Online Exam System

Software Architecture Design

Record of change

\*A - Added M - Modified D - Deleted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Effective Date | Changed Items | A\* M, D | Change Description | New Version |
| 16/03/2017 | The whole document | A | Initiate version | 1.0 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1. Discussion of chosen Design 3](#_Toc382768990)

[2. Description of System Interface 4](#_Toc382768991)

[3. Component diagram 5](#_Toc382768992)

[3.1 Server 5](#_Toc382768993)

[3.2 Client 6](#_Toc382768994)

[4. Details Description of components 7](#_Toc382768995)

[4.1 Client 7](#_Toc382768996)

[4.1.1 UI 7](#_Toc382768997)

[4.1.1.1 Brief Description 7](#_Toc382768998)

[4.1.1.2 Class diagram 8](#_Toc382768999)

[4.1.1.3 Class diagram explanation 8](#_Toc382769000)

[4.1.2 Utilities 12](#_Toc382769001)

[4.1.2.1 Brief Description 12](#_Toc382769002)

[4.1.2.2 Class Diagram 13](#_Toc382769003)

[4.1.2.3 Class diagram explanation 13](#_Toc382769004)

[4.1.3 DbAcess 14](#_Toc382769005)

[4.1.3.1 Brief Description 14](#_Toc382769006)

[4.1.3.2 Class diagram 15](#_Toc382769007)

[4.1.3.3 Class diagram explanation 15](#_Toc382769008)

[4.1.4 Preferences 17](#_Toc382769009)

[4.1.4.1 Brief Description 17](#_Toc382769010)

[4.1.4.2 Class diagram 17](#_Toc382769011)

[4.1.4.3 Class diagram explanation 17](#_Toc382769012)

[4.2. Server 18](#_Toc382769013)

[5. Sequence Diagram 19](#_Toc382769014)

[5.1 Update new content 19](#_Toc382769015)

[5.2 Enter Study Mode 20](#_Toc382769016)

[5.3 Enter Relax Mode/Next content 20](#_Toc382769017)

[5.4 Open/Change Setting 21](#_Toc382769018)

# 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:

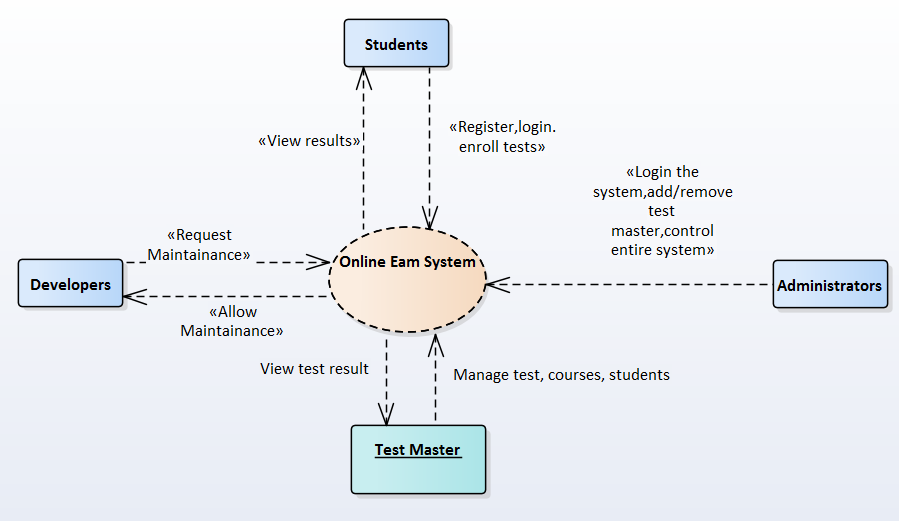
* This diagram will describe and define boundary between the Online Store System and its environment with actors interacting with it.
* There are 4 main actors at the time the decision of this design is made:

+) 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.



# 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

##### 4.1.3.1. 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) |

##### 4.1.3.2 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) |

##### 4.1.3.3 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) |

##### 4.1.3.4 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) |

##### 4.1.3.5 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) |

##### 4.1.3.6 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) |

##### 4.1.3.7 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) |

##### 4.1.3.8 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) |

##### 4.1.3.9 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 |