

**STUDENT PROJECT MANAGEMENT**

**Software Design Specification**

– Hanoi, May 2022 –

Record of changeS

|  |  |  |  |
| --- | --- | --- | --- |
| Date | A\* M, D | In charge | Change Description |
| 20/5/2022 | A | TraiCV | I.1- Code packages; II.2 – User login, II.5 - Register |
| 20/5/2022 | A | HoangVM | II.3 – Setting list |
| 20/5/2022 | A | KhanhTQ | I.2 – Database design, II.4 – Setting details |
| 20/5/2022 | A | HuongNC | II.6 – User profile |
| 20/5/2022 | A | DatPT | II.1 – Home, Dashboard |
| 2/6/2022 | M | KhanhTQ | Modified II.4 – Setting details |
| 2/6/2022 | A | KhanhTQ | II.13 – Subject Setting List, II.14 – Subject Setting Details |
| 4/6/2022 | M | TraiCV | I.1- Code packages |
| 4/6/2022 | A | TraiCV | II.7. Reset password, II.8. Change password |
| 4/6/2022 | A | HuongNC | II.9. User list, II.10. User details |
| 4/6/2022 | A | HoangVM | II.11 – Subject list, II.12 – Subject details |
| 4/6/2022 | A | DatPT | II.15 - Iteration List, II.16 – Iteration details |
| 4/6/2022 | M | KhanhTQ | Update I.2.a – database schema |
| 12/6/2022 | A | KhanhTQ | II.17,18,19 - Criteria List, Criteria Details, Class List |
| 17/6/2022 | A | HoangVM | II.20,21,22 – Class Details, Milestone List, Milestone Details |
| 17/6/2022 | A | TraiCV | II. 23. Class User List 24.Class User detail 25. Import/Export Class user |
| 17/6/2022 | A | HuongNC | II.26,27– Team List, Team details |
| 17/6 | A | DatPT | II.28,39- Feature List, Feature details |
| 8/7/2022 | A | HuongNC | II.30. Issue list, II.31. Issue details |
| 8/72022 | A | TraiCV | II.32 Function List, II.33 Function details , II.34, Function Import/Export |
| 8/7/2022 | A | KhanhTQ | II. 35,36,37 – Loc Evaluation, Team Evaluation, User Evaluation |
| 8/7/2022 | A | HoangVM | II.38 – Milestone Sync, II.39 – Issue Sync |
| 18/7/2022 | M | TraiCV | Modified I.1 – Code pakages |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 4](#_Toc96516286)

[1. Code Packages 4](#_Toc96516287)

[2. Database Design 5](#_Toc96516288)

[a. Database Schema 5](#_Toc96516289)

[b. Table Description 5](#_Toc96516290)

[II. Code Designs 5](#_Toc96516291)

[1. <Common /Home, Dashboard>](#_Toc96516292) 5

[a. Class Diagram](#_Toc96516293) 6

[b. Class Specifications 6](#_Toc96516294)

[c. Sequence Diagram(s) 6](#_Toc96516295)

[d. Database queries 6](#_Toc96516296)

[2. <Common/ User Register> 7](#_Toc96516297)

3. <Admin / Setting List> …………………………………………………………………………………………………………………….8

4. <Admin / Setting Details> ……………………………………………………………………………………………………………..10

5. <Common / User Login> ……..………………………………………………………………………………………………………..12

6. <Common/User Profile>………………………………………………………………………………………………………………..14

7. <Common/Reset Password>………………………………………………………………………………………………………….15

8. <Common/Change Password>………………………………………………………………………………………………………16

9. <Admin/User List>………………………………………………………………………………………………………………………..18

10. <Admin/User Details>…………………………………………………………………………………………………………………20

11. <Admin/Subject List>…………………………………………………………………………………………………………………..23

12. <Admin/Subject Details>……………………………………………………………………………………………………………..25

13. <Author/Subject Setting List>………………………………………………………………………………………………………27

14. <Author/Subject Setting Details>…………………………………………………………………………………………………29

15. < Author /Iteration List>………………………………………………………………………………………………………………31

16. < Author /Iteration Details> ……………………………………………………………………………………………………….32

17.<Author/Criteria List> …………………………………………………………………………………………………………………36

18.<Author/Criteria details>……………………………………………………………………………………………………………..38

19.<Author/Class List>………………………………………………………………………………………………………………………40

20.<Author/Class Details>……………………………………………………………………………………………………………….42

21.<Trainer/Milestone List>…………………………………………………………………………………………………………….44

22.<Trainer/Milestone Details>……………………………………………………………………………………………………….47

23.<Trainer/Class User List>……………………………………………………………………………………………………………49

24.<Trainer/Class User Details>……………………………………………………………………………………………………..52

25.<Trainer/Class User Import Export>……………………………………………………………………………………………54

26.<Trainer/Team List>…………………………………………………………………………………………………………………..56

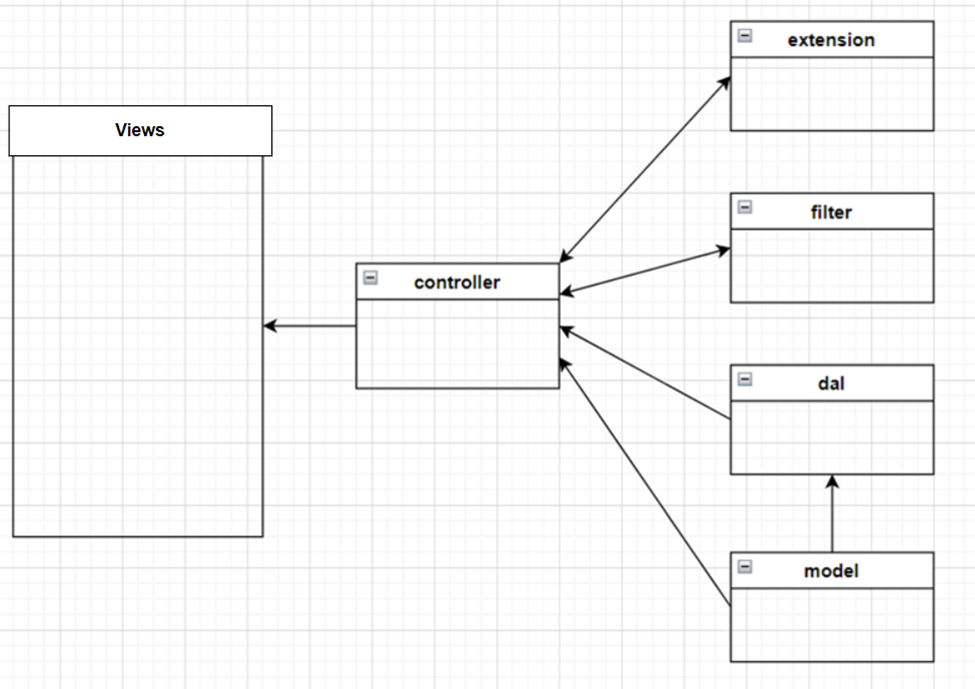
27.<Trainer/Team Details>…………………………………………………………………………………………………………….58

28.<Student/Feature List>……………………………………………………………………………………….…………………….60

29.<Student/Feature Details>…………………………………………………………………………………..……………………61

# I. Overview

## 1. Code Packages

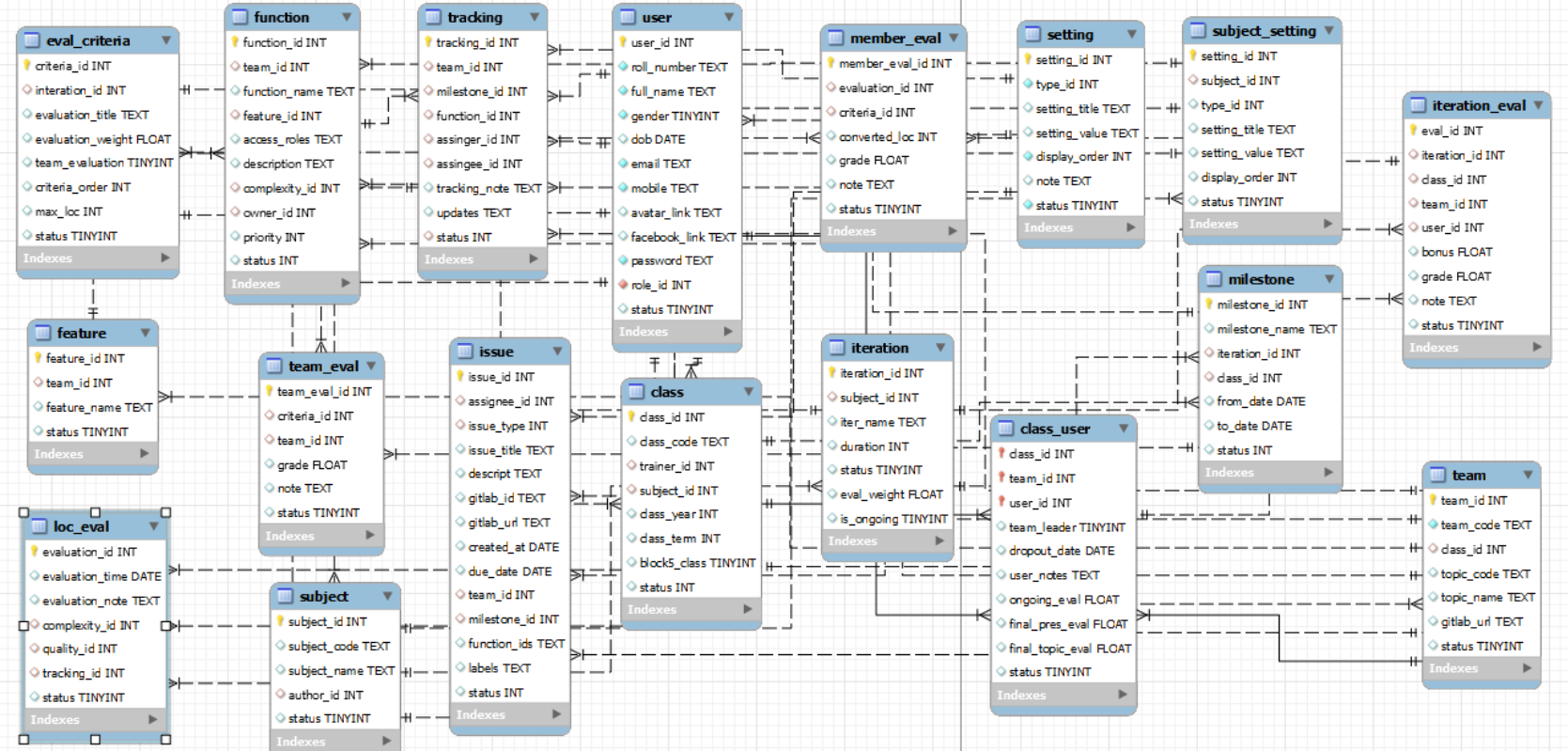


***Package descriptions & package class naming conventions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| *01* | *model* | *- Class Naming Convention: User, Setting.*  *- This is layer acts as an inventory for storing system object* |
| *02* | *dal* | *- Class Naming Convention: DBContext, SettingDB, UserDB.*  *- This is the data layer which contains the business logic of the system, and also represents the state of the application. It’s independent of the presentation layer, the controller fetches the data from the Model layer and sends it to the View layer.* |
| *03* | *extension* | *-Class Naming Convention: Checkmail, Encode.*  *- Package contains classes that implement data processing logic.* |
| *04* | *controller* | *-Class Naming Convention: HomeController, CheckmailController, EditProfileServlet, LoginController, LogoutController, ProfileController, RegistrationController, SettingAddController, SettingDetailController, SettingListController, UserControlle.*  *- The controller layer acts as an interface between View and Model. It receives requests from the View layer and processes them, including the necessary validations.* |
| *05* | *filter* | *-Class Naming Convention: AuthorityAdminAuthorFilter, AuthorityAdminFilter,* *AuthorityUserFilter.*  *- Filters are used to assign user permissions before accessing the severlet.* |

## 2. Database Design

### a. Database Schema :



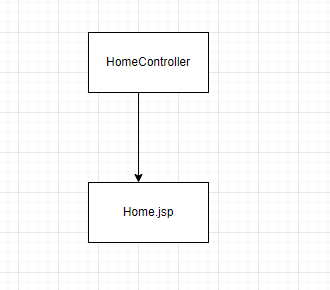
### b. Table Description

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| 01 | iteration | Table contain iteration information for each subject |
| 02 | team | Information of team in each class |
| 03 | class | Table of all class |
| 04 | function | Table of all function |
| 05 | user | Table contain all user information |
| 06 | tracking | Table for tracking progress of function |
| 07 | feature | Table of feature of team in a project |
| 08 | setting | Table contain setting information for some other tables |
| 09 | subject\_setting | Table contain subject setting about complexity, quality, … |
| 10 | eval\_criteria | Contain evaluation criteria for a project in each iteration |
| 11 | class\_user | Many-many relationship of table class, team and user |
| 12 | team\_eval | Table for team evaluation |
| 13 | iteration\_eval | Table for iteration evaluation |
| 14 | milestone | Table contains milestone |
| 15 | loc\_eval | Line of code evaluation |
| 16 | issue | Table contain issue in project of each team |
| 17 | member\_eval | Table for member evaluation |
| 18 | subject | Table contain all subjects information |

# II. Code Designs

**1. <Common/Home, Dashboard>**

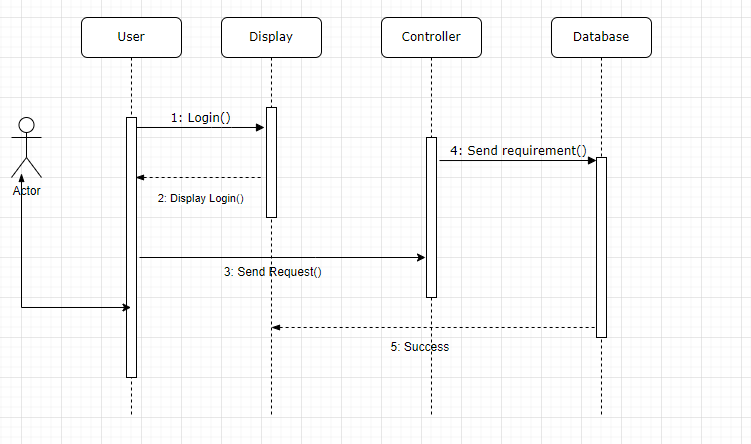
a. Class Diagram



b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet() | run to request forward to “Home.jsp” |

c. Sequence Diagram(s)

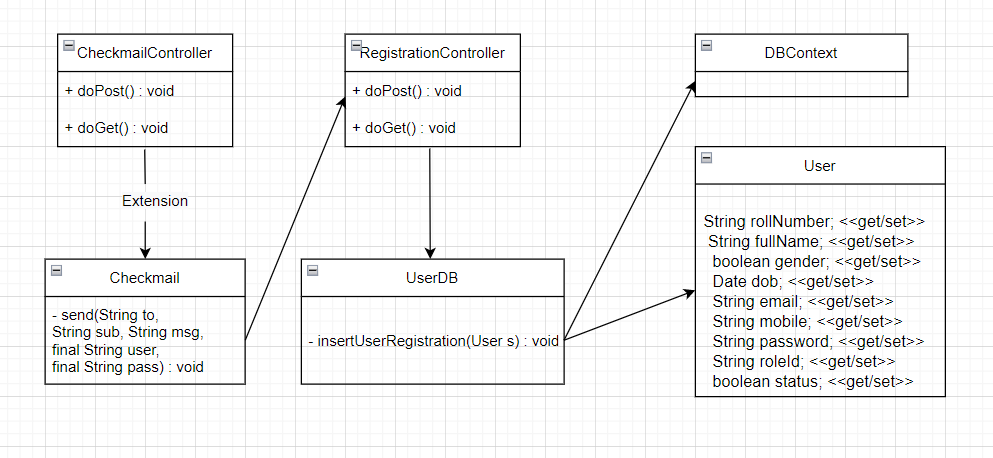


d. Database queries

No queries used.

**2. <Common/User Register>**

**a.** *Class Diagram*

****

*b. Class Specifications*

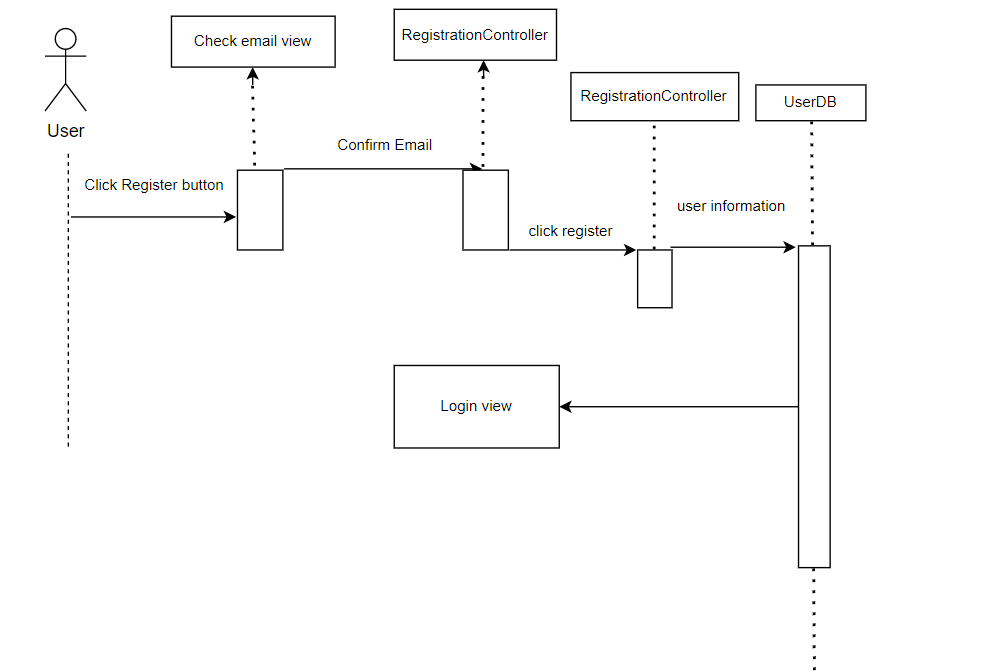
***CheckmailController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doPost() | run to check mail screen and enter the email you want to register. |
| 2 | doGET() | Use Extension class to checkmail and send email to registered email |

**RegistrationController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doPost() | after the user confirms the email, run to registration screen and enter user information. |
| 2 | doGET() | Process and check user information and then add it to the database |

### c. Sequence Diagram(s)

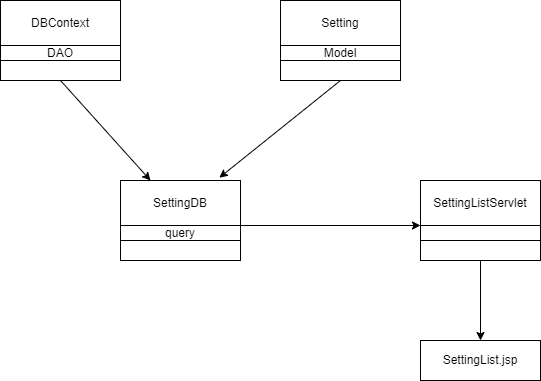


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | insertUserRegistration(User s) :  void | INSERT INTO `user` (`roll\_number`, `full\_name`,`gender`, `dob`, `email`, `mobile`, `password`, `role\_id`, `status`)  VALUES (?, ?,?,?,?,?,?,?,?) |

