question list | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES15 | | | |
| Use Case Name | Add test | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can add test | | |
| Preconditions | Admin logged in | | |
| Postconditions | Test is added to database | | |
| Normal Flow | 1. Admin clicks Add test 2. Admin fills in question list, student list, start and end time, private or public. 3. Admin click button Add 4. Test is added to database | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES16 | | | |
| Use Case Name | Remove test | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can remove his/her own test | | |
| Preconditions | Admin logged in | | |
| Postconditions | Test is removed from database | | |
| Normal Flow | 1. Admin clicks button delete in Test record 2. Admin chooses Yes in confirm popup 3. Test is removed from database | | |
| Alternative Flows | 2.a Admin chooses No in confirm popup  3.a Back to test list | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES17 | | | |
| Use Case Name | Update test | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can update his/her own test | | |
| Preconditions | Admin logged in | | |
| Postconditions | Test is updated in database | | |
| Normal Flow | 1. Admin clicks Update in test record 2. Admin updates test information 3. Admin click button Update 4. Test is updated in database | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES18 | | | |
| Use Case Name | View tests | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can view his/her own test | | |
| Preconditions | Admin logged in | | |
| Postconditions | Testlist is displayed to test master | | |
| Normal Flow | 1. Admin clicks View test 2. Test table is displayed to test master | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID : OES19 | | | |
| Use Case Name | View test report | | |
| Created By | CanhKD | Last Updated By | CanhKD |
| Date Created | February 15th, 2017 | Date Last Updated | February 17th, 2017 |
| Actors | Admin | | |
| Description | Admin can view test report ( score, answers, time, etc) | | |
| Preconditions | Admin logged in | | |
| Postconditions | Report is displayed to test master | | |
| Normal Flow | 1. Admin clicks View report 2. Report is displayed to test master | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | High | | |
| Special Requirements | N/A | | |
| Notes and Issues | N/A | | |

* 1. Non-functional requirements
* Security: secure assess to user’s confidential data.
* Reliability: 24/7 availability.
* Maintainability: coding conventions, follow-up documents for maintenance activities.

## Performance requirements

90% of responses should be within 1 second, except for Update content for which more time is acceptable.

* 1. Technical issues

This system will work on client-server architecture. The system should support common browsers such as IE, mozilla firefox, chrome, etc.

1. **INTERFACE REQUIREMENTS**
   1. User interface

A graphic user interface will be available in a workflow scenarios to assess to all features of the website. Any occuring error or exception catching should be displayed to user with friendly messages.

* 1. Software requirement

Any operating system with browser and internet connection.

* 1. Hardware requirement

No special hardware requirement is needed for the system

1. **SOFTWARE DESIGN SPECIFICATION**

# Discussion of chosen design

Client Browser:

Dynamic HTML Pages

Web CONTAINER (Tomcat)

JSP Page

Web application

Hibernate

Database MS Azure

*Figure 1: System Context Overview*

* 1. **Overview**
* The whole system will follow the MVC pattern in order to develop and maintain in the future much easier.
* As shown in the figure above, Microsoft Azre is chosen as Database, and will use Microsoft Azure Server to create the Database server.
* The website will have 2 separated parts: the ‘view’ (which is the client browser) and the ‘model‘ (which is the Database). These two parts will communicate with each other through the ‘controller’ (which is the one that has many services and dynamic web apps in it).
* In addition to the parts above, any class(es) that have same properties will be put in the same ‘util’ package and can be used and called oftenly.
* Client and Server will communicate with each other using HTTP request-response methods.
* The web application will communicate with the database by using Hibernate framework. This will lead to easier approach to the database since the structure of it in this project is considered sophisticated.
  1. **Server**
* Tomcat 8.0.27 will be used as the website’s Web container.
* Server may use Cookie, Session to store users when they first log in the system -> Making it easier when they want to log in next time.
* Because this project implemented mostly with java code and since this is a web application, using Servlet as Dynamic Web App is the most suitable.
* The server will be running on a local machine during the development phase. It is expected that the project will be deployed in a static host server.
  1. **Client**
* NOTE: Most users may use a up-to-dated version of a specific webrowser to render content responsed from the server properly.
* Most display page will use HTML and JSP to render in browser.
* All JSP and html pages will all use bootstrap and javascript in order to enhance the display and to help manage java code easier.

# System Layer Architecture (Architecture Diagram)

Web Users

Servlet

Hibernate Framework

Data Sources

Database Server Azure

Google API

JSP Pages

Each layer has its own responsibility:

* **Client Layer (Presentation Layer):**
  + Presentation Layer has responsibility for rendering the web application’s GUI dynamically depending on what kind of response the server sends everytime the users interact with the Online Exam System.
  + Presentation Layer will accept data given from users and makes a request to the server. Then this layer will take respones from the server give off the result to users. This layer contains HTML and JSP files with the uses of bootstrap and javascript, Jquery,.. for more comfortable Looks And Feels display.
* **Business Layer:**
  + Business Layer is the most important layer; it acts as a controller in the MVC pattern where view is Presentation layer, and model is the Database layer. Whenever a request is sent by an user, the servlet will receive it and bases on what contents of the request is, the servlet will a “ask” the Data Access to connect it to the database and process what that user needs. After all operations within the server finished, the Servlet will response back to the user.
  + This layer will try to generate appropriate HTML/JSP pages and send response back to user to render the results expected by the user. Apart from that, we also use Google API to authen the login with fpt.edu.vn mail.
* **Data Access Layer:**
  + Data Access Layer will act as a plugger, connect the web apps implemented in the server to the database layer. The main purpose of this layer is to CRUD (create, read, update, delete) datas within the database. It use Hibernate Framework instead of jdbc for easier access within the database
* **Database Layer (Database server):**
  + Database Layer is where all datas of the Online Exam System is stored. This layer contains many tables (having many relationship to each other) that have been created during the developement phase.
* **Server OS:** Microsoft Azure.
* **Web Container:** Tomcat 8.0.27.
* **Hibernate 4.3.x**
* **JSTL, GSON.**
* **Bootstrap:** getbootstrap, adminLTE, font awesome, ionicons, dataTables, icheck, ckeitor, pace, morris, chartjs.
* **Java platform:** Java SDK 8.1, JavaEE 7.0 API Lib

