==



**Online learn**

**Software Design Specification**

– Hanoi, August 2022 –

Record of changeS

|  |  |  |  |
| --- | --- | --- | --- |
| Date | A\* M, D | In charge | Change Description |
| 18/02 | Add | ThangPM | Add code packages,database design |
| 02/03 | Add | ThangPM | Add code designs(feature,function) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 4](#_Toc96516286)

[1. Code Packages 4](#_Toc96516287)

[2. Database Design 4](#_Toc96516288)

[a. Database Schema 4](#_Toc96516289)

[b. Table Description 4](#_Toc96516290)

[II. Code Designs 5](#_Toc96516291)

[1. <Feature/Function Name1> 5](#_Toc96516292)

[a. Class Diagram 5](#_Toc96516293)

[b. Class Specifications 5](#_Toc96516294)

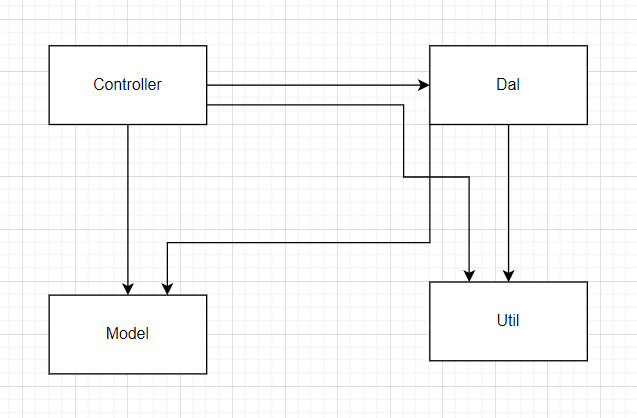
[c. Sequence Diagram(s) 5](#_Toc96516295)

[d. Database queries 6](#_Toc96516296)

[2. <Feature/Function Name2> 6](#_Toc96516297)

# I. Overview

## 1. Code Packages



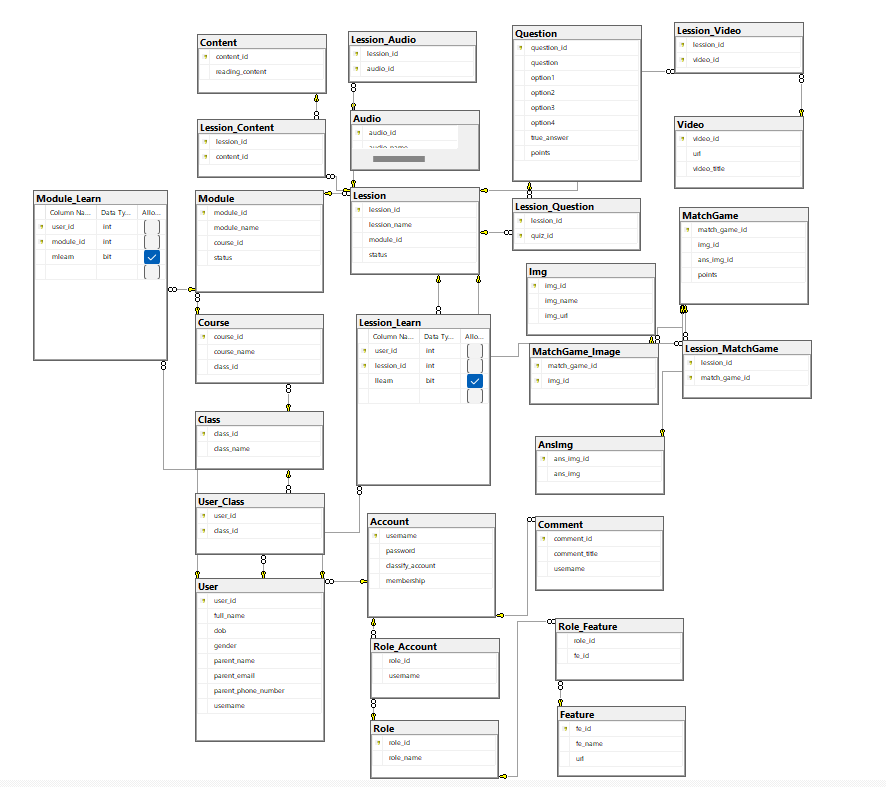
***Package descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| *01* | *Controller* | *Handle back-end, drive flow of code* |
| *02* | *Dal* | *Handle database* |
| *03* | *Model* | *Create object suitable database* |
| *04* | *Util* | *Support* |
|  |  |  |
|  |  |  |
|  |  |  |

## 2. Database Design

### a. Database Schema

*[Provide the tables relationship like example below – following MySQL database naming convention]*



### b. Table Description

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| *01* | *<Table name>* | *<Description of the table>* |

# II. Code Designs

## 1. Login

### a. Class Diagram

### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response). | guide the input data stream. |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response). | move to processRequest(HttpServletRequest request, HttpServletResponse response). |
| 03 | processRequest(HttpServletRequest request, HttpServletResponse response). | Handle the data stream about login. |

### c. Sequence Diagram(s)

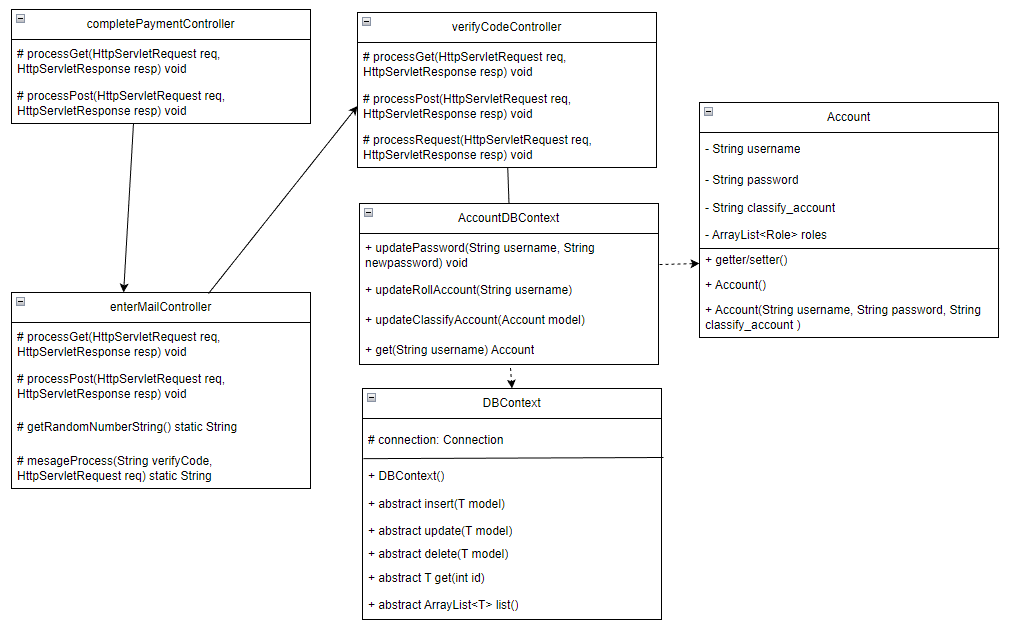
### d. Database Queries

"SELECT \* FROM Account \n" + "WHERE username = ? \n" + "and password = ?"

## 2. <Feature/Function Name2>

…

**Complete Payment**

****

b. Class Specifications

#### completePaymentController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processGet(HttpServletRequest, HttpServletResponse): void | To handle request, set attribute and send redirect to CompletePayment.jsp. |
| 02 | processPost(HttpServletRequest, HttpServletResponse): void | Get data and check format, if the format is correct send redirect to EmailAcception.jsp. |

#### enterMailController

#### **Class Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processGet(HttpServletRequest, HttpServletResponse): void | To handle request, set attribute and send redirect to EmailAcception. |
| 02 | processPost(HttpServletRequest, HttpServletResponse): void | Get data, get random string from function getRandomString() and message from messageProcess(, check format email, if the format is correct send redirect to EncodeEmail.jsp. Otherwise, back to EmailAcception.jsp. |
| 03 | getRandomNumberString() String | Random 1 string consisting of 6 digits, return this string |
| 04 | messageProcess(String, HttpServletRequest) | Returns 1 message, this message will be used as the body of the email |

#### VerifyCodeController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get verifyCode from session name “verifyCode”  And verifyCode from input of user. If it matches, update classify\_account to “premium”, update Roll\_Account send redirect to PaymentSuccesful.html. On the other hand, send redirect to paymentfail.html |
| 02 | processPost(HttpServletRequest, HttpServletResponse): void | Call processRequest() |
| 03 | processGet(HttpServletRequest, HttpServletResponse): void | Call processRequest() |

#### AccountDBContext

***Class Methods***

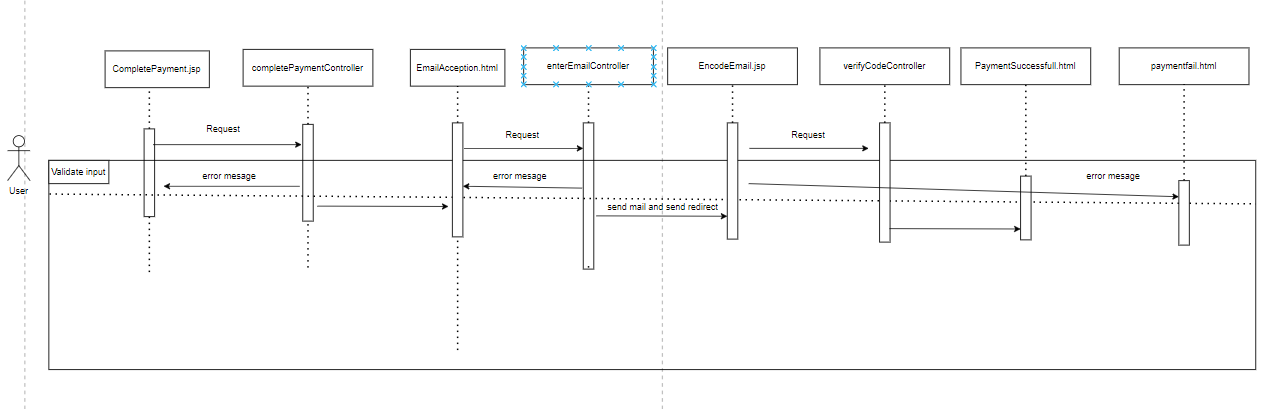
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | updateClassifyAccount(Account model) | Update classify\_account from normal to premium |
| 02 | Get(String name) | Get an Account by userName |

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | insert(T Model) | Abstract class (insert a model) |
| 03 | update(T Model) | Abstract class (update a model) |
| 02 | delete(T Model) | Abstract class (delete a model) |
| 02 | T get(int id) | Abstract class (get a model by id of this model) |
| 02 | ArrayList<T> list() | Abstract class (get an ArrayList model) |

c. Sequence Diagram(s)



d. Database Queries

// Update classify account

UPDATE [Account]

SET

[classify\_account] = 'premium'

WHERE username = ?