**3. <Admin/Setting list>**

### a. Class Diagram



### b. Class Specifications

#### DBContext Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *DBContext()* | *Setup a connection to MySql database* |

#### Setting Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructor* | *Construct an object with provided value* |
| *02* | *Getter* | *Get a field value* |
| *03* | *Setter* | *Set value for a field* |

#### SettingDB Class

***Class Methods***

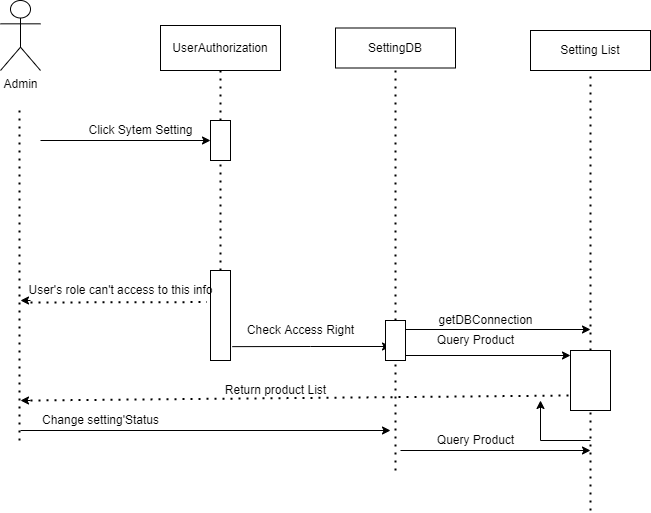
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Search(String,String,String)* | *Search for a set of specific Setting that meet all 3 requirement parameter* |
| *02* | *Update(Setting)* | *Update a Setting to the database* |

#### SettingListServlet Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Query a list of setting* |

### c. Sequence Diagram(s)



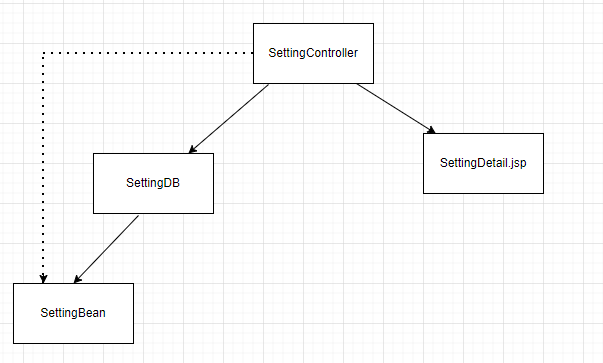
### d. Database queries

*Select: SELECT \* FROM setting WHERE TRUE AND type\_id=?, status=?, setting\_name like ?*

*Update; Update setting SET status=? WHERE setting\_id=?*

## 4. <Admin/Setting details>

### a. Class Diagram



### b. Class Specifications

#### SettingController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input is the settingId.*  *Output: a setting in database with that settingId*  *And then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input is attribute of setting model admin want to update: typeId, displayOrder, settingValue, status, note*  *And then update new information to database.*  *After update success redirect to setting list jsp page* |

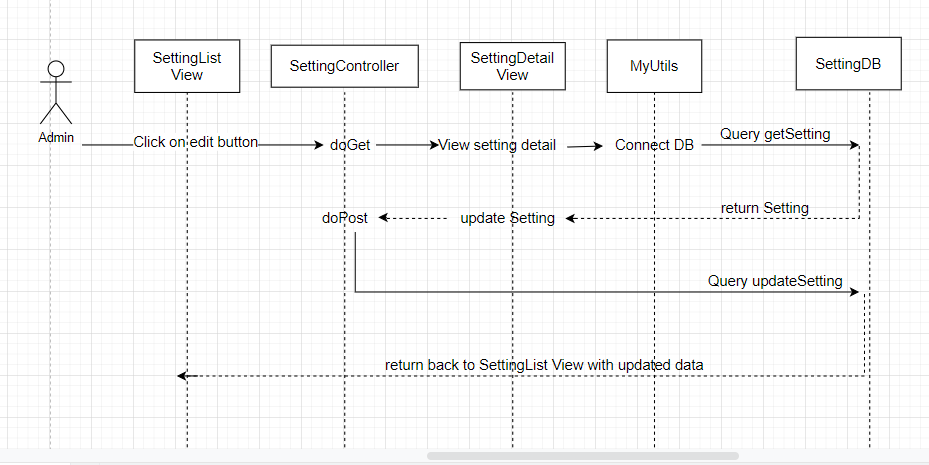
#### SettingDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getSettingById(int id)* | *Input: settingId*  *Output: a setting that take from database with all information* |
| *02* | *getListSettingType* | *Input: none*  *Output: a List<String> of all typeId in setting table in database* |
| *03* | *updateSetting(Setting s)* | *Input: an object Setting*  *Run sql statement update to update new information to database* |

***Setting Class***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Setting()* | *Constructor with no parameter* |
| *02* | *Setting(int settingId, int typeId, String settingTitle, String settingValue, int displayOrder, String note, boolean status)* | *Constructor with parameters* |
| *03* | *getter* | *Getter function for all attribute in this setting model* |
| *04* | *setter* | *Setter function for all attribute in this setting model* |

### c. Sequence Diagram(s)



### d. Database queries

*+) getSettingById(int id):*

*select \* from setting where setting\_id = ?*

*+) getListSettingType():*

*SELECT distinct type\_id FROM setting*

*+) updateSetting(Setting s):*

*update setting set `type\_id` = ?,* `*setting\_title`= ?, setting\_value` = ?, `display\_order` = ?,`note` = ?,`status`=? where `setting\_id` = ?*

## 5. <Common/User login>

*a. Class Diagram*

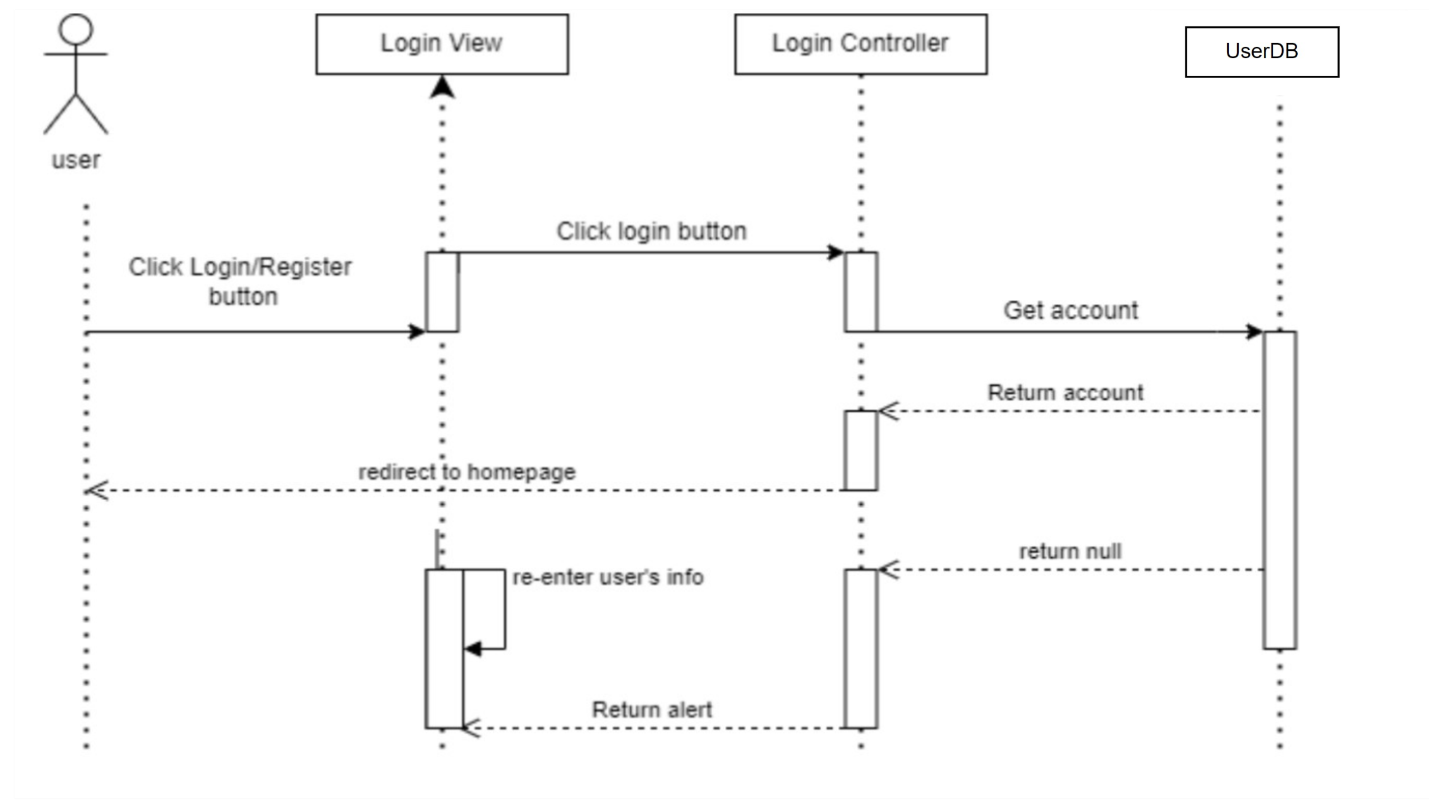
# 

*b. Class Specifications*

***LoginController***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doPost() | Run to the login screen and the user fills in the email and password to log in. |
| 2 | doGET() | Check email and password in database. Successful login will send user information to Session. |

### c. Sequence Diagram(s)

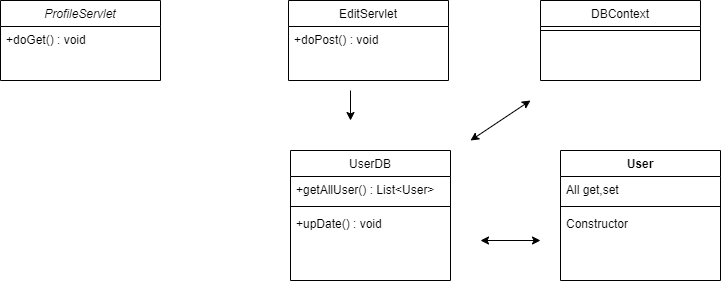


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | getUserAccount(String email, String password) : User | select \* from user where email = ? and password = ? |

# 6. <Common/User profile>

### a. Class Diagram



### b. Class Specifications

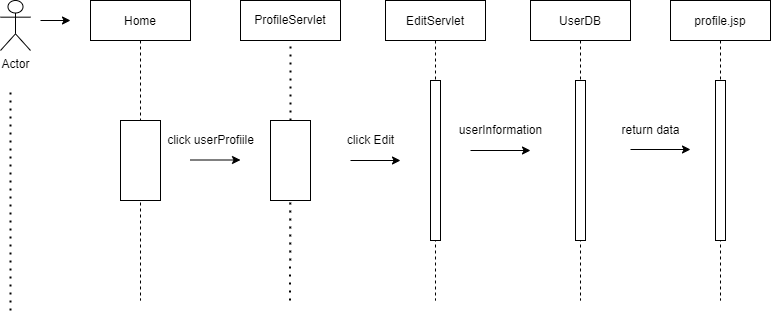
### ProfileServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get User information* |

EditServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Allow User to change information* |

### c. Sequence Diagram(s)



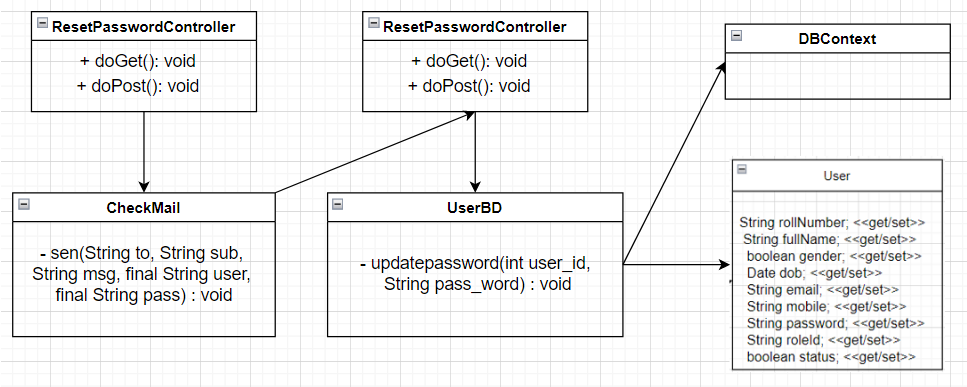
### d. Database queries

"select \* from user"

"update user set roll\_number=?,full\_name=?,gender=?,dob=?,email=?,mobile=?,avatar\_link=? where user\_id = ?"

**7. <Common/Reset password>**

**a.***Class Diagram*

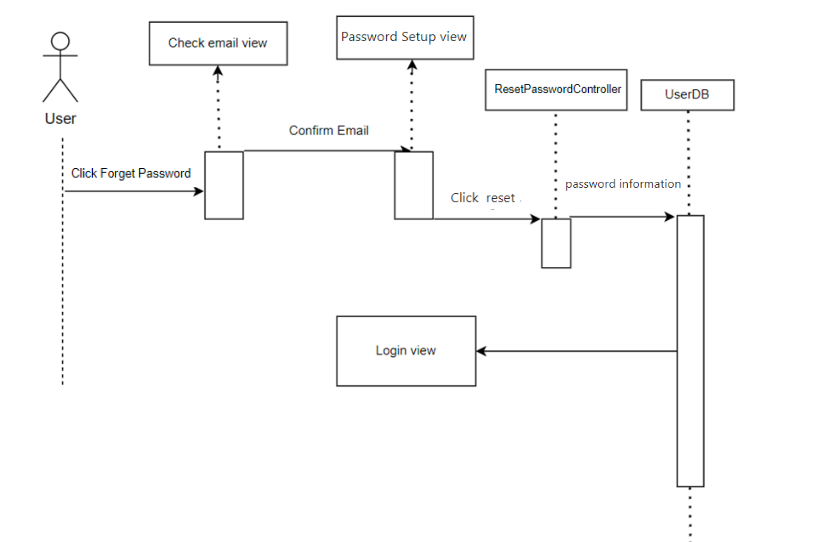
****

*b. Class Specifications*

**ResetPasswordController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGET() | - Send a confirmation email to reset password to the user.  - After confirming the email, you will go to the page to enter a new password. |
| 2 | doPost() | - Update new password in database. |

### c. Sequence Diagram(s)

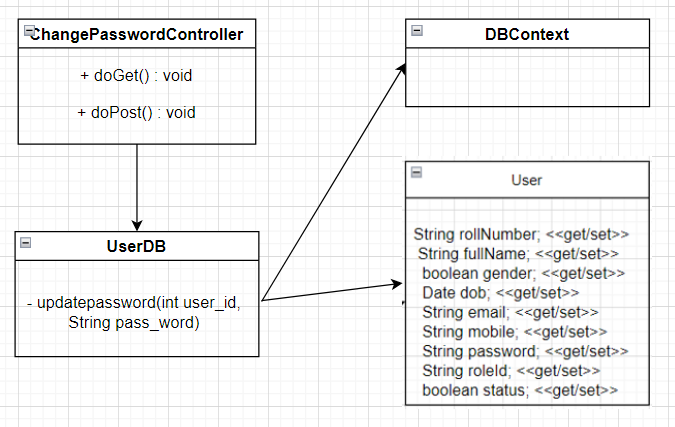


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | updatepassword(int user\_id, String pass\_word):  void | UPDATE user SET  `password` = ?  WHERE `user\_id` = ? |

**8. <Common /Change Password>**

**a.***Class Diagram*

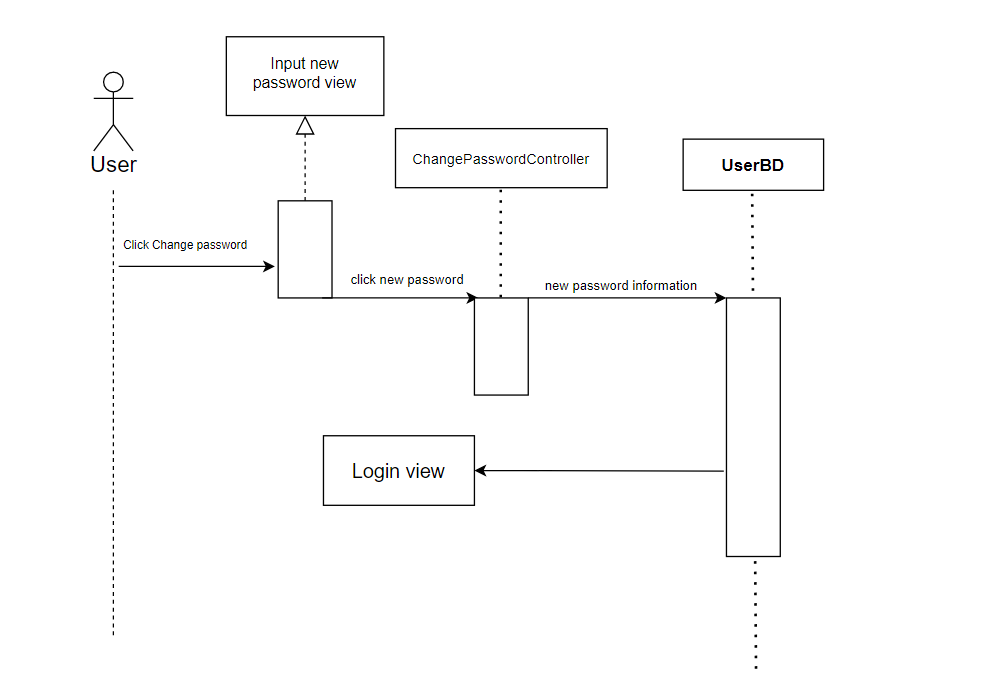
****

*b. Class Specifications*

**ChangePasswordController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGET() | - Go to the display page to change password |
| 2 | doPost() | - Update new password in database. |

### c. Sequence Diagram(s)

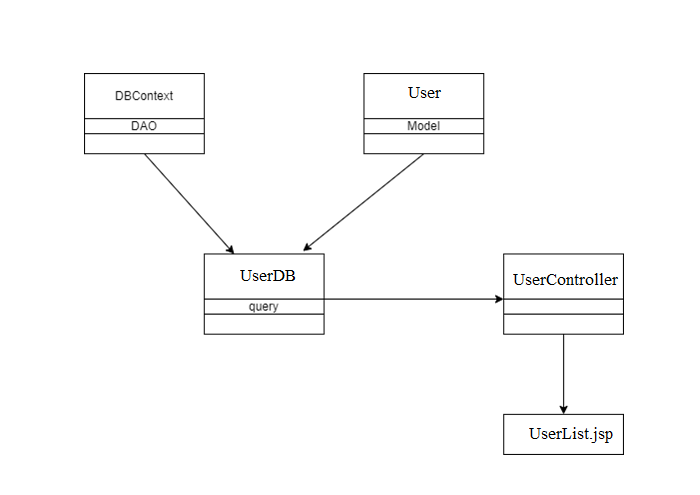


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | updatepassword(int user\_id, String pass\_word):  void | UPDATE user SET  `password` = ?  WHERE `user\_id` = ? |

