

**student project management**

**Software Design Specification**

– Hanoi, August 2022 –

**Record of changeS**

| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- |
| 9/Aug | A | TungDS | Home + Dashboard |
| 9/Aug | A | SonDH | User Register |
| 9/Aug | A | CuongNM | User Login |
| 9/Aug | A | DatDX | Setting List |
| 9/Aug | A | TuanNQ | Setting Details |
| 13/Aug | A | TungDS | Add Iteration List + Iteration Details |
| 14/Aug | A | DatDX | Add code package + database schema (with out description) |
| 14/Aug | A | DatDXatDX | Add User List + User DetailsDetails |
| 14/Aug | M | TungDS | Modified Iteration Details |
| 14/Aug | M | TuanNQ | Subject List + Subject Details |
| 14/Aug | M | DatDX | Class Setting List + Class Setting Details |
| 16/Aug | M | TungDS | Update Iteration List |
| 16/Aug | A | TungDS | Add User authorization + Criteria List+ Criteria Details |
| 17/Aug | A | DatDX | Milestone List +MileStone Details |
| 18/Aug | A | DatDX | Milestone Submit + Student Evaluation |
| 20/Aug | A | TungDS | Add Tracking Update + Tracking Submit |
| 20/Aug | A | TuanNQ | Class List + Class Details |
| 25/Aug | M | DatDX | Modified Student Evaluation |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[**I. Overview**](#_gjdgxs) **9**