//Select top 8 of promotional product

SELECT a.username,a.classify\_account

,r.role\_id, r.role\_name,

f.fe\_id,f.fe\_name,f.[url]

FROM Account a

LEFT JOIN Role\_Account ra ON a.username = ra.username

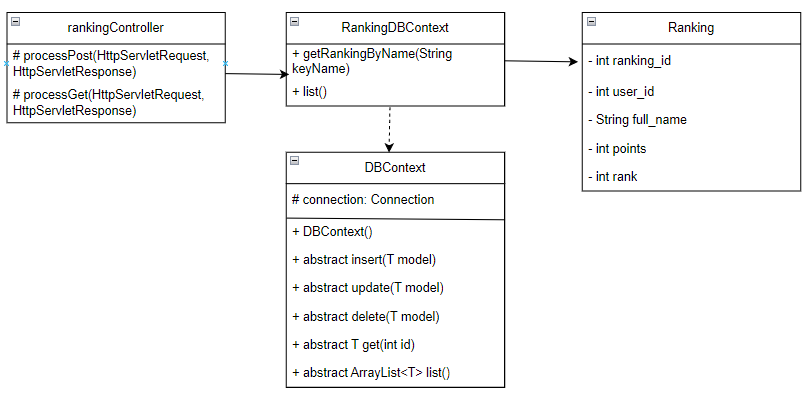
LEFT JOIN [Role] r ON r.role\_id= ra.role\_id

LEFT JOIN Role\_Feature rf ON rf.role\_id = r.role\_id

LEFT JOIN Feature f ON f.fe\_id = rf.fe\_id\

WHERE a.username = ?

**Ranking**

****

b. Class Specifications

#### rankingController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processGet(HttpServletRequest, HttpServletResponse): void | Load data from DB and display to jsp |
| 02 | processPost(HttpServletRequest, HttpServletResponse): void | Get data, load data from DB display to jsp |

#### RankingDBContext

***Class Methods***

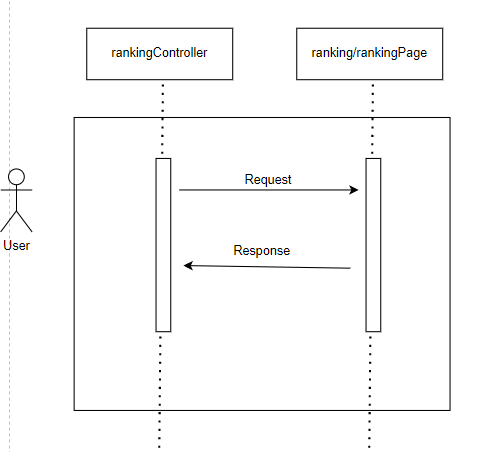
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | List() | Load ranking of all student and sort by point |
| 02 | getRankingByName (String keyName) | Load ranking of student (contain keyName in Name) and sort by point |

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | insert(T Model) | Abstract class (insert a model) |
| 03 | update(T Model) | Abstract class (update a model) |
| 02 | delete(T Model) | Abstract class (delete a model) |
| 02 | T get(int id) | Abstract class (get a model by id of this model) |
| 02 | ArrayList<T> list() | Abstract class (get an ArrayList model) |

c. Sequence Diagram(s)



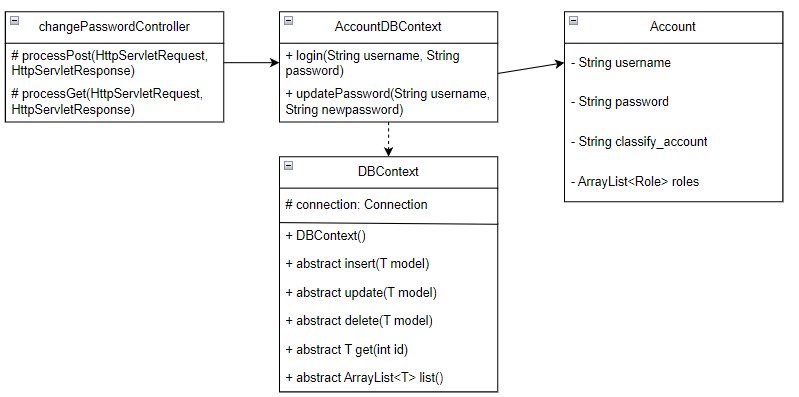
d. Database Queries

// Load ranking of all student and sort by point

SELECT ROW\_NUMBER() OVER(ORDER BY r.points DESC) AS [rank], r.ranking\_id,r.[user\_id],r.full\_name,r.points\n"

+ "FROM Ranking r

**Change password**

****

b. Class Specifications

#### changePasswordController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processGet(HttpServletRequest, HttpServletResponse): void | send redirect to changePassword.jsp |
| 02 | processPost(HttpServletRequest, HttpServletResponse): void | Check account with old pass existed, check input new password and repeat password, update password for this account. Other, throw error message to changePassword.jsp |

#### AccountDBContext

***Class Methods***

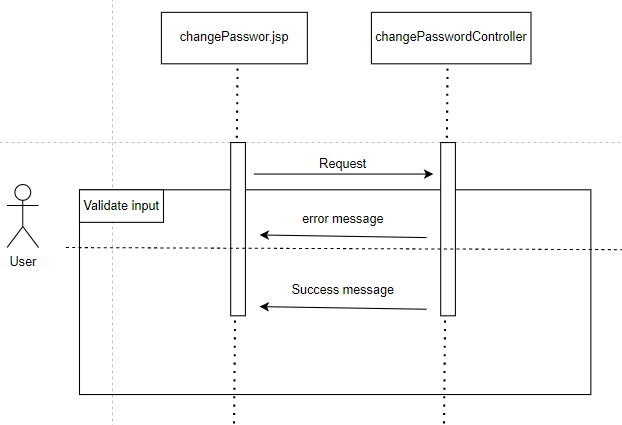
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | login(String username, String password) | check in the DB if there is an account with this username and password |
| 02 | updatePassword(String username, String newpassword) | Update password for this account |

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | insert(T Model) | Abstract class (insert a model) |
| 03 | update(T Model) | Abstract class (update a model) |
| 02 | delete(T Model) | Abstract class (delete a model) |
| 02 | T get(int id) | Abstract class (get a model by id of this model) |
| 02 | ArrayList<T> list() | Abstract class (get an ArrayList model) |

c. Sequence Diagram(s)



d. Database Queries

// Update password

UPDATE [dbo].[Account]

SET [password] = ?

WHERE username = ?

//get an account with this info

SELECT a.username,a.classify\_account

,r.role\_id, r.role\_name,

f.fe\_id,f.fe\_name,f.[url]

FROM Account a

LEFT JOIN Role\_Account ra ON a.username = ra.username

LEFT JOIN [Role] r ON r.role\_id= ra.role\_id

LEFT JOIN Role\_Feature rf ON rf.role\_id = r.role\_id

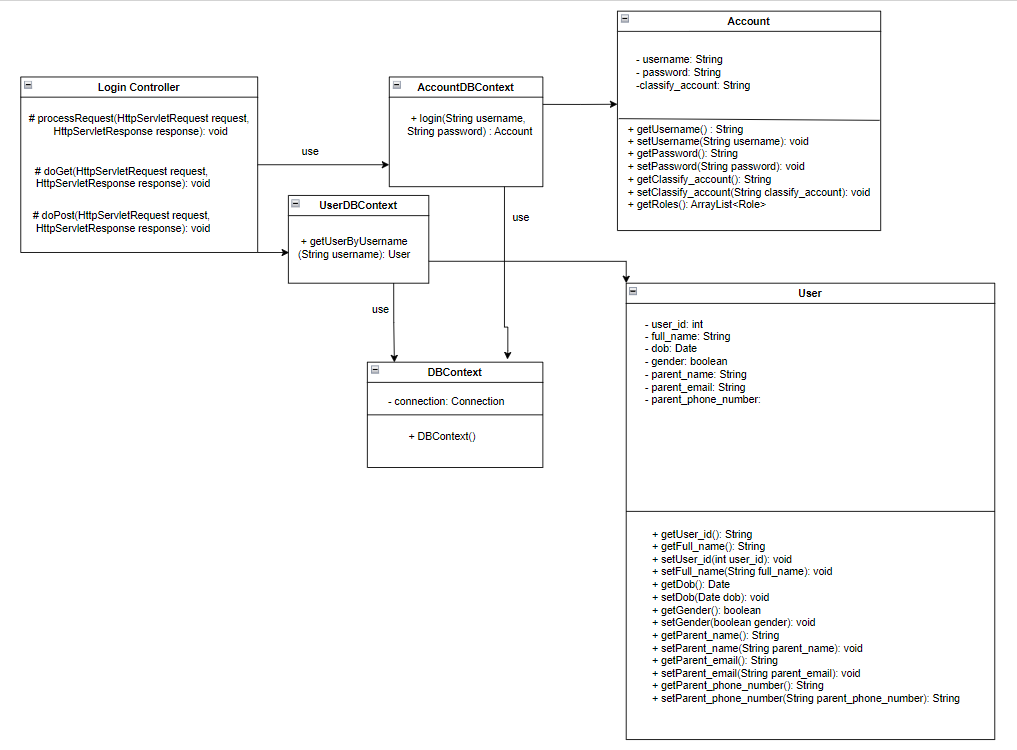
LEFT JOIN Feature f ON f.fe\_id = rf.fe\_id

WHERE a.username = ? and a.password = ?

II. Code designs

## 1. Login

a. Class Diagram



b. Class Specifications

#### controller.auth

***SignupController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### dal

***Class AccountDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | login(String username, String password): Account | check username and password is correct or not |

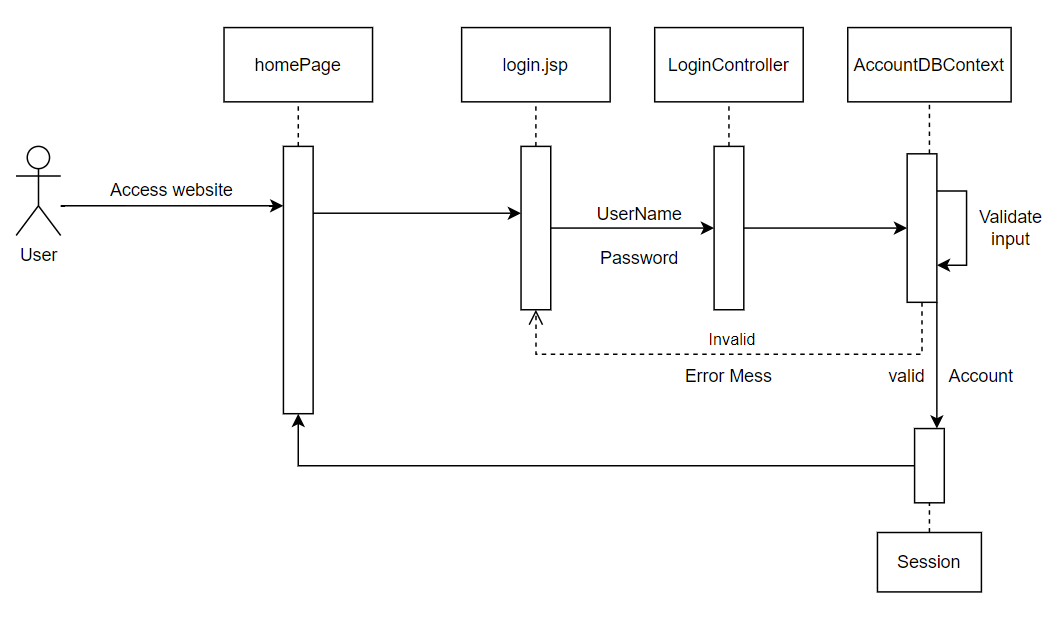
***Class UserDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getUserByUsername(String username): User | Get an user base on username |