**9. <Admin/User list>**

### a. Class Diagram



### b. Class Specifications

#### DBContext Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *DBContext()* | *Setup a connection to MySql database* |

#### User Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructor* | *Construct an object with provided value* |
| *02* | *Getter* | *Get a field value* |
| *03* | *Setter* | *Set value for a field* |

#### UserDB Class

***Class Methods***

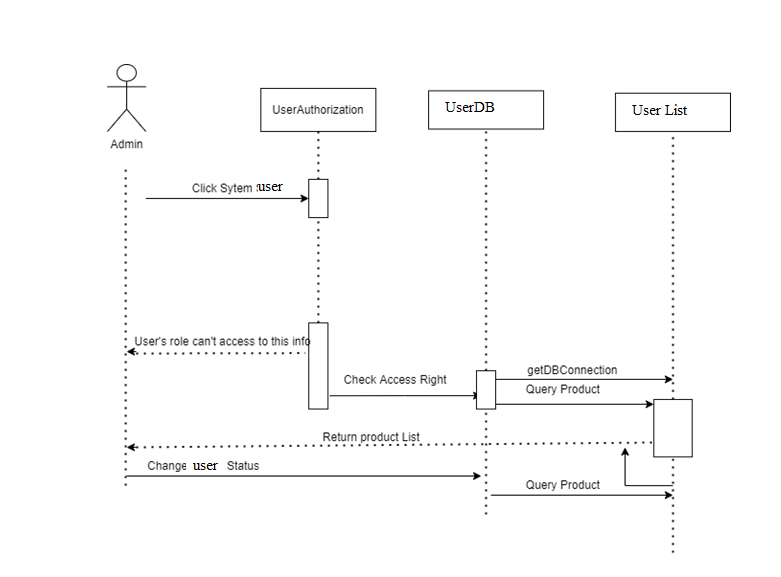
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Search(String,String,String)* | *Search for a set of specific User that meet all 3 requirement parameter* |
| *02* | *Update(User)* | *Update a User to the database* |

#### SettingListServlet Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Query a list of User* |

### c. Sequence Diagram(s)



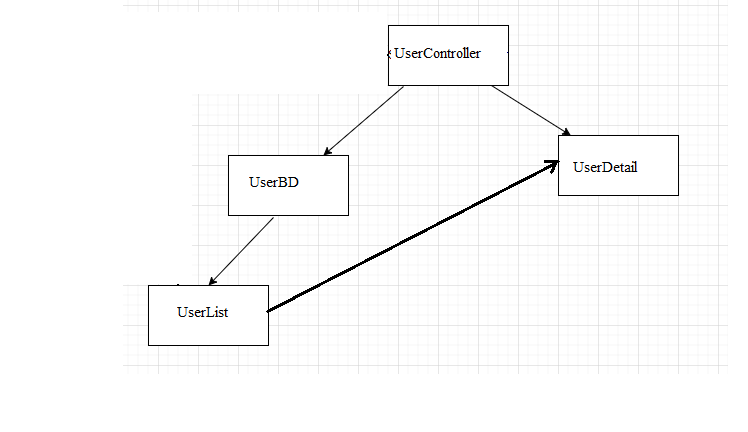
### d. Database queries

*Select: select \* from user where true and role\_id=? and status=? and full\_name like ?;*

*Update: UPDATE user SET status = ? WHERE user\_id  = ?;*

## 10. <Admin/User details>

### a. Class Diagram



### b. Class Specifications

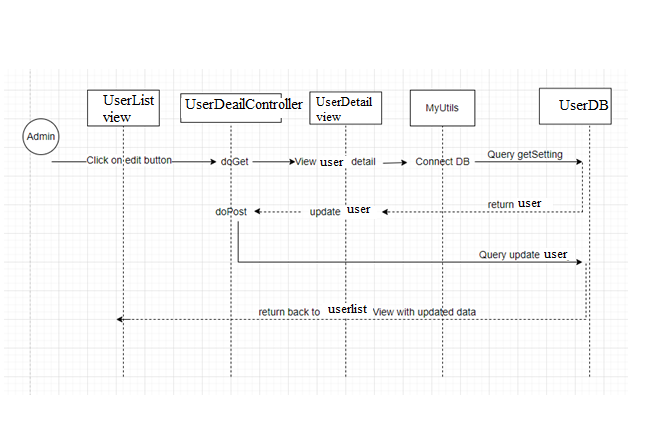
#### SettingDetail Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input is the userId.*  *Output: a setting in database with that UserId*  *And then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input is attribute of setting model admin want to update:* full\_name*,* email, mobile, facebook, gender, status, role  *And then update new information to database.*  *After update success redirect to setting list jsp page* |

#### UserDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getSettingById(int id)* | *Input: userId*  *Output: a user that take from database with all information* |
| *02* | *getListUserId* | *Input: none*  *Output: a List<String> of all UserId in setting table in database* |
| *03* | *updateUser(User s)* | *Input: an object Setting*  *Run sql statement update to update new information to database* |

### c. Sequence Diagram(s)



### d. Database queries

*+) Update UserDetail*

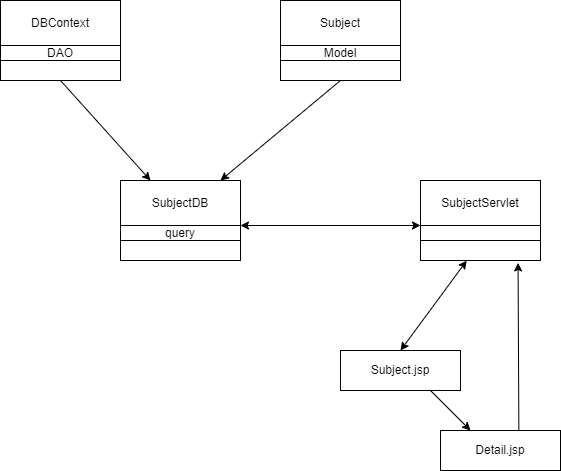
*UPDATE user SET full\_name = ?, email = ?,mobile=? ,facebook\_link = ?,gender= ?,status = ?,role\_id = ?  WHERE user\_id = ?;*

*+)UserDetail-AllRole*

*SELECT setting\_title FROM setting limit 4;*

## 11. <Admin/Subject List>

### a. Class Diagram



### b. Class Specifications

#### DBContext Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *DBContext()* | *Setup a connection to MySql database* |

#### Subject Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructor* | *Construct an object with provided value* |
| *02* | *Getter* | *Get a field value* |
| *03* | *Setter* | *Set value for a field* |

#### SubjectDB Class

***Class Methods***

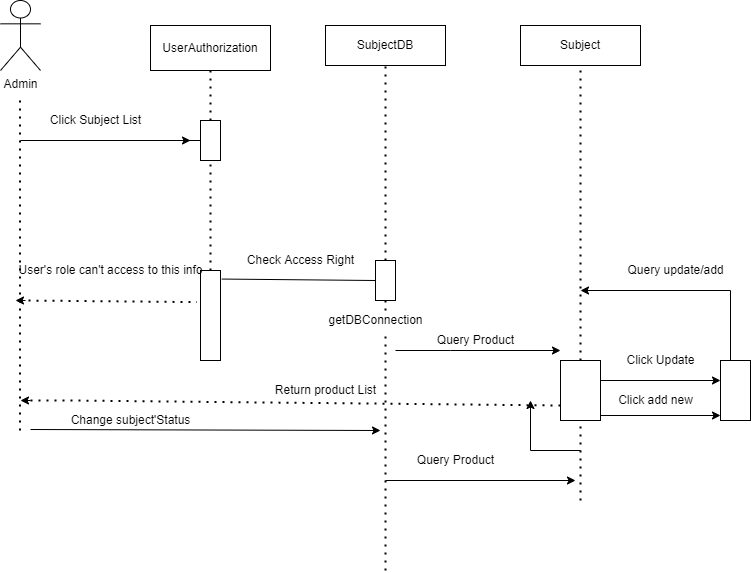
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Search(String,String,String)* | *Search for a set of specific Subject that meet all 3 requirement parameter, return all Subject if 3 string is empty* |
| *02* | *changeStatus(Subject)* | *Change subject status* |
| *03* | *GetListByPage(List,int,int)* | *Pagination support* |

#### SettingListServlet Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Query a list of subject, search, change status* |
| *02* | *doPost()* | *Add new status if id not exist, update otherwise* |

### c. Sequence Diagram(s)



### d. Database queries

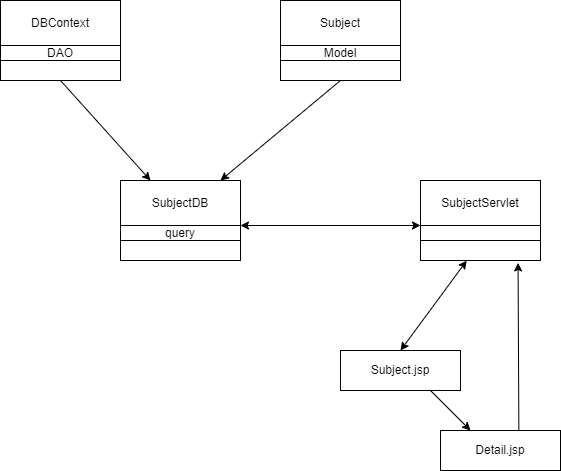
Select: SELECT \* FROM setting WHERE TRUE AND subject\_code like ?, status=?, subject\_name like ?

Update; Update setting SET status=? WHERE setting\_id=?

Change status: update subject set status=? where subject\_id=?

## 12. <Admin/Subject Details>

### a. Class Diagram



### b. Class Specifications

#### DBContext Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *DBContext()* | *Setup a connection to MySql database* |

#### Subject Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructor* | *Construct an object with provided value* |
| *02* | *Getter* | *Get a field value* |
| *03* | *Setter* | *Set value for a field* |

#### SubjectDB Class

***Class Methods***

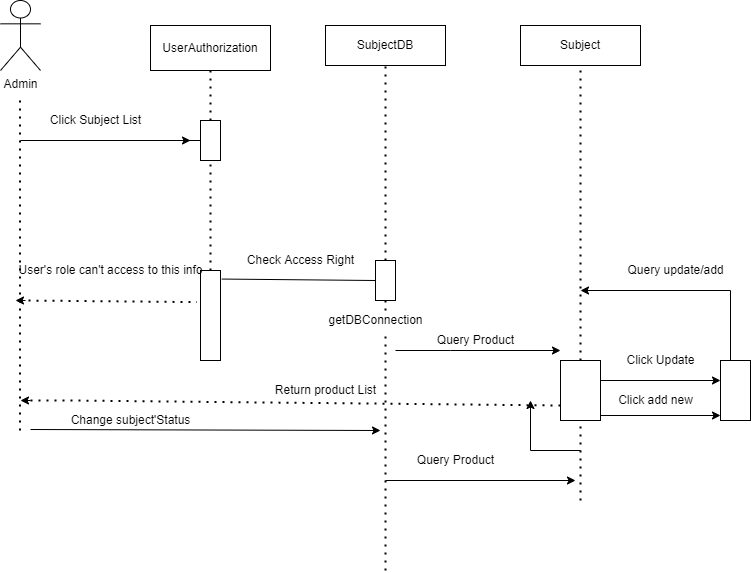
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *02* | *Update(Subject)* | *Update a Subject to the database* |
| *03* | *Add(Subject)* | *Add Subject to database* |
| *04* | *AuthorList()* | *List all user with role author* |

#### SettingListServlet Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Query a list of subject, search, change status* |
| *02* | *doPost()* | *Add new status if id not exist, update otherwise* |

### c. Sequence Diagram(s)



### d. Database queries

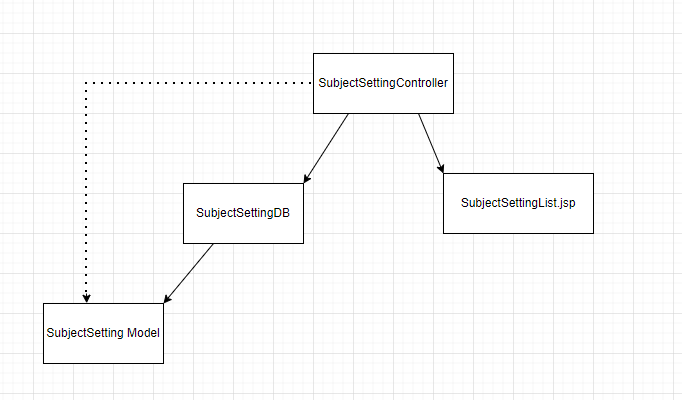
Add :INSERT INTO `subject` (`subject\_code`,`subject\_name`,`author\_id`,`status`)  VALUES(?,?,?,?);

Update: update subject set `subject\_code` = ?,`subject\_name` = ?,’status` = ?,`author\_id` = ?\n" where `subject\_id` = ?";

List of author: SELECT u.user\_id, u.full\_name from setting s inner join user u where s.setting\_id= u.role\_id and s.type\_id=1 and s.setting\_title='author' and u.status=1

# 13. <Author/Subject Setting List>

### a. Class Diagram



### b. Class Specifications

#### SubjectSettingController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag null, add or update*  *Output: a setting in database with that tag to specific jsp page (list or detail or add page)*  *And then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input: get update information*  *And then update new information to database.*  *After update success redirect to SubjectSettingList.jsp page* |

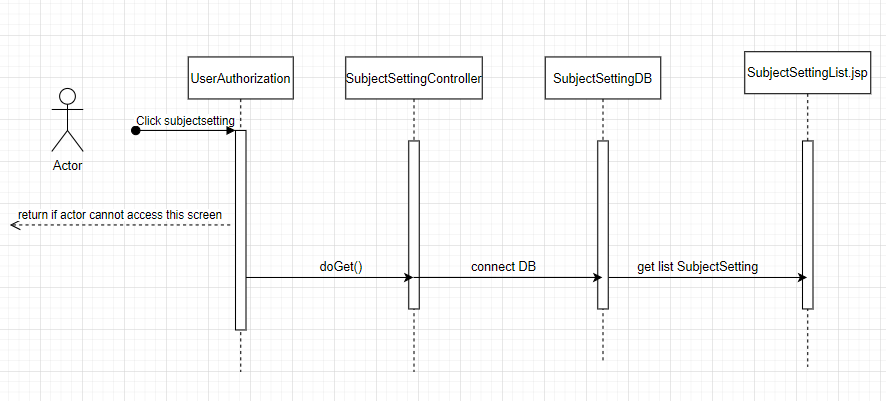
#### SubjectSettingDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *search(String type, String status, String search\_value, String search\_subject\_id, String author)* | *Input: String type, String status, String search\_value, String search\_subject\_id, String author*  *Output: list of SubjectSetting* |
| *02* | *getListByPage(List<SubjectSetting> list, int start, int end)* | *Input: List<SubjectSetting> list, int start, int end*  *Output: list of SubjectSetting from start to end* |
| *03* | *getListType()* | *Input: none*  *Output: list String of listType* |
| *04* | *getAll()* | *Input: none*  *Output: list of all SubjectSetting* |
| *05* | *getSSById(String id, String user\_id)* | *Input: String id, String user\_id*  *Output: a SubjectSetting* |
| *06* | *getAllSubject()* | *Input: none*  *Output: list of all Subject* |
| *07* | *addSubjectSetting(SubjectSetting ss)* | *Input: SubjectSetting ss*  *Output: void* |

#### SubjectSetting Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *SubjectSetting()* | *Constructor with no parameter* |
| *02* | *SubjectSetting(int settingId, int subjectId, int typeId, String settingTitle, String settingValue, int displayOrder, boolean status)* | *Constructor with all parameters* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



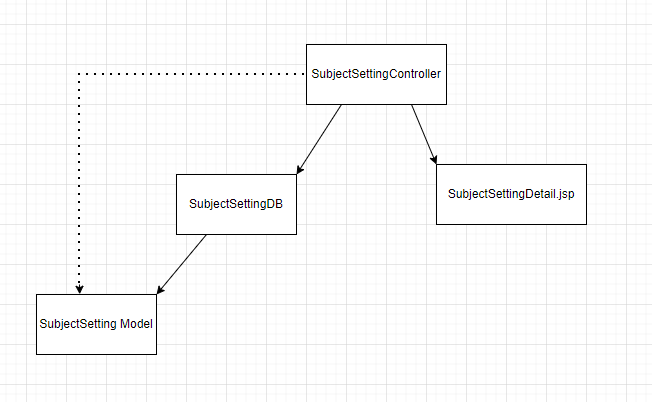
### d. Database queries

* search: select ss.setting\_id, ss.subject\_id, ss.type\_id, ss.setting\_title, ss.setting\_value, ss.display\_order, ss.`status` from user as u, subject as s, subject\_setting as ss where u.user\_id = s.author\_id and ss.subject\_id = s.subject\_id and s.status=1
* getListType: SELECT distinct type\_id FROM subject\_setting
* getAll: select \* from subject\_setting
* getSSById: select ss.setting\_id, ss.subject\_id, ss.type\_id, ss.setting\_title, ss.setting\_value, ss.display\_order, ss.`status` from user as u, subject as s, subject\_setting as ss where u.user\_id = s.author\_id and ss.subject\_id = s.subject\_id and ss.setting\_id = ?
* getAllSubject: select s.subject\_id, s.subject\_code, s.subject\_name, s.author\_id, s.status from user as u, subject as s where u.user\_id = s.author\_id and s.status=1
* addSubjectSetting: INSERT INTO `subject\_setting` (`subject\_id`,`type\_id`,`setting\_title`,`setting\_value`,`display\_order`,`status`)

VALUES (?,?,?,?,?,?)