[1. Code Packages](#_30j0zll) 9

[2. Database Design](#_1fob9te) 10

[a. Database Schema](#_3znysh7) 10

[b. Table Description](#_2et92p0) 10

[**II. Code Designs**](#_tyjcwt) **11**

[1. Homepage](#_69o2w75exmc9) 11

[2. Dashboard](#_nrytp82ha1li) 11

[3. <Login>](#_3dy6vkm) 11

[a. Class Diagram](#_1t3h5sf) 11

[b. Table Description](#_4d34og8) 12

[c. Sequence Diagram(s)](#_2s8eyo1) 12

[d. Database queries](#_17dp8vu) 13

[4. User Register](#_3rdcrjn) 13

[a. Class Diagram](#_h4dwgnm56x8a) 13

[b. Table Description](#_3h8pc8oqoiqg) 14

[c. Sequence Diagram(s)](#_hao23wjssw0w) 15

[d. Database queries](#_veiwggfqvdyw) 15

[5. Setting Management](#_rbtoqeyab5d3) 15

[5.1. Show Setting List](#_8clhr1tjrk0d) 15

[5.1.a. Class Diagram](#_77326rngnmm) 15

[5.1.b. Table Description](#_oorfbyxkelw0) 15

[5.1.c. Sequence Diagram(s)](#_dg8h2gp3w3gm) 16

[5.1.d. Database queries](#_8qvblnyvse58) 16

[5.2. Show Setting Details](#_jndy43cpc9kd) 17

[5.2.a. Class Diagram](#_hrlgobuvjk4) 17

[5.2.b. Table Description](#_vjb6a2qaibfn) 17

[5.2.c. Sequence Diagram(s)](#_42om98vr120v) 17

[5.2.d. Database queries](#_1qgo4a337bm) 17

[6. Subject Management](#_wvfofbngak2v) 18

[6.1. Subject List](#_dmz72ard2ucz) 18

[6.1.a. Class Diagram](#_f4kxzjgr6qv1) 18

[6.1.b. Class Specification](#_3i10o9bxzkfb) 18

[6.1.c. Sequence Diagram](#_wvbtaui1jhu0) 18

[6.1.d. Database queries](#_tummbu46r7uu) 18

[6.2. Subject Details](#_ep6oxgbabelw) 18

[6.2.a. Class Diagram](#_y373t428wu7l) 18

[6.2.b. Class Specification](#_pjcihknu7djc) 18

[6.2.c. Sequence Diagram](#_ugrax9q7nfk9) 18

[6.2.d. Database queries](#_2cqwtudhkb68) 18

[7. Iterations Management](#_1gyabmj8o9bb) 19

[Class Diagram](#_ruaj9jcjx74j) 19

[Class Specification](#_qfq9t4305lkk) 19

[7.1. Iteration List](#_f67v3lm3i7vz) 23

[7.1.a. Sequence Diagram](#_7gu95u1vuft4) 23

[d7.1.b. Database queries](#_bji9zy5erqnu) 23

[7.2. Iteration Details](#_zb0pvgqawuu2) 24

[7.2.a Sequence Diagram](#_uxvojevmgexv) 24

[7.2.b. Database queries](#_atl164ffadtf) 24

[8. User Profile:](#_6mt1frw128zc) 25

[a. Class Diagram:](#_ps3gsq5wz5zo) 25

[b. Class Specification:](#_8uv45vrjadd4) 25

[c. Sequence Diagram:](#_crfg6445wpcd) 26

[d. Database Queries:](#_4cgou5jn1oo1) 26

[9. Change password:](#_e26megk5zlyq) 27

[a. Class Diagram:](#_w22lz9n0mr8x) 27

[b. Class Specification:](#_oiv60kw8z2i8) 27

[c. Sequence Diagram:](#_bd3du7exqa0x) 28

[d. Database Queries:](#_6h149evupjn) 28

[10. User Management](#_waasw5moyhzs) 29

[10.1. User List:](#_8t1hobewghsl) 29

[10.1.a Class Diagram](#_fnzaeabfp4kq) 29

[10.1.b Table Description](#_ts9wj84l83ci) 29

[10.1.c Sequence Diagram(s)](#_iy7wzar5n84i) 31

[10.1.d Database queries](#_aqv48r2qu2oe) 31

[10.2. User Detail:](#_e85q6np11sws) 32

[10.2.a Class Diagram](#_ar2i0pdvu6cg) 32

[10.2.b Table Description](#_2umr1v95gzv) 32

[10.2.c Sequence Diagram(s)](#_py43iqk4gg3h) 34

[10.2.d Database queries](#_zyz8jtdp3r2) 34

[11. Subject Management](#_mp9rq0xwj7w8) 35

[a. Subject Setting List](#_ghmbxf5pu2l) 35

[a. Class Diagram](#_hfdsl0v3enkm) 35

[b. Table Description](#_hfdsl0v3enkm) 35

[b. Subject Setting Details](#_3uf0f5ee0wr1) 38

[12. User Authorization](#_3lvt73zdyrmj) 41

[12.a. Class Diagram](#_r8c66yt20hx) 41

[12.b. Class Specification](#_cougbzhxsw2q) 41

[13. Criteria Management](#_gh6qvsguxn44) 41

[Class Diagram](#_7rtt54bwu1yp) 41

[Class Specification](#_yabwrfdit8jp) 41

[13.1. Criteria List](#_979z5metuaso) 46

[13.1.a. Sequence Diagram](#_karxppijuxe) 46

[13.1.b. Database Queries](#_wcug1pbd83b1) 46

[13.2 Criteria Details](#_qpwnj1n75agh) 48

[13.2.a. Sequence Diagram](#_jp4fzpfhp4gp) 48

[13.2.b. Database Queries](#_ccmlb7ktt11v) 48

[14. Class Management](#_xsvacw211opl) 49

[Class Diagram](#_b5wlgu4yuazl) 49

[Class Specification](#_uphkjzj8d03t) 49

[14.1 Class List](#_s2rqotknlohq) 53

[14.1.a. Sequence Diagram](#_1jbru5f5sngl) 53

[14.1.b. Database queries](#_mx88901xiegl) 53

[14.2 Class Details](#_z9qng5j2dl8h) 54

[14.2.a. Sequence Diagram](#_q3ic42126wdz) 54

[14.2.b. Database queries](#_fexba48odnil) 54

[15. Class Setting Management](#_9p18pe226as0) 54

[15.1 Class Setting List](#_chw2dqxqwelf) 54

[15.1.a Class diagram](#_jkic40gwqlu4) 54

[15.1.b Table Description](#_idi899eij816) 54

[15.1.c Sequence Diagram(s)](#_faooyex6w6ag) 56

[15.1.d Database queries](#_o7bf0ldoal23) 56

[15.2 Class Setting Details](#_g6htoie39m6q) 57

[15.2.a Class diagram](#_36upxvfg2vdg) 57

[15.2.b Table Description](#_evtdx0lpg6jb) 57

[15.2.c Sequence Diagram(s)](#_jxyy8uktnojw) 59

[15.2.d Database queries](#_j8xsl7gwmje) 59

[16. Milestone Management](#_9pctustlipkx) 60

[16.1. Milestone List](#_9fczj855fkbm) 60

[16.1.a. Class Diagram](#_rtqpb1brm77l) 60

[16.1.b. Table Description](#_k2yn1bp32pma) 60

[16.1.c. Sequence Diagram(s)](#_fnmgpjoomxk7) 62

[16.1.d. Database queries](#_pjqhgxzawxxx) 62

[16.2.b. Milestone Details](#_9kc5f6yosaz6) 63

[16.2.a. Class Diagram](#_h42oz97gfe3q) 63

[16.2.b. Table Description](#_x8u0h1wsli7n) 63

[16.2.c. Sequence Diagram(s)](#_d7anivg17d49) 65

[16.2.d. Database queries](#_aafyz56i9ayy) 65

[16.3. Milestone Submits](#_26sh5xad3yju) 66

[16.3.a Class Diagram](#_ap6i8qlypsog) 66

[16.3.b Table Description](#_2r4s7pkojtbe) 66

[16.3.c Sequence Diagram](#_yzvsu8t6swol) 68

[16.3.d Database queries](#_s2g3x1holcev) 68

[17. Student Evaluation](#_ozy3cqzfjch8) 69

[17.a Class Diagram](#_jetile2sfc6p) 69

[17.b Table Description](#_1c331wigra89) 69

[17.c Sequence Diagram](#_327nlkxc3j3o) 70

[17.d Database queries](#_frjb5hm0wwc) 70

[18. Tracking Update](#_biz6qj2p8l33) 71

[Class Diagram](#_nbdplaefo6ty) 71

[Table Description](#_g7ibluqb0fxr) 71

[Sequence Diagram](#_h9yji4j0ppp8) 75

[Database Queries](#_nktlsouetzmx) 77

[19. Tracking Submit](#_nzdvi7osfxas) 77

[Class Diagram](#_y7f16n6ai6oi) 77

[Table Description](#_ewi859gb5ccp) 77

[Sequence Diagram](#_sukw1eurmzcr) 77

[Database Queries](#_rsv0fv2piq57) 77

[20. Team List](#_gih7dcgfj409) 77

[Class Diagram](#_rzxbfkl2ftue) 77

[Table Description](#_l18ua68e5lmi) 77

[Sequence Diagram](#_kk9b5khpi5r) 78

[Database Queries](#_idkgicudnjz3) 78

[21. Team Detail](#_b2f4jt5k7pow) 79

[Class Diagram](#_7k9uc5k4ha6l) 79

[Table Description](#_1mvvhn9ylrf4) 79

[Sequence Diagram](#_p82ef2vxycp3) 80

[Database Queries](#_rirdhu5opd5m) 80

[22. Issue Management](#_5u521ubxke6) 80

[Class Diagram](#_xu72jxn6gpts) 80

[Class Specification](#_hncsha9junfp) 80

[22.1 Class List](#_unbjmfg1myyx) 80

[22.1.a. Sequence Diagram](#_doagfpl4o188) 80

[22.1.b. Database queries](#_cprxhzu076su) 80

[22.2 Class List](#_o8j9eekqr6eu) 80

[22.2.a. Sequence Diagram](#_1r643zwfv1np) 80

[22.2.b. Database queries](#_7n43oqaj7n47) 80

[23. Iteration Evaluation](#_z9afadn42s3y) 80

[24. Web Contact](#_p17f4zdlruwi) 80

[Class Diagram](#_buqks9v6uxeq) 80

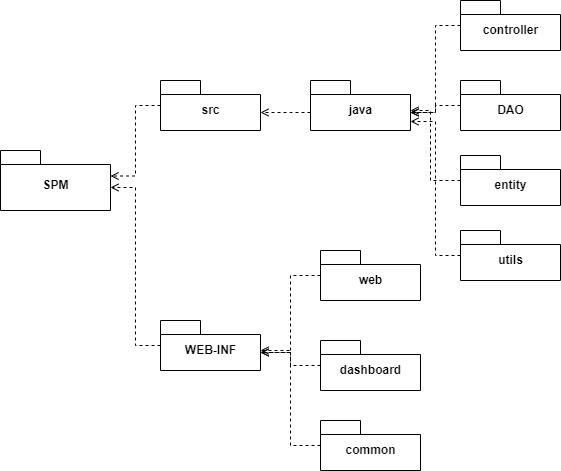
[Class Specification](#_5wpslhqz70du) 81

[Sequence Diagram](#_eduxi5lct07r) 82

[Database Queries](#_p8mr1t1ynia2) 82

# I. Overview

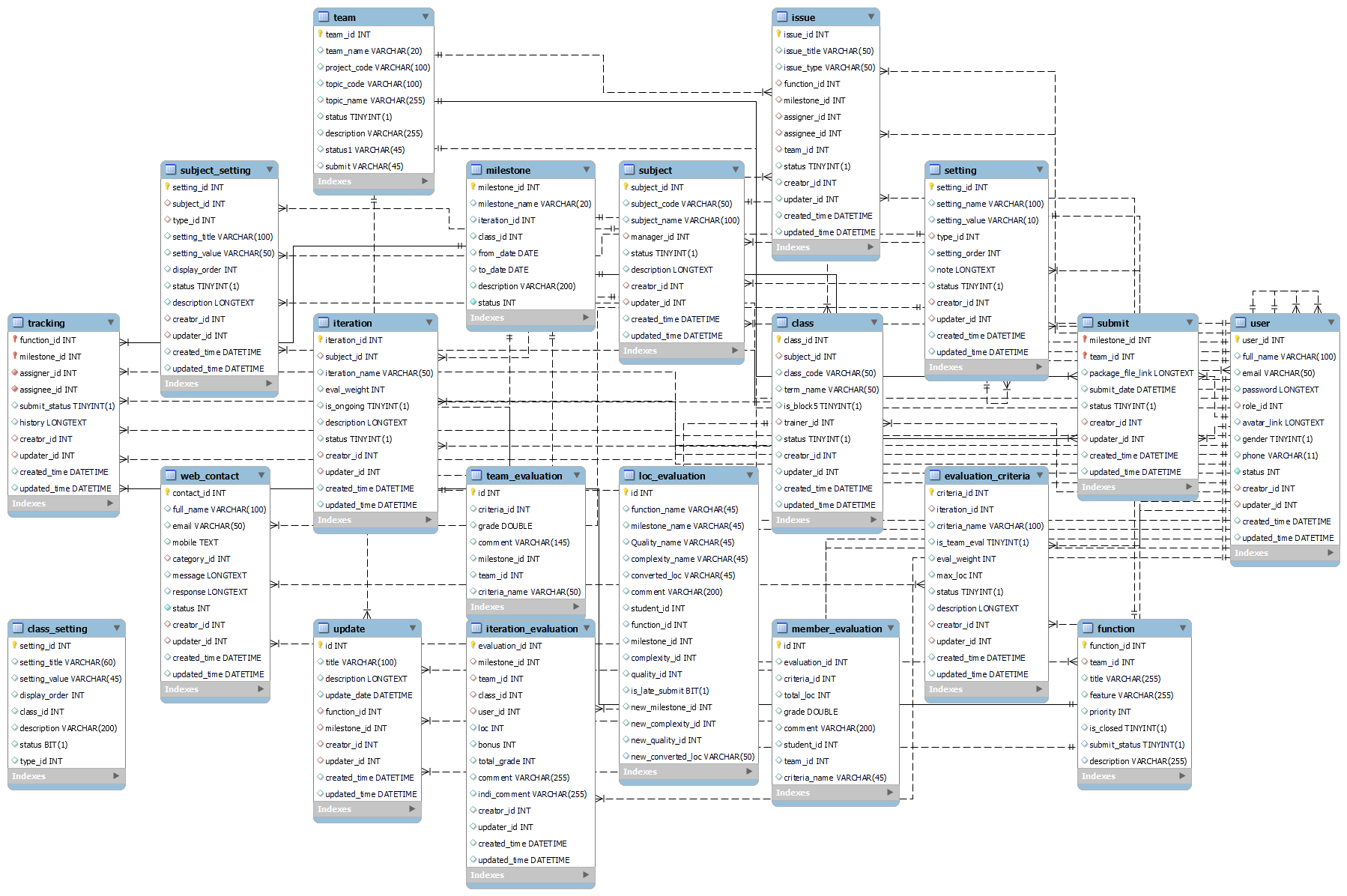
## 1. Code Packages



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

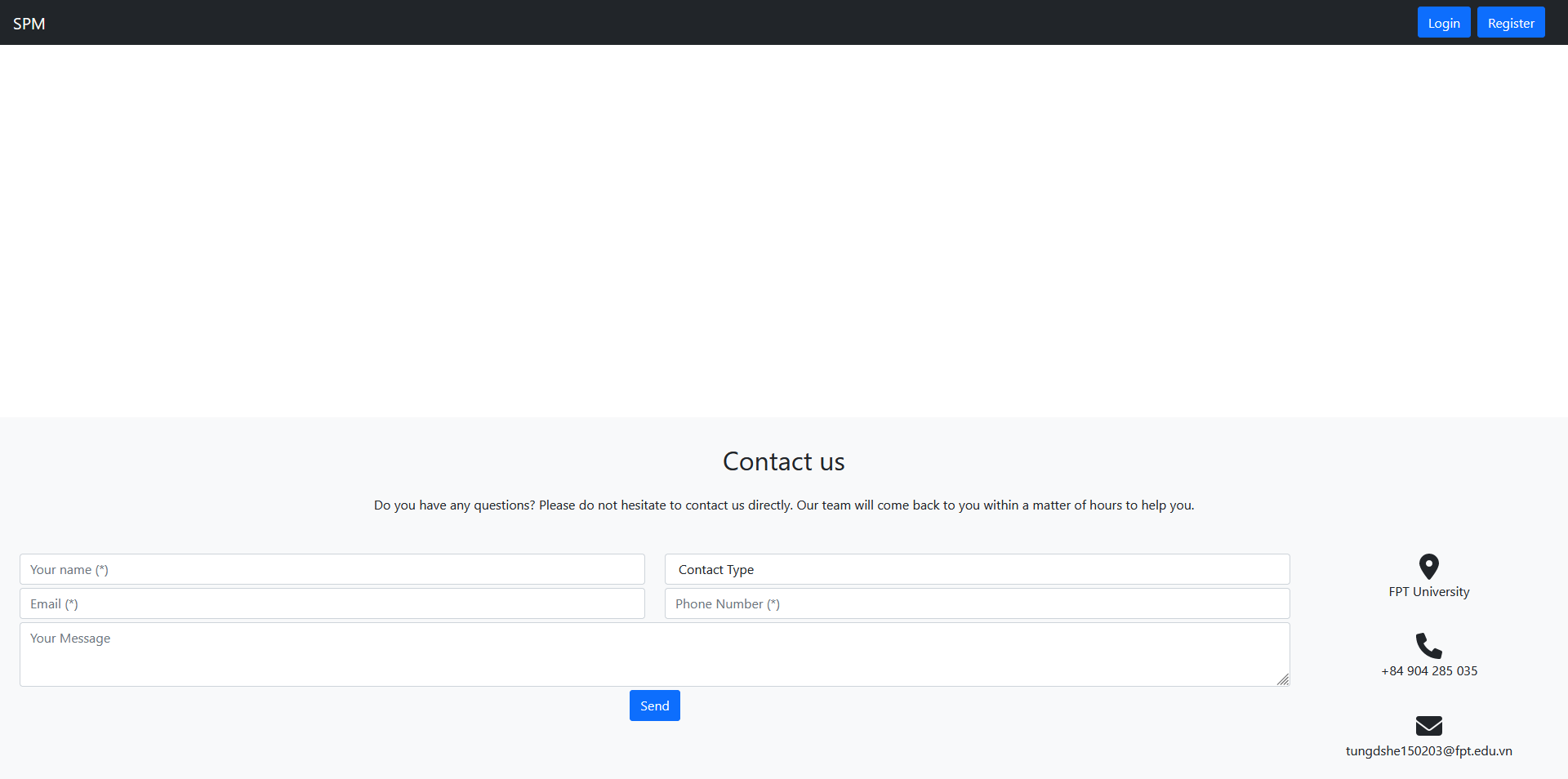
| **No** | **Package** | **Description** |
| --- | --- | --- |
| *01* | *dao* | *Used to contain DAO files.* |
| *02* | *controller* | *Used to contain controller files.* |
| *03* | *entity* | *Used to contain entity/model files.* |
| *04* | *utils* | *Used to contain utilities files.* |

