

**<<QUIZ PRACTICE>>**

**Software Design Document**

– Hanoi, August 2019 –

**Table of Contents**

[I. Overview 3](#_gjdgxs)

[1. Code Packages 3](#_30j0zll)

[2. Database Schema 3](#_1fob9te)

[II. Code Designs 4](#_3znysh7)

[1. <Feature/Function Name1> 4](#_2et92p0)

[a. Class Diagram 4](#_tyjcwt)

[b. Class Specifications 4](#_3dy6vkm)

[c. Sequence Diagram(s) 4](#_1t3h5sf)

[d. Database queries 5](#_4d34og8)

[2. <Feature/Function Name2> 5](#_2s8eyo1)

[III. Database Tables 5](#_17dp8vu)

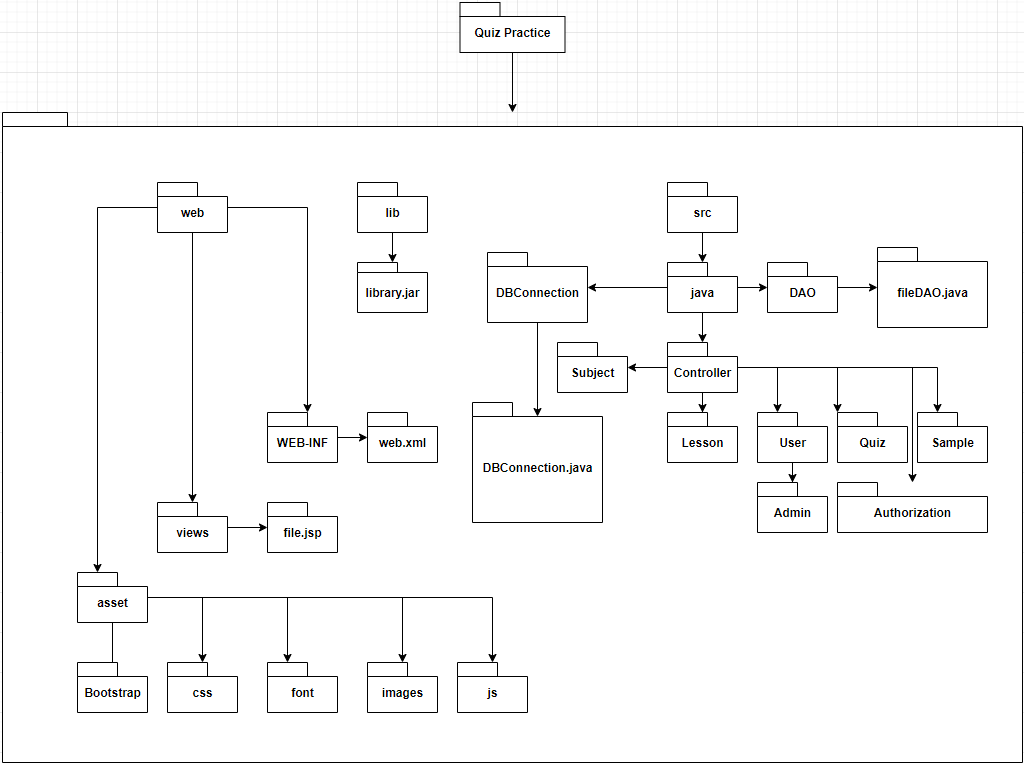
[1. <Table name 1> 5](#_3rdcrjn)

[2. <Table name 2…> 5](#_26in1rg)

# I. Overview

## 1. Code Packages/Namespaces

*[Provide the package diagram for each sub-system. The content of this section including the overall package diagram, the explanation, package and class naming conventions in each package. Please see the sample and description table format below – following Java project naming convention]*