# 14. <Author/Subject Setting Details>

### a. Class Diagram



### b. Class Specifications

#### SubjectSettingController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag null, add or update*  *Output: a setting in database with that tag to specific jsp page (list or detail or add page)*  *And then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input: get update information*  *And then update new information to database.*  *After update success redirect to SubjectSettingList.jsp page* |

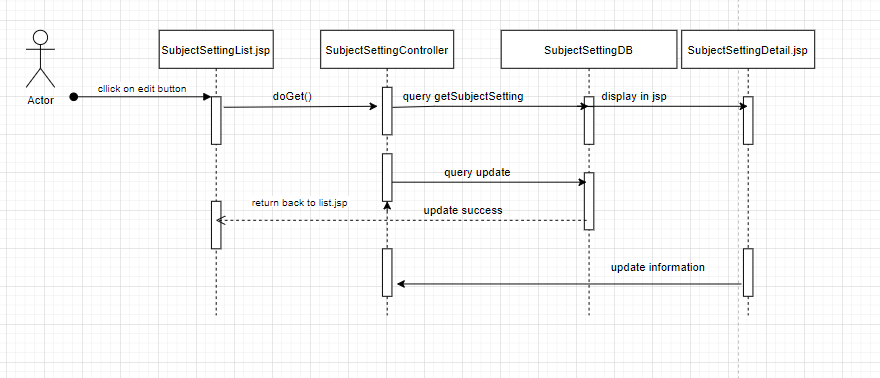
#### SubjectSettingDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *updateSubjectSetting(SubjectSetting ss)* | *Input: SubjectSetting ss*  *Output: void* |
| *02* | *getSSById(String id, String user\_id)* | *Input: String id, String user\_id*  *Output: a SubjectSetting* |

#### SubjectSetting Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *SubjectSetting()* | *Constructor with no parameter* |
| *02* | *SubjectSetting(int settingId, int subjectId, int typeId, String settingTitle, String settingValue, int displayOrder, boolean status)* | *Constructor with all parameters* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

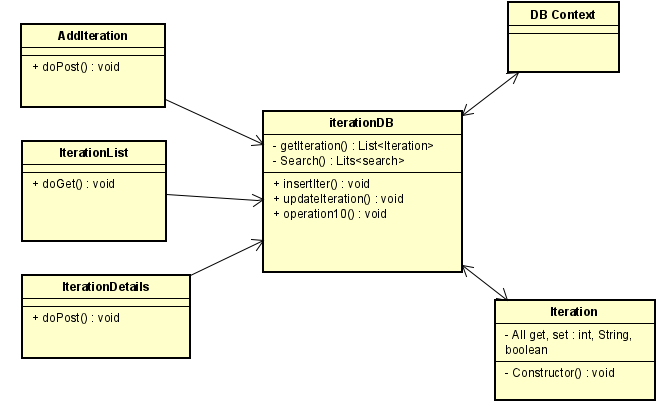
*updateSubjectSetting: UPDATE subject\_setting SET `subject\_id` = ?, type\_id` = ?, `setting\_title` = ?,"`setting\_value` = ?, `display\_order` = ?, `status` = ? WHERE `setting\_id` = ?*

getSSById: select ss.setting\_id, ss.subject\_id, ss.type\_id, ss.setting\_title, ss.setting\_value, ss.display\_order, ss.`status` from user as u, subject as s, subject\_setting

## 15. <Author/ Iteration List>

### After user(Admin or Author) login user can choose to click the button Iteration List allow user to see Information and edit it.

### a. Class Diagram



### b. Class Specifications

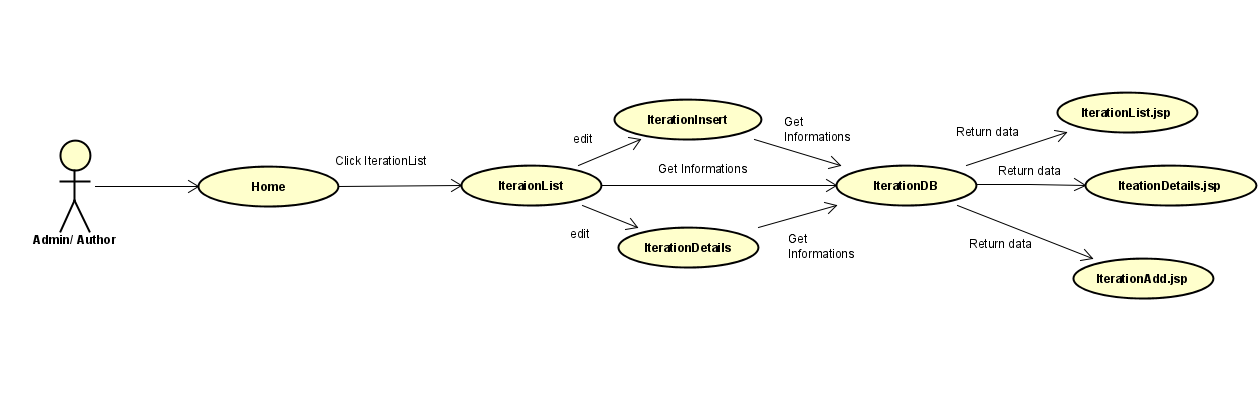
Iteration Servlet:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get List Iteration information* |

AddIteration Servlet:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Allow add new the iteration to the list* |

### c. Sequence Diagram(s)



### d. Database queries

"SELECT \* FROM student\_project\_management.iteration"

"SELECT \* FROM student\_project\_management.iteration where iteration\_id = ?"

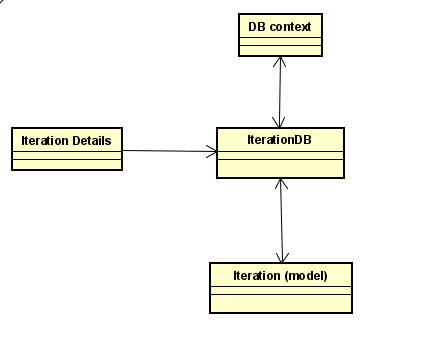
"INSERT INTO `student\_project\_management`.`iteration` (`subject\_id`,"`iter\_name`,"`duration`,`status`) VALUE(?,?,?,?);

"select \* from iteration where true ";

## 16. <Author/ Iteration Details>

When Author or Admin want to see details, they can click button Details and can change the information

### a. Class Diagram



### b. Class Specifications

InterationDetails Servlet:

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Get Iteration information of each and can change it.* |

### c. Sequence Diagram(s)

### 

### d. Database queries

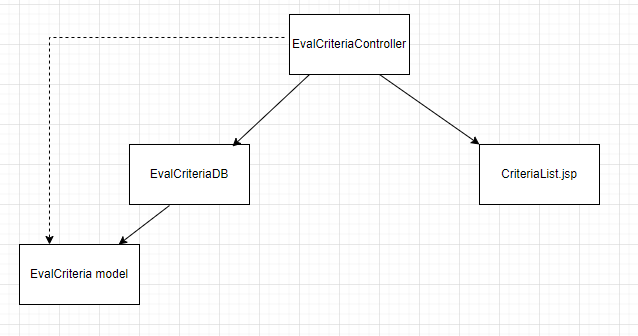
"UPDATE `student\_project\_management`.`iteration`

SET `subject\_id` = ?, iter\_name` = ?,`duration` = ?,`status` = ?

WHERE `iteration\_id` = ?";

## 17. <Author/Criteria List >

### a. Class Diagram



### b. Class Specifications

#### EvalCriteriaController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag null, add or details*  *Output: a EvalCriteria in database with that tag to specific jsp page (list or detail or add page)*  *And then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input: get update information or add new*  *Output: Update or add new, connect to DB*  *Send redirect to CriteriaList.jsp page* |

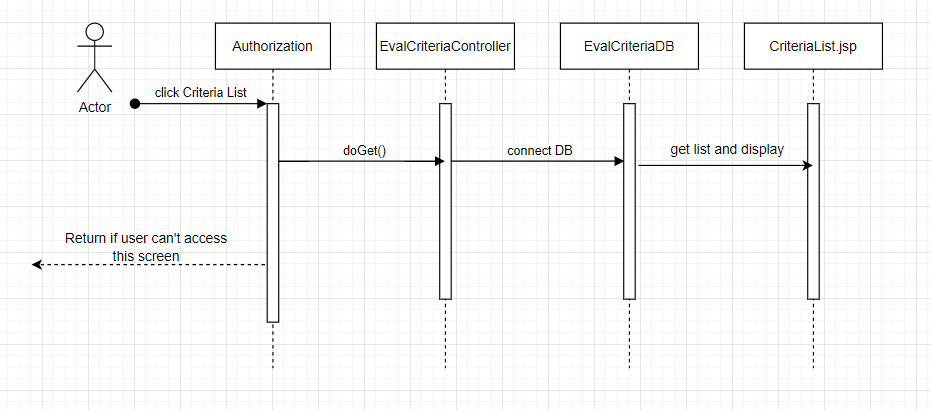
#### EvalCriteriaDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *count* | *Input: search value with author id*  *Output: int number of EvalCriteria in database that follow the search value or id* |
| *02* | *search* | *Input: String search\_teamEval, String status, String search\_value, String author, int curPage, int numPerPage*  *Output: List<EvalCriteria>* |
| *03* | *getIterList* | *Input: String author*  *Output: List<Iteration>* |
| *04* | *getTotalWeight* | *Input: int iterationId, boolean teamEval*  *Output: double result* |

#### EvalCriteria Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *EvalCriteria* | *Input: none*  *Just a no-parameter constructor* |
| *02* | *EvalCriteria* | *Input: int criteriaId, Iteration iteration, String evalTitle, double evalWeight, boolean teamEval, int criteriaOrder, int maxLoc, boolean status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

* *count:*

*select count(\*)*

*from eval\_criteria as ec, iteration as i, `subject` as s*

*where ec.interation\_id = i.iteration\_id and i.subject\_id = s.subject\_id and i.status = 1 and s.status = 1*

* *search:*

*select \* from*

*(select ec.criteria\_id, ec.interation\_id, ec.evaluation\_title, ec.evaluation\_weight, ec.team\_evaluation,ec.criteria\_order, ec.max\_loc, ec.`status`, row\_number() over*

*(order by ec.criteria\_id) as row\_index from eval\_criteria as ec, iteration as i, `subject` as s*

*where ec.interation\_id = i.iteration\_id and i.subject\_id = s.subject\_id and i.status = 1 and s.status = 1 )*

*as t where row\_index >= (?-1)\*?+1 and row\_index <= ?\*?*

* *getIterList:*

*SELECT i.iteration\_id, i.subject\_id, i.iter\_name, i.duration, i.`status`*

*FROM iteration as i, subject as s where i.subject\_id=s.subject\_id and s.status=1*

* *getTotalWeight*

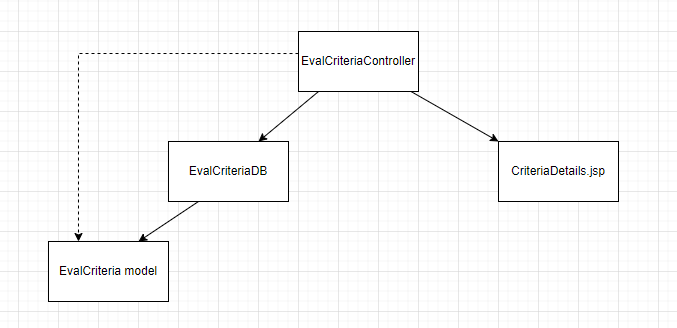
*select sum(evaluation\_weight)*

*from eval\_criteria where status = 1 and interation\_id = ? and team\_evaluation = ?*

*group by interation\_id, team\_evaluation*

## 18. <Author/Criteria Details>

### a. Class Diagram



### b. Class Specifications

#### EvalCriteriaController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag null, add or details*  *Output: a EvalCriteria in database with that tag to specific jsp page (list or detail or add page)*  *And then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input: get update information or add new*  *Output: Update or add new, connect to DB*  *Send redirect to CriteriaList.jsp page* |

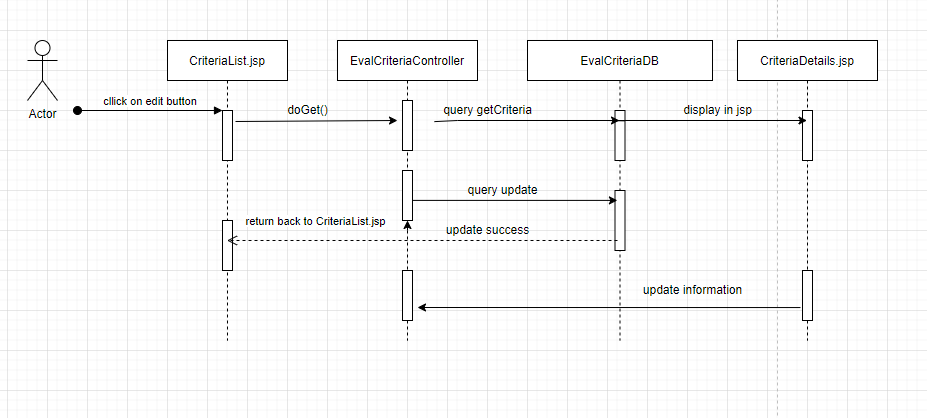
#### EvalCriteriaDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getEvalCriById* | *Input: String id, String author*  *Output:* *EvalCriteria* |
| *02* | *updateEvalCri* | *Input: EvalCriteria e*  *Output: void (update to DB)* |
| *03* | *getMaxPossibleWeight* | *Input: int id, int iterationId, boolean teamEval*  *Output: double result (max weight can be for a Criteria)* |
| *04* | *addEC* | *Input: EvalCriteria ec*  *Output: void (add new Eval Criteria to DB)* |

#### EvalCriteria Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *EvalCriteria* | *Input: none*  *Just a no-parameter constructor* |
| *02* | *EvalCriteria* | *Input: int criteriaId, Iteration iteration, String evalTitle, double evalWeight, boolean teamEval, int criteriaOrder, int maxLoc, boolean status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

* *getEvalCriById:*

*SELECT \* from eval\_criteria as ec, iteration as i, `subject` as s where criteria\_id = ?*

*and ec.interation\_id = i.iteration\_id and i.subject\_id = s.subject\_id*

* *updateEvalCri:*

*UPDATE `eval\_criteria`*

*SET*

*`interation\_id` = ?, `evaluation\_title` = ?, `evaluation\_weight` = ?, `team\_evaluation` = ?, `criteria\_order` = ?, `max\_loc` = ?, `status` = ?*

*WHERE `criteria\_id` =?*

* *getMaxPossibleWeight:*

*select sum(evaluation\_weight)*

*from eval\_criteria where status = 1 and interation\_id = ? and team\_evaluation = ? "*

*and criteria\_id != ?*

*group by interation\_id, team\_evaluation*

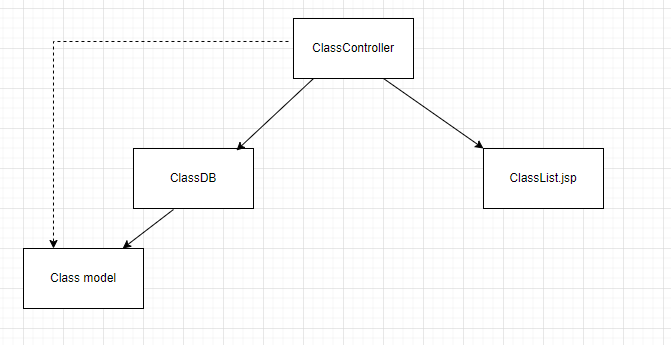
* *addEC:*

*INSERT INTO `eval\_criteria`*

*"(`interation\_id`,`evaluation\_title`,`evaluation\_weight`,`team\_evaluation`,`criteria\_order`,`max\_loc`,`status`) VALUES (?,?,?,?,?,?,?)*

## 19. <Author/Class List>

### a. Class Diagram



### b. Class Specifications

#### ClassController Classs

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag null, add or details*  *Output: a Class in database with that tag to specific jsp page (list or detail or add page)*  *And then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input: get update information or add new*  *Output: Update or add new, connect to DB*  *Send redirect to ClassList.jsp page* |

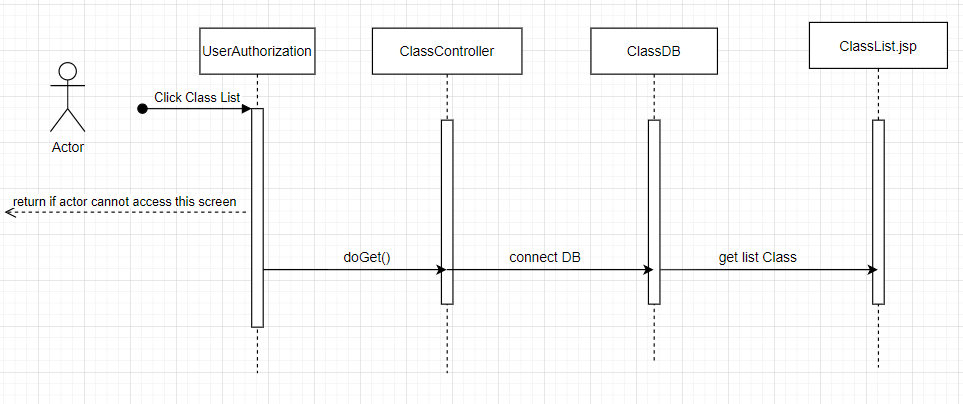
#### ClassDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *search* | *Input: String term, String status, String search\_value, String author, String trainer, int curPage, int numPerPage*  *Output: List<Class>* |
| *02* | *count* | *Input: String term, String status, String search\_value, String author, String trainer*  *Output: int number that is the number of row on class table in DB satisfy input value* |

#### Class Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Class* | *Input: none*  *Just a no-parameter constructor* |
| *02* | *Class* | *Input: int classId, String classCode, User trainer, Subject subject, int classYear, int term, boolean block5Class, int status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

* *search:*

*select \* from*

*(select c.class\_id, c.class\_code, c.trainer\_id, c.subject\_id, c.class\_year, c.class\_term, c.block5\_class, c.`status`, row\_number() over*

*(order by c.class\_id) as row\_index from class as c, subject as s*

*)as t where row\_index >= (?-1)\*?+1 and row\_index <= ?\*?*

* *count:*

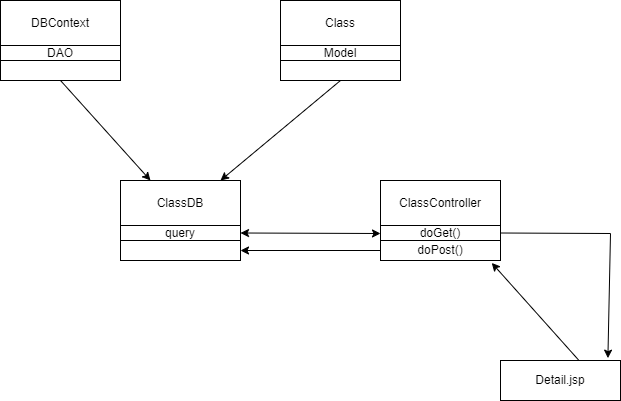
*select count(\*)*

*from class as c, subject as s*

*where c.subject\_id=s.subject\_id*

## 20. <Author/Class details>

### a. Class Diagram



### b. Class Specifications

#### ClassController Classs

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag add or details*  *Output: a Class with id in database with that tag to deatails jsp page to display* |
| *02* | *doPost* | *Input: get update information or add new*  *Output: Update or add new, connect to DB*  *Send redirect to ClassList.jsp page* |

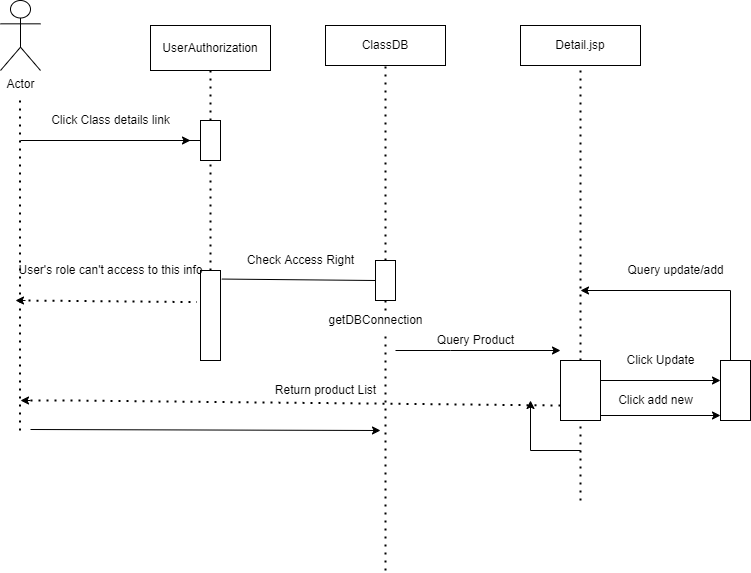
#### ClassDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getClassById(String id, String user\_id)* | *Input: String id, String user\_id(if user is role admin then use\_rid==””)*  *Output: Class* |
| *02* | *TrainerList()* | *Input:*  *Output: List of trainer name and id* |
| *03* | *SubjectList(int id)* | *Input: user id;*  *Output: List of subject code and id where assign author=id* |
| *04* | *UpdateClass(String id, Class c)* | *Input:String id, Class c*  *Output: Class c updated to database* |
| *05* | *AddClass(Class c)* | *Input: Class c;*  *Output: Add class c to database* |
| *06* | *CheckClassAndSubject(String code,String subject)* | *Input: String classCode, String subjectId*  *Output: true if classCode already have this subjectId, false if none* |

#### Class Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Class* | *Input: none*  *Just a no-parameter constructor* |
| *02* | *Class* | *Input: int classId, String classCode, User trainer, Subject subject, int classYear, int term, boolean block5Class, int status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

select \* from class where class\_id = ?