## 2. Database Design

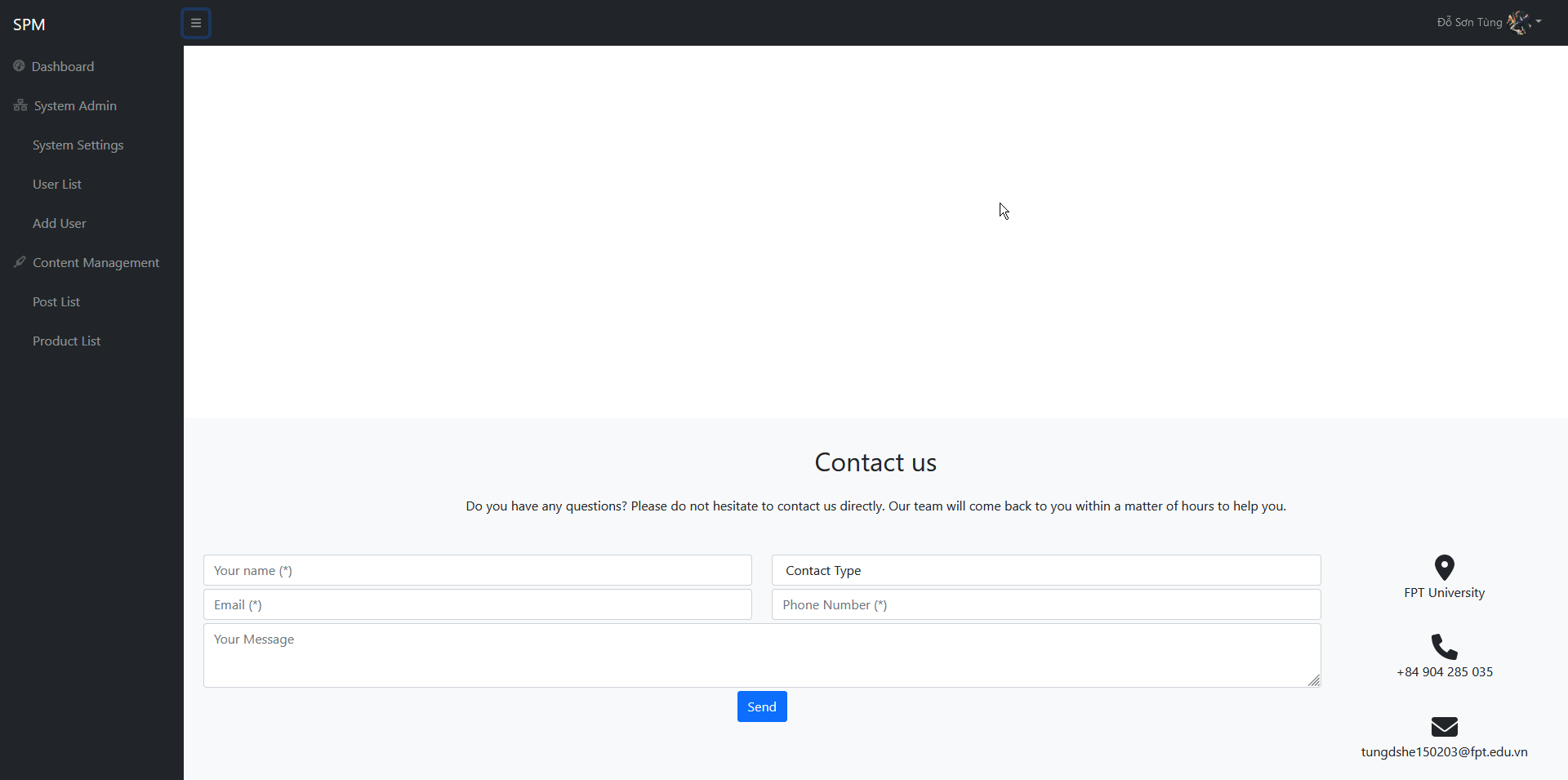


# II. Code Designs

## 1. Homepage



## 2. Dashboard

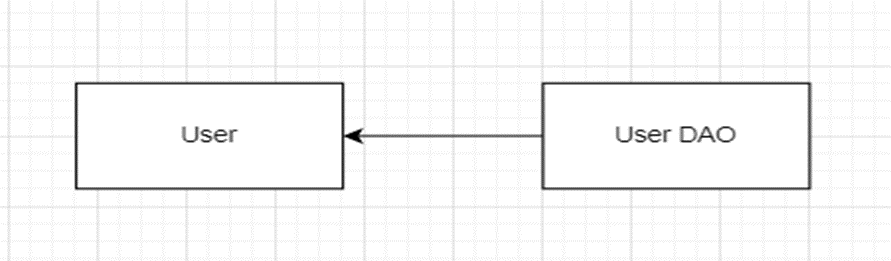


## 3. <Login>

*[Provide the detailed design for the function <Feature/Function Name1>. It include Class Diagram, Class Specifications, and Sequence Diagram(s)]*

### a. Class Diagram

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



### b. Table Description

*[Provide the description for each class and the methods in each class, following the table format as below]*

***LoginController***

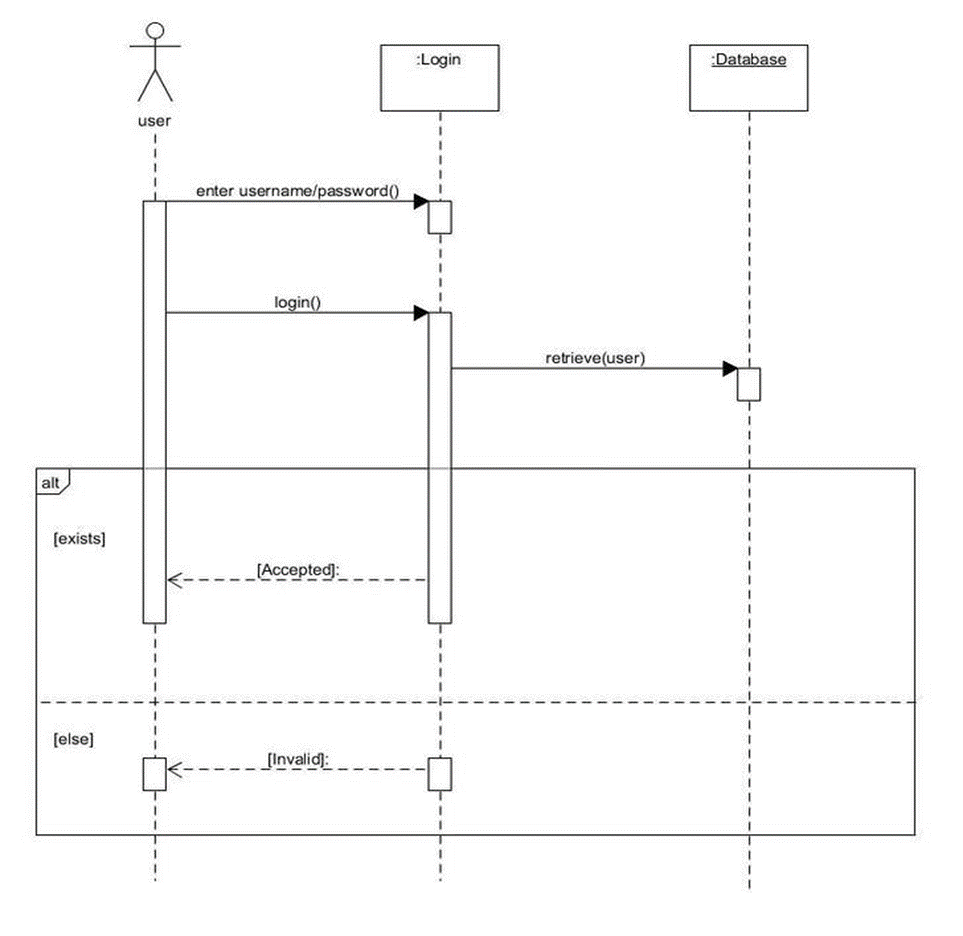
| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGET* | *Redirect request to page login.jsp* |
| *02* | *doPOST* | *Check if email and password are correct* |

***UserDao***

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGET* | *Redirect request to page login.jsp* |
| *02* | *doPOST* | *Check if email and password are correct* |

### c. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*

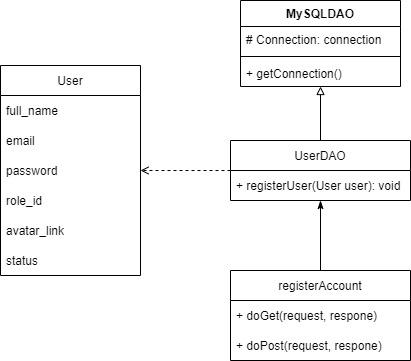
**

### d. Database queries

*[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]*

## 4. User Register

### a. Class Diagram



### b. Table Description

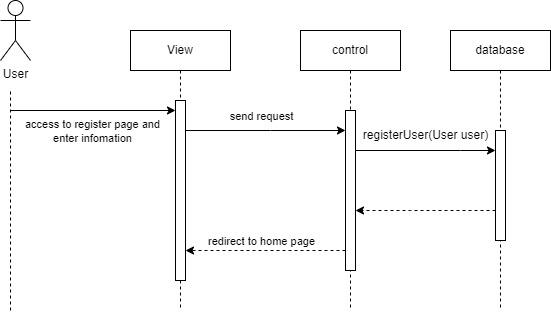
*registerAccount*

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGET* | *Load register view* |
| *02* | *doPOST* | *Get request from view and handle data to add user in database* |

UserDAO

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *registerUser(User user)* | *add user into database* |

### c. Sequence Diagram(s)



### d. Database queries

INSERT INTO user (full\_name, email, password, role\_id, status, created\_time) VALUES(?, ?, ?, ? ,?, ?)