***Class DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |

c. Sequence Diagram(s)



d. Database Queries

// Select an account has username and password correct

SELECT a.username,a.classify\_account\n"

+ ",r.role\_id, r.role\_name,\n"

+ "f.fe\_id,f.fe\_name,f.[url]\n"

+ "FROM Account a\n"

+ "LEFT JOIN Role\_Account ra ON a.username = ra.username\n"

+ "LEFT JOIN [Role] r ON r.role\_id= ra.role\_id\n"

+ "LEFT JOIN Role\_Feature rf ON rf.role\_id = r.role\_id\n"

+ "LEFT JOIN Feature f ON f.fe\_id = rf.fe\_id\n"

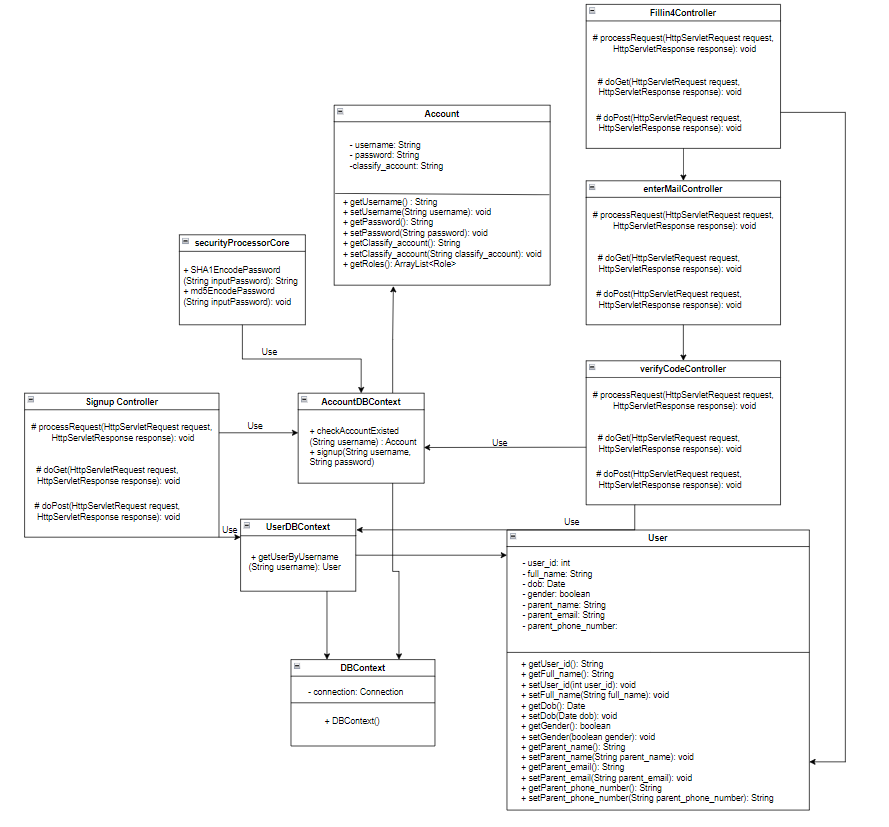
+ "WHERE a.username = ? AND a.password = ?"

//Select user base on username

SELECT \* FROM [User] WHERE username = ?

## 2. Signup

a. Class Diagram



b. Class Specifications

#### controller.auth

#### **Class** SignupController

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class Fillin4.Controller

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |

|  |  |  |
| --- | --- | --- |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class sendmail.Controller

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getRandomNumberString(): String | Create OTP code |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class verifyCode.Controller

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class sercurityProcessorCore

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | SHA1EncodePassword(String inputPassword): String. | Change a string to SHA1 code. |
| 02 | md5EncodePassword(String inputPassword): String. | Change a string to md5 code. |

#### dal

***Class AccountDBContext***

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | signup(String username, String password): void | Add new account into database. |
| 02 | checkAccountExisted(String username): Account | Check account has in database yet or not. |

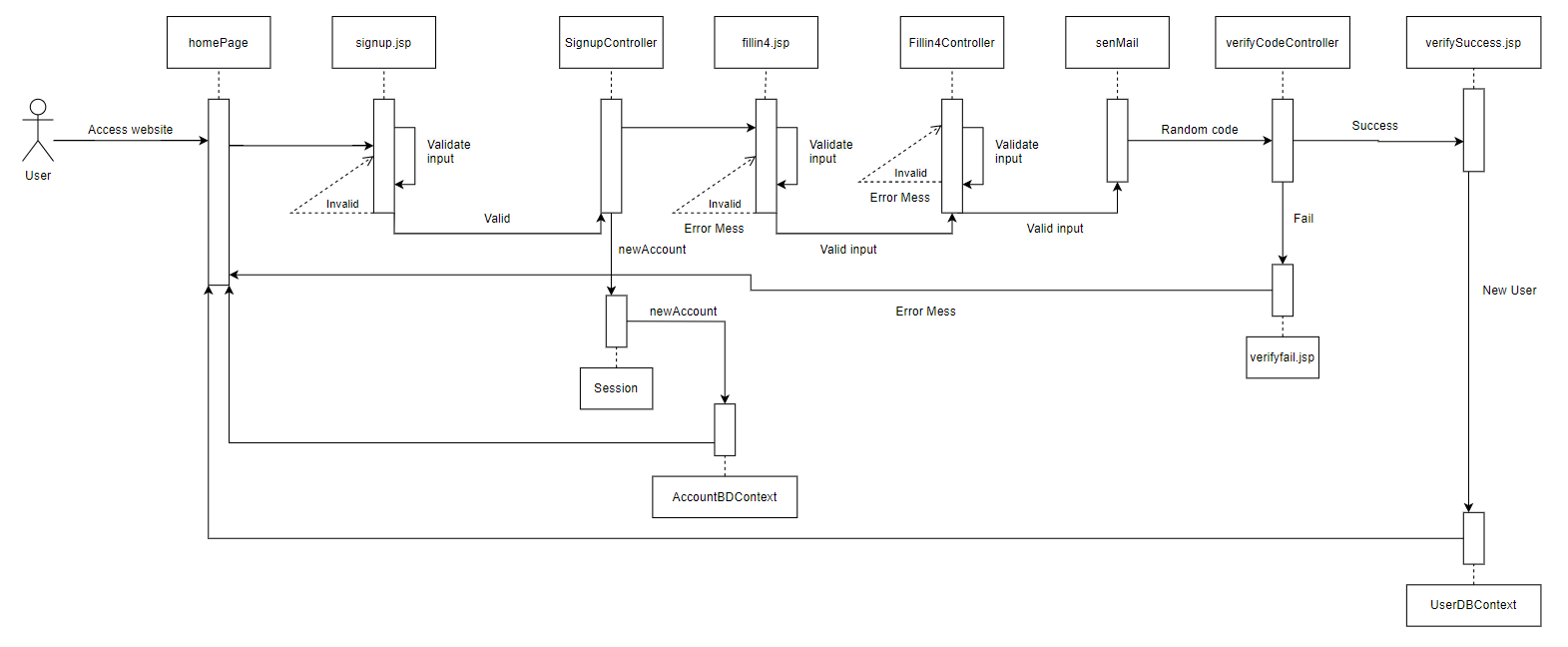
***Class UserDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getUserByUsername(String username): User | Get an user base on username |

***Class DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |

c. Sequence Diagram(s)



d. Database Queries

// Insert an account into database

"INSERT INTO [dbo].[Account]\n"

+ " ([username]\n"

+ " ,[password]\n"

+ " ,[classify\_account])\n"

+ " VALUES\n"

+ " (?,\n"

+ " ?,\n"

+ " 'normal')"

//Insert information for new user

"INSERT INTO [dbo].[Account]\n"

+ " ([username]\n"

+ " ,[password]\n"

+ " ,[classify\_account])\n"

+ " VALUES\n"

+ " (?,\n"

+ " ?,\n"

+ " 'nomal')"

//Update roll-account

"UPDATE [dbo].[Role\_Account]\n"

+ " SET [role\_id] = 1\n"

+ " WHERE username = ?"

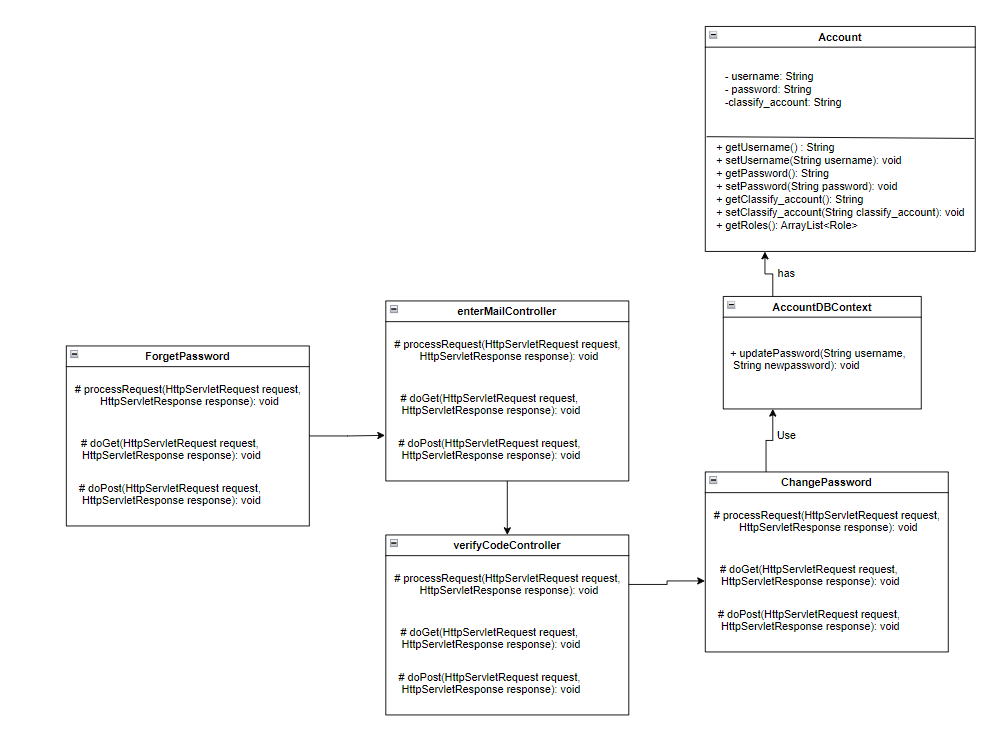
//Select user base on username

SELECT \* FROM Account \n"

+ "WHERE username = ?