SELECT u.user\_id, u.full\_name from setting s inner join user u where s.setting\_id= u.role\_id and s.type\_id=1 and s.setting\_title='trainer' and u.status=1

select distinct subject\_id, concat(subject\_code,' (',subject\_name,')') from subject where author\_id=?

select \* from class as c, subject as s where c.class\_id=? and c.subject\_id=s.subject\_id and s.author\_id=? and c.trainer\_id=?

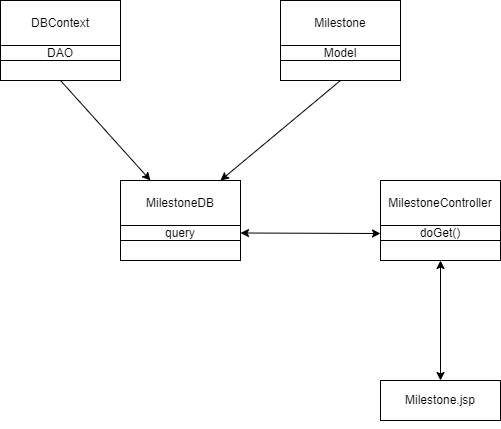
INSERT INTO `student\_project\_management`.`class`(`class\_code`,`trainer\_id`,`subject\_id`,`class\_year`,`class\_term`,‘block5\_class`,`status`) VALUES (?,?,?,?,?,?,?)

UPDATE `student\_project\_management`.`class` SET "`class\_code` = ?,`trainer\_id` = ?,`subject\_id` = ?,`class\_year` = ?,‘class\_term` = ?,`block5\_class` = ?,`status` = ? WHERE `class\_id` = ?

SELECT \* FROM student\_project\_management.class where class\_code=? and subject\_id=?

## 21. <Trainer/Milestone List>

### a. Class Diagram



### b. Class Specifications

#### ClassController Classs

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag null, add or details*  *Output: a List of Milestone in database with that tag to Milestone jsp page*  *And then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Null* |

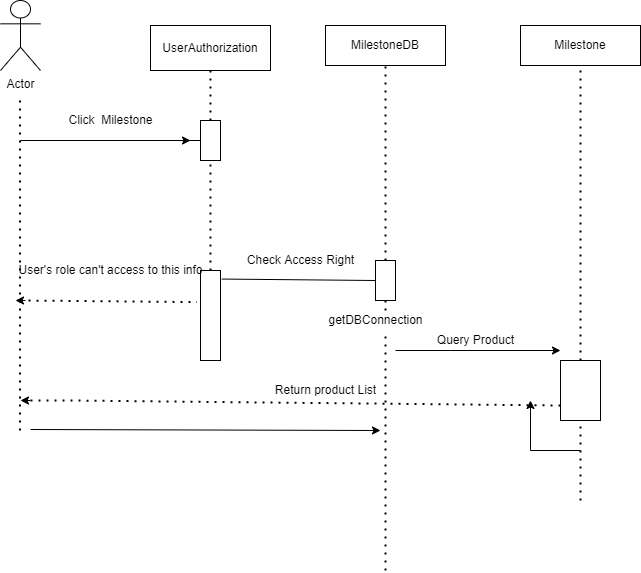
#### MilestoneDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *search* | *Input: String User\_id, String status, String Class, String from, String to, int curPage, int numPerPage*  *Output: List<Milestone> where trainer\_id=UserId and satisfy input value* |
| *02* | *count* | *Input: String User\_id, String status, String Class, String from, String to*  *Output: a number that is the number of row on Milestone table in DB satisfy input value* |

#### Milestone Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Milestone* | *Input: none*  *Just a no-parameter constructor* |
| *02* | *Milestone* | *Input: int milestoneId, String milestoneName, Iteration iteration, Class classId, Date fromDate, Date toDate, int status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

select \* from

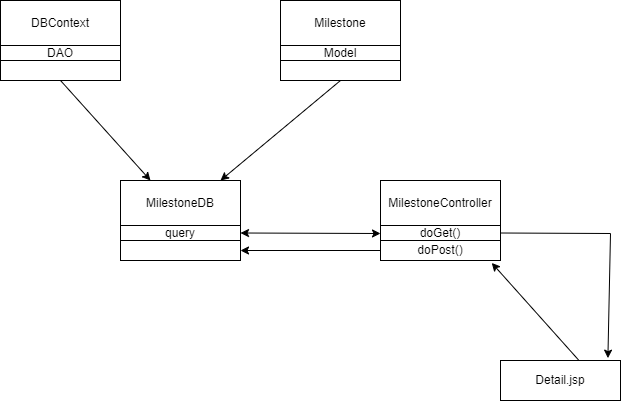
(select m.milestone\_id,m.milestone\_name, m.iteration\_id, m.class\_id, m.from\_date, m.to\_date,m.status, row\_number() over (order by m.milestone\_id) as row\_index from milestone as m, iteration as i, class as c where m.iteration\_id = i.iteration\_id and m.class\_id = c.class\_id and i.status = 1 and c.status = 1 and c.trainer\_id=? and m.class\_id=? and (m.from\_date between ? and ?) and (m.to\_date between ? and ?) and m.status =? )

as t where row\_index >= (?-1)\*?+1 and row\_index <= ?\*?

SELECT count(\*) FROM milestone m inner join class c on m.class\_id=c.class\_id inner join iteration i on m.iteration\_id=i.iteration\_id where c.trainer\_id=? and i.status=1 and c.status=1 and m.class\_id=? and (m.from\_date between ? and ?) and (m.to\_date between ? and ?) and m.status =?

## 22. <Trainer/Milestone Details>

### a. Class Diagram



### b. Class Specifications

#### ClassController Classs

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag add or details*  *Output: a Milestone in database with that tag to Detail.jsp page* |
| *02* | *doPost* | *Input:get Updated information or add new*  *Output: Milestone update or add to database* |

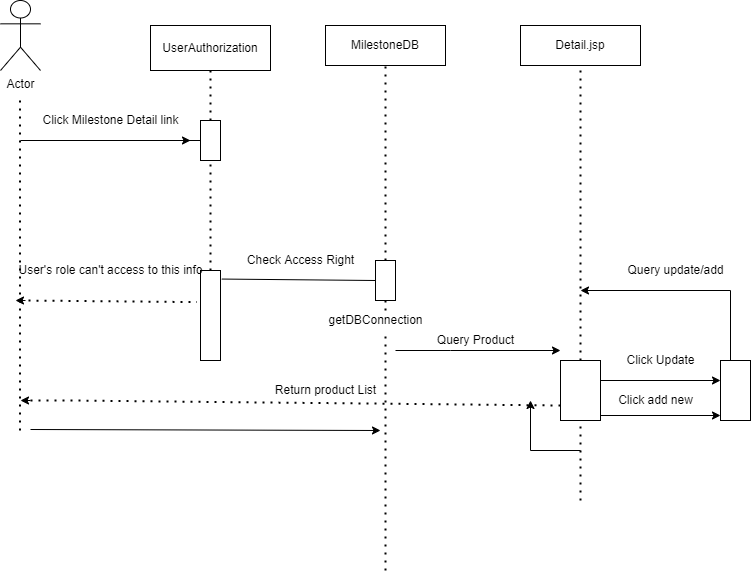
#### MilestoneDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getMilestoneById* | *Input: String Milestone\_id, int user\_id*  *Output: a Milestone where trainer\_id=user\_id* |
| *02* | *ClassList()* | *Input: int user\_id*  *Output: a List<Class> where trainer\_id=user\_id and Class\_status=true* |
| *03* | *getIterationByClass()* | *Input:Class\_id*  *Output a List<Iteration> of said class\_id* |
| *04* | *Update* | *Input:String id, String name, String from, String to, String status*  *Output: update Milestone to database* |
| *05* | *Add* | *Input:* *String name, String iteration, String Class, String from,String to ,String status*  *Output: add new Milestone to database* |
| *06* | *CheckUniqueMilestoneName()* | *Input: String name, String iteration\_id, String Class\_id*  *Output: true if in that iteration of that class, a milestone name already exist* |

#### Milestone Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Milestone* | *Input: none*  *Just a no-parameter constructor* |
| *02* | *Milestone* | *Input: int milestoneId, String milestoneName, Iteration iteration, Class classId, Date fromDate, Date toDate, int status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

select \* from milestone m inner join class c on m.class\_id=c.class\_id where m.milestone\_id=? and c.trainer\_id=?

SELECT \* FROM class where trainer\_id=? and status=1 order by class\_code

SELECT \* FROM iteration i inner join class c on c.subject\_id=i.subject\_id where c.class\_id=?

UPDATE `student\_project\_management`.`milestone` SET `milestone\_name` = ?,`from\_date` = ?,`to\_date` = ?, `status` = ? WHERE `milestone\_id` = ?;

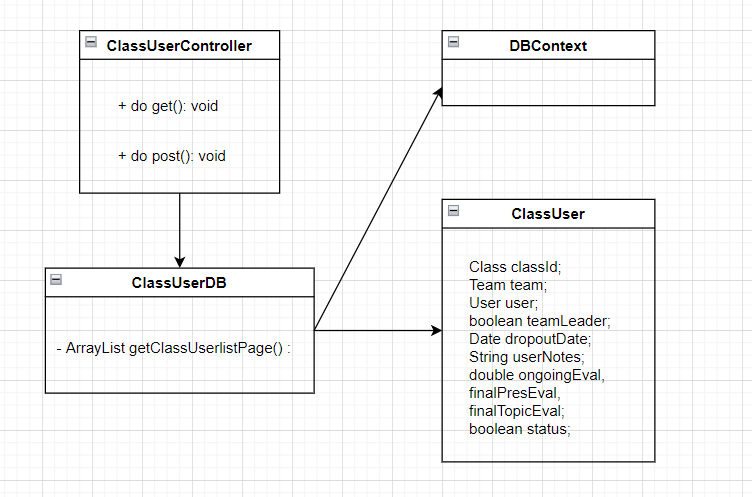
INSERT INTO `student\_project\_management`.`milestone`

(`milestone\_name`,`iteration\_id`,`class\_id`,`from\_date`,`to\_date`,`status`) VALUES(?,?,?,?,?,?)

SELECT \* FROM student\_project\_management.milestone where milestone\_name=? and iteration\_id=? and class\_id=?

## 23. <Trainer/Class User List>

### a. Class Diagram



### b. Class Specification

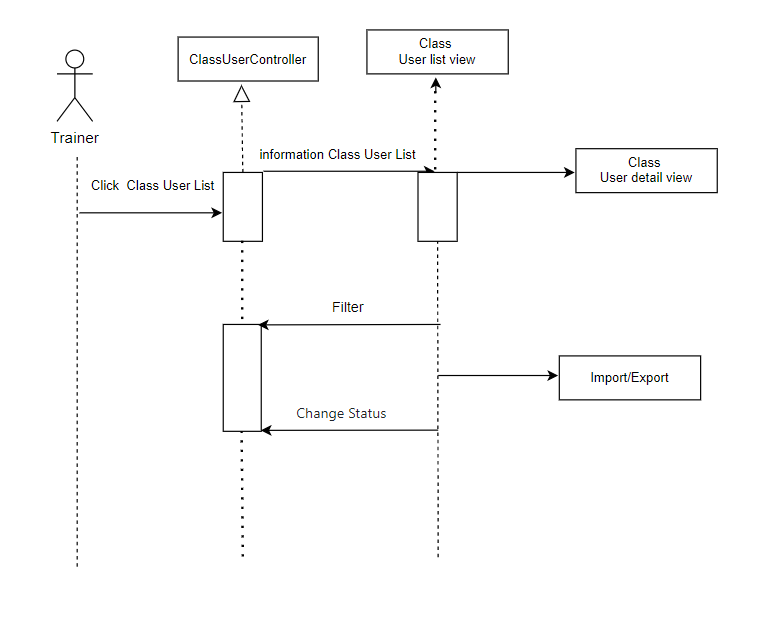
**ClassUserController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doPost() | send request data class user to the display class user list. |

**ClassUserDB**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getClassUserlistPage()* | *Input: ClassUser*  *Output: get user class information by page.* |
| *02* | *getClasslistTrainer()* | *Input: Trainer id*  *Output: Retrieve all classes of a trainer* |

### *c*. *Sequence* Diagram

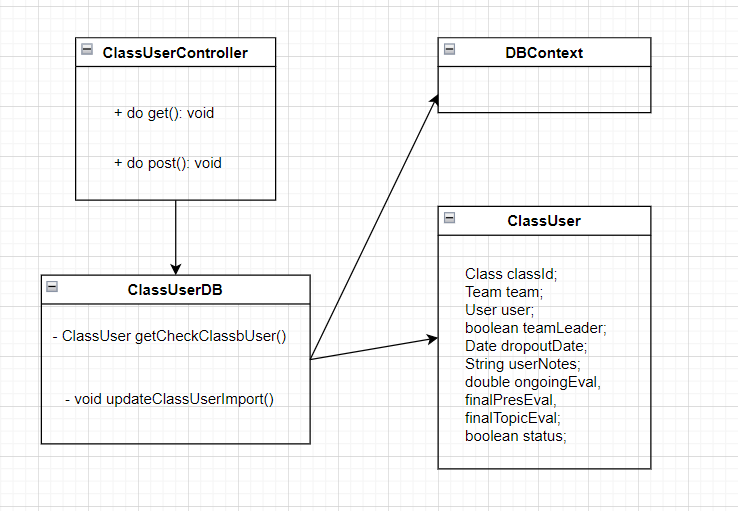
****

### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | ArrayList<ClassUser> getClassUserlistPage(): | SELECT \* from(SELECT class\_user.class\_id, class\_user.team\_id, class\_user.user\_id, class\_user.team\_leader, class\_user.dropout\_date, user\_notes, class\_user.ongoing\_eval, class\_user.final\_pres\_eval, class\_user.final\_topic\_eval,class\_user.status,ROW\_NUMBER() OVER "(ORDER BY class\_user.class\_id ASC) as row\_index FROM class\_user  INNER JOIN class ON class\_user.class\_id = class.class\_id  INNER JOIN user ON user.user\_id = class\_user.user\_id  where (class\_user.team\_leader "+team\_leader+" and class.class\_code like '%"+class\_code+"%' "+dropout\_date+" and  class\_user.status "+status+" and class.class\_year "+class\_year+" and class.class\_term "+class\_term+" ) and  (user.full\_name like '%"+full\_name+"%' OR user.roll\_number like '%"+roll\_number+"%') ) as t  WHERE row\_index >= (? -1)\*? + 1 AND row\_index <= ?\*? |

## 24. <Trainer/Class User Details>

### a. Class Diagram



### 

### b. Class Specifications

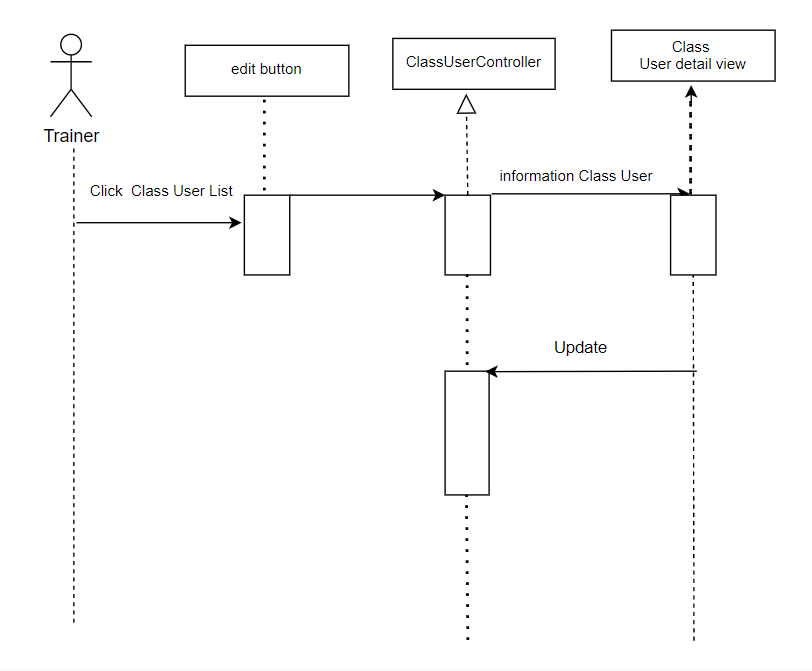
**ClassUserController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doPost() | send the class user information to the Class User Detail user of the class displayed. |
| 2 | doGet() | Update user information in database table "Class User". |

**ClassUserDB**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Void getCheckClassbUser()* | *Input: ClassID, TeamID, UserId*  *Output: get Student information in Class User.* |
| *02* | *void updateClassUserImport()* | *Input: ClassID, TeamID, UserId*  *Output: Update user information in database table "Class User"* |

### c. Sequence Diagram(s)

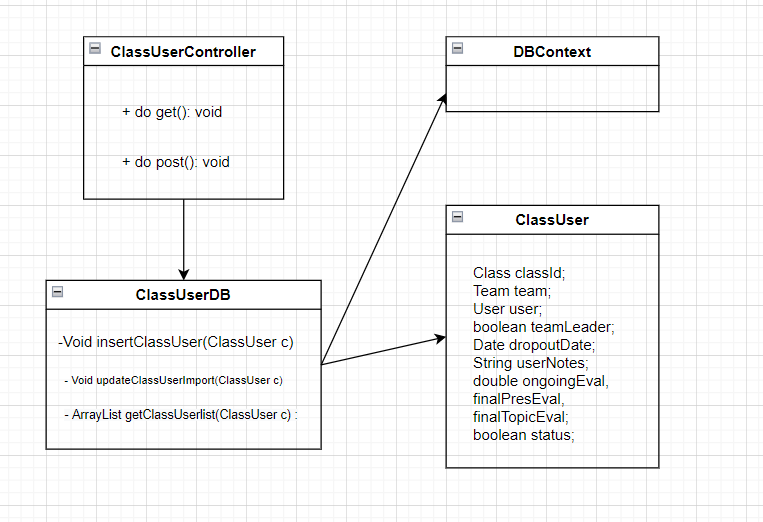


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | void updateClassUserImport(): | UPDATE `class\_user`  SET (`team\_leader` = ?,  `dropout\_date` = ?,  `user\_notes` = ?,  `ongoing\_eval` = ?,  `final\_pres\_eval` = ?,  `final\_topic\_eval` = ?)  WHERE `class\_id` = ? AND `team\_id` = ? AND `user\_id` = ? |