***Package descriptions & package class naming conventions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| *01* | *web* | *Hold all front-end feature*  *Naming follow HTML/CSS naming style* |
| *02* | *asset* | *Hold library which support for front-end* |
| *03* | *views* | *Hold all jsp file* |
| *04* | *WEB-INF* | *Hold only web.xml file* |
| *05* | *lib* | *Hold all addition library which define as jar type* |
| *06* | *src/java* | *Hold all java file* |
| *07* | *DBConnection* | *Hold only DBConnection.java file* |
| *08* | *DAO* |  |

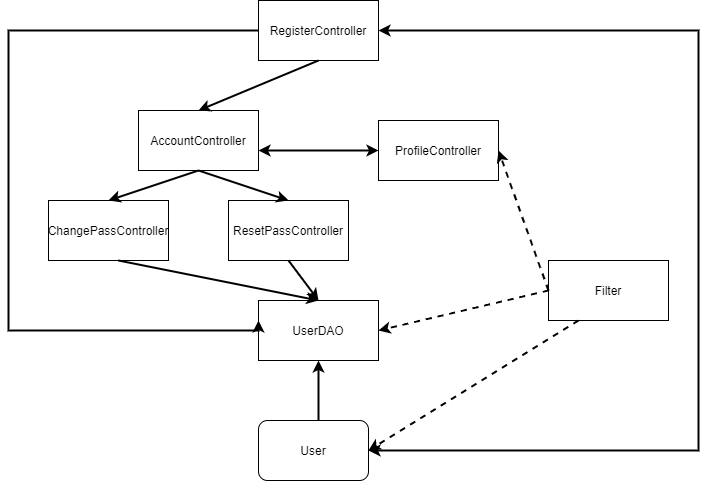
## 2. Coding Conventions

# II. Code Designs

## 1. <Common Feature>

### a. Class Diagram

*[This part presents the class diagram for the relevant feature]*



### b. Class Specifications

#### User Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Get User’s properties | Get data of the properties which belong to the user class |
| 2 | Set User’s properties | Set data of the properties which belong to the user class |

#### UserDAO Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | register(User user) | set data of the user in the Database |
| 2 | updateProfile(User user) | Update profile of the user in the Database |
| 3 | changePass(String pass,int userID ) | Change user password and update in the Database |
| 4 | isExisted(User u) | Check whether the user account or email existed or not |
| 5 | getUser(User u) | Get the user name and password and status |
| 6 | GetUsers() | Get all the user from database |

#### AccountController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | Get account and compare the input and the data in the data base |
| 2 | Request.setAtribute | Set the message for the alert of the login |
| 3 | Cookies.setcookies | Set the cookies of the username and password |
| 4 | Request.getRequestDispacher | Redirect the user to the screen after login |

#### EditProfileController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | Get account and compare the input and the data in the data base |
| 2 | Request.setAtribute | Set the message for the alert of the login |
| 3 | Cookies.setcookies | Set the cookies of the username and password |
| 4 | Request.getRequestDispacher | Redirect the user to the screen after login |
| 5 | updateProfile(newUser) | Update the user to the database |

#### RegisterController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | Get account and compare the input and the data in the data base |
| 2 | Request.setAtribute | Set the message for the alert of the login |
| 3 | Cookies.setcookies | Set the cookies of the username and password |
| 4 | Request.getRequestDispacher | Redirect the user to the screen after login |
| 5 | isExisted(User u) | Check whether the user account or email existed or not |
| 6 | register(u) | Insert new user profile to the Database |