## 3. ForgetPassword

a. Class Diagram



b. Class Specifications

#### controller.forgetPassword

Class ForgetPassword

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class SendMail

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |

|  |  |  |
| --- | --- | --- |
| 01 | getRandomNumberString(): String | Create OTP code |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class verifyCode

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### dal

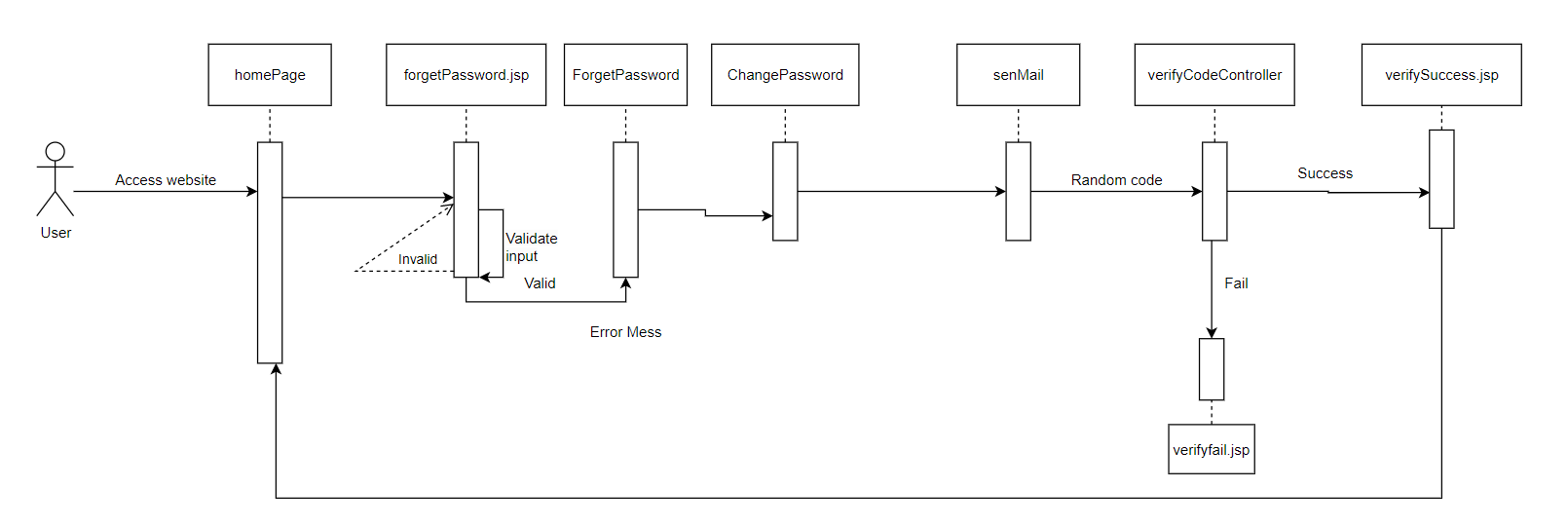
***Class AccountDBContext***

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | updatePassword(String username, String newpassword) | Update new password for account. |

***Class DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |

c. Sequence Diagram(s)



d. Database Queries

// Update new password

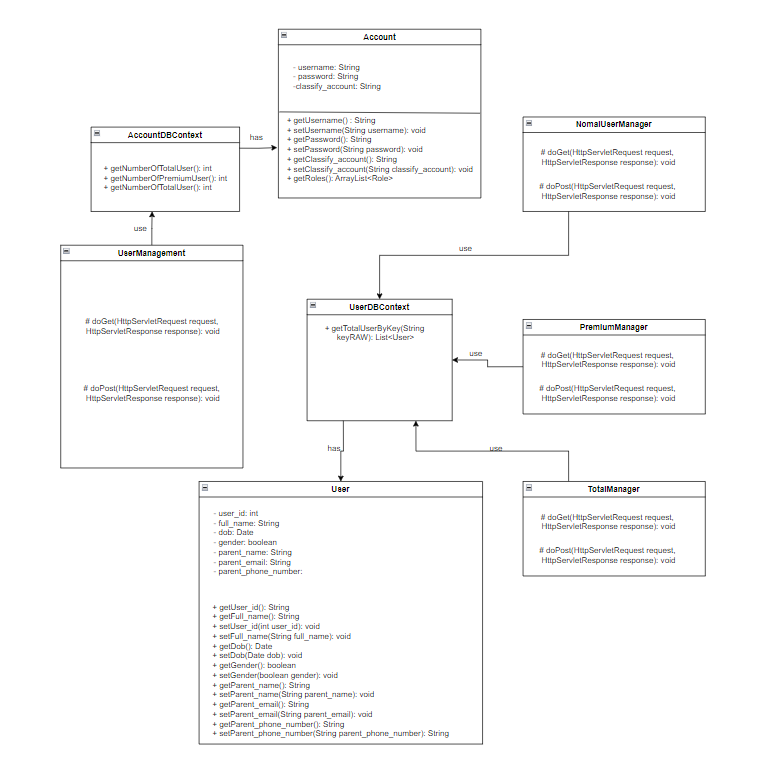
"UPDATE [dbo].[Account]\n"

+ " SET [password] = ?\n"

+ " WHERE username = ?" + " 'normal')"

## 4. UserManagement

a. Class Diagram



b. Class Specifications

#### controller.admin.viewUser

Class UserManagerment

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class NomalUserManager

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class PremiumManager

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

Class TotalManager

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### dal

***Class AccountDBContext***

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getNumberOfPremiumUser(): int | Return number of premium user |
| 02 | getNumberOfNormalUser(): int | Return number of premium user |
| 03 | getNumberOfTotalUser(): int | Return number of total user |

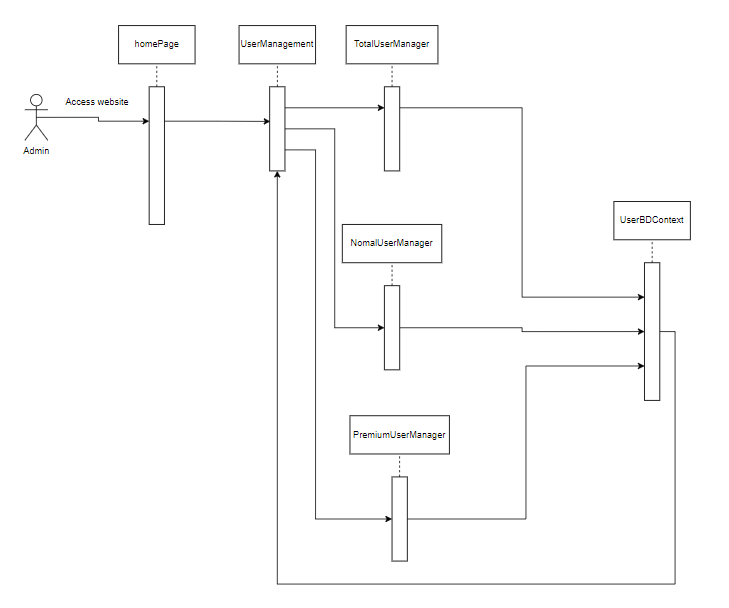
ClassUserDBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getPremiumUserByKey(String keyRAW): List<User> | Return a list of premium user |
| 02 | getNormalUserByKey(String keyRAW): List<User> | Return a list of nomal user |
| 03 | getTotalUserByKey(String keyRAW): List<User> | Return a list of total user |

***Class DBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |

c. Sequence Diagram(s)



d. Database Queries

// Get a list of premium user

"SELECT u.full\_name,u.dob,u.gender,u.parent\_email,u.parent\_name,u.parent\_phone\_number,u.username FROM [User] u \n"

+ "JOIN Account a ON a.username = u.username\n"

+ "WHERE u.full\_name LIKE ? AND a.classify\_account = 'premium'"

// Get a list of nomal user

"SELECT u.full\_name,u.dob,u.gender,u.parent\_email,u.parent\_name,u.parent\_phone\_number,u.username FROM [User] u \n"

+ "JOIN Account a ON a.username = u.username\n"

+ "WHERE u.full\_name LIKE ? AND a.classify\_account = 'normal'"

// Get a list of total user

"SELECT u.full\_name,u.dob,u.gender,u.parent\_email,u.parent\_name,u.parent\_phone\_number,u.username FROM [User] u \n"

+ "JOIN Account a ON a.username = u.username\n"

+ "WHERE u.full\_name LIKE ?"

// Get number of total user

"SELECT COUNT(\*) AS total FROM [User] u"

// Get number of premium user

"SELECT COUNT(\*) AS premium FROM [User] u\n"

+ "JOIN Account a ON a.username = u.username AND a.classify\_account = 'premium'"

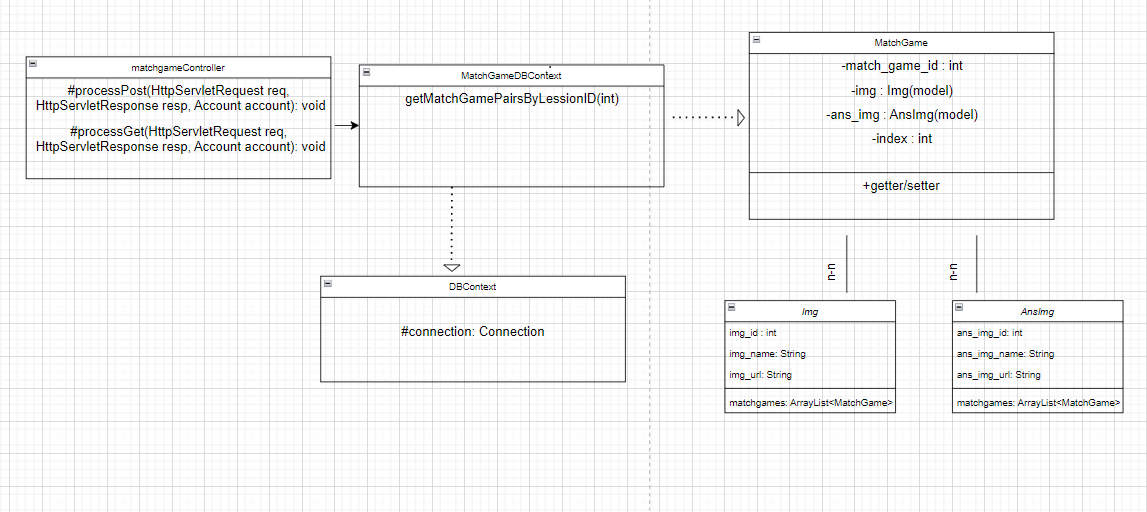
// Get number of nomal user

"SELECT COUNT(\*) AS normal FROM [User] u \n"

+ "JOIN Account a ON a.username = u.username AND a.classify\_account = 'normal'"

1.Match Game Page

a.Class Diagram



b.Class Specifications

#### matchgameController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

#### MatchGameDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getMatchGamePairsByLessionID(int) ArrayList<MatchGame> | Get a list of match game based on: what is id of lesson\_id |