## 25. <Trainer/Class User Import/Export>

### a. Class Diagram



### b. Class Specifications

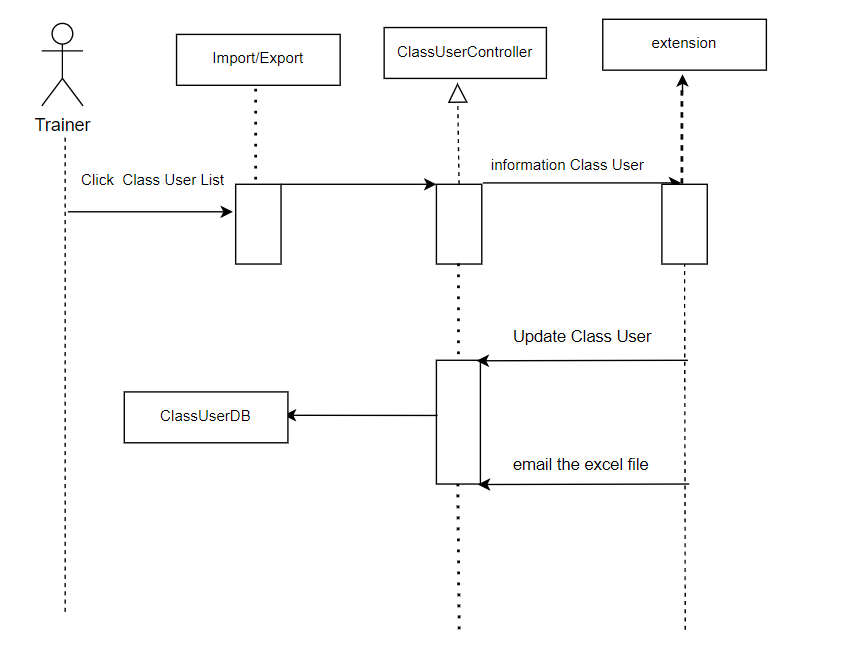
**ClassUserController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doPost() | -send email containing excel file to Trainer.  -Read excel file to get user class data and update data in database. |

**ClassUserDB**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Void getCheckClassbUser()* | *Input: ClassID, TeamID, UserId*  *Output: get Student information in Class User.* |
| *02* | *void updateClassUserImport()* | *Input: ClassID, TeamID, UserId*  *Output: Update user information in database table "Class User"* |

### c. Sequence Diagram(s)

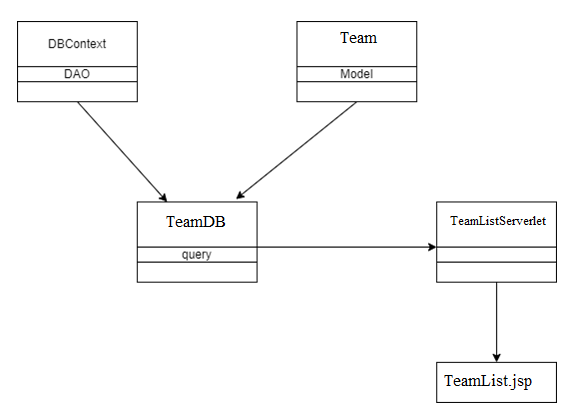


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | void updateClassUserImport(): | UPDATE `class\_user`  SET (`team\_leader` = ?,  `dropout\_date` = ?,  `user\_notes` = ?,  `ongoing\_eval` = ?,  `final\_pres\_eval` = ?,  `final\_topic\_eval` = ?)  WHERE `class\_id` = ? AND `team\_id` = ? AND `user\_id` = ? |

## 26. <Trainer/Team List>

### a. Class Diagram



### b. Class Specifications

#### DBContext Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *DBContext()* | *Setup a connection to MySql database* |

#### Team Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructor* | *Construct an object with provided value* |
| *02* | *Getter* | *Get a field value* |
| *03* | *Setter* | *Set value for a field* |

#### TeamDB Class

***Class Methods***

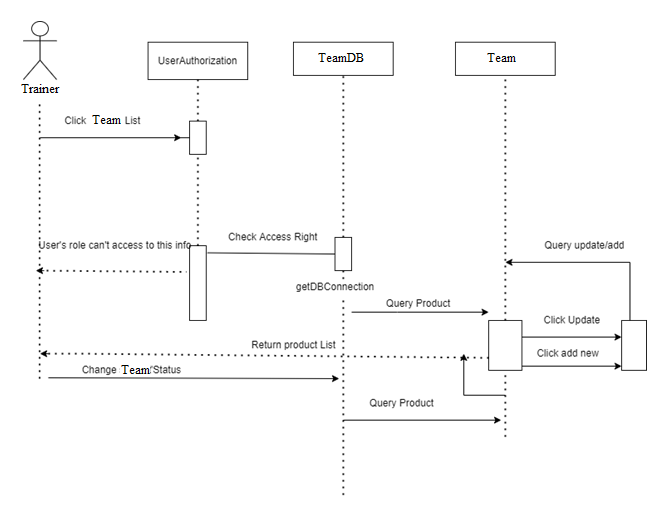
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Search(String,String,String)* | *Search for a set of specific Team that meet all 3 requirement parameter, return all Subject if 3 string is empty* |
| *02* | *changeStatus(Team)* | *Change Team status* |
| *03* | *GetListByPage(List,int,int)* | *Pagination support* |

#### SettingListServlet Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Query a list of Team, search, change status* |
| *02* | *doPost()* | *Add new status if id not exist, update otherwise* |

### c. Sequence Diagram(s)



### d. Database queries

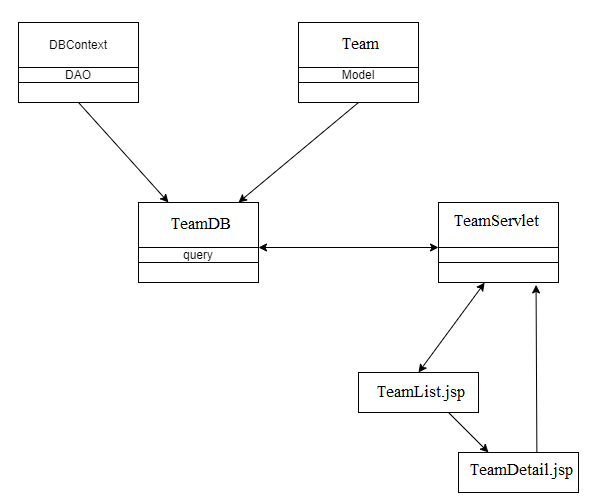
Select: SELECT \* FROM student\_project\_management.team where true and class\_id=? and status=? and topic\_name like ?;

Update: UPDATE student\_project\_management.team SET team\_code =? ,class\_id = ?,topic\_code = ?,topic\_name = ?, gitlab\_url = ?, status = ? WHERE team\_id = ?;

Change status: UPDATE team SET status = ? WHERE team\_id = ?;

## 27. < Trainer/Team Details>

### a. Class Diagram



### b. Class Specifications

#### DBContext Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *DBContext()* | *Setup a connection to MySql database* |

#### Team Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructor* | *Construct an Team with provided value* |
| *02* | *Getter* | *Get a field value* |
| *03* | *Setter* | *Set value for a field* |

#### TeamDB Class

***Class Methods***

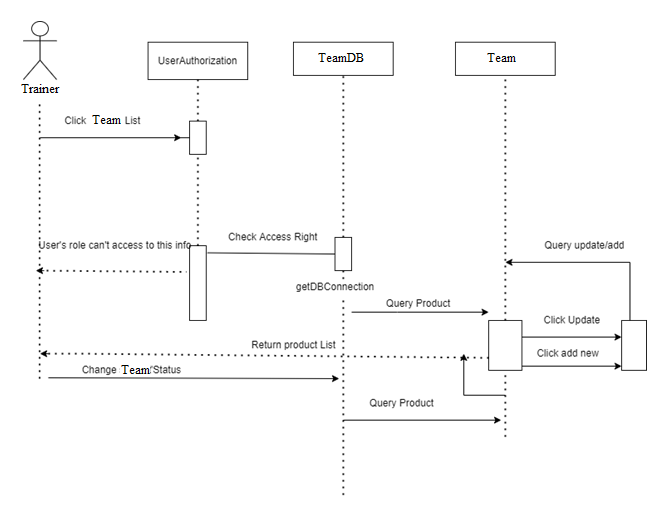
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *02* | *Update(Team)* | *Update a Team to the database* |
| *03* | *Add(Team)* | *Add Team to database* |

#### TeamListServlet Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Query a list of Team, search, change status* |
| *02* | *doPost()* | *Add new status if id not exist, update otherwise* |

### c. Sequence Diagram(s)



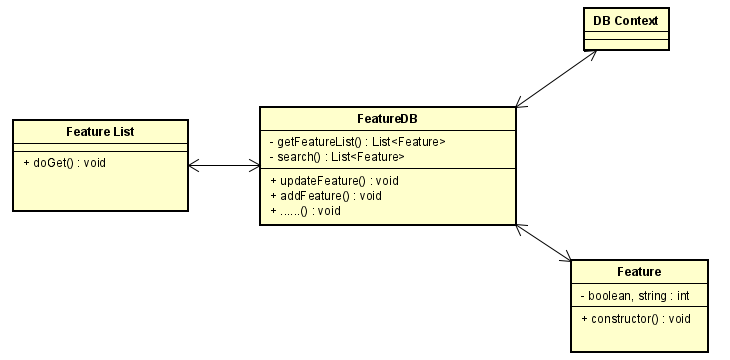
### d. Database queries

Add : INSERT INTO `student\_project\_management`.`team (team\_code, class\_id, topic\_code, topic\_name, gitlab\_url, status) VALUES (?, ?, ? ,?, ?, ?);

Update: UPDATE student\_project\_management.team SET team\_code =? ,class\_id = ?,topic\_code = ?,topic\_name = ?, gitlab\_url = ?, status = ? WHERE team\_id = ?;

## 28. <Student/Feature List>

1. **Class diagram**

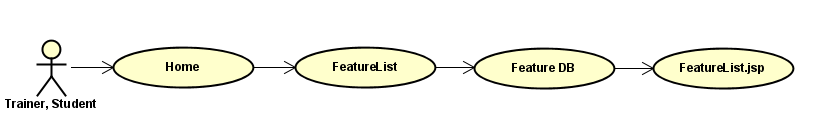


1. **Class Specifications:**

Feature Servlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get List Feature information* |

### c. Sequence Diagram(s)



### d. Database queries

“ select f.feature\_id, concat(t.topic\_name,'(', t.topic\_code,')') as topic, f.feature\_name,t.team\_id, f.status from feature as f, team as t where t.team\_id = f.team\_id “

"INSERT INTO `student\_project\_management`.`feature`\n"

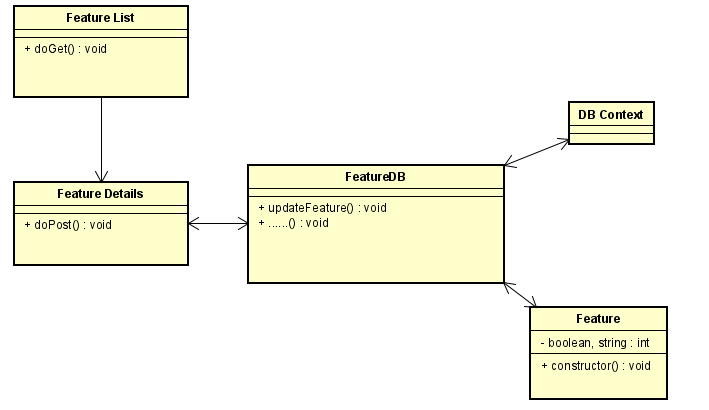
                + "(`feature\_id`, `team\_id`, `feature\_name`,`status`)\n"

                + "VALUES\n"

                + "(?,?,?,?)"

## 29. <Student/Feature Details>

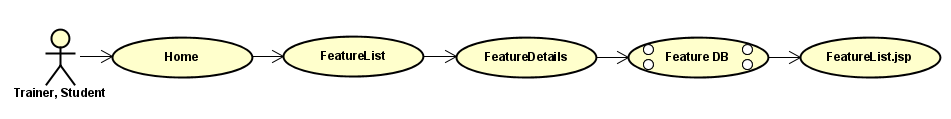
**a. Class diagrams**



1. **Class Specifications:**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Get Feature information that you chose.* |

### c. Sequence Diagram(s)

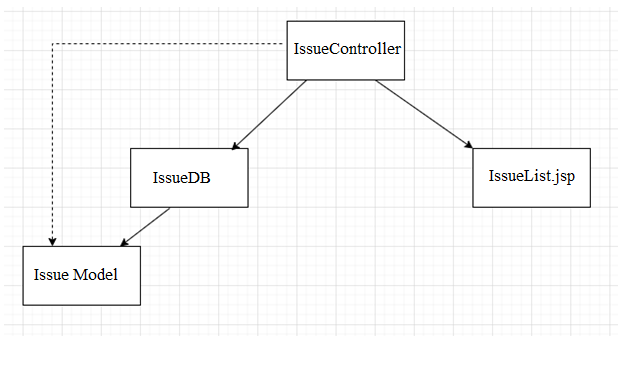


**d. Database queries**

"UPDATE `student\_project\_management`.`feature` SET `team\_id` = ?,`feature\_name` = ?,’status` = ? WHERE `feature\_id` = ?

## 30. <Student /Issue List>

### a. Class Diagram



### b. Class Specifications

#### DBContext Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *DBContext()* | *Setup a connection to MySql database* |

#### Issue Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructor* | *Construct an object with provided value* |
| *02* | *Getter* | *Get a field value* |
| *03* | *Setter* | *Set value for a field* |

#### TeamDB Class

***Class Methods***

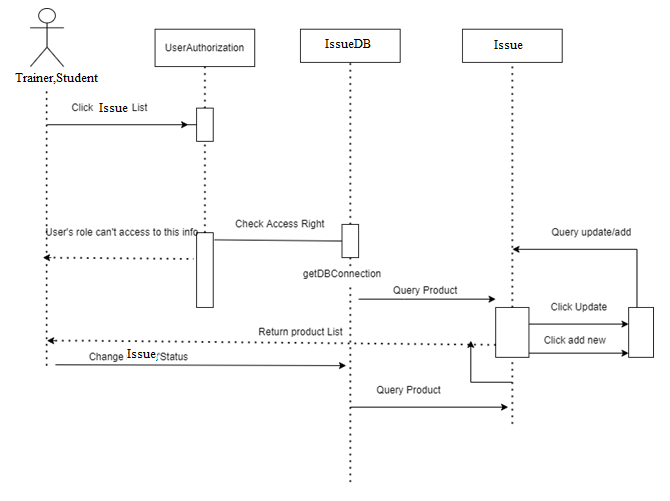
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Search(String,String,String)* | *Search for a set of specific Issue that meet all 3 requirement parameter, return all Subject if 3 string is empty* |
| *02* | *changeStatus(Team)* | *Change Issue status* |
| *03* | *GetListByPage(List,int,int)* | *Pagination support* |

#### SettingListServlet Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Query a list of Issue, search, change status* |
| *02* | *doPost()* | *Add new status if id not exist, update otherwise* |

### c. Sequence Diagram(s)



### d. Database queries

Select: select \* from issue;

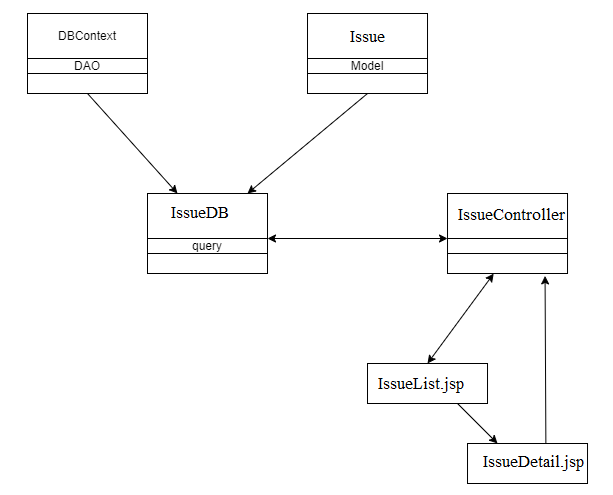
Search: select \* from student\_project\_management.issue;

Select x.\*, y.full\_name from student\_project\_management.issue x join student\_project\_management.user y on x.assignee\_id = y.user\_id where true and x.status = ? and issue\_title like ? or full\_name like ?