# Context Diagram

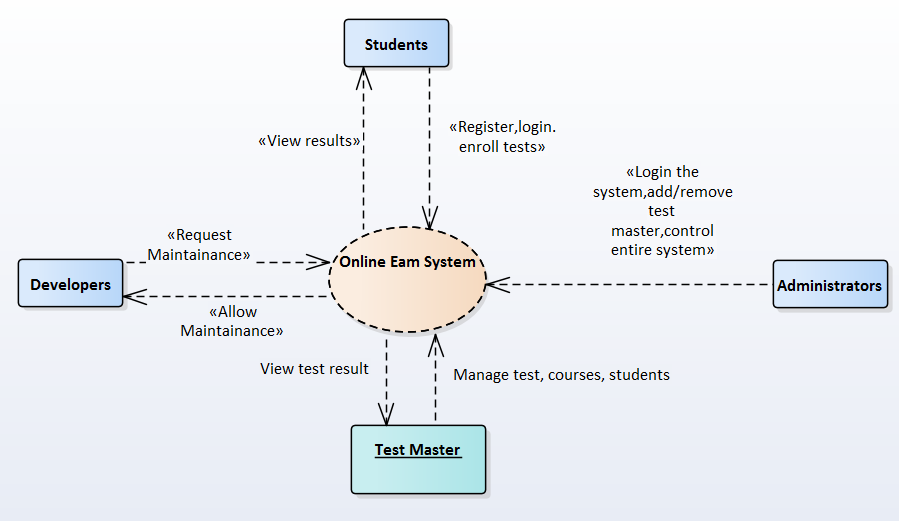
* This diagram will describe and define boundary between the Online Store System and its environment with actors interacting with it.
* There are 4 main actors at the time the decision of this design is made:

+) Administrator (Primary actor).

+) Student/Test Master (Primary actor).

+) Developers (Secondary actor).

* The primary actors will be the main user for this website, which means that they just only need to have an internet connection and a browser in order to use the website normally. On the contrary, developers will perform any maintainance in the “backstage”, and will not be represented in any diagrams except the context diagram in this whole document.



# 4 Project Structural Design

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Package name | Description | Naming convention & Notes |
| 1 | model | - Contain all Plain Old Java Object (POJO) classes that map with many tables in the database. Generated using Hibernate Framework.  - Additionally, there will be many xml files (also are generated by Hibernate) corresponding to the above POJO. | * Class name will be similar to one used in the database. * All fields in each class must be in lower case and must be formed by the following formula: (nameOfClass) + field name with the first letter in Upper case. For example: If the class name is ‘Student’ then the variable storing id must be ‘studentId’ |
| 2 | view (Web Pages folder) | Contain many JSP and HTML files (may have some CSS files as well if needed) that render response from client. This package will be used mostly for displaying friendly UI for users. | * All JSP, HTML files should have their names in lower case only. * All css, js, boostraps libraries are also included here. |
| 3 | dal | Contain some classes that can interact with the database and perform CRUD operations. | * With each correspoding object (or table), there will be a class managing all operations that can happen within that object. For instance, ‘AccountManager.java’ control any operations within the ‘Account’ object. |
| 4 | controller | - Contains many servlets, each have different functions to control request, response between clients and server. | * The servlets should have their names choosen under the following formula: (absolute path of the corresponding jsp file) + ‘Controller’. For example: SubjectListController.java indicates that this servlet will control functions related to the display of the subjects, with the jsp file should be located in Web Pages/WEB-INF/subject/list.jsp |
| 5 | util | - Contains many java classes that is frequently used throughout the entire project. These includes some like: GoogleAPI, Hibernate, ManagedServlet, Paging, … | * There will be many util packages, each package will be named differently based on the general properties that classes in that package have. For example: util.googleapi contains classes using API of Google. |

## Package model

### Brief Description

* + Contain all Plain Old Java Object (POJO) classes that map with many tables in the database.
  + JPA Annotations will be used in each class.

### Classes explanation

### 4.1.2.1. Account.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **username**  Private | String | username of the Acount |
| **password**  Private | String | password of the account |
| **roles**  Private | Set<Role> | Role of the account |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

### 4.1.2.2 AccountProfile.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **username**  Private | String | Username of the account |
| **account**  Private | Account | An Account object |
| **fullName**  Private | String | Real name of the user |
| **gender**  Private | boolean | Gender of the user |
| **birthDate**  Private | Date | Birth date of the user |
| **email**  Private | String | Email of the user |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

### 4.1.2.3 Attempt.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **id**  Private | Long | Id of the attempt (Each student take part in a particular exam, that aciton is called an attempt) |
| **examinee**  Private | Account | An Account object (student) |
| **test**  Private | Test | A Test object. |
| **startTime**  Private | Date | Start time of the attempt of the student |
| **endTime**  Private | Date | End time of the attempt of the student |
| **score**  Private | Doublr | Score of the student after the test |
| **choices**  Private | Set<Choice> | choices related to the attempt |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

### 4.1.2.4 Choice.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **id**  Private | Long | Id of the choice |
| **question**  Private | Question | Each question Object has many choices |
| **content**  Private | String | Content of the choice |
| **correct**  Private | Boolean | Mark if this choice is the correct answer. |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