### c. Sequence Diagram(s)

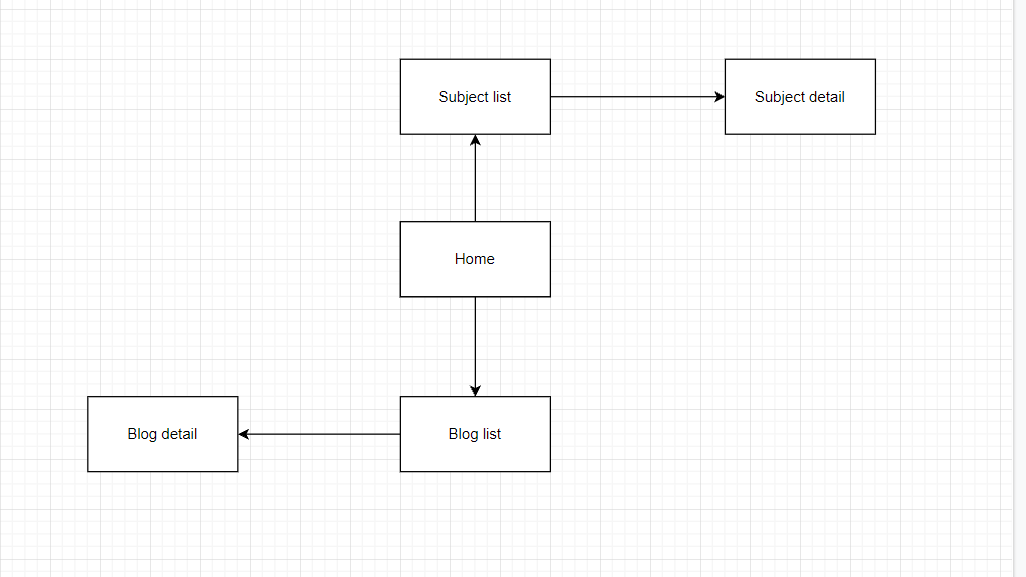


### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Query | Action |
| 1 | register | insert into [User]([role], username,[password],[status], first\_name,last\_name,birth,gender,phone,avatar,email,[description],[address], visa,paymentMethod,position,[sign],company,school)  values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) | Insert new user profile into the database |
| 2 | updateProfile | update [User] set first\_name=?,  last\_name=?,birth=?,gender=?,phone=?,avatar=?,email=?,  [description]=?,[address]=?,[visa]=?,[paymentMethod]=?,[position]=?,  [sign]=?,company=?,school=? where u\_id = ?" | Update profile of the user into the database |
| 3 | changePass | update [User] set [password]=? where u\_id = ? | Change user password into the database |
| 4 | GetUsers | Select \* from [QuizPractice].[dbo].[User] | Get all user profile from the database |

**2. <Public Feature>**

### a. Class Diagram



### b. Class Specifications

#### Subject Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Get Subject’s properties | Get data of the properties which belong to the subject class |
| 2 | Set Subject’s properties | Set data of the properties which belong to the subject class |

#### Blog Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Get Blog’s properties | Get data of the properties which belong to the blog class |
| 2 | Set Blog’s properties | Set data of the properties which belong to the blog class |

#### Topic Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Get Topic’s properties | Get data of the properties which belong to the topic class |
| 2 | Set Topic’s properties | Set data of the properties which belong to the topic class |

#### SubjectDAO Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getSubjects(String condition) | Get data of the subject base on condition in the Database |
| 2 | getSubject(String sjtId) | Get data of one subject |
| 3 | getTopicBySubejectId(int s\_id) | Get data of the topic by subject id in the Database |
| 4 | getCompleteSubjects(String condition) | Get data of the subject (add price, expert name,…) base on condition in the Database |

#### BlogDAO Class

#### Home Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | Get blog, subject, expert from database |
| 2 | getRandomEntity(List totalList, List displayList, int n) | Generate random entity to get random expert, blog, subject |
| 3 | getExpert | Get data of random expert from the database |
| 4 | getBlogTags | Get data of random blog from the database |
| 5 | getRecommenedSubjectTags | Get data of recommened subject from the database |
| 6 | getLatestSubjectTags | Get data of new updated subject from the database |