Change status: UPDATE issue SET status = ? WHERE issue\_id = ?;

## 31. < Student/Issue Details>

### a. Class Diagram



### b. Class Specifications

#### DBContext Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *DBContext()* | *Setup a connection to MySql database* |

#### Issue Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Constructor* | *Construct an Team with provided value* |
| *02* | *Getter* | *Get a field value* |
| *03* | *Setter* | *Set value for a field* |

#### IssueDB Class

***Class Methods***

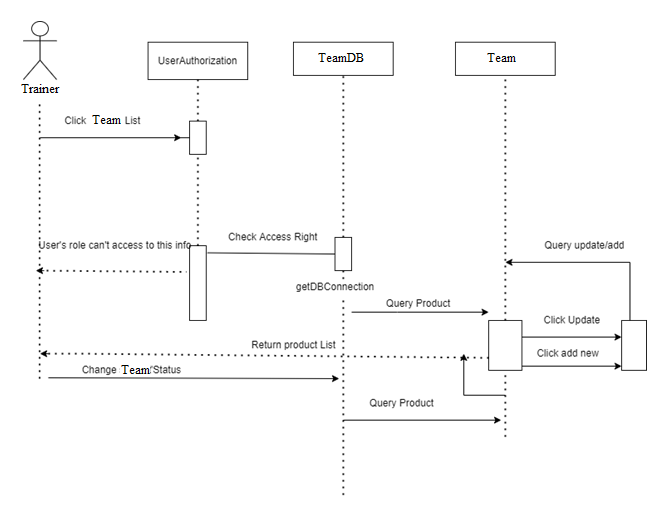
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *02* | *Update(Issue)* | *Update a Issue to the database* |
| *03* | *Add(Issue)* | *Add Issue to database* |

#### IssueListServlet Class

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Query a list of issue, search, change status* |
| *02* | *doPost()* | *Add new status if id not exist, update otherwise* |

### c. Sequence Diagram(s)



### d. Database queries

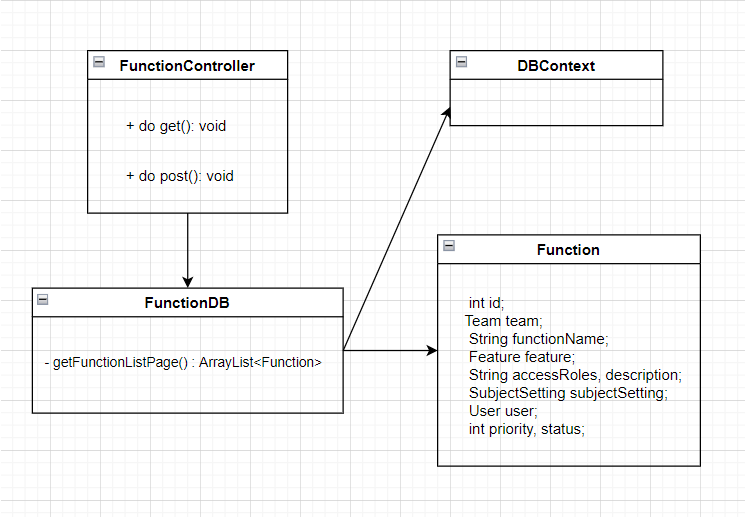
Add New: INSERT INTO student\_project\_management.issue assignee\_id , issue\_type , issue\_title , descript, gitlab\_id, gitlab\_url, created\_at, due\_date ,team\_id , milestone\_id , function\_ids, labels, status

VALUES (?, ?, ? ,?, ?, ?, curdate(), ?, ? ,?, ?, ?, ?);

Update: UPDATE student\_project\_management.issue SET issue\_id = ?, assignee\_id = ?, issue\_type = ?, issue\_title = ?, descript = ?, gitlab\_id= ?, gitlab\_url =?, created\_at =?, due\_date = ?, team\_id = ?, milestone\_id = ?, function\_ids = ?, labels = ?, status = ? where issue\_id = ?;

## 32.<Student/Function List>

### a. Class Diagram



### b. Class Specification

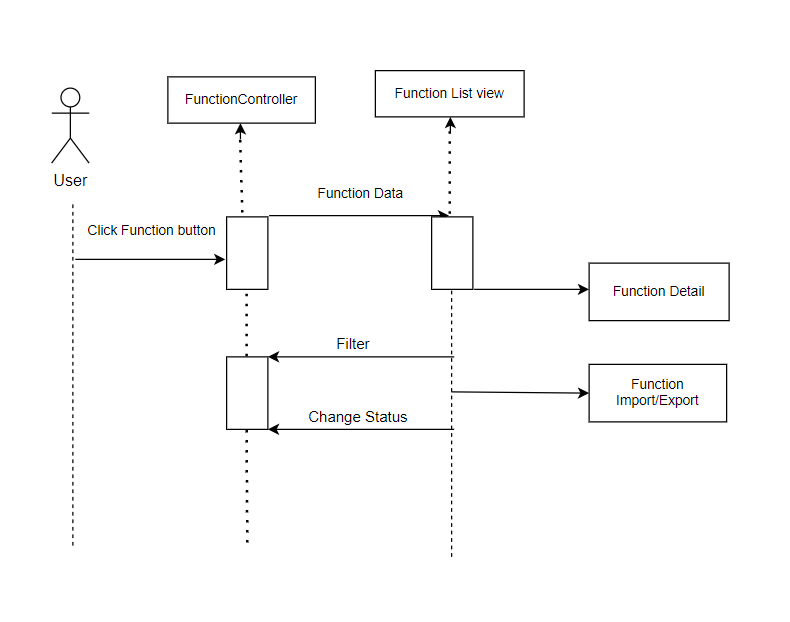
**FunctionController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet() | send request data Function to the display Function list. |

**FunctionDB**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getFunctionlistPage()* | *Input: Function*  *Output: get Function information by page.* |
| *02* | *getClasslist ()* | *Input: Trainer id or Student*  *Output: Retrieve all Function of a Trainer or Student* |

### *c*. *Sequence* Diagram

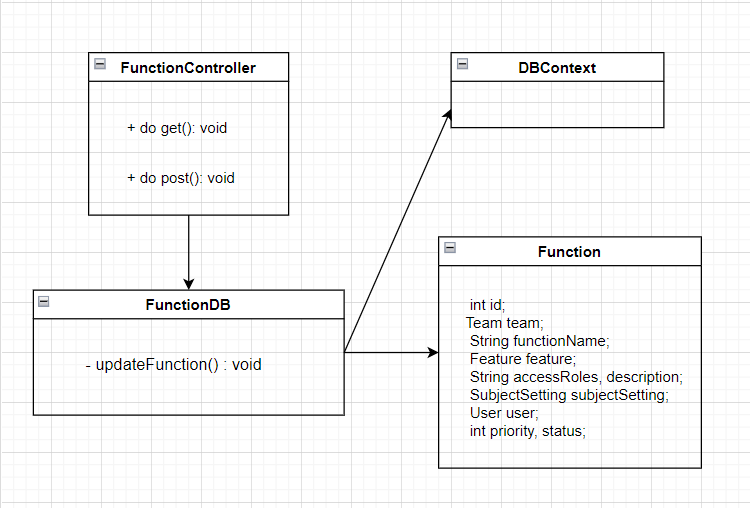


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | ArrayList<Function> getFunctionlistPage(): | select \* from (select fu.function\_id,fu.team\_id,fu.function\_name,fu.feature\_id,fu.access\_roles,  fu.description,fu.complexity\_id,fu.owner\_id,fu.priority,fu.status ,  ROW\_NUMBER() OVER (ORDER BY function\_id ASC) as row\_index from  `function` as fu  INNER JOIN feature as fe on fe.feature\_id = fu.feature\_id  WHERE (fu.function\_name like '%?%' or fe.feature\_name like '%?%' or fu.access\_roles like '%trainer%') and fu.team\_id like '%?%'  and fu.complexity\_id like '%?%' and fu.priority like '%?%' and fu.status like '%?%' ) as t  WHERE row\_index >= (? -1)\*? + 1 AND row\_index <= ?\*? |

## 23. <Trainer/Function Details>

### a. Class Diagram



### b. Class Specifications

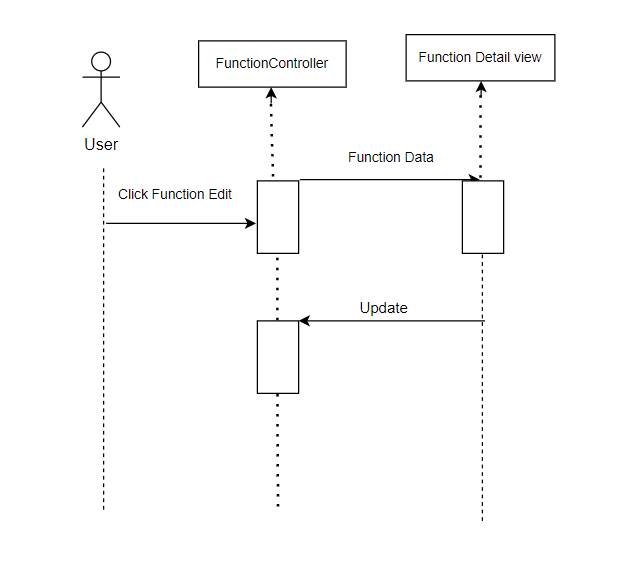
**FunctionController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet () | send the Function information to the Function Detail user of the Function displayed. |
| 2 | doPost() | Update Function information in database table "Function". |

**FunctionDB**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Void updateFunction()* | *Input: model Function*  *Output: get Function information* table "Function". |

### c. Sequence Diagram(s)

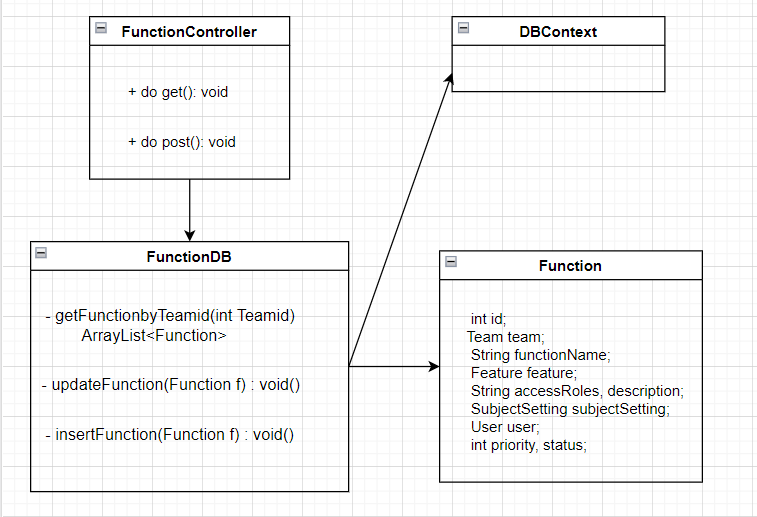


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | void updateFunction (function f): | UPDATE `function`  SET `team\_id` = ?, `function\_name` = ?,  `feature\_id` = ?,  `access\_roles` = ?,  `description` = ?,  `complexity\_id` = ?,  `owner\_id` = ?,  `priority` = ?,  `status` = ? WHERE `function\_id` = ?; |

## 34. <Trainer/Function Import/Export>

### a. Class Diagram



### b. Class Specifications

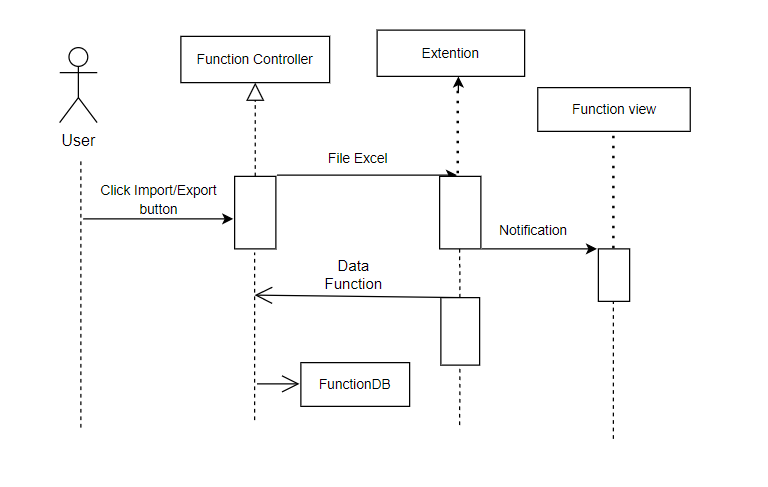
**FunctionController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doPost() | - Download the excel file to your computer.  - Read excel file to get user class data and update data in database. |

**FunctionDB**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Void* updateFunction(Function f)*()* | *Input: model Function*  *Output: Update Function information in table “Function”.* |
| *02* | *Void* insertFunction(Function f)*()* | *Input: model Function*  *Output: Insert Function information in table “Function”.* |

### c. Sequence Diagram(s)

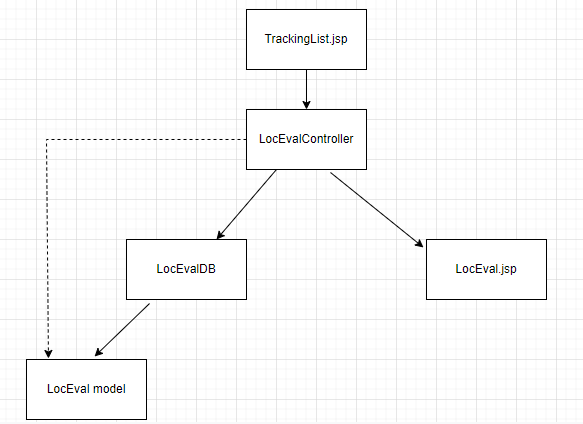


### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Script** |
| 1 | void updateFunction (Function f): | UPDATE `function`  SET `team\_id` = ?, `function\_name` = ?,  `feature\_id` = ?,  `access\_roles` = ?,  `description` = ?,  `complexity\_id` = ?,  `owner\_id` = ?,  `priority` = ?,  `status` = ? WHERE `function\_id` = ?; |
| 2 | Void insertFunction(Function f): | INSERT INTO `function` (`team\_id`, `function\_name`, `feature\_id`, `access\_roles`, `description`, `complexity\_id`, `owner\_id`, `priority`, `status`)  VALUES (?,?,?,?,?,?,?,?,?) |

## 35. <Trainer/Loc Evaluation>

### a. Class Diagram



### b. Class Specifications

#### LocEvalController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag view, add or update and id*  *Output: get related information and then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input: get update information or add new*  *Output: Update or add new, connect to DB*  *Send redirect to former Tracking list page* |

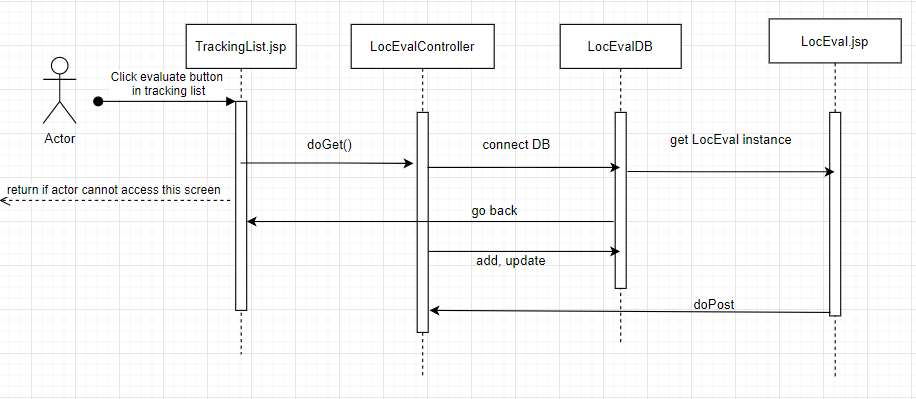
#### LocEvalDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getMilestoneById* | *Input: String id*  *Output: a Milestone instance with that id* |
| *02* | *getTrackingById* | *Input: int trackingId, String stu, String tra*  *Output: a Tracking with that id and follow by studentId or trainerId* |
| *03* | *getLocEvalByTrackingId* | *Input: int trackingId, String stu, String tra*  *Output: a LocEval instance with id, studentId or trainerId* |
| *04* | *updateTrackingStatus* | *Input: int id, int status*  *Output: void* |
| *05* | *update* | *Input: LocEval le*  *Output: void* |
| *06* | *add* | *Input: LocEval le*  *Output: void* |
| *07* | *calculateConvertedLoc* | *Input: LocEva le*  *Output: int convertedLoc which is calculated by complexity and quality in table subject\_setting* |

#### LocEval Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *LocEval* | *Input: none*  *Just a no-parameter constructor* |
| *02* | *LocEval* | *Input: int id, Date time, String note, SubjectSetting complexity, SubjectSetting quality, Tracking tracking, boolean status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

* *getMilestoneById:*

*select \* from milestone where milestone\_id=?*

* *getTrackingById:*

*SELECT \* FROM tracking tr, class c, team t, class\_user cu*

*where c.class\_id = t.class\_id and tr.team\_id = t.team\_id*

*and t.team\_id = cu.team\_id and tr.tracking\_id = ?*

* *getLocEvalByTrackingId:*

*SELECT\_SQL + " where tracking\_id = ?*

* *updateTrackingStatus:*

*UPDATE `tracking`*

*SET `status` = ?*

*WHERE `tracking\_id` = ?*

* *update:*

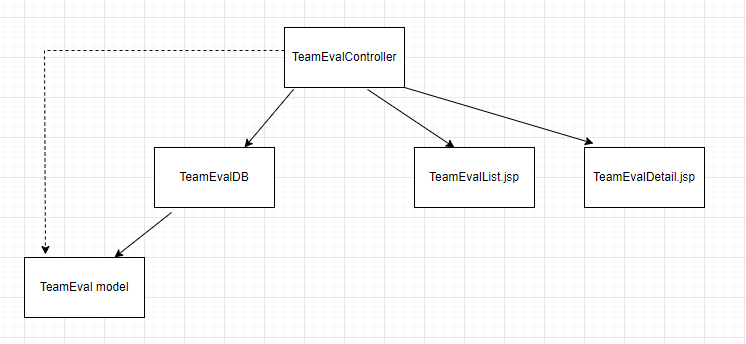
*UPDATE\_SQL*

* *add:*

*INSERT\_SQL*

## 36. <Trainer / Team Evaluation>

### a. Class Diagram



### b. Class Specifications

#### TeamEvalController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag null, add or details*  *Output: get related information and then setAttribute by request and forward to jsp page to display* |
| *02* | *doPost* | *Input: get update information or add new*  *Output: Update or add new, connect to DB*  *Send redirect to former TeamEvalList.jsp* |

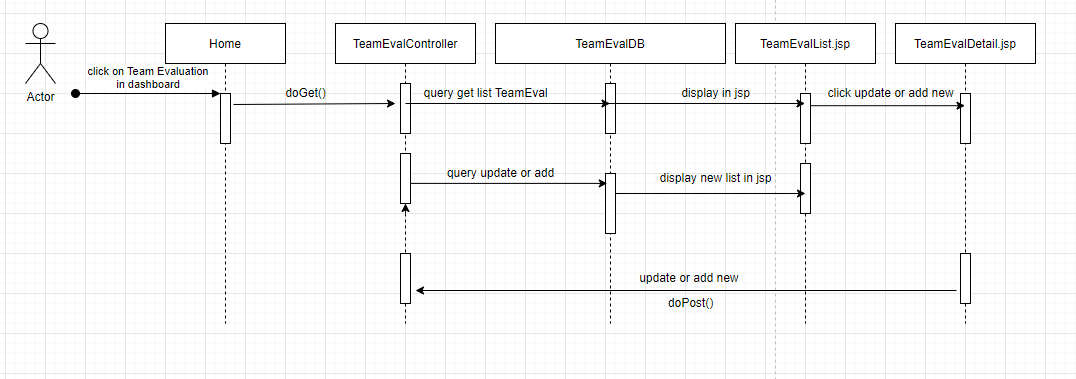
#### TeamEvalDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *search* | *Input: String search\_team, String search\_value, String stu, String tra, int curPage, int numPerPage*  *Ouput: List<TeamEval> with user search and page* |
| *02* | *count* | *Input: String search\_team, String search\_value, String stu, String tra*  *Output: int which is number of TeamEval in DB satisfy* |
| *03* | *getListTeam* | *Input: String stu, String tra*  *Output: List<Team>* |
| *04* | *getTEById* | *Input: String id*  *Output: TeamEval with that id or null if not exist* |
| *05* | *isTEExist* | *Input: String criteria, String team*  *Output: int which is it id if exist or -1 if not exist* |
| *06* | *update* | *Input: TeamEval te*  *Output: void* |
| *07* | *add* | *Input: TeamEval te*  *Output: void* |

#### TeamEval Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | TeamEval | *Input: none*  *Just a no-parameter constructor* |
| *02* | TeamEval | *Input: int id, EvalCriteria evalCriteria, Team team, double grade, String note, boolean status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

* *search:*

*select \* from*

*(select te.team\_eval\_id, te.criteria\_id, te.team\_id, te.grade, te.note, te.status ,row\_number() over (order by te.team\_eval\_id) as row\_index from team\_eval as te, team as t, class\_user as cu, class as c, subject as s, eval\_criteria as ec*

*where te.status = 1 and te.team\_id = t.team\_id and t.class\_id = cu.class\_id and t.class\_id = c.class\_id and c.subject\_id = s.subject\_id and te.criteria\_id = ec.criteria\_id*

* *count:*

*with T as*

*(select distinct te.team\_eval\_id, te.criteria\_id, te.team\_id, te.grade, te.note, te.status*

*from team\_eval as te, team as t, class\_user as cu, class as c, subject as s, eval\_criteria as ec*

*where te.team\_id = t.team\_id and t.class\_id = cu.class\_id and t.class\_id = c.class\_id and c.subject\_id = s.subject\_id and te.criteria\_id = ec.criteria\_id*

* *getListTeam:*

*SELECT distinct te.team\_id FROM team\_eval te, class c, team t, class\_user cu*