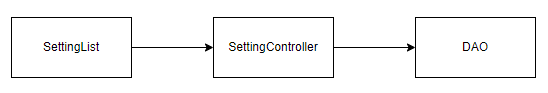
## 

## 

## 5. Setting Management

### 5.1. Show Setting List

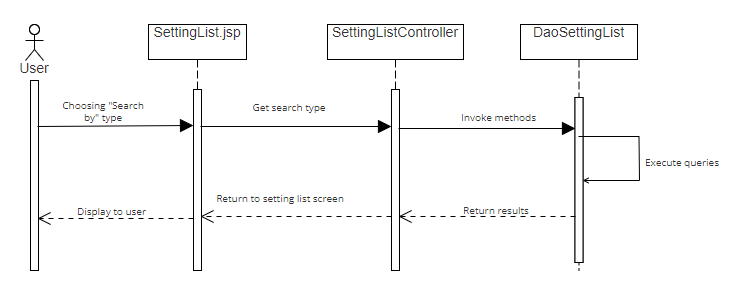
#### 5.1.a. Class Diagram

******

#### 5.1.b. Table Description

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGET* | *Get setting data based on id* |
| *02* | *doPOST* | *Check submit is success or fail* |

#### 5.1.c. Sequence Diagram(s)



#### 5.1.d. Database queries

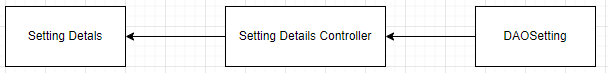
"SELECT \* FROM spm.setting ";

“UPDATE spm.setting SET spm.setting.status = ? where setting\_id =?”;

## 

### 5.2. Show Setting Details

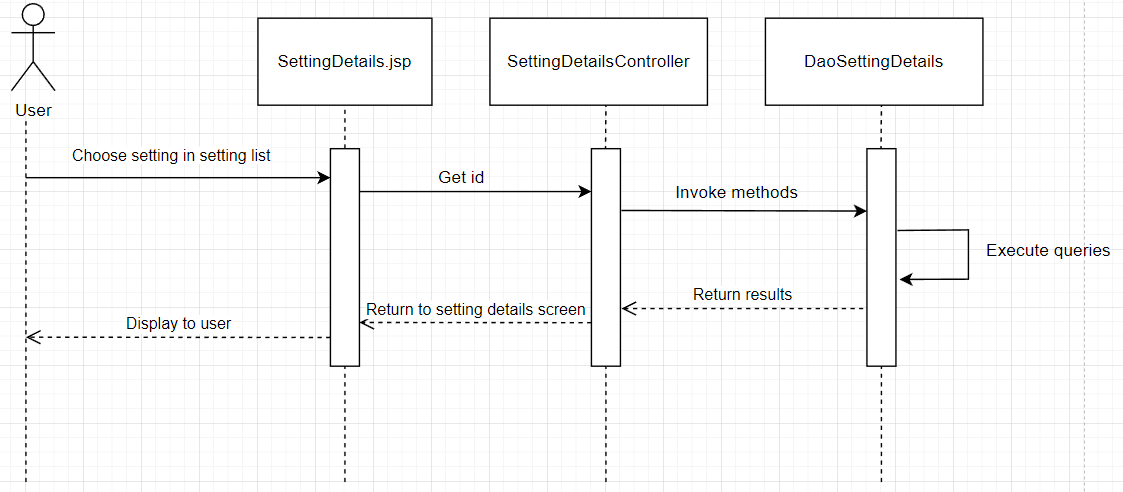
#### 5.2.a. Class Diagram



#### 5.2.b. Table Description

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGET* | *Get setting data based on id* |
| *02* | *doPOST* | *Check submit is success or fail* |

#### 5.2.c. Sequence Diagram(s)

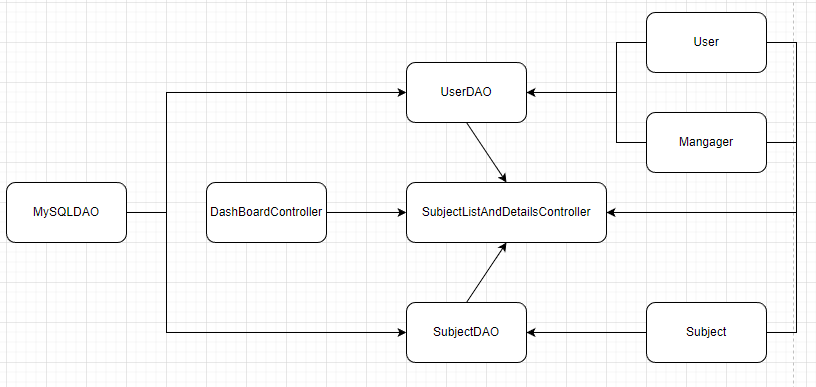


#### 5.2.d. Database queries

SELECT \* FROM setting where setting\_id = ?;

## 6. Subject Management

#### Class Diagram



#### Class Specification

MySQLDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getConnection* | *Connect to Database* |

SubjectDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getSubjects* | *Query all subjects from subject table with given query and return list of subjects corresponding to query* |
| *02* | *updateSubjectStatus* | *Input the subject id, the id of the updater as well as the status, then get current time. After that, update the status, updater id and update time of the subject based on the input value.* |
| *03* | *getSubjectDetails* | *Input the id of the subject , then query from the database with that subject id. After that, return an subject object with queried values.* |
| *04* | *updateSubjectDetails* | *Input an subject object and user id, then get current time. After that, update the details, updater id and update time of the subject based on the values of the input subject object and the 2 remaining values.* |
| *05* | *addNewSubject* | *Input an subject object and user id, then get current time. After that, insert into the database with the inputted data.* |

UserDAO class

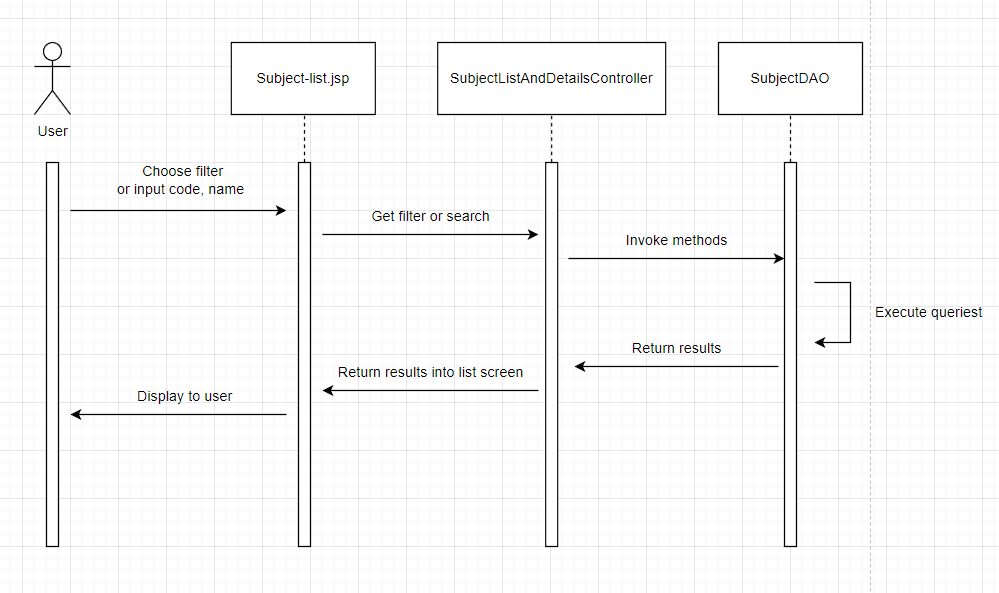
| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getManagers* | *Query information of users with role is manager* |

SubjectListAndDetailsController class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGet* | *Check current path value, then call functions based on that value.* |
| *02* | *doPost* | *Check current path value, then call functions based on that value.* |
| *03* | *SubjectList* | *Get filter value, pagination page value, and query then paginate data from the database based on those values, after that, display the list of subjects to subject -list.jsp.* |
| *04* | *updateSubjectStatus* | *Get current user id and status value, then call updateSubjectStatus. After that redirect the user to the subject list.* |
| *05* | *subjectDetails* | *Get the requested subject id and query the details from the database with that id. Then populated the subject -details.jsp with the queried information.* |
| *06* | *updateSubjectDetails* | *Get all information sent by subject details form, then update the database with the received information.* |

### 6.1. Subject List

#### 6.1.a. Sequence Diagram



#### 6.1.b. Database queries

SELECT s.subject\_id, s.subject\_code, s.subject\_name, (select u.full\_name from spm.user u where u.user\_id = s.manager\_id) AS manager\_name, s.status FROM spm.subject s

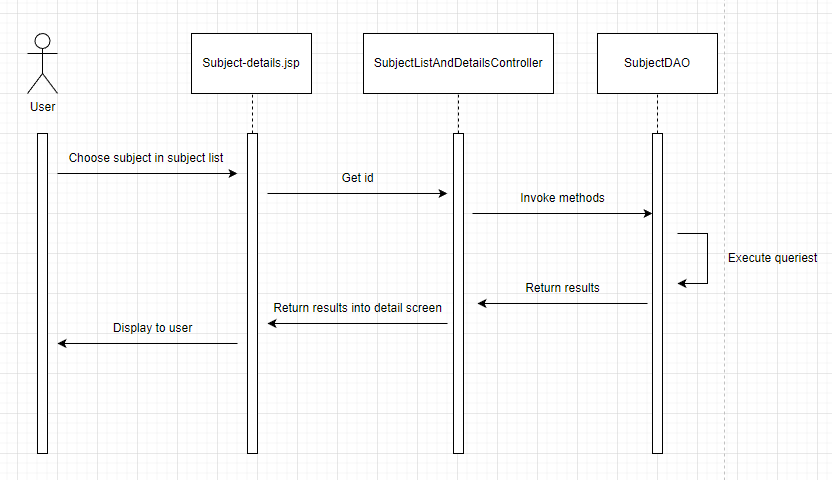
UPDATE spm.subject SET status = ?, updater\_id = ?, updated\_time = ? WHERE subject\_id = ?

INSERT INTO spm.subject (subject\_id, subject\_code, subject\_name, manager\_id, status, description, creator\_id, created\_time) VALUES (?,?,?,?,?,?,?,?)

SELECT user\_id, full\_name FROM spm.user where role\_id = 3

### 6.2. Subject Details

#### 6.2.a. Sequence Diagram



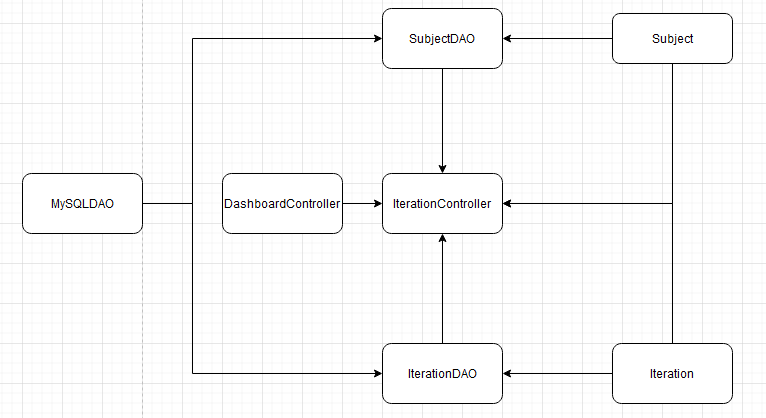
#### 6.2.b. Database queries

SELECT \* FROM spm.subject WHERE subject\_id =

UPDATE spm.subject SET subject\_id = ?, subject\_code = ?, subject\_name = ?, manager\_id = ?, status = ?, description = ?, updater\_id = ?, updated\_time = ? WHERE subject\_id = ?

## 7. Iterations Management

#### Class Diagram



#### Class Specification

MySQLDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getConnection* | *Connect to Database* |

IterationDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getTotalIterationsSubjectId* | *Input the subject id, then count the number of iterations in iteration table with that subject id, then return the value.* |
| *02* | *getIterationsSubjectId* | *Input the current page value, the number of entries per page, and the subject id, then query an iteration list from database based on those 3 values, then return the list.* |
| *03* | *getTotalIterationsSubjectIdAndName* | *Input the iteration name and subject id, then count the number of iterations in iteration table with that subject id and that name pattern, then return the value.* |
| *04* | *getIterationsSubjectIdAndName* | *Input the current page value, the number of entries per page, the subject id, and the iteration name, then query an iteration list from database based on those 4values, then return the list.* |
| *05* | *getTotalIterationsSubjectIdAndStatus* | *Input the iteration status and the subject id, then count the number of iterations in iteration table with that status and subject id, then return the value.* |
| *06* | *getIterationsSubjectIdAndStatus* | *Input the current page value, the number of entries per page, the subject Id, and the iteration status, then query an iteration list from database based on those 4 values, then return the list.* |
| *07* | *getIterationsSubjectIdAndOrder* | *Input the current page value, the number of entries per page, the subject id, and the order for the list, then query an iteration list from database based on those 4 values, then return the list.* |
| *08* | *updateIterationStatus* | *Input the iteration id, the id of the updater as well as the status, then get current time. After that, update the status, updater id and update time of the iteration based on the input value.* |
| *09* | *getIterationDetails* | *Input the id of the iteration, then query from the database with that iteration id. After that, return an Iteration object with queried values.* |
| *10* | *updateIterationDetails* | *Input an Iteration object and user id, then get current time. After that, update the details, updater id and update time of the iteration based on the values of the input iteration object and the 2 remaining values.* |
| *11* | *addNewIteration* | *Input an Iteration object and user id, then get current time. After that, insert into the database with the inputted data.* |

SubjectDAO class

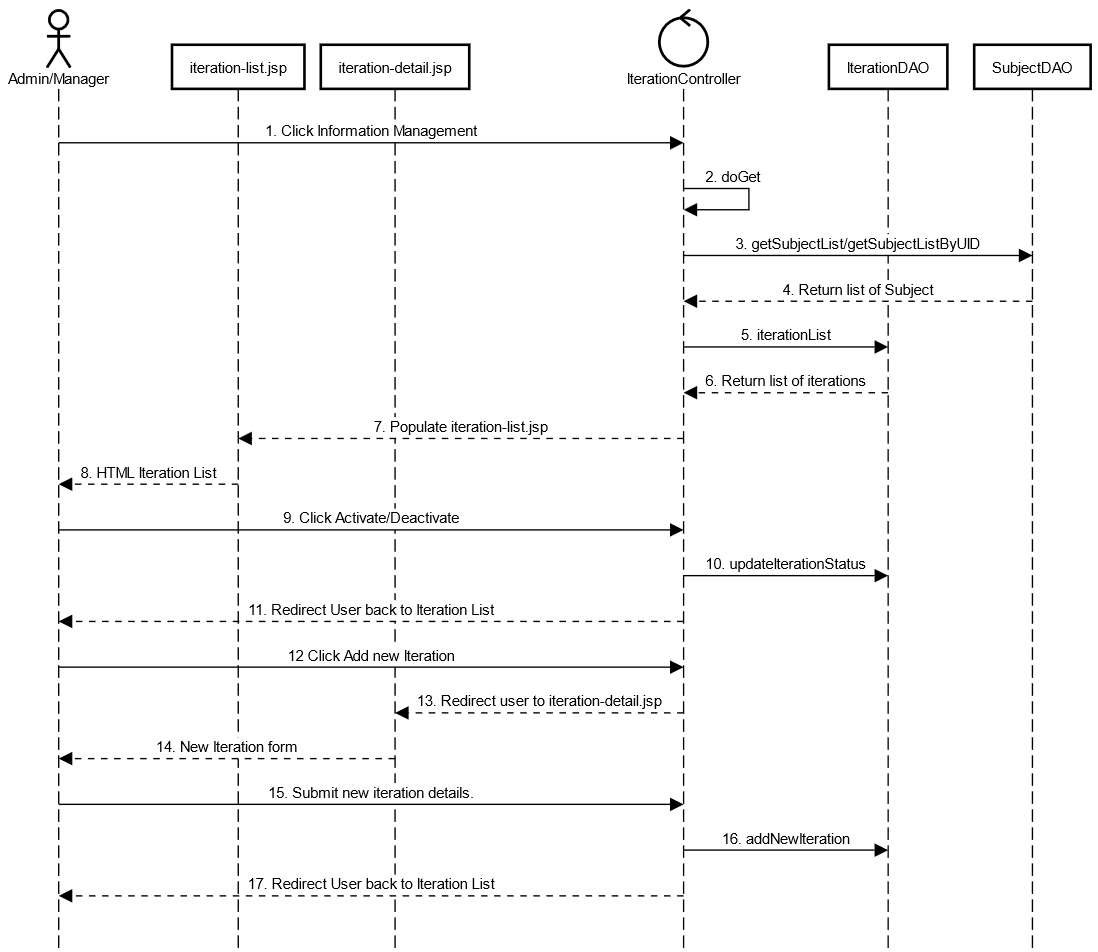
| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getSubjectList* | *Query all subject from subject table in database and return a list of subject.* |
| *02* | *getSubjectListByUID* | *Query all subject from subject table with the input user id and return a list of subject.* |

IterationController class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGet* | *Check current path value, then call functions based on that value.* |
| *02* | *doPost* | *Check current path value, then call functions based on that value.* |
| *03* | *iterationList* | *Get filter value, pagination page value, and query then paginate data from the database based on those values, after that, display the list of iterations to iteration-list.jsp.* |
| *04* | *updateIterationStatus* | *Get current user id and status value, then call updateIterationStatus. After that redirect the user to iteration list.* |
| *05* | *iterationDetails* | *Get the requested iteration id and query the details from database with that id. Then populated the iteration-detail.jsp with the queried information.* |
| *06* | *updateIterationDetails* | *Get all information sent by iteration details form, then update the database with the received information.* |
| *07* | *toAddIteration* | *Send user to add new iteration screen.* |
| *08* | *addNewIteration* | *Get all information sent by new iteration form, then insert into the database with the received information.* |

### 7.1. Iteration List

#### 7.1.a. Sequence Diagram



#### d7.1.b. Database queries

SELECT COUNT(\*) FROM iteration WHERE subject\_id = ?;

SELECT \* FROM iteration WHERE subject\_id = ? LIMIT ? OFFSET ?

SELECT \* FROM iteration WHERE subject\_id = ? AND status = ? LIMIT ? OFFSET ?

SELECT COUNT(\*) FROM iteration WHERE subject\_id = ? AND iteration\_name LIKE ?

SELECT \* FROM iteration WHERE subject\_id = ? AND iteration\_name LIKE ? LIMIT ? OFFSET ?

SELECT \* FROM iteration WHERE subject\_id = ? ORDER BY eval\_weight ASC LIMIT ? OFFSET ?

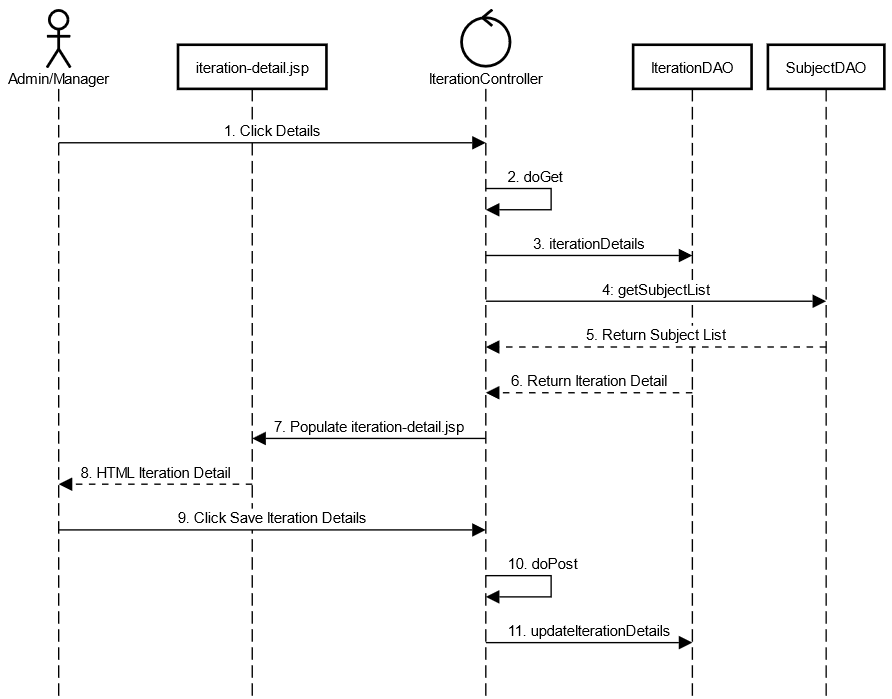
SELECT \* FROM iteration WHERE subject\_id = ? ORDER BY eval\_weight DESC LIMIT ? OFFSET ?

UPDATE iteration SET status = ?, updater\_id = ?, updated\_time = ? WHERE iteration\_id = ?

INSERT INTO iteration (subject\_id, iteration\_name, eval\_weight, is\_ongoing, status, description, creator\_id, created\_time) VALUES (?,?,?,?,?,?,?,?)

### 7.2. Iteration Details

#### 7.2.a Sequence Diagram



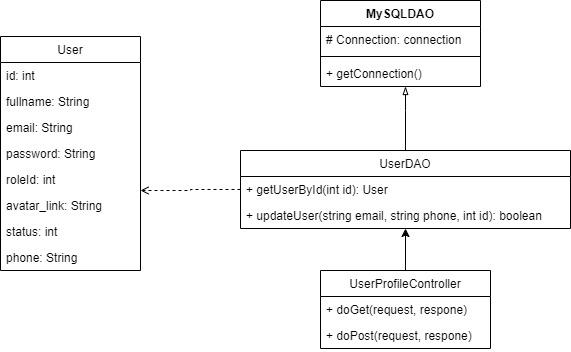
#### 7.2.b. Database queries

SELECT \* FROM iteration WHERE iteration\_id = ?

UPDATE iteration SET subject\_id = ?, iteration\_name = ? ,eval\_weight = ?, is\_ongoing = ?, status = ?, description = ?, updater\_id = ?, updated\_time = ? WHERE iteration\_id = ?

## 8. User Profile:

### a. Class Diagram:



### b. Class Specification:

MySQLDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection | Connect to Database |

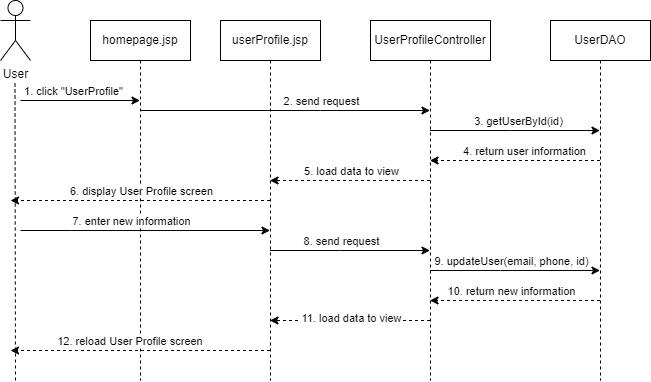
UserDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getUserById(int id) | get user from database following user id |
| 02 | updateUser(String email, String phone, int id) | update information of user to database following user id |

UserProfileController class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(request, response) | load data of the user from database to User Profile screen. |
| 02 | doPost(request, response) | process the data that is transmitted down when the user updates the information and put it in the database |

### c. Sequence Diagram:



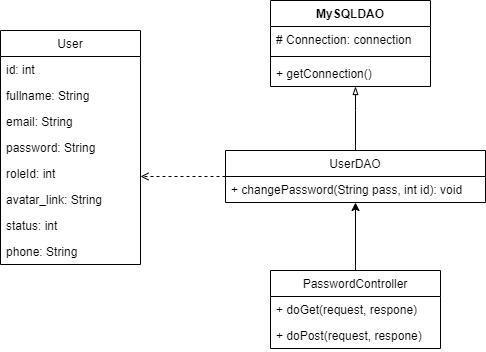
### d. Database Queries:

SELECT \* FROM user WHERE user\_id = ?

Update user set email = ?, phone = ? where user\_id = ?

## 9. Change password:

### a. Class Diagram:



### b. Class Specification:

MySQLDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection | Connect to Database |

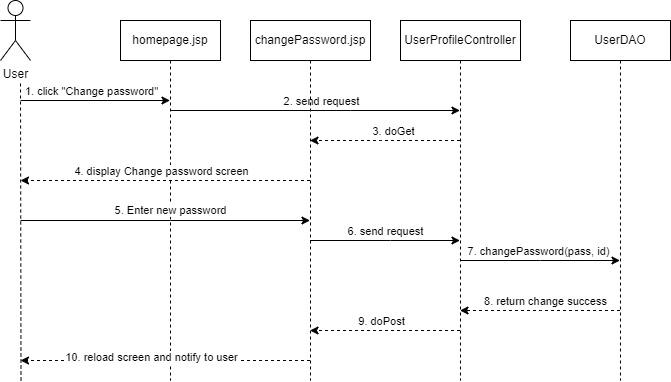
UserDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | changePassword(String pass,int id) | Update new password to database following user id |

PasswordController class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(request, response) | redirect to Change Password screen. |
| 02 | doPost(request, response) | Update new password in database following user id |

### c. Sequence Diagram:



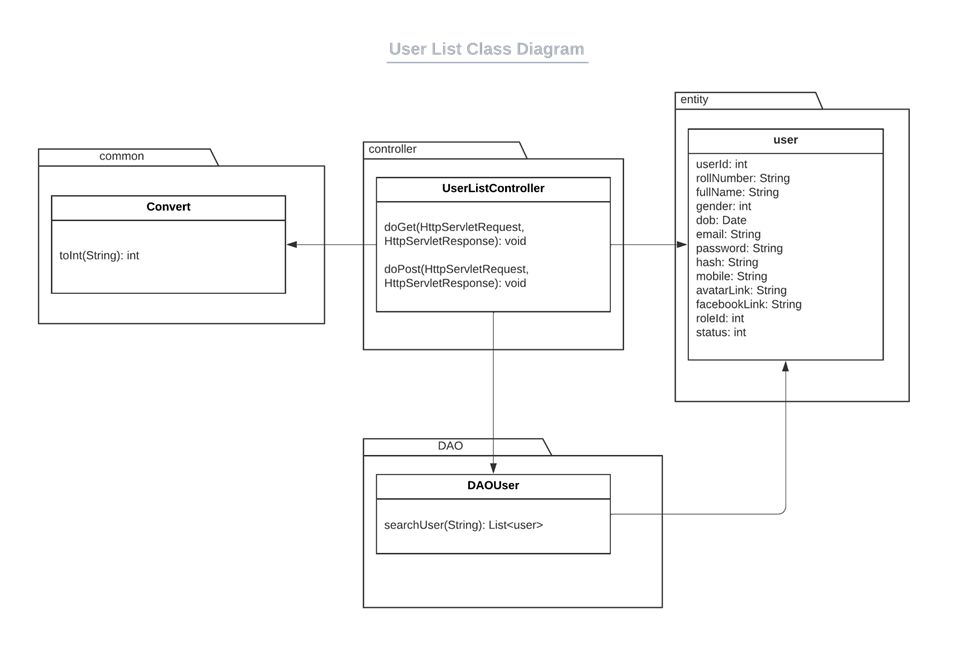
### d. Database Queries:

update user set password = ? where user\_id = ?

## 10. User Management

### 10.1. User List:

#### 10.1.a Class Diagram



#### 10**.1.b** Table Description

***UserListController***

*[Displays user list to view]*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Return User list to View* |
| *02* | *doPost* | *Receive keyword, process and return searched results to view* |

***Convert***

*[Convert data types]*

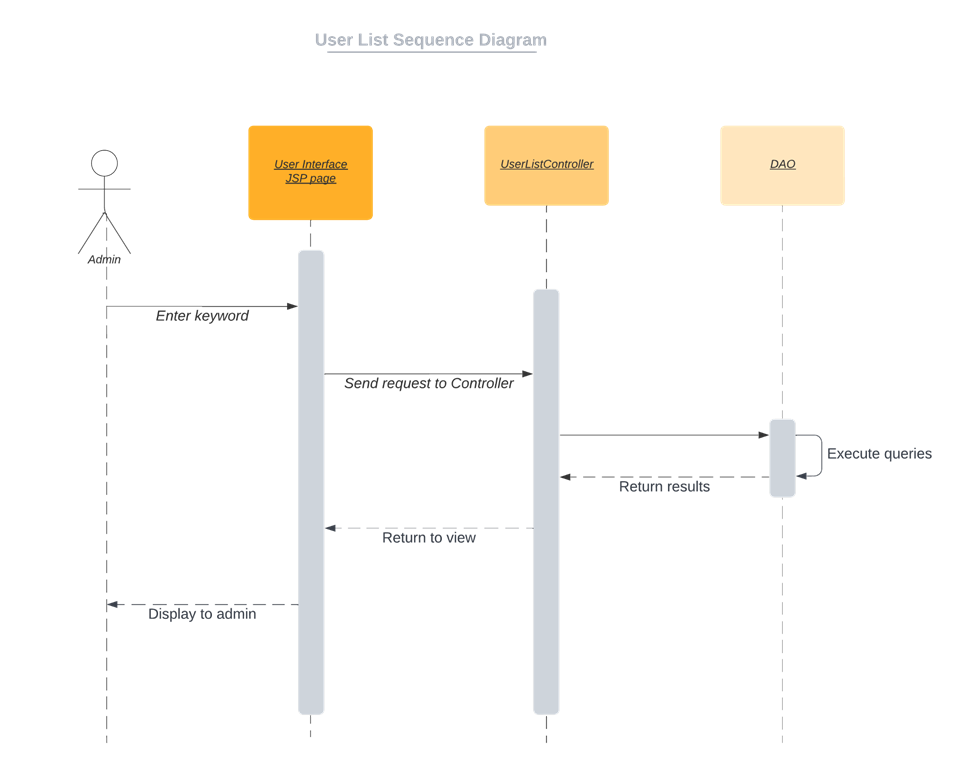
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *toInt* | *Convert input String to int number* |
|  |  |  |

***DAOUser***

*[Connect and take action with database]*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *searchUser* | *Find users whom name contains keywords in database and return a list* |
|  |  |  |

#### 10.1.c Sequence Diagram(s)



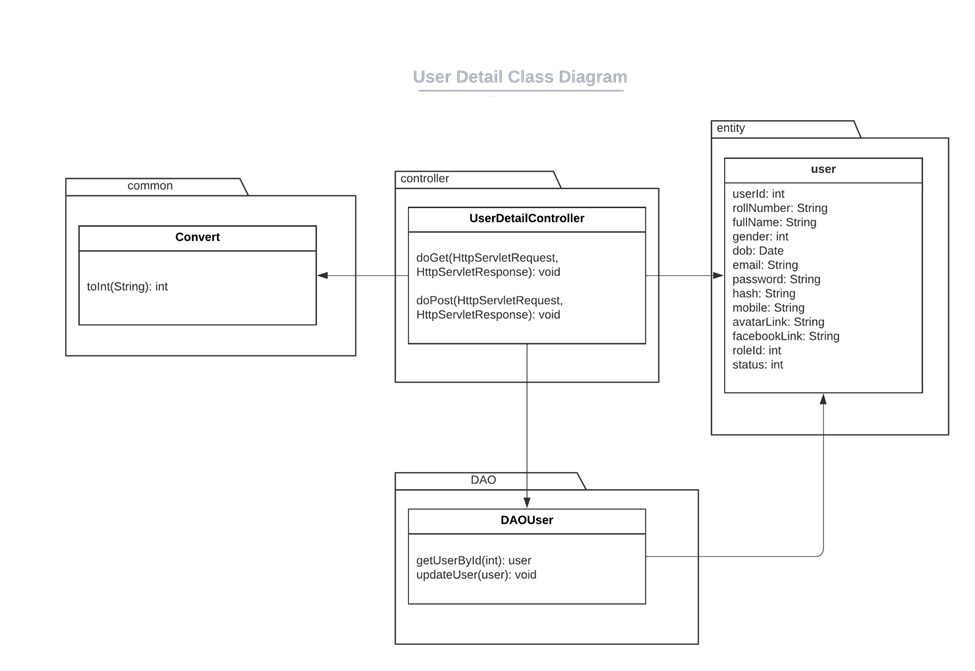
#### 10.1.d Database queries

select \* from user where full\_name like %?%

order by full\_name ascending

### 10.2. User Detail:

#### 10.2.a Class Diagram



#### 10.2.b Table Description

***UserDetailController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Get data from database and Push to view* |
| *02* | *doPost* | *Update entered data from fields to database* |

***UserDAO***

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| ***01*** | ***getUsers*** | ***Get a specific ClassUser from database by specific IDs*** |
| ***02*** | ***updateClassUser*** | ***Update the Class User informations to database*** |
| ***03*** | ***changeStatus*** |  |
| ***04*** | ***updateUser*** |  |

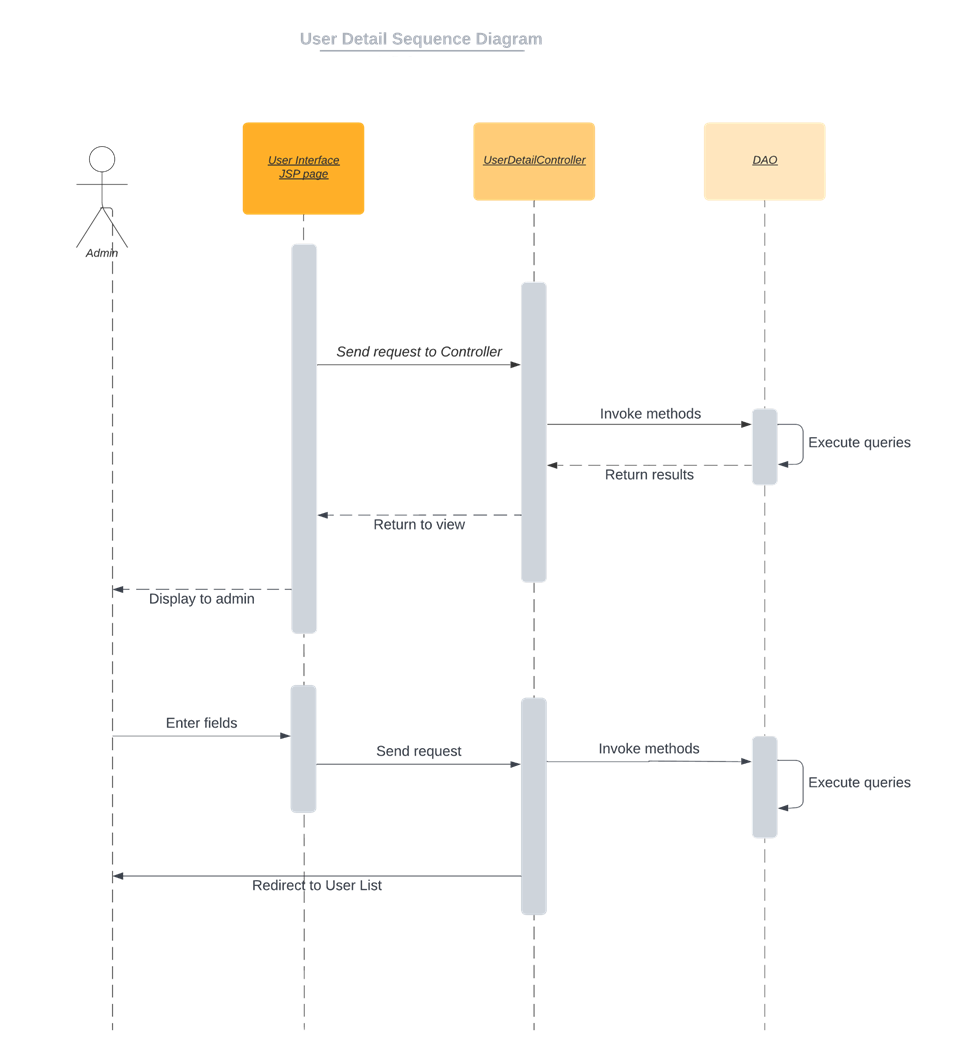
***DaoClass***

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| ***01*** | ***searchClassByCondition*** | ***Get the list of Class that match the conditions*** |

***DaoUser***

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| ***01*** | ***getUserById*** | ***Get a specific User by User ID in the database*** |

#### 10.2.c Sequence Diagram(s)



#### 10.2.d Database queries

SELECT \* FROM user WHERE email = ?

UPDATE `spm`.`user` SET `status` = ? WHERE (`user\_id` = ?);

UPDATE `spm`.`user` SET `full\_name` = ?, `email` = ?, `password` = ?,"

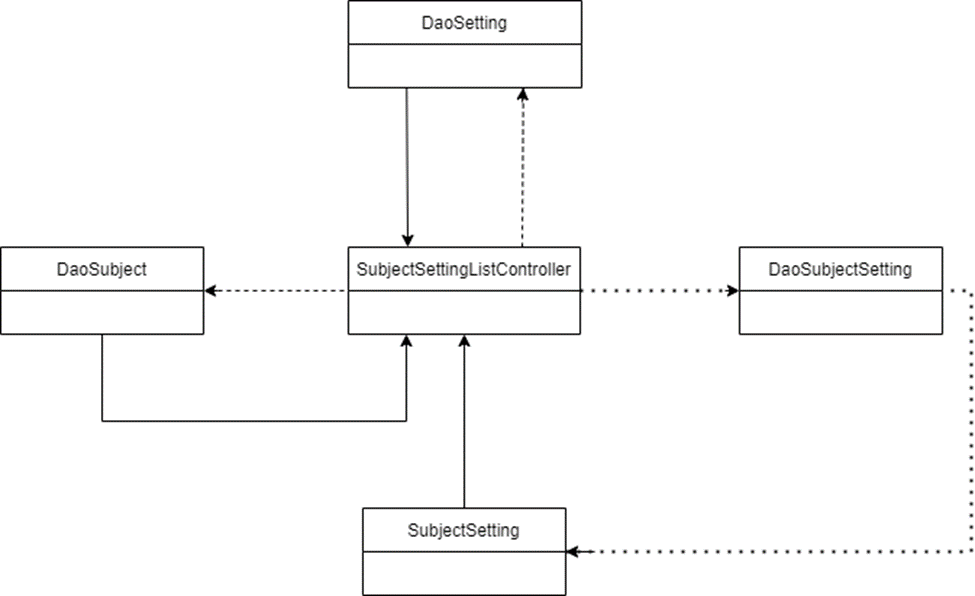
+ " `role\_id`=? ,`phone` = ?, `gender`=? WHERE (`user\_id` = ?);

### 

## 11. Subject Management

### a. Subject Setting List

#### a. Class Diagram



#### b. Table Description

**SubjectSettingListController**

## 

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest | Get data, update data, Delete data to database |

## 

DaoSubjectSetting

## 

| No | Method | Description |
| --- | --- | --- |
| 01 | viewAllSubjectSetting | Get data, update data, Delete data to database |
| 02 | getTotal | Get the number of Subject Setting in the database |
| 03 | changeStatus | Update the status of Subject Setting in the database |

## 

DaoSetting

## 

| No | Method | Description |
| --- | --- | --- |
| 01 | getSettingToFilter | Get a list of Setting from the database |

## 

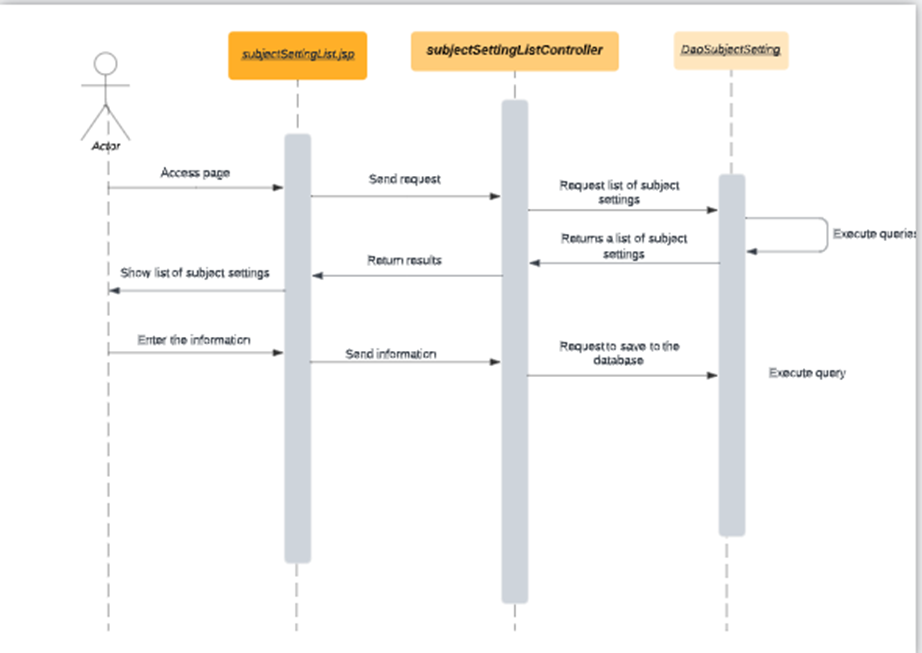
DaoSubject

## 

| No | Method | Description |
| --- | --- | --- |
| *01* | *getAllSubject* | *Get a list of all Subject from the database* |

### 

***c. Sequence Diagram(s)***



***d. Database queries***

"SELECT \n"

+ "su.subject\_code,s.setting\_id,s.type\_id,s.setting\_title,s.setting\_value,s.display\_order,s.status,s.description, count(\*) over() as RowOfSubject\n"

+ "FROM spm.subject\_setting s,spm.subject su \n"

+ "where s.subject\_id = su.subject\_id\n"

+ "LIMIT ? OFFSET ?";

SELECT su.subject\_code,s.setting\_id,s.type\_id,s.setting\_title,s.setting\_value,s.display\_order,s.status,s.description FROM spm.subject\_setting s,spm.subject su where s.subject\_id = su.subject\_id and s.setting\_title like ?"

"SELECT su.subject\_code,s.setting\_id,s.type\_id,s.setting\_title,s.setting\_value,s.display\_order,s.status,s.description FROM spm.subject\_setting s,spm.subject su where s.subject\_id = su.subject\_id and s.setting\_value like ?"

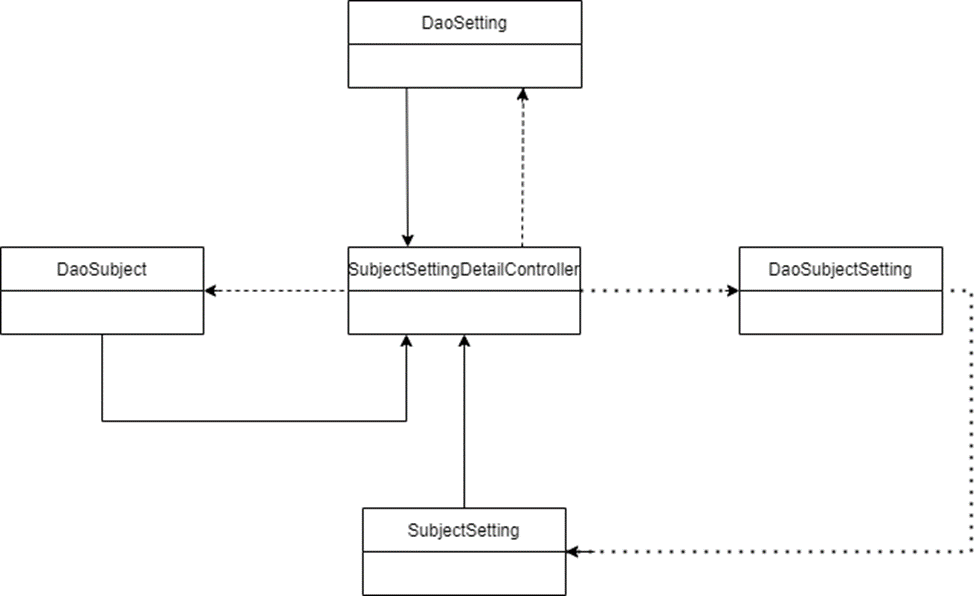
"SELECT distinct su.subject\_code FROM spm.subject\_setting s,spm.subject su where s.subject\_id = su.subject\_id";

"SELECT su.subject\_code,s.setting\_id,s.type\_id,s.setting\_title,s.setting\_value,s.display\_order,s.status,s.description FROM spm.subject\_setting s,spm.subject su where s.subject\_id = su.subject\_id and su.subject\_code like ?";

"SELECT su.subject\_code,s.setting\_id,s.type\_id,s.setting\_title,s.setting\_value,s.display\_order,s.status,s.description FROM spm.subject\_setting s,spm.subject su where s.subject\_id = su.subject\_id and s.status like ?";

### b. Subject Setting Details

***a. Class Diagram***

******

***b. Table Description***

***SubjectSettingDetailController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest* | *Get data, update data, Delete data to database* |

***DaoSubjectSetting***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSubjectSettingById* | *Find subject setting in database by subject setting id* |
| *02* | *addSubjectSetting* | *Add Subject Setting to the database* |
| *03* | *updateSubjectSetting* | *Update Subject Setting to the database* |

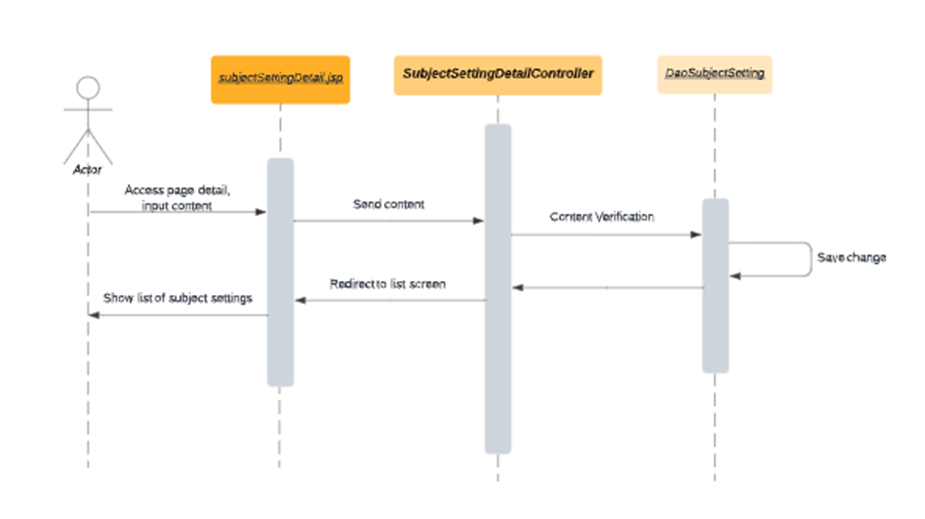
***DaoSetting***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSettingToFilter* | *Get a list of Setting from the database* |

***DaoSubject***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllSubject* | *Get a list of all Subject from the database* |

***c. Sequence Diagram(s)***

******

***d. Database queries***

"UPDATE subject\_setting set setting\_title=?, setting\_value=?, display\_order=?, status=?, description=?, updated\_time=CURDATE() \n"

+ "where setting\_id = ?";

"UPDATE subject\_setting\n"

+ "SET status = ?\n"

+ "WHERE setting\_id = ?";

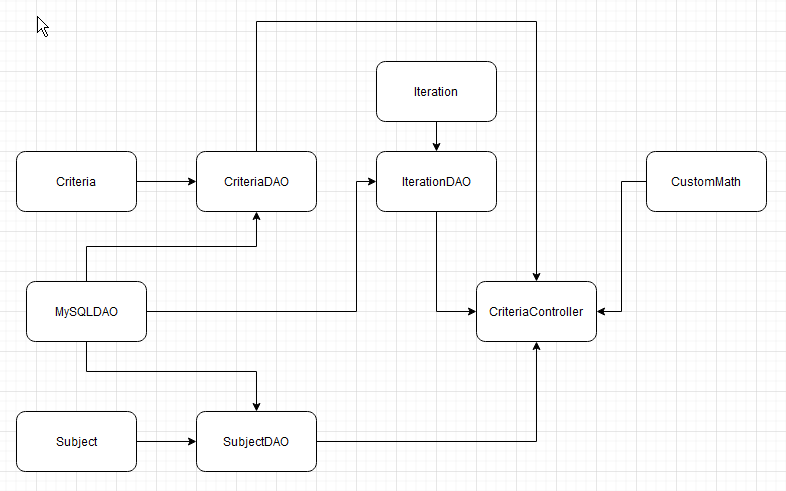
## 12. User Authorization

### 12.a. Class Diagram

### 12.b. Class Specification

## 13. Criteria Management

### Class Diagram



### Class Specification

MySQLDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getConnection* | *Connect to Database* |

SubjectDAO Class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getSubjectList* | *Get all subject from database then return a list of subjects.* |
| *02* | *getSubjectListByUID* | *Input an user id, then get all subject from database based on that id then return a list of subjects.* |

IterationDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getAllIteration* | *Get all iterations from database then return a list of iterations.* |
| *02* | *getAllIterationsSubjectId* | *Input a subject id, get all iterations from database based on that subject id then return a list of iterations.* |

CriteriaDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getCriteriaListBySubjectIdAndIterationId* | *Input a page number, the number of entries per page, and a subject id, then return a list of subjects from database based on those 3 parameter.* |
| *02* | *getTotalCriteriaBySubjectIdAndIterationId* | *Input a subject id and an iteration id, then count all records with those 2 parameter from database.* |
| *03* | *getCriteriaListBySubjectIdAndIterationIdAndStatus* | *Input a page number, the number of entries per page, a subject id, and a true/false status then return a list of subjects from database based on those 4 parameter.* |
| *04* | *getTotalCriteriaBySubjectIdAndIterationIdAndStatus* | *Input a subject id, an iteration id, and a true/false status then count all records with those 3 parameter from database.* |
| *05* | *getCriteriaListBySubjectIdAndIterationIdAndName* | *Input a page number, the number of entries per page, a subject id, and a string then return a list of subjects from database based on those 4 parameter.* |
| *06* | *getTotalCriteriaBySubjectIdAndIterationIdAndName* | *Input a subject id, an iteration id, and a string then count all records with those 3 parameter from database.* |
| *07* | *getCriteriaListBySubjectIdOrdered* | *Input a page number, the number of entries per page, a subject id, and a string then return an ordered list of subjects from database based on those 4 parameter.* |
| *08* | *getIterationCriteriaDetails* | *Input an id, then return a criteria entity based on that id from the database.* |
| *09* | *getSubjectNameByCriteriaId* | *Input a criteria id, then return a subject name having that criteria from the database.* |
| *10* | *getIterationNameByCriteriaId* | *Input a criteria id, then return an iteration name having that criteria from the database.* |
| *11* | *updateCriteriaStatus* | *Input a criteria id, the current user id and the criteria status, then update the criteria records with those information.* |
| *12* | *updateCriteriaDetails* | *Input a criteria entity and the current user id, then update the criteria records with those information.* |
| *13.* | *addNewCriteria* | *Input a new criteria entity and the current user id, then insert those information into the database.* |

CustomMath class