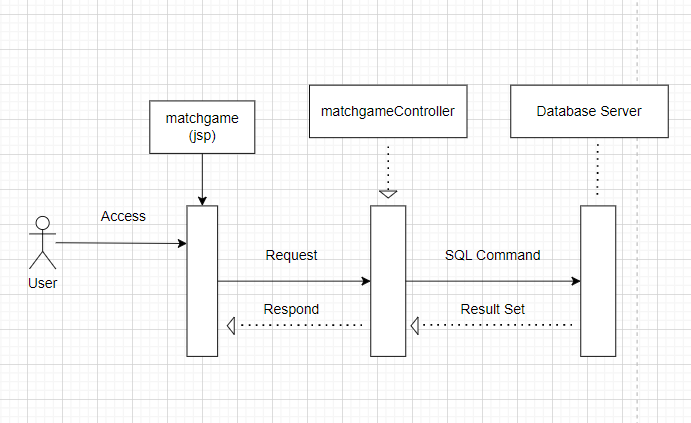
#### 

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |

c. Sequence Diagram(s)



d. Database Queries

//Select match game by lesson id

SELECT m.match\_game\_id, m.img\_id, m.ans\_img\_id, m.[index],

                        i.img\_name, i.img\_url,

                     a.ans\_img\_name, a.ans\_img\_url

                     FROM MatchGame m

                   Inner join Img i on i.img\_id = m.img\_id

                   Inner join AnsImg a on a.ans\_img\_id = m.ans\_img\_id

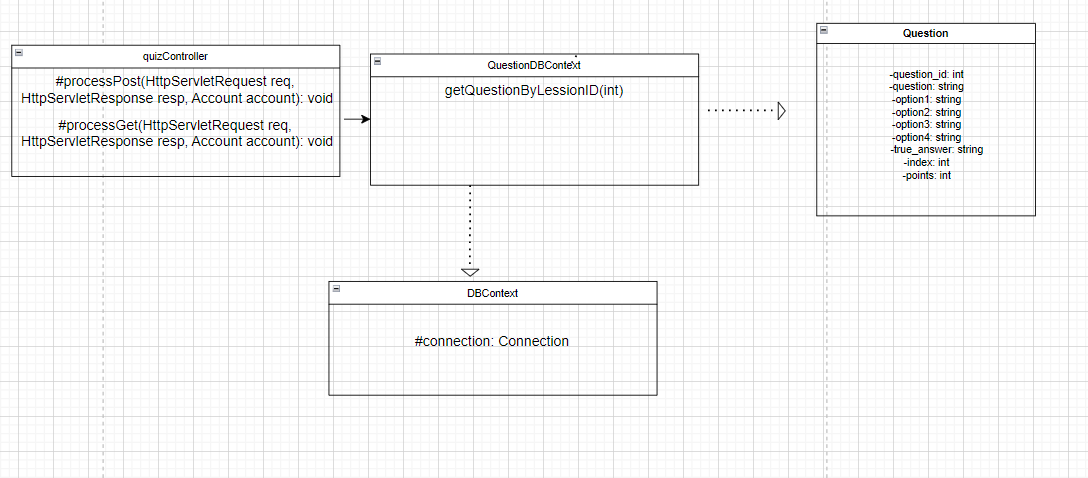
                   Inner join Lession\_MatchGame lm on lm.match\_game\_id = m.match\_game\_id

                   Inner join Lession l on l.lession\_id = lm.lession\_id

                   WHERE l.lession\_id = ?

2.Quiz Page

a.Class Diagram



b.Class Specifications

#### quizController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

#### QuestionDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getQuestionByLessionID(int):ArrayList<Question> | Get a list of question based on: what is id of lesson\_id |

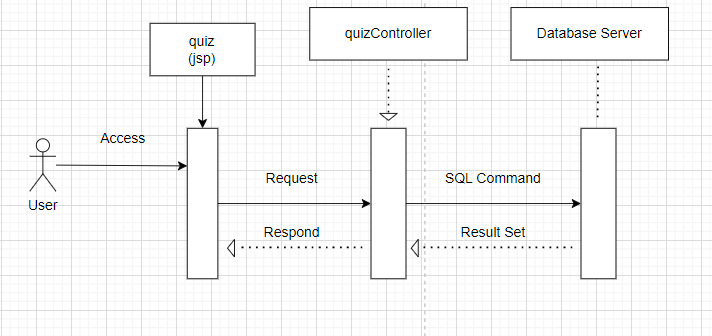
#### 

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |

c. Sequence Diagram(s)



d. Database Queries

//Select question by lesson id

SELECT qu.[index], qu.question\_id, qu.question, qu.option1, qu.option2,

                    qu.option3, qu.option 4, qu.true\_answer, qu.points

                   FROM Question qu

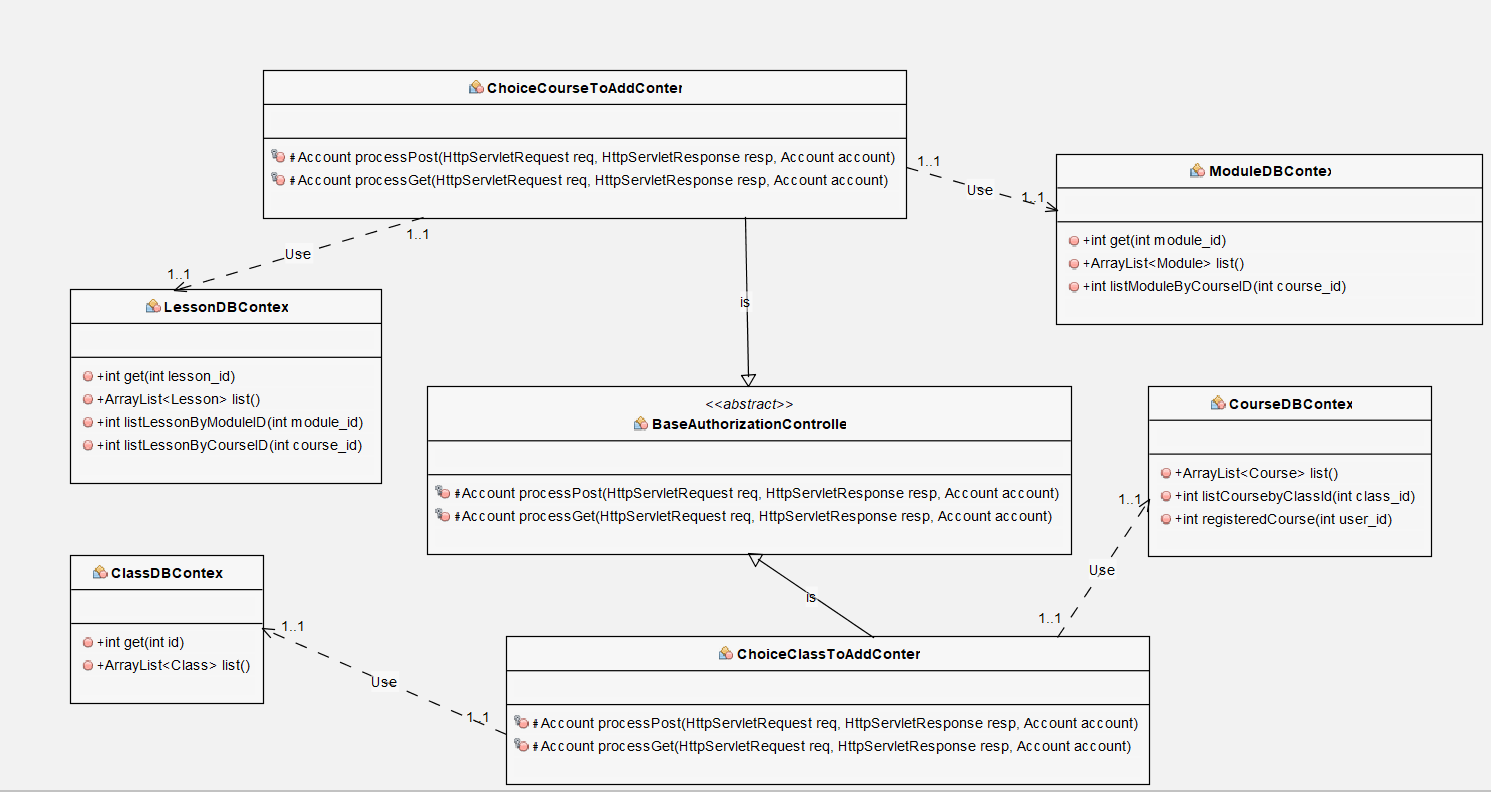
                    inner join Lession\_Question lq on lq.question\_id = qu.question\_id

                   inner join Lession l on l.lession\_id = lq.lession\_id

                    WHERE l.lession\_id = ?

3.choose to add lesson for content page

a, Class Diagram



b.Class Specifications

***ChoiceClassToAddContent***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

***ClassDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | list(): ArrayList<Class> | Get list of classes |

***CourseDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | listCoursebyClassId(int class\_id): ArrayList<Class> | Get list of courses by class id |

***ChoiceCourseToAddContent***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

***ModuleDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | listModuleByCourseID(int course\_id): ArrayList<Module> | Get list of modules by course id |

***LessonDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | listLessonByModuleID(int module\_id): ArrayList<Lesson> | Get list of lessons by module id |

4. Admin content page

a. Class Diagram

b.Class Specifications

***AddContentController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

***ContentDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getContentListByLessionID(int lession\_id): ArrayList<Content> | Get list of contents by lesson id |
| 02 | insert(Content model, int lesson\_id): void | Insert new content into database |
| 03 | update(Content model): void | Update content into database |
| 04 | deleteLessonContent(int content\_id): void | Delete table that have foreign key of content id in database |
| 05 | delete(int content id): void | Delete content in database |

***LessonDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | get(int lesson\_id): Lesson | Get lesson by lesson id |

***actionContentController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

c. Sequence Diagram(s)

d. Database Queries

// Delete content

DELETE FROM [dbo].[Content]  WHERE content\_id = ?

//Delete lesson content

 DELETE FROM [dbo].[Lession\_Content]   WHERE content\_id = ?

//Update content

UPDATE [dbo].[Content]

                SET [reading\_content] = ?

                WHERE content\_id = ?

// Insert content

INSERT INTO [dbo].[Content]

                    ([reading\_content])

                 VALUE (?)

//Get list of contents by lesson id

SELECT c.content\_id, c.reading\_content

                   FROM Content c

                  inner join Lession\_Content lc on lc.content\_id = c.content\_id

                inner join Lession l on l.lession\_id = lc.lession\_id

                 WHERE l.lession\_id = ?