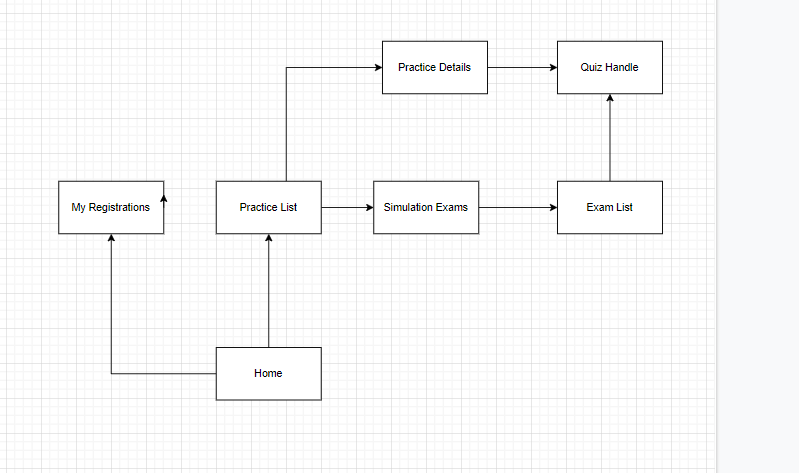
### c. Sequence Diagram(s)

### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | Query | Action |
| 1 | getCompleteSubjects | Select [Subject].\*, [PricePackage].price, [PricePackage].currency\_unit, [PricePackage].discount, [User].first\_name, [User].last\_name from [Subject] join [PricePackage] on [Subject].s\_id=[PricePackage].s\_id join [User] on [Subject].instructor\_id=[User].u\_id where [PricePackage].[status]=1 and [Subject].[status]=1 | Get Complete Subject |
| 2 | getSubjects | select \* from [QuizPractice].[dbo].[Subject] | Get All Subject base on condition |
| 3 | getTopicBySubejectId | select \* from [LessonTopic] where s\_id =? | Change user password into the database |

**3. <Customer Feature>**

### a. Class Diagram



### b. Class Specifications

#### Quizz Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Get Quizz’s properties | Get data of the properties which belong to the quizz class |
| 2 | Set Quizz’s properties | Set data of the properties which belong to the quizz class |

#### Exam Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Get Exam’s properties | Get data of the properties which belong to the exam class |
| 2 | Set Exam’s properties | Set data of the properties which belong to the exam class |

#### ExamDAO Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getExamsByLessonId(int s\_id, String condition) | Get data of the exam base on subject id and condition in the Database |

#### PracticeListController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | Get previous practices of user from database |
| 2 | newPracticeSelection | Get subject and topic base on subject from database |

#### ExamListController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | Get exam list base on customer id from database |

### c. Sequence Diagram(s)

### d. Database queries

## 8. <Admin Feature>

a. Class Diagram

b. Class Specifications

#### User Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Get User’s properties | Get data of the properties which belong to the user class |
| 2 | Set User’s properties | Set data of the properties which belong to the user class |

#### UserDAO Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | register(User user) | set data of the user in the Database |
| 2 | updateProfile(User user) | Update profile of the user in the Database |
| 3 | changePass(String pass,int userID ) | Change user password and update in the Database |
| 4 | isExisted(User u) | Check whether the user account or email existed or not |
| 5 | getUser(User u) | Get the user name and password and status |
| 6 | GetUsers() | Get all the user from database |

#### EditProfileController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | Get account and compare the input and the data in the data base |
| 2 | Request.setAtribute | Set the message for the alert of the login |
| 3 | Cookies.setcookies | Set the cookies of the username and password |
| 4 | Request.getRequestDispacher | Redirect the user to the screen after login |
| 5 | updateProfile(newUser) | Update the user to the database |