### 4.1.2.5 Course.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **id**  Private | String | Id of the course |
| **name**  Private | String | Name of the course |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

### 4.1.2.6 Question.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **owner**  Private | Account | Account object (the test master response for managing this question). |
| **orderDate**  Private | date | Date ordered |
| **course**  Private | Course | Course that this question belongs to |
| **content**  Private | String | Content of the question |
| **choices**  Private | Set<Choice> | Choices belongs to this question |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

### 4.1.2.7 Role.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **name**  Private | String | name of the role |
| **description**  Private | String | description of the role |
| **accounts**  Private | Set<Account> | many-to-many relationship with Acocunt class |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

### 4.1.2.8 Test.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **id**  Private | Long | Id of the test |
| **owner**  Private | Account | Test master account response for managing the test |
| **name**  Private | String | Name of the test |
| **joinStartTime**  Private | Date | Start Time allow to attempt. |
| **joinEndTime**  Private | Date | End Time allow to attempt |
| **timeLength**  Private | Integer | Test duration (between join start time and join end time) |
| **attemptLimit**  Private | Integer | limit attempts of the test |
| **restricted**  Private | Boolean | specify if the test is private or not |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

### 4.1.2.9 Rate.java

Attributes

| **Attribute** | **Type** | **Description** |
| --- | --- | --- |
| **rateId**  Private | int | Id of the rating by an user |
| **rating**  Private | float | Rating of the product |
| **comment**  Private | String | Comment made by an user for the product |
| **datetime**  Private | date | Date that this rate is submitted |
| **productId**  Private | int | Product id that this rating for. |
| **userId**  Private | int | The id of the user who submit the rate |

Operations

| **Method** | **Purpose** | **Parameters** |
| --- | --- | --- |
| **Getter methods**  Public | Many method to get the above attributes |  |
| **Setter methods**  Public | Many method to set the above attributes | (will be defined) |

## Package view

### Brief Description

* This package contains JSP and HTML files which provide user interface to render for users in web browsers.
* This package is actually located in Web Pages in the Project.
* All jsp and html files will be put in specifics sub-folder of WEB-INF/jsp.
* Additional library (such as bootstrap, javascript lib,..) will be placed out side of the WEB-INF.
* Headers, footers will be put in WEB-INF/jspf.

### Expexted Files:

|  |  |
| --- | --- |
| File Name | Description |
| viewtest/report.jsp | UI displaying report after test |
| add\_test.jsp | UI displaying form to add test |
| manage\_questions.jsp | UI displaying panel to manage questions |
| manage\_test\_masters.jsp | UI displaying panel to manage test masters |
| index.jsp | Homepage of the Web app |
| …/index.jsp | UI displaying many indexes of the corresponding object |
| test.jsp | UI displaying panel where student doing test. |
| course.jsp | UI displaying courses list |
| manage\_course.jsp | UI displaying panel used to manage courses |
| login.jsp | UI displaying login form. |

## Package dal

### Brief Description

* This package contain all classes and function that involve interacting with database.
* Classes in this package will use Hibernate to perform many CRUD operations.
* All classes must extend from TransactionPerformer.java (located in util.hibernate).

### Expected Classes:

|  |  |
| --- | --- |
| File Name | Description |
| AccountManager.java | Control CRUD of accounts |
| AttemptManager.java | Control CRUD of attempts |
| ChoiceManager.java | Control CRUD of choices |
| CourseManager.java | Control CRUD of courses |
| QuestionManager.java | Control CRUD of questions |
| RoleManager.java | Control CRUD of roles |
| TestManager.java | Control CRUD of tests |

## Package controller

### Brief Description:

* + All classes will extend form the ManagedServlet.java(located in util.servlet package).
  + For the reason that HttpServlet is a pre-built class which comes within the JavaEE, we will not make any explanation about it in the section below.
  + Classes extending HttpServlet will need to override 2 methods: doGet and doPost.
  + Classes extending ManagedServlet.java (this class extends from HttpServlet) will need to override 2 methods: doGet and doPost.

### Expected Classes:

|  |  |
| --- | --- |
| File Name | Description |
| LoginController.java | Control login of the web app |
| LogoutController.java | Control logout of the web app |
| CourseController.java | Control course (add/update/remove and view list of courses) |
| AttemptController.java | Control attempts of students’ requests |
| DashboardController.java | Interacts with the dashboard view |
| SecurityController.java | Manage security of login/ logout and account used in the web application |
| SettingController.java | Interacts with the setting view |
| AddMasterController.java | Manage adding test master |
| DeleteMasterController.java | Manage deleting test master |
| ListTestMasterController.java | Manage listing test masters |
| TestController.java | Manage all operations related to the test object |
| ViewTestReportController.java | Retreive data form DB and listing test reports |
| QuestionController.java | Manage CRUD with questions |
|  |  |

1. **SCREENSHOT GUIDELINE**

# Login

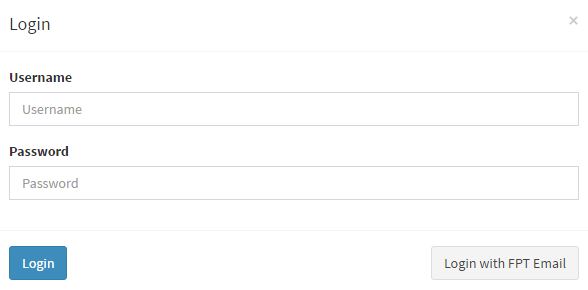


Figure 1. Login

User can login manually by filling in the login form or login via FPT email. Manual login currently is meant for admin usage. Manual register and login form for users will be developed in the future.