*where c.class\_id = t.class\_id and te.team\_id = t.team\_id "*

*and t.team\_id = cu.team\_id*

* *getTEById:*

*select \* from team\_eval where team\_eval\_id = ?*

* *isTEExist:*

*select \* from team\_eval where criteria\_id = ? and team\_id = ?*

* *update:*

*UPDATE `team\_eval`*

*SET `criteria\_id` = ?, `team\_id` = ?,`grade` = ?,`note` = ?,`status` = ?*

*WHERE `team\_eval\_id` = ?*

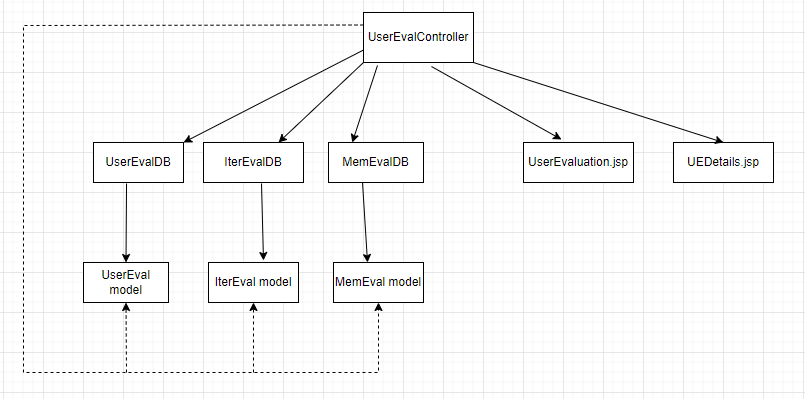
* *add:*

*INSERT INTO `team\_eval`(`criteria\_id`,`team\_id`,`grade`,`note`)*

*VALUES (?,?,?,?)*

## 37. <Trainer/User Evaluation>

### a. Class Diagram



### b. Class Specifications

#### UserEvalController Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet* | *Input: tag null, details, export*  *Output: get related information and then setAttribute by request and forward to jsp page to display or export data to .xlsx file* |
| *02* | *doPost* | *Input: tag details, import*  *Output: Update or add new, connect to DB*  *Import data from xlsx file to DB* |
| *03* | *readExcel* | *Input: InputStream inputStream, String import\_class, String import\_iter*  *Output: List<UserEval> read from xlsx file* |
| *04* | *getCellValue* | *Input: Cell cell*  *Output: Object value of a cell in xlsx file* |

#### IterEvalDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getIEOfTracking* | *Input:* *int trackingId*  *Output: IterEval instance* |
| *02* | *getListIEByTE* | *Input: TeamEval te*  *Output: List<IterEval>* |
| *03* | *isIEExist* | *Input: IterEval ie*  *Output: int* |
| *04* | *add* | *Input: IterEval ie*  *Output: void* |
| *05* | *update* | *Input: IterEval ie*  *Output: void* |
| *06* | *getIEById* | *Input: String id*  *Output: IterEval instance* |
| *07* | *teamGradeOfIE* | *Input:* *IterEval ie*  *Output: double* |
| *08* | *weightOfTeamGradeOfIE* | *Input: IterEval ie*  *Output: double* |
| *09* | *memGradeOfIE* | *Input: IterEval ie*  *Output: double* |

#### MemEvalDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getMemEvalOfLoc* | *Input: int evalId*  *Output: MemEval instance* |
| *02* | *isMEExist* | *Input:* *MemEval me*  *Output: int* |
| *03* | *getMEById* | *Input: String id*  *Output: MemEval instance* |
| *04* | *add* | *Input: MemEval me*  *Output: void* |
| *05* | *update* | *Input: MemEval me*  *Output: void* |
| *06* | *getGrade* | *Input: String ieId, String ecId*  *Output: double* |
| *07* | *getMEOfIE* | *Input: IterEval ie*  *Output: List<MemEval>* |

#### UserEvalDB Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *search* | *Input:* *String search\_class, String search\_team, String search\_iter, String search\_value, String stu, String tra, int curPage, int numPerPage*  *Output:* *List<UserEval>* |
| *02* | *count* | *Input:* *String search\_class, String search\_team, String search\_iter, String search\_value, String stu, String tra*  *Output: int* |
| *03* | *importData* | *Input: List<UserEval> listUE*  *Output:void* |

#### MemEval Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | MemEval | *Input: none*  *Just a no-parameter constructor* |
| *02* | MemEval | *Input: int id, IterEval iterEval, EvalCriteria evalCriteria, int convertedLoc, double grade, String note, boolean status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

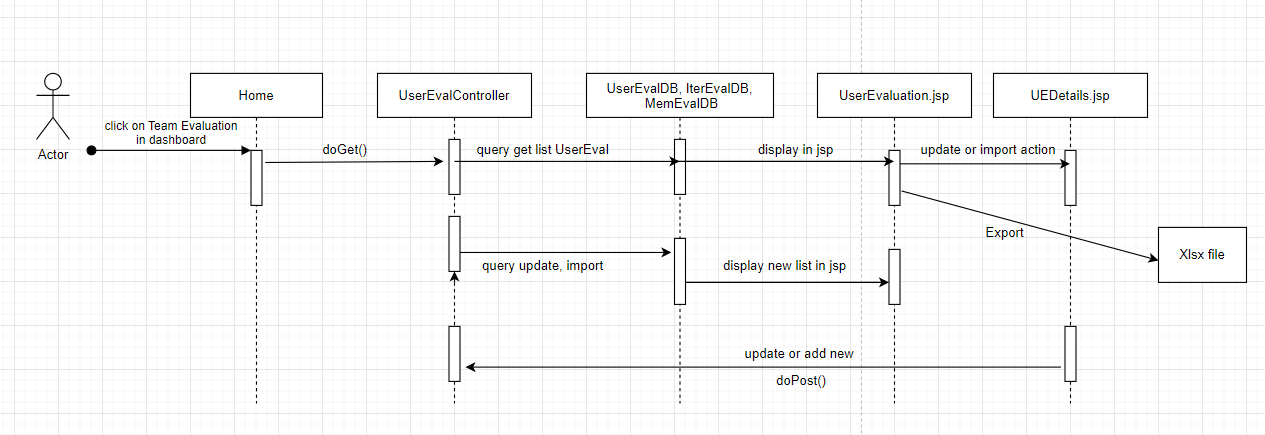
#### IterEvalClass

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | IterEval | *Input: none*  *Just a no-parameter constructor* |
| *02* | IterEval | *Input: int id, Iteration iteration, Class classObj, Team team, User user, double bonus, double grade, String note, boolean status*  *A full-parameter constructor* |
| *03* | *getter* | *Get value of attribute in class* |
| *04* | *setter* | *Set value for attribute in class* |

#### UserEval Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | UserEval | *Input: none*  *Just a no-parameter constructor* |
| *02* | UserEval | *Input: IterEval ie*  *Just one parameter constructor* |
| *03* | UserEval | *Input: IterEval ie, List<MemEval> me*  *A full-parameter constructor* |
| *04* | *getter* | *Get value of attribute in class* |
| *05* | *setter* | *Set value for attribute in class* |

### c. Sequence Diagram(s)



### d. Database queries

* *getIEOfTracking:*

*select m.iteration\_id, t.class\_id, t.team\_id, tr.assingee\_id from loc\_eval le, tracking tr, milestone m, team t*

*where le.tracking\_id = tr.tracking\_id and tr.milestone\_id = m.milestone\_id and tr.team\_id = t.team\_id and tr.tracking\_id = ?*

*group by m.iteration\_id, t.class\_id, t.team\_id, tr.assingee\_id*

* *getListIEByTE:*

*select ec.interation\_id, t.class\_id, t.team\_id, cu.user\_id from team\_eval te, eval\_criteria ec, team t, class\_user cu*

*where te.criteria\_id = ec.criteria\_id and te.team\_id = t.team\_id and t.team\_id = cu.team\_id*

*and te.team\_id = ? and te.criteria\_id = ?*

* *isIEExist:*

*select \* from iteration\_eval where iteration\_id = ? and class\_id = ? and team\_id = ? and user\_id=?*

* *add:*

*INSERT INTO `student\_project\_management`.`iteration\_eval`*

*(`iteration\_id`,`class\_id`,`team\_id`,`user\_id`,`bonus`,`grade`,`note`)*

*VALUES (?,?,?,?,?,?, ?)*

* *update:*

*UPDATE `iteration\_eval`*

*SET `iteration\_id` = ?,`class\_id` = ?,`team\_id` = ?,`user\_id` = ?,`bonus` = ?,`grade` = ?,`note` = ?,`status` = ? WHERE `eval\_id` = ?*

* *getIEById:*

*select \* from iteration\_eval where eval\_id = ?*

* *teamGradeOfIE:*

*select sum(distinct te.grade \* ec.evaluation\_weight/100)*

*as team\_eval\_grade from team\_eval te, eval\_criteria ec, iteration\_eval ie*

*where te.criteria\_id = ec.criteria\_id and ec.interation\_id = ie.iteration\_id and te.team\_id = ie.team\_id and ie.iteration\_id = ? and ie.team\_id = ?*

*group by ie.team\_id, ie.iteration\_id, ie.team\_id*

* *weightOfTeamGradeOfIE:*

*select sum(distinct evaluation\_weight)*

*as team\_weight from team\_eval te, eval\_criteria ec, iteration\_eval ie*

*where te.criteria\_id = ec.criteria\_id and ec.interation\_id = ie.iteration\_id and te.team\_id = ie.team\_id and ie.iteration\_id = ? and ie.team\_id = ?*

*group by ie.team\_id, ie.iteration\_id, ie.team\_id*

* *memGradeOfIE:*

*select (me.grade \* ec.evaluation\_weight/100) as mem\_grade from member\_eval me, eval\_criteria ec*

*where me.evaluation\_id = ? and me.criteria\_id = ec.criteria\_id*

* *getMemEvalOfLoc:*

*select \* from member\_eval me, eval\_criteria ec*

*where me.criteria\_id = ec.criteria\_id and ec.max\_loc != 0 and me.evaluation\_id = ?*

* *isMEExist:*

*select \* from member\_eval where evaluation\_id = ? and criteria\_id = ?*

* *getMEById:*

*select \* from member\_eval where member\_eval\_id = ?*

* *add:*

*INSERT INTO `student\_project\_management`.`member\_eval`*

*(`evaluation\_id`,`criteria\_id`,`converted\_loc`,`grade`,`note`)*

*VALUES (?,?,?,?,?)*

* *update:*

*UPDATE `student\_project\_management`.`member\_eval`*

*SET `evaluation\_id` = ?, `criteria\_id` = ?,`converted\_loc` = ?,`grade` = ?,`note` = ?,`status` = ?*

*WHERE `member\_eval\_id` = ?*

* *getGrade:*

*select \* from member\_eval where evaluation\_id = ? and criteria\_id = ?*

* *getMEOfIE:*

*select \* from member\_eval me, iteration\_eval ie, eval\_criteria ec*

*where me.evaluation\_id = ie.eval\_id and ie.iteration\_id = ec.interation\_id and ec.team\_evaluation = 0 and ec.status = 1 and ie.iteration\_id = ? and ie.team\_id = ?*

* *search:*

*select \* from*

*(select ie.eval\_id, row\_number() over*

*(order by ie.iteration\_id, ie.team\_id) as row\_index from iteration\_eval ie , class c, user u*

*where ie.class\_id = c.class\_id and ie.user\_id = u.user\_id*

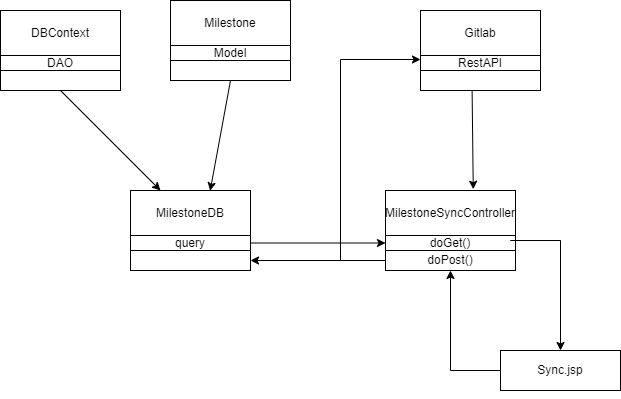
* *count:*

*select count(\*) from iteration\_eval ie , class c, user u*

*where ie.class\_id = c.class\_id and ie.user\_id = u.user\_id*

## 38. <Trainer/Milestone Sync>

### a. Class Diagram



### b. Class Specifications

**MilestoneSyncController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet () | Get a class’s milestones from SPM and gitlab |
| 2 | doPost() | Update SPM milestone and post milestone on gilab |

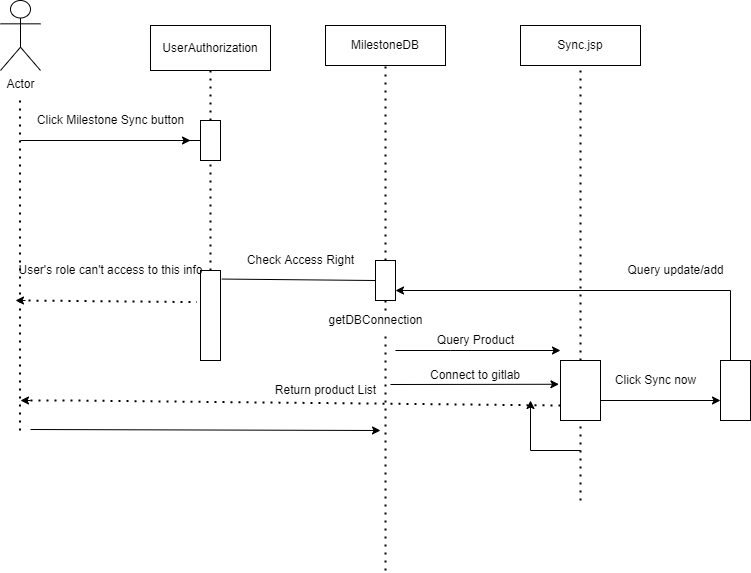
**MilestoneDB**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | List<Milestone> search | Input: user id and class.  Output: List of class’s milestone on SPM. |
| 02 | Void Add() | Input: information of a milestone.  Output: Milestone add to database. |
| 03 | Boolean CheckUniqueMilestoneName (String name, String iter, String Class) | Input: milestone name, milestone iteration and milestone class  Output: true if a milestone have input’s class and iteration and milestone name exist. False otherwise. |

**GitlabSync**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | List<Milestone>getMilestone  (String class\_code) | Input: class\_code  Output: List of class’s milestone on gitlab. |
| 02 | Void UploadMilestone(String class\_code, Milestone m) | Input:class\_code and a milestone infomation  Output: Milestone add to group on gitlab |
| 03 | String GetGroupId(String class\_code) | Input: a class’code  Output: id of group which have same name on gitlab |

### c. Sequence Diagram(s)



### Database queries

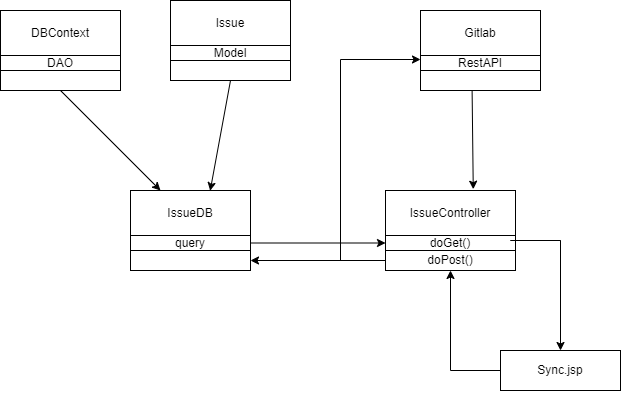
SELECT \* from (select m.milestone\_id,m.milestone\_name, m.iteration\_id, m.class\_id, m.from\_date, m.to\_date,m.status, row\_number() over (order by m.milestone\_id) as row\_index from milestone as m, iteration as i, class as c where m.iteration\_id = i.iteration\_id and m.class\_id = c.class\_id and i.status = 1 and c.status = 1 and c.trainer\_id=? and m.class\_id=? and (m.from\_date between ? and ?) and (m.to\_date between ? and ?) and m.status =? ) as t where row\_index >= (?-1)\*?+1 and row\_index <= ?\*?

SELECT \* FROM student\_project\_management.milestone where milestone\_name=? and class\_id=? and iteration\_id =?

INSERT INTO `student\_project\_management`.`milestone` (`milestone\_name`,`iteration\_id`,`class\_id`,`from\_date`,`to\_date`, status VALUES(?,?,?,?,?,?)

## 39. <Student/Issue Sync>

### a. Class Diagram



### b. Class Specifications

**MilestoneSyncController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet () | Get a class’s milestones from SPM and gitlab |
| 2 | doPost() | Update SPM milestone and post milestone on gilab |

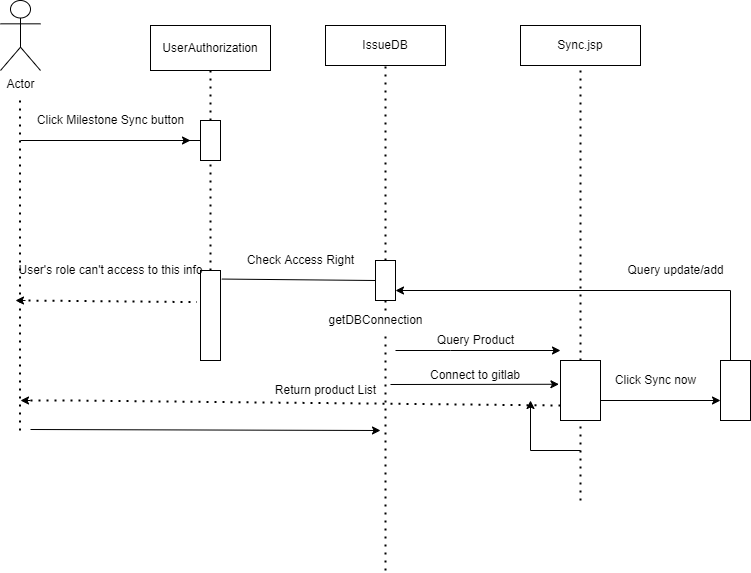
**MilestoneDB**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | List<Team> getTeamByUser(int id, Boolean isTrainer) | Input: user id and Boolean(true if user is trainer, false if student).  Output: List of class’s issue on SPM. |
| 02 | Void sync() | Input: information of a issue.  Output: Issue add to database. |
| 03 | List<Issue> getIssueByTeam(int id) | Input: team id  Output : list of Issue by team where gitlab id is null |

**GitlabSync**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Void getIssueByTeam() | Input: class\_code, team code, team id, user id  Output: add all issue found on gitlab to SPM, assignee= current user |
| 02 | Void UploadIssue() | Input:class\_code , team code and all issue infomation  Output: post issue to gitlab |
| 03 | String GetSubGroupId(String class\_code, String group name) | Input: a class’code and group name  Output: id of subgroup which have same name on gitlab |

### c. Sequence Diagram(s)



### Database queries

INSERT INTO `student\_project\_management`.`issue` (assignee\_id ,issue\_title, descript, gitlab\_id, gitlab\_url, created\_at, due\_date ,team\_id, status) VALUES (?, ?, ?, ? ,?, ?, ?, ?, ?)

select \* from issue where team\_id=? and gitlab\_id is null