#### RegisterController Class

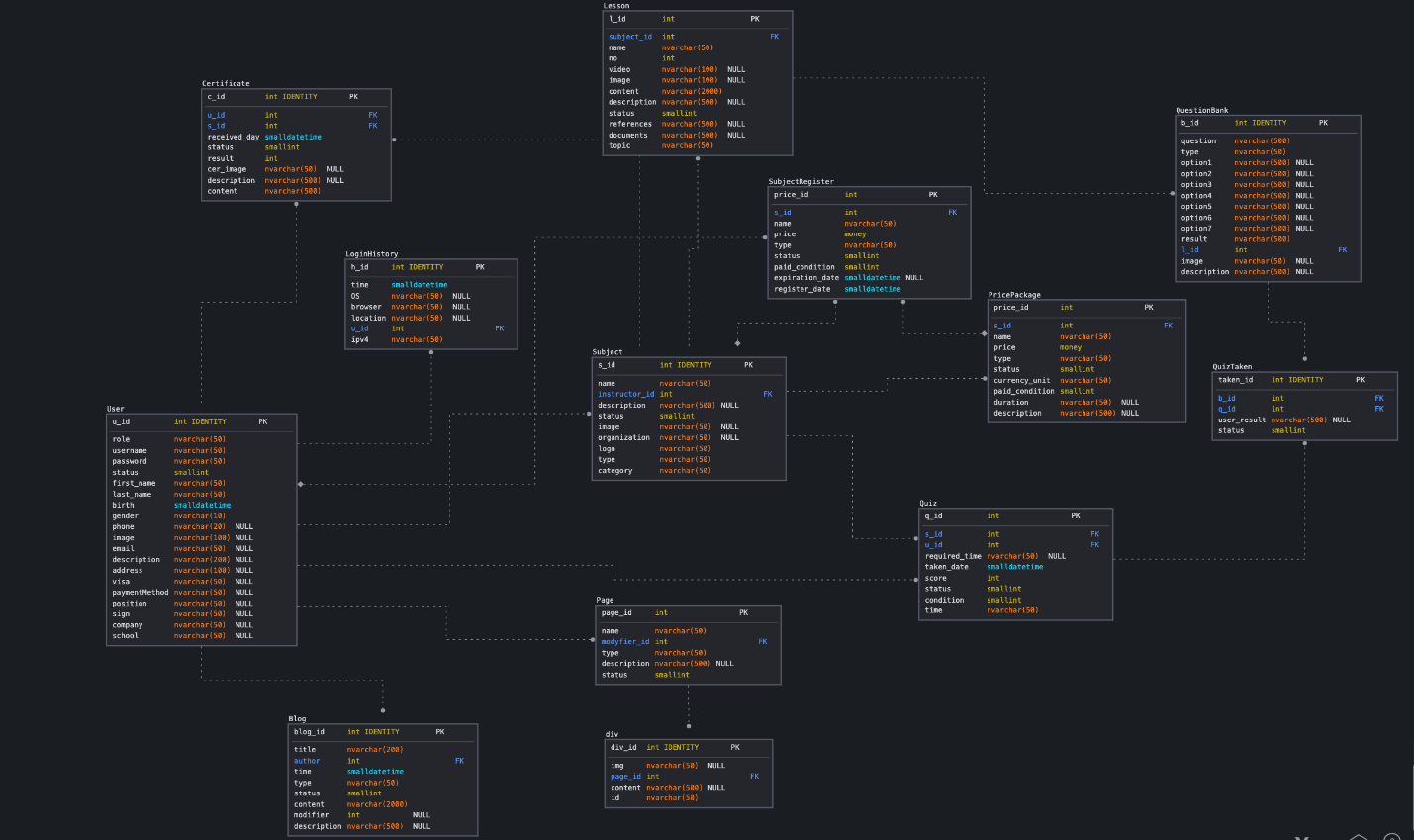
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | Get account and compare the input and the data in the data base |
| 2 | Request.setAtribute | Set the message for the alert of the login |
| 3 | Cookies.setcookies | Set the cookies of the username and password |
| 4 | Request.getRequestDispacher | Redirect the user to the screen after login |
| 5 | isExisted(User u) | Check whether the user account or email existed or not |
| 6 | register(u) | Insert new user profile to the Database |

c. Sequence Diagram(s)

d. Database queries

# III. Database Design

## 1. Database Schema



## 2. Table Description

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| *01* | *<Table name>* | *<Description of the table>*  *- Primary keys: <<list of primary key fields>>*  *- Foreign keys: <<list of foreign key fields>>* |
| *02* |  |  |