5. Add match game image page (admin)

a. Class Diagram

b.Class Specifications

***AddPairsOfImageController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

***MatchGameDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | insertImg(Img model): void | Insert new image into Database |
| 02 | insertAnsImg(AnsImg model): void | Insert new answer image content into database |
| 03 | updateImg(Img model): void | Update image into database |
| 04 | updateAnsImg(Img model): void | Update answer image into database |
| 05 | insertMatchGame(MatchGame model, int lesson\_id, Img model, AnsImg model): void | Insert new match game pairs into database |
| 06 | getMatchGamePairsByLessionID(int lesson\_id):ArrayList<MatchGame> | Get list of match game image pairs |

***actionMatchGameController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

c. Sequence Diagram(s)

d. Database Queries

// Get match game pairs by lesson id

SELECT m.match\_game\_id, m.img\_id, m.ans\_img\_id, m.[index],

                       i.img\_name, i.img\_url,

                       a.ans\_img\_name, a.ans\_img\_url

                      FROM MatchGame m

                     inner join Img i on i.img\_id = m.img\_id

                     inner join AnsImg a on a.ans\_img\_id = m.ans\_img\_id

                    + "     inner join Lession\_MatchGame lm on lm.match\_game\_id =m.match\_game\_id

                   inner join Lession l on l.lession\_id = lm.lession\_id

                   WHERE l.lession\_id = ?

// Update Match game

"UPDATE [dbo].[MatchGame]\n"

                + "   SET [img\_id] = ?\n"

                + "      ,[ans\_img\_id] = ?\n"

                + "      ,[points] = ?\n"

                + "      ,[index] = ?\n"

                + " WHERE match\_game\_id = ?";

// Insert Match Game

INSERT INTO [dbo].[MatchGame]

                 ([img\_id]

                 ,[ans\_img\_id]

                 ,[points]

                 ,[index])

                 VALUES (?, ?, ?, ?)

// Insert Lesson Match Game

"INSERT INTO [dbo].[Lession\_MatchGame]\n"

                + "           ([lession\_id]\n"

                + "           ,[match\_game\_id])\n"

                + "     VALUES\n"

                + "           (?\n"

                + "           ,?)";

// Insert image

"INSERT INTO [dbo].[Img]

                + "           ([img\_name]

                + "           ,[img\_url])

                + "     VALUES

                + "           (?

                + "           ,?)

//Insert answer image

"INSERT INTO [dbo].[AnsImg]\n"

                + "           ([ans\_img\_name]

                + "           ,[ans\_img\_url])

                + "     VALUES

                + "           (?

                + "           ,?)

6. Add module  page (admin)

a. Class Diagram

b.Class Specifications

***actionModuleController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processPost(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |
| 02 | processGet(HttpServletRequest req, HttpServletResponse resp, Account account): void | To handle request, get data from database and display to jsp and get authorization |

***ModuleDBContext***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | get(int module\_id): Module | Get a module by module id |
| 02 | listModuleByCourseID(int course\_id): ArrayList<Module> | Get list of modules by course id |
| 03 | insert(Module model): void | Insert new module into database |
| 04 | update(Module model): void | Update module into database |

c. Sequence Diagram(s)

d. Database Queries

// insert

"INSERT INTO [dbo].[Module]

                     ([module\_name]

                    ,[course\_id]

                      ,[status])

                + "     VALUES

                + "           (?

                + "           ,?

                + "           ,?)

// Update

"UPDATE [dbo].[Module]\n"

                 SET [module\_name] = ?

                     ,[course\_id] = ?

                     ,[status] = ?

                WHERE module\_id = ?

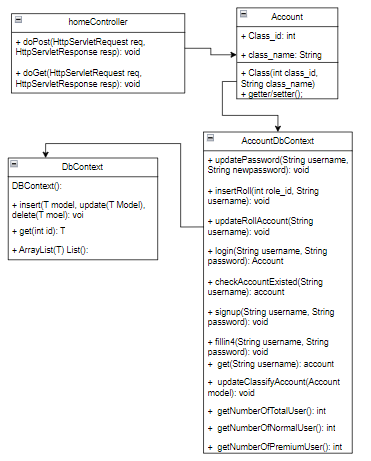
// Get list of modules by course id

SELECT \* FROM Module WHERE course\_id = ?

II. Code designs

## 1. Home Page

a. Class Diagram



b. Class Specifications

#### HomeController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call itself |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | req.getRequestDispatcher("./home.jsp").forward(req, resp); |

#### AccountDBContext

***Class Methods***

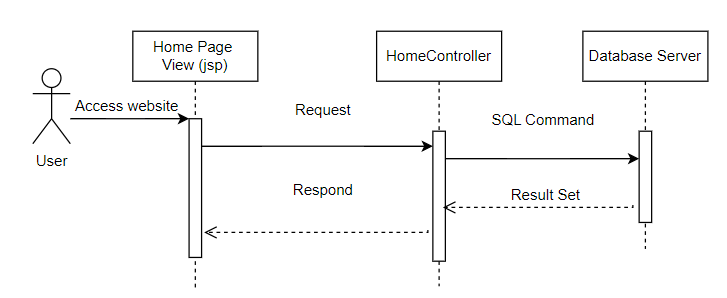
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | updatePassword(String username, String newpassword) | Update password |
|  | insertRoll(int role\_id, String username) | Insert roll for account |
|  | updateRollAccount(String username) | Update roll |
|  | login(String username, String password) | User login |
|  | checkAccountExisted(String username) | Check account existed or not |
|  | signup(String username, String password) | Sign up username and password |
|  | fillin4(String username, String password) | Insert information of user |
|  | get(String username) | Get account with username |
|  | updateClassifyAccount(Account model) | Update classify of account |
|  | getNumberOfTotalUser() | Get total user of web |
|  | getNumberOfNormalUser() | Get total account normal |
|  | getNumberOfPremiumUser() | Get total account premium |

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

c. Sequence Diagram(s)



d. Database Queries

//update password account

UPDATE [dbo].[Account

SET [password] = ?,

WHERE username = ?

//insert roll account

INSERT INTO [dbo].[Role\_Account

([role\_id],

[username])

VALUES

(?,?)

// updateRollAccount

UPDATE [dbo].[Role\_Account]

SET [role\_id] = 1

WHERE username = ?

//login(String username, String password)

SELECT a.username,a.classify\_account

+ ",r.role\_id, r.role\_name,

+ "f.fe\_id,f.fe\_name,f.[url]

+ "FROM Account a

+ "LEFT JOIN Role\_Account ra ON a.username = ra.username

+ "LEFT JOIN [Role] r ON r.role\_id= ra.role\_id

+ "LEFT JOIN Role\_Feature rf ON rf.role\_id = r.role\_id

+ "LEFT JOIN Feature f ON f.fe\_id = rf.fe\_id

+ "WHERE a.username = ? AND a.password = ?

// checkAccountExisted(String username)

SELECT \* FROM Account

WHERE username = ?

// signup(String username, String password)

INSERT INTO [dbo].[Account]

+ " ([username]

+ " ,[password]

+ " ,[classify\_account])

+ " VALUES

+ " (?,

+ " ?,

+ " 'normal')

//fillin4(String username, String password)

INSERT INTO [dbo].[Account]

([username]

,[password]

,[classify\_account])

VALUES

(?,

?,

'nomal')"

// get(String username)

SELECT a.username,a.classify\_account

r.role\_id, r.role\_name,

f.fe\_id,f.fe\_name,f.[url]

FROM Account a

LEFT JOIN Role\_Account ra ON a.username = ra.username

LEFT JOIN [Role] r ON r.role\_id= ra.role\_id

+ "LEFT JOIN Role\_Feature rf ON rf.role\_id = r.role\_id

+ "LEFT JOIN Feature f ON f.fe\_id = rf.fe\_id

+ "WHERE a.username = ?

// updateClassifyAccount(Account model)

UPDATE [Account]

SET

[classify\_account] = 'premium'

WHERE username = ?

// getNumberOfTotalUser()

SELECT COUNT(\*) AS total FROM [User] u

// getNumberOfNormalUser()

SELECT COUNT(\*) AS normal FROM [User] u

JOIN Account a ON a.username = u.username AND a.classify\_account = 'normal'

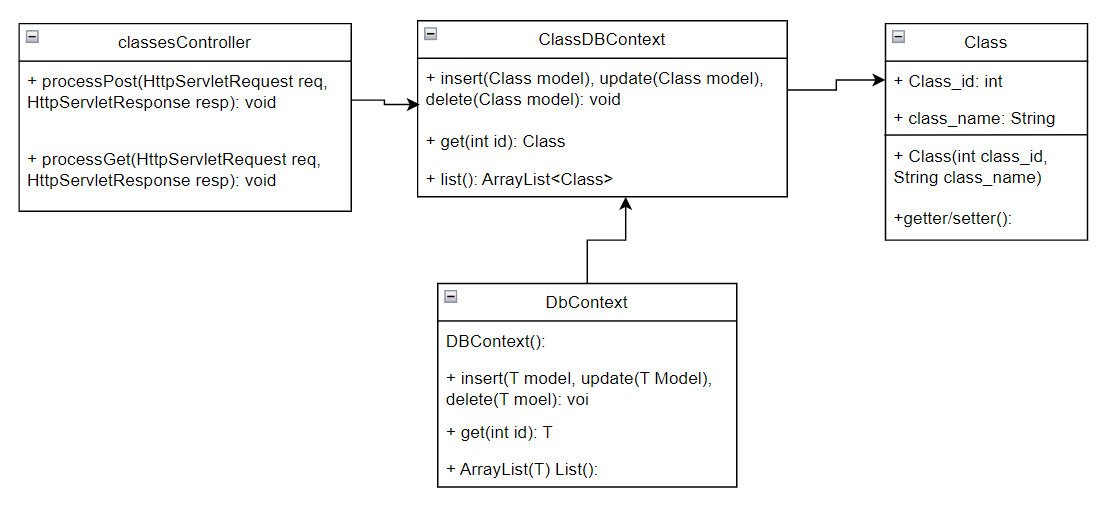
// getNumberOfPremiumUser()

SELECT COUNT(\*) AS premium FROM [User] u

JOIN Account a ON a.username = u.username AND a.classify\_account = 'premium'

## Chooses class

1. Class diagram:



1. Class specification

Class method

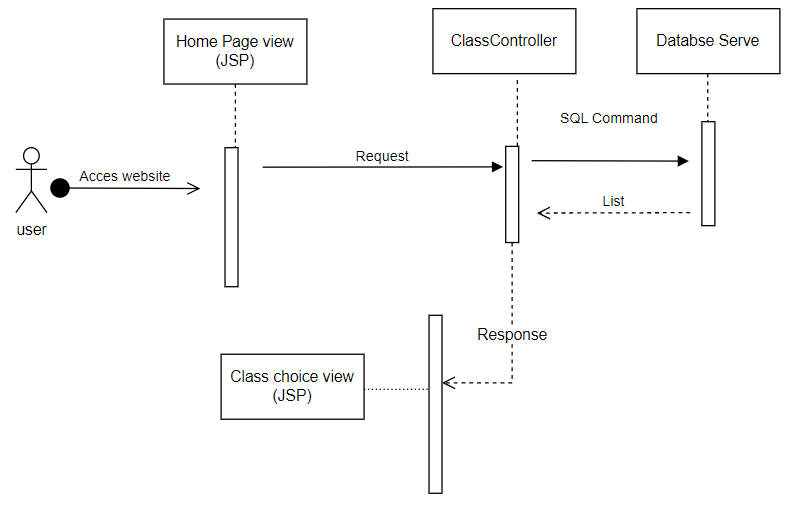
* classesController:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call itself |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Get list classes and req.getRequestDispatcher("./classes.jsp").forward(req, resp); |