# Homepage for student

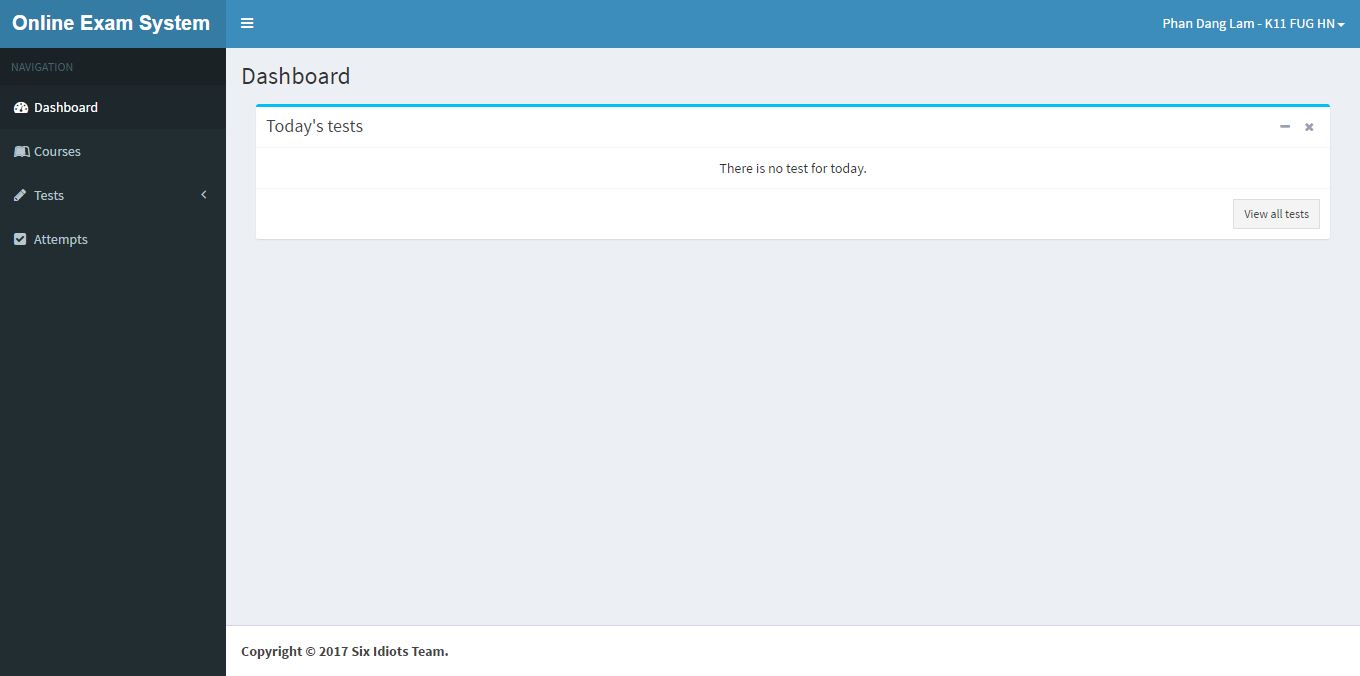


Figure 2. Homepage for student

This is the homepage when student logged in the website. In the dashboard, all the incoming test in current date will be displayed. The side bar on the left contains all the navigators to all main functions of OES.

# List all available tests

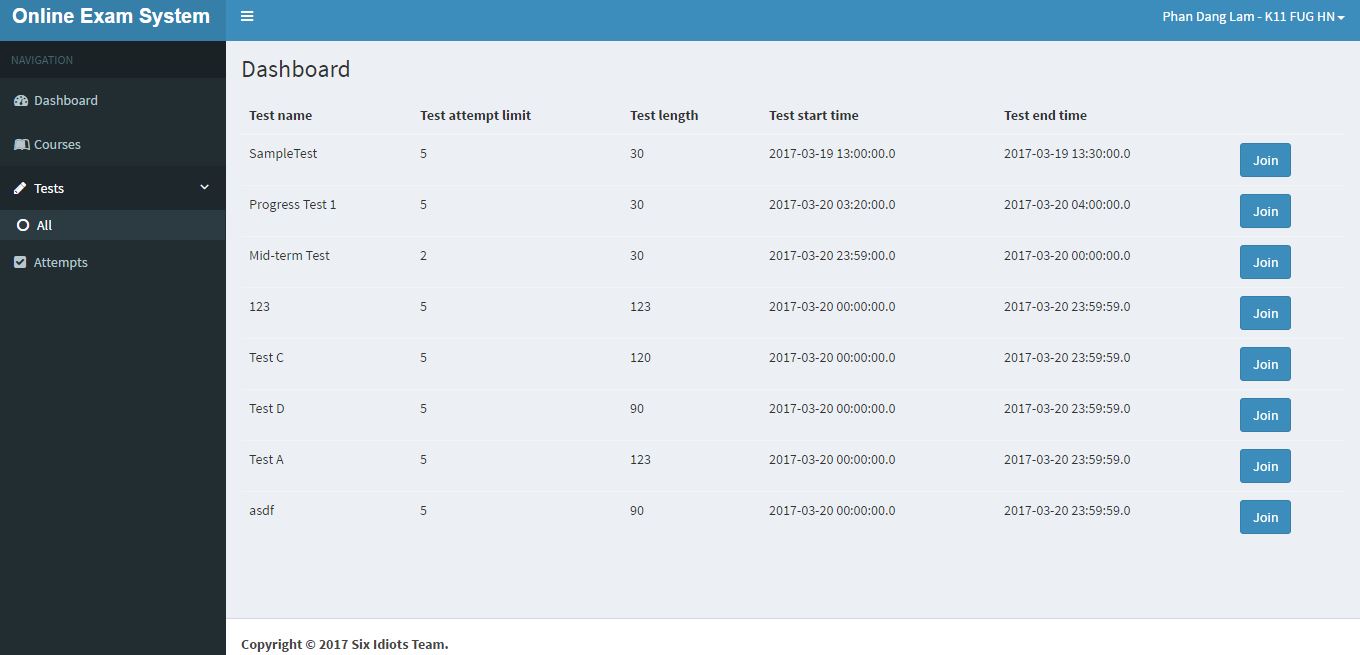


Figure 3. List test

List of all avalable tests is displayed when user clicks the “All test” button. Student can click Join to join the assigned test.

# Join Test

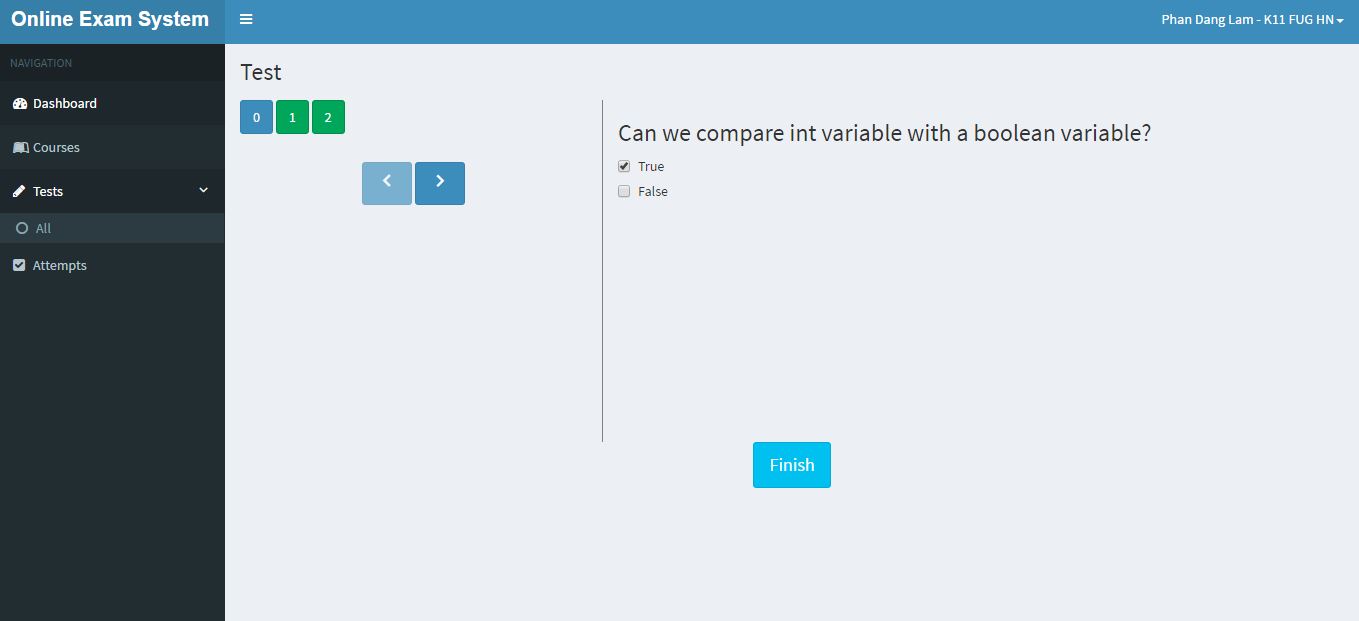


Figure 4. Join test

Student choices are auto saved.

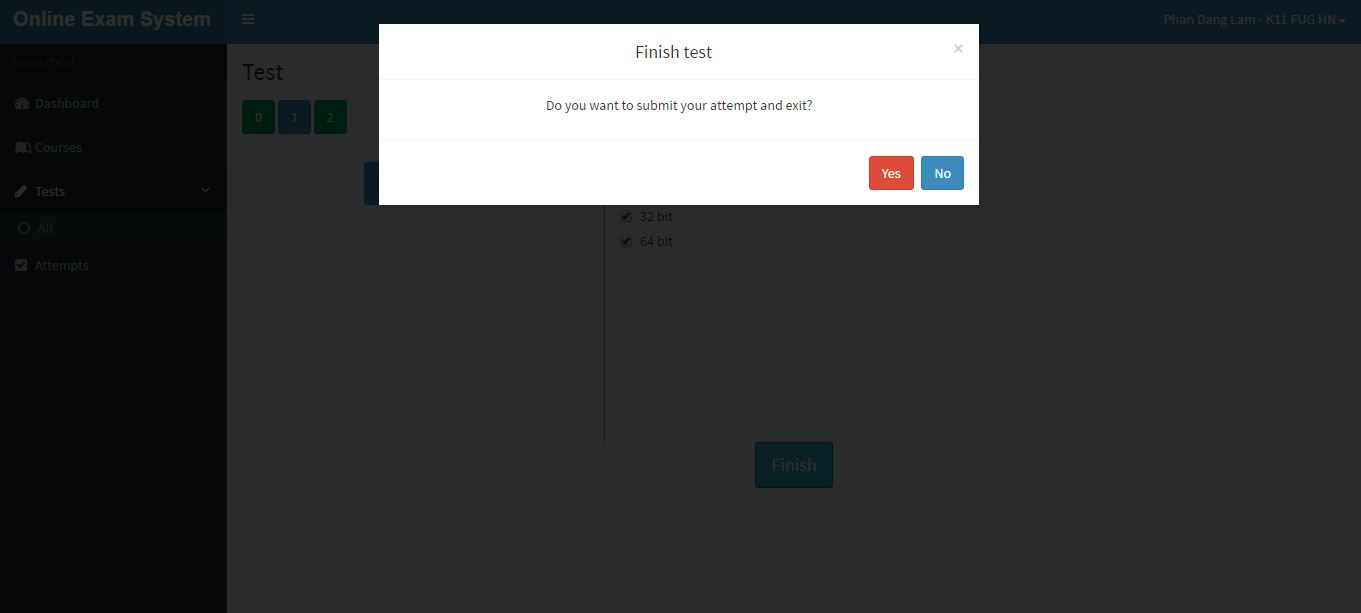


Figure 5. Confirm submit test

# Review attempt

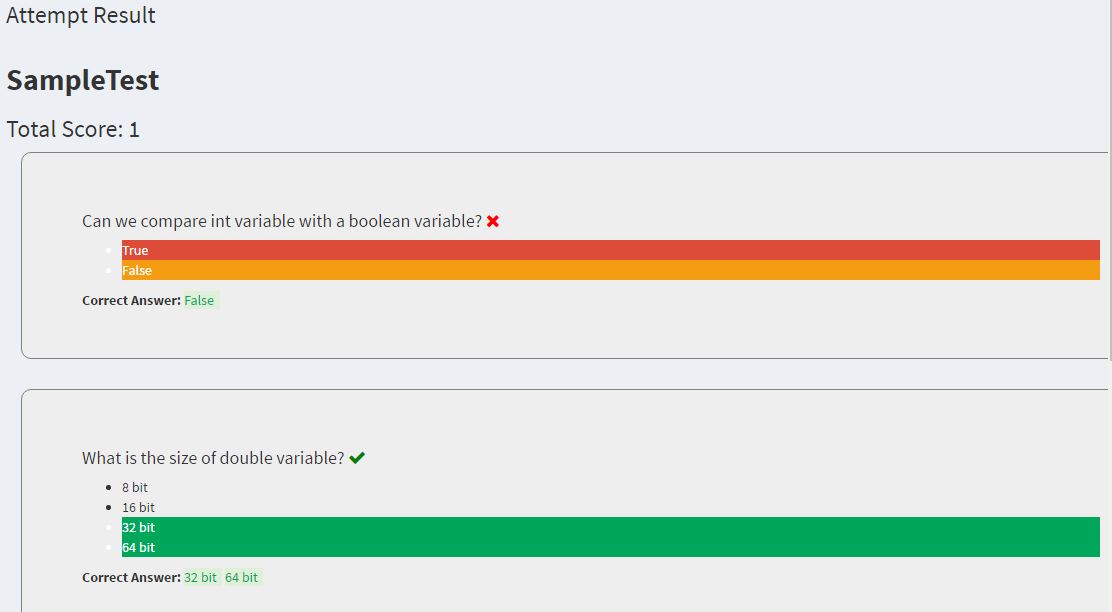


Figure 6. Student review attempt

Attempt review is loaded after student submits test.

# Update profile

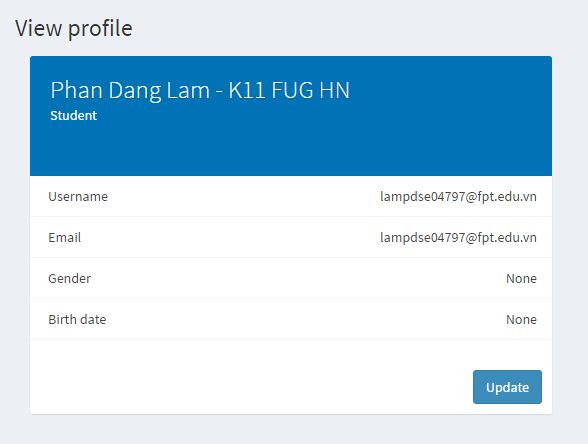


Figure 7. User update profile

# Test master manages test

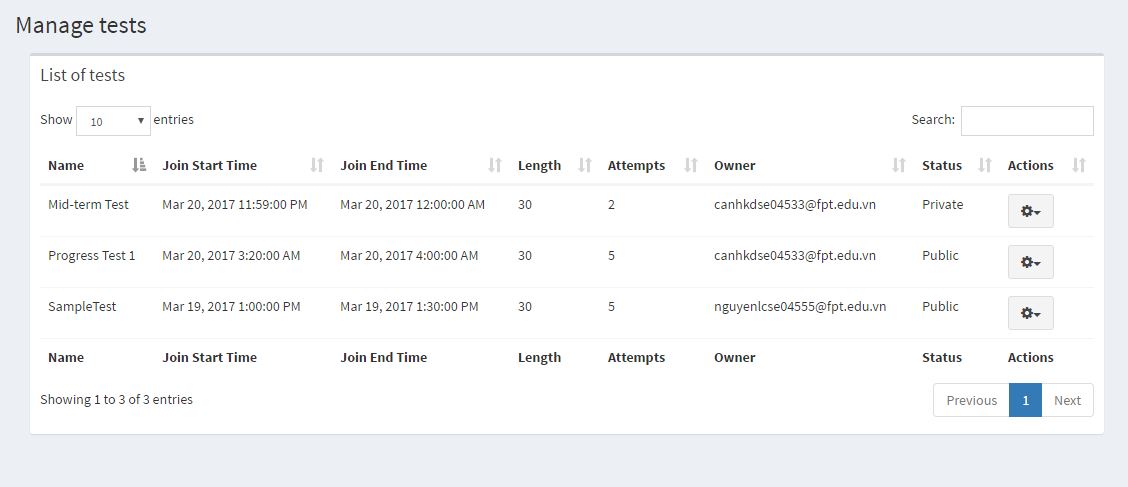


Figure 8. Test master lists test

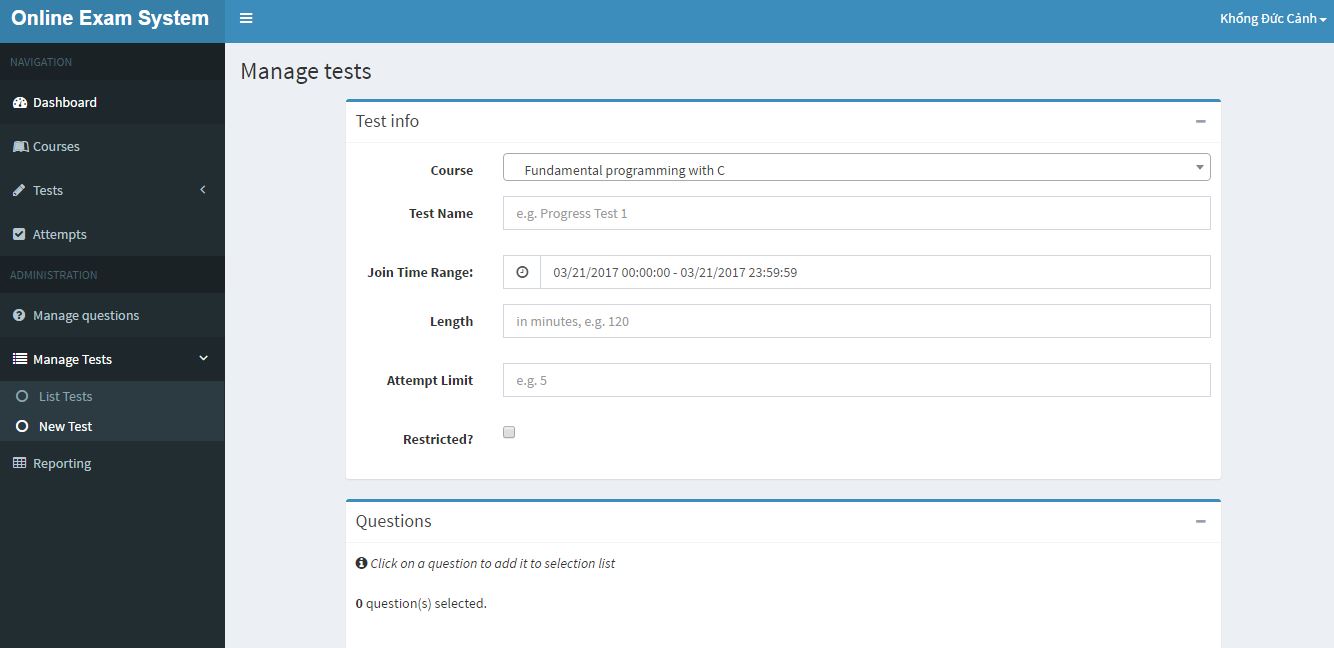


Figure 9. Test master adds test

# Test master reviews own test

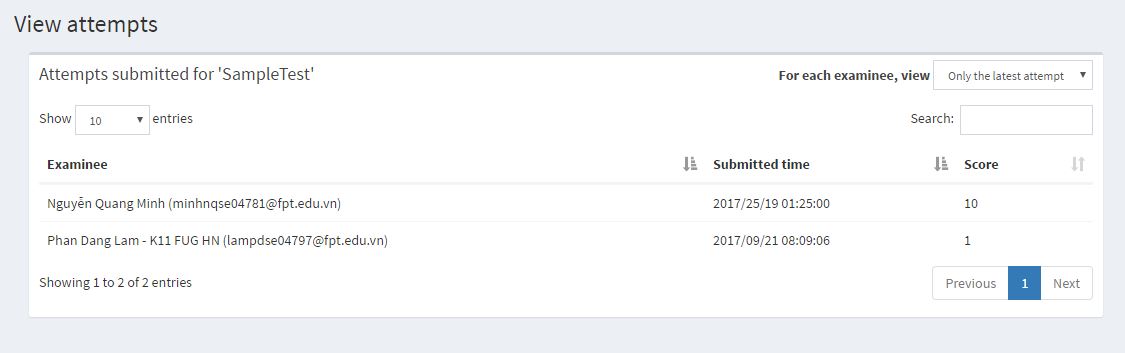


Figure 10. Testmaster reviews own test

# Admin manages courses

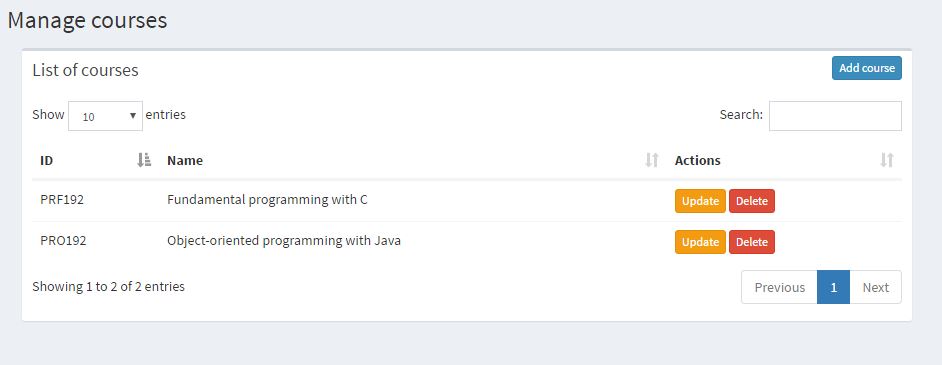


Figure 11. Admin lists courses

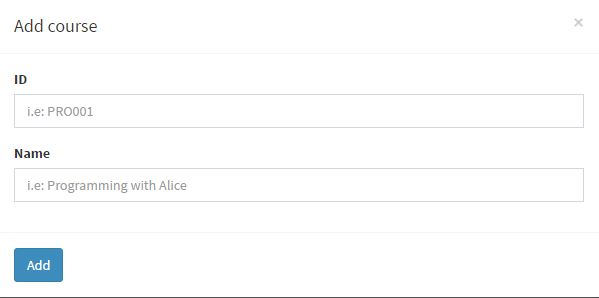


Figure 12. Admin adds courses

# Admin manages accounts

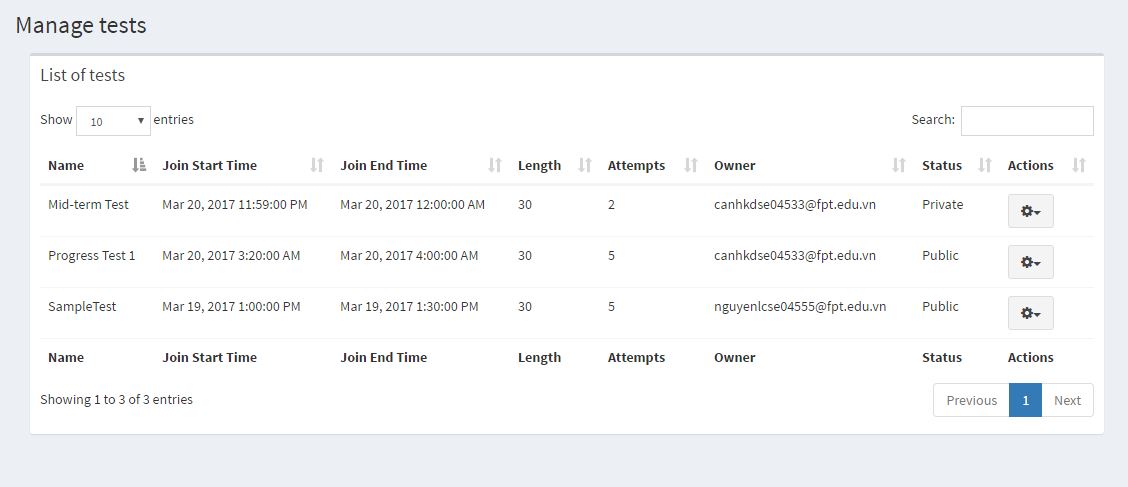


Figure 13. Admin lists test

1. **CONCLUSION AND DISCUSSION**

# System review

Online Exam System is a system that provides a studying environment between students and examiners without geographical limitations. The system has many benefits to be professionally developed. However there are apparently some problems that we have not solved due to limitations related to time, human resource, etc. Those pros and cons are analysed in the following table.

|  |  |  |
| --- | --- | --- |
| **Actions** | **PROS** | **CONS** |
| Using Bootstrap, AdminLTE template and various web interface libraries | easier to design beautiful and user-friendly web pages. | Inflexible layout, need time to learn how to use the libraries |
| Using online database (Microsoft Azure Service) | necessary for team development | slow speed |
| Using pure JSP + servlet for backend | No need to learn a web framework, benefit studying in class | harder to develop & maintain |
| Using Hibernate to access database | easier to write management code | Need time to learn how to use Hibernate, errors happened due to inadequate knowledge |
| Using Tomcat as web server | easier to configure, support in-app configuration for data source and security | Slow and tedious deployment process, not bundled with full Java EE 7 API and JSTL. |

Apparently, the system still has some problems due to time limitation. In the future, we will spend time to optimize some features such as randomly generating questions, thoroughly test the system, indentify bug list and complete the website. Re-engineering code convention also needs to be implemented. If possible, we will rent a cloud service such as Microsoft Azure Service or Amazon Web Service to solve current speed performance and enhance security to the system.

# Member reflections

What we have learned from this project.

|  |  |
| --- | --- |
| **Name** | **Reflections** |
| Khổng Đức Cảnh | For the first time ever I have got the chance to experience working in a course project as a project manager. Difficulties and inexperience come along with many valuable lessons. I acquired the working process in developing a software product, learned how to allocate tasks to suitable team members, track team members’ workloads, etc. Also, as a developer in the team, I had the chance to work with many talented team members, who helped me a lot in adapting to web development technologies, getting accustomed to web frameworks and libraries, which was, at the same time, beneficial to my academical study in class. Last but not least, I learned how to manage my time and schedule in this busy semester when I had to study 5 courses at the same time, have a part-time job and several extra-curricular activities. Busy and exhausted semester as it might be, it was completely worth with many valuable lessons and experience that I had. |
| Lê Cao Nguyên |  |
| Nguyễn Duy Hải |  |
| Phan Đăng Lâm |  |
| Nguyễn Quang Minh |  |
| Công Tôn Minh | At first, working with a group with full of talented people seems a bit hard to me. However, until now if it was without their helps, I could not imagine what I had learned anything related to our specialization. Each individual in my team has different ideas and ways of programming, but none of these are reduntdant knowledges. I want to sum up what I have learned throughout the project:  - Ways of thinking and working in group.  - Ways of following ideal of the leader.  - How to formating and refactor codes.  - Get used to using Hibernate instead of pure JDBC.  - Knowledge to use many bootstrap.  - Know how to use Github and subversion  - Applying full EL and jstl insteads of embedding Java code in JSP pages |

# Contribution

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Topic** | **Team**  **Effort** | CanhKD | NguyenLC | HaiND | LamPQ | MinhNQ | MinhCT |
| Case Study Analysis | 100% | 16.67 | 16.67 | 16.67 | 16.67 | 16.67 | 16.67 |
| Database design | 100% | 12.5 | 25 | 12.5 | 12.5 | 12.5 | 25 |
| System design | 100% | 15 | 25 | 15 | 15 | 15 | 15 |
| Implementation | 100% | 11.67 | 25 | 20 | 20 | 11.67 | 11.67 |
| Documentation | 100% | 20 | 15 | 15 | 15 | 15 | 20 |