* ClassDBContext:

|  |  |  |
| --- | --- | --- |
| * **No** | **Method** | **Description** |
| 01 | + insert(T model, update(T Model), delete(T moel) | Abstract method |
| 02 | list(): | Get list classes |

1. Sequence Diagram



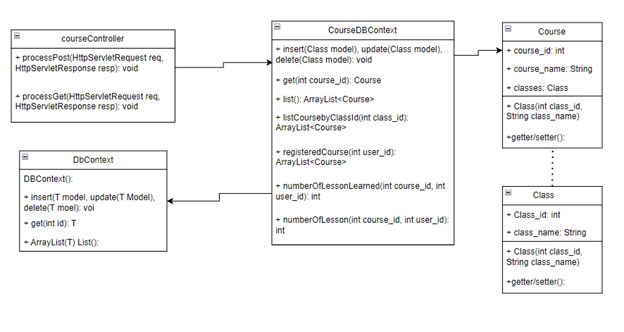
1. Database queries

* // list():

SELECT \* FROM Class

## 3. Choice Course

* 1. Class diagram



* 1. Class specification

Class method

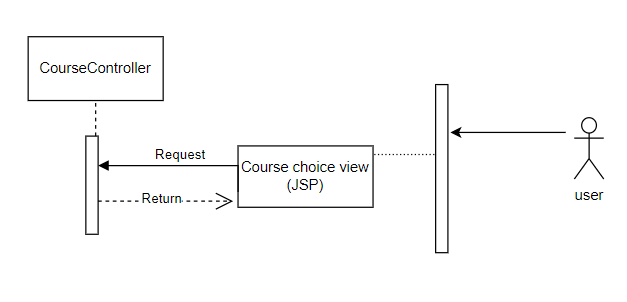
* CourseController:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call itself |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Get list course and req.getRequestDispatcher("./course.jsp").forward(req, resp); |

* CourseDBContext:

|  |  |  |
| --- | --- | --- |
| * **No** | **Method** | **Description** |
| 01 | + insert(T model, update(T Model), delete(T moel) | Abstract method |
| 02 | list() : | Get list course |
| 03 | listCoursebyClassId(int class\_id) | Get list course by class id |
| 04 | numberOfLessonLearned(int course\_id, int user\_id) | Get number of lesson learned |
| 05 | numberOfLesson(int course\_id, int user\_id) | Get number of lesson |
| 06 | registeredCourse(int user\_id) | Register course |
| 07 | get(int course\_id) | Get course by id |

* 1. Sequence diagram



* 1. Database queries

-// get(int course\_id):

SELECT \* FROM Course WHERE course\_id = ?

// list():

SELECT \* FROM Course

// listCoursebyClassId(int class\_id):

SELECT \* FROM Course WHERE class\_id = ?

// numberOfLessonLearned(int course\_id, int user\_id):

SELECT COUNT(\*) AS numberOfLessonLearned

FROM Lession l

JOIN Module m ON m.module\_id = l.module\_id

JOIN Course c ON c.course\_id = m.course\_id

JOIN Class cl ON cl.class\_id = c.class\_id

JOIN Lession\_Learn ll ON ll.lession\_id = l.lession\_id

WHERE c.course\_id = ? AND ll.llearn = 'true' AND ll.[user\_id] =?

// numberOfLesson(int course\_id, int user\_id):

SELECT COUNT(\*) AS numberOfLessonLearned\

FROM Lession l

JOIN Module m ON m.module\_id = l.module\_id

JOIN Course c ON c.course\_id = m.course\_id

JOIN Class cl ON cl.class\_id = c.class\_id

JOIN Lession\_Learn ll ON ll.lession\_id = l.lession\_id

WHERE c.course\_id = ? AND ll.[user\_id] =?

// registeredCourse(int user\_id):

SELECT DISTINCT c.course\_id, c.course\_name,c.class\_id, cl.class\_name FROM Course c

JOIN Class cl ON cl.class\_id = c.class\_id

JOIN Module m ON m.course\_id = c.course\_id

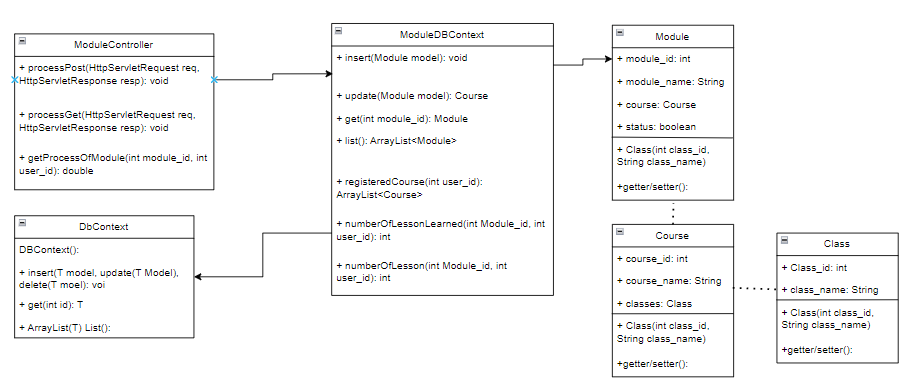
JOIN Lession l ON l.module\_id = m.module\_id

JOIN Lession\_Learn ll ON l.lession\_id = ll.lession\_id

WHERE ll.[user\_id] = ?

## 4. Choice Module

1. Class diagram



1. Class specification

Class method

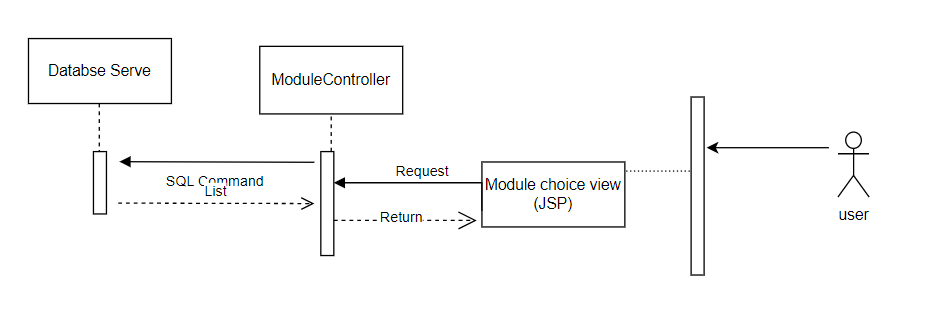
* ModuleController:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call itself |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Get list classes and req.getRequestDispatcher("./classes.jsp").forward(req, resp); |
| 03 | getProcessOfModule(int module\_id, int user\_id) | Get process module |

* ModuleDBContext:

|  |  |  |
| --- | --- | --- |
| * **No** | **Method** | **Description** |
| 01 | insert(Module model) | Abstract method |
| 02 | update(Module model) | Get list classes |
|  | delete(Module model) |  |
|  | get(int module\_id) | Get module by id |
|  | list() |  |
|  | listModuleByCourseID(int course\_id) | Get list module by course id |
|  | numberOfLessonLearned(int module\_id, int user\_id) | Get lesson learned |
|  | numberOfLesson(int module\_id, int user\_id) | Get lesson |

1. Sequence diagram



1. Database queries

// insert(Module model)

INSERT INTO [dbo].[Module]

([module\_name]

,[course\_id]

,[status])

VALUES

(?

,?

,?)

// update(Module model)

UPDATE [dbo].[Module]

SET [module\_name] = ?

,[course\_id] = ?

,[status] = ?

WHERE module\_id = ?

// get(int module\_id):

SELECT \* FROM Module WHERE module\_id = ?

// list():

SELECT \* FROM Module

// listModuleByCourseID(int course\_id):

SELECT \* FROM Module WHERE course\_id = ?

// numberOfLessonLearned(int module\_id, int user\_id):

SELECT COUNT(\*) AS numberOfLessonLearned FROM Lession\_Learn ll

JOIN Lession l ON ll.lession\_id = l.lession\_id

WHERE l.module\_id = ? AND ll.llearn = 'true' AND ll.[user\_id] = ?

// numberOfLesson(int module\_id, int user\_id):

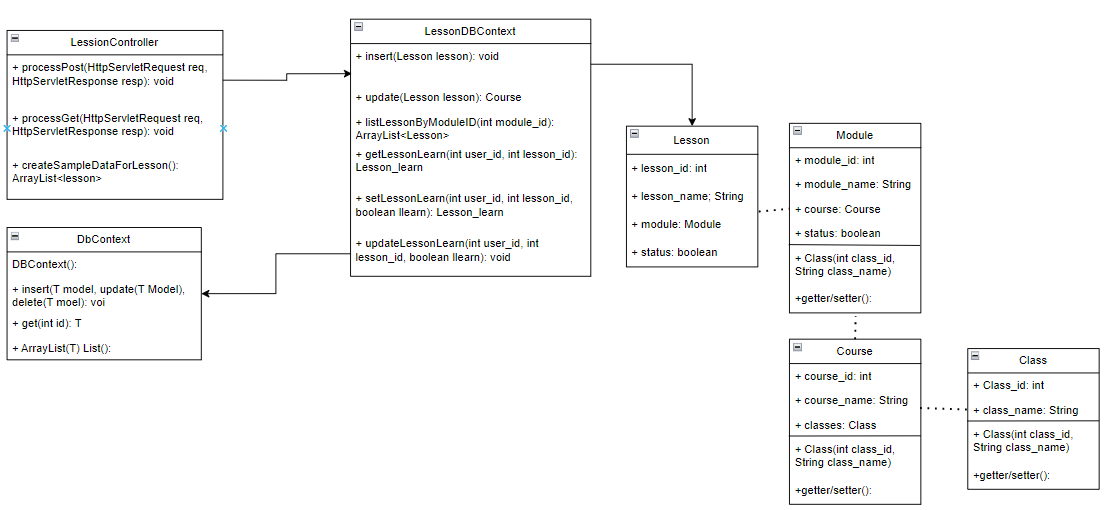
SELECT COUNT(\*) AS numberOfLessonLearned FROM Lession\_Learn ll

JOIN Lession l ON ll.lession\_id = l.lession\_id

WHERE l.module\_id = ? AND ll.[user\_id] = ?

## 5. Choice lesson

1. Class diagram



1. Class specification

Class method

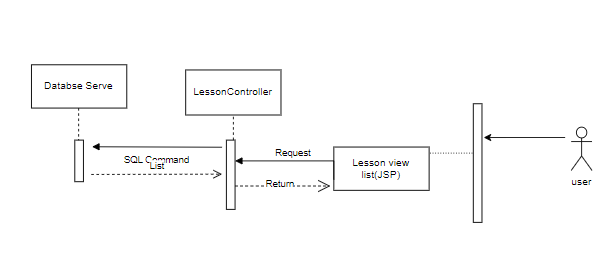
* LessonController:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call itself |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Get list classes and req.getRequestDispatcher("./classes.jsp").forward(req, resp); |
| 03 | createSampleDataForLesson() | Create Sample Data forl |

* LessonDBContext:

|  |  |  |
| --- | --- | --- |
| * **No** | **Method** | **Description** |
| 01 | insert(Lesson model) | Abstract method |
| 02 | update(Lesson model) | Get list classes |
|  | delete(Lesson model) |  |
|  | get(int lesson\_id) | Get lesson by id |
|  | ArrayList<Lesson> list() | Get list lesson |
|  | listLessonByModuleID(int module\_id) | Get list lesson by module id |
|  | listLessonByCourseID(int course\_id) | Get list lesson by course id |
|  | getLessonLearn(int user\_id, int lesson\_id) | Get list lesson learned |
|  | setLessonLearn(int user\_id, int lesson\_id, boolean llearn) | Set up lesson learn |
|  | updateLessonLearn(int user\_id, int lesson\_id, boolean llearn) | Update lessonlearn |

1. Sequence diagram



1. Database queries

// insert(Lesson model)

INSERT INTO [dbo].[Lession]

([lession\_name]

,[module\_id]

,[status])

VALUES

(?

,?

,?)

// update(Lesson model):

UPDATE [dbo].[Lession]

SET [lession\_name] = ?

,[module\_id] = ?

,[status] = ?

WHERE lession\_id = ?

// get(int lesson\_id):

SELECT \* FROM Lession WHERE lession\_id = ?

// list():

SELECT \* FROM Lession WHERE module\_id = ?

// listLessonByCourseID(int course\_id):

SELECT l.lession\_id,l.lession\_name,l.module\_id,l.[status],

m.module\_id,m.module\_name,m.course\_id,m.[status],

c.class\_id,c.course\_id,c.course\_name,

cl.class\_id,cl.class\_name

FROM Lession l

JOIN Module m ON m.module\_id = l.module\_id

JOIN Course c ON c.course\_id = m.course\_id

JOIN Class cl ON cl.class\_id = c.class\_id

WHERE c.course\_id = ?

// getLessonLearn(int user\_id, int lesson\_id):

SELECT ll.[user\_id], ll.lession\_id,ll.llearn \n"

+ "FROM Lession\_Learn ll WHERE ll.[user\_id] = ? AND ll.lession\_id = ?

// setLessonLearn(int user\_id, int lesson\_id, boolean llearn):

INSERT INTO [dbo].[Lession\_Learn]

([user\_id]

,[lession\_id]

,[llearn])

VALUES

(?

,?)

,?)

// updateLessonLearn(int user\_id, int lesson\_id, boolean llearn):

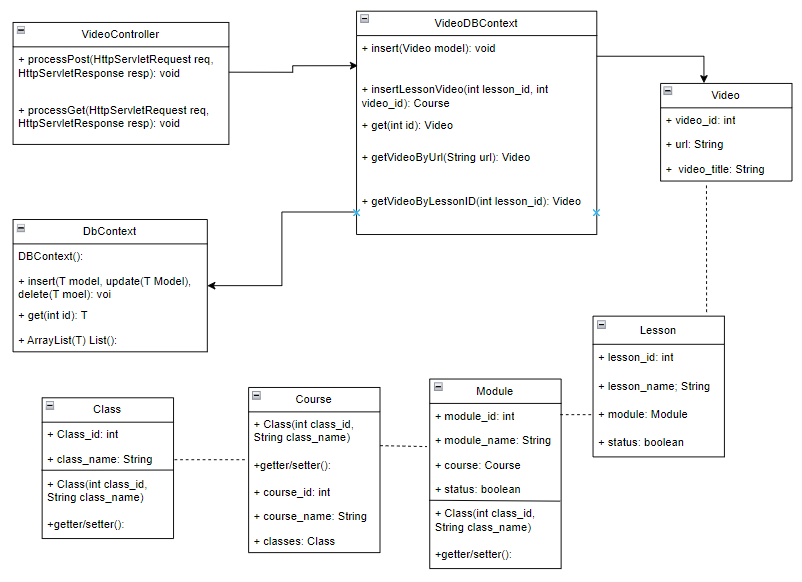
UPDATE [dbo].[Lession\_Learn]

SET [llearn] = ?

WHERE [lession\_id] = ? AND [user\_id] = ?

## 6. View video

1. Class diagram



1. Class specification

Class method

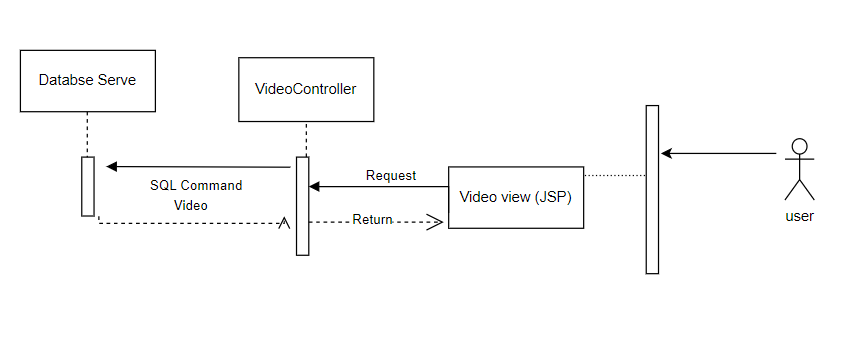
* VideoController:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call itself |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Get video of lesson and req.getRequestDispatcher("./videoLesson.jsp").forward(req, resp); |
|  |  |  |

* VideoDBContext:

|  |  |  |
| --- | --- | --- |
| * **No** | **Method** | **Description** |
| 1 | insert(Video model) | Insert video db |
| 2 | insertLessonVideo(int lesson\_id, int video\_id) | Insert video for lesson |
| 3 | update(Video model) | Update video db |
| 4 | get(int id) | Get video by id |
| 5 | getVideoByUrl(String url) | Get video by url |
| 6 | getVideoByLessonID(int lesson\_id) | Get video by lesson id |

1. Sequence diagram



1. Database queries

// insert(Video model):

INSERT INTO [dbo].[Video]

([url]

,[video\_title])

VALUES

(?

,?)

// insertLessonVideo(int lesson\_id, int video\_id):

INSERT INTO [dbo].[Lession\_Video]

([lession\_id]

,[video\_id])

VALUES

(?

,?)

// update(Video model):

UPDATE [dbo].[Video]

SET [url] = ?

,[video\_title] = ?

WHERE video\_id = ?

// get(int id):

SELECT \* FROM Video WHERE video\_id = ?

// getVideoByUrl(String url):

SELECT \* FROM Video WHERE [url] = ?

// getVideoByLessonID(int lesson\_id):

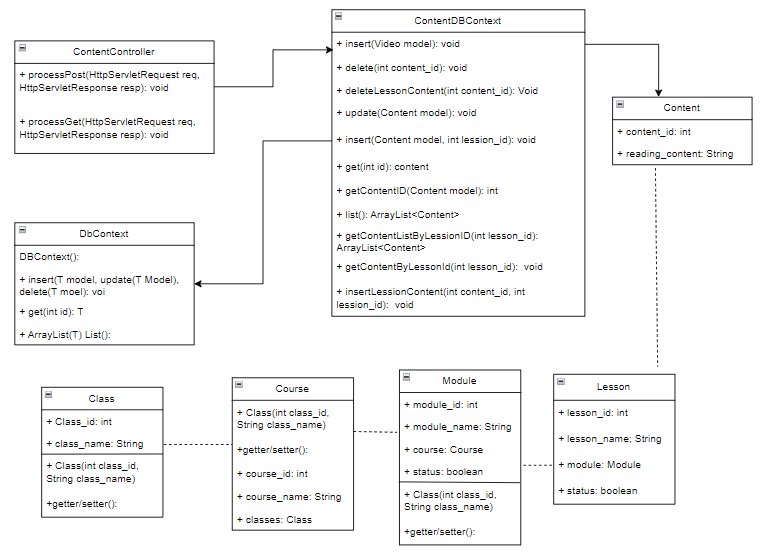
SELECT v.video\_id,v.[url],v.video\_title FROM Video v

JOIN Lession\_Video lv ON lv.video\_id = v.video\_id

WHERE lv.lession\_id = ?

## 7. View content

1. Class diagram



1. Class specification

Class method

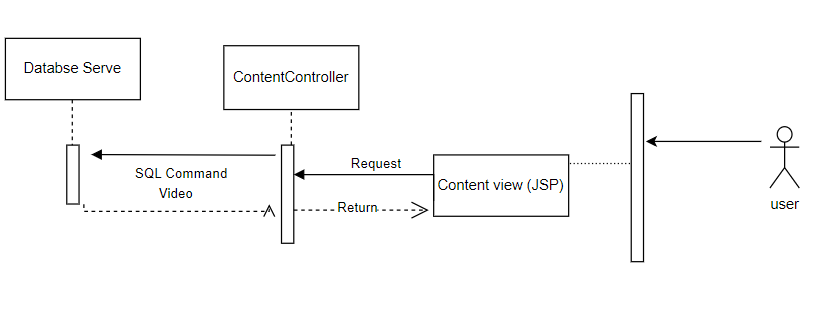
* ModuleController:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Call itself |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Call itself |
| 03 | doGetProcess(HttpServletRequest req, HttpServletResponse resp) | Get content lesson by id and req.getRequestDispatcher("./contentLesson.jsp").forward(req, resp); |

* ContentDBContext:

|  |  |  |
| --- | --- | --- |
| * **No** | **Method** | **Description** |
| 01 | insert(Content model) | Abstract method |
| 02 | update(Content model) |  |
|  | delete(Content model) |  |
|  | deleteLessonContent(int content\_id) | Delete content in lesson |
|  | insert(Content model, int lession\_id) | Insert content for lesson |
|  | get(int id) | Get content by id |
|  | getContentID(Content model) | Get content |
|  | list() |  |
|  | getContentListByLessionID(int lesson\_id) | Get list content by lesson id |
|  | insertLessionContent(int content\_id, int lession\_id) | Insert content for lesson |

1. Sequence diagram



1. Database queries

// delete(int content\_id):

DELETE FROM [dbo].[Content]

WHERE content\_id = ?

// deleteLessonContent(int content\_id):

DELETE FROM [dbo].[Lession\_Content]

WHERE content\_id = ?

// update(Content model):

UPDATE [dbo].[Content]

SET [reading\_content] = ?

WHERE content\_id = ?

// insert(Content model, int lession\_id):

INSERT INTO [dbo].[Content]

([reading\_content])

VALUES\n"

(?)

// get(int id):

SELECT \* FROM Content WHERE content\_id = ?

// getContentID(Content model):

SELECT content\_id FROM Content WHERE reading\_content = ?

// list():

SELECT \* FROM Content

// getContentListByLessionID(int lesson\_id):

SELECT c.content\_id, c.reading\_content

FROM Content c

inner join Lession\_Content lc on lc.content\_id = c.content\_id

inner join Lession l on l.lession\_id = lc.lession\_id

WHERE l.lession\_id = ?

// getContentByLessonId(int lesson\_id):

SELECT c.content\_id,c.reading\_content

FROM Content c, Lession\_Content lc

WHERE lc.lession\_id = ? AND c.content\_id = lc.content\_id

// insertLessionContent(int content\_id, int lession\_id) :

INSERT INTO [dbo].[Lession\_Content]

([lession\_id]

,[content\_id])

VALUES

(?

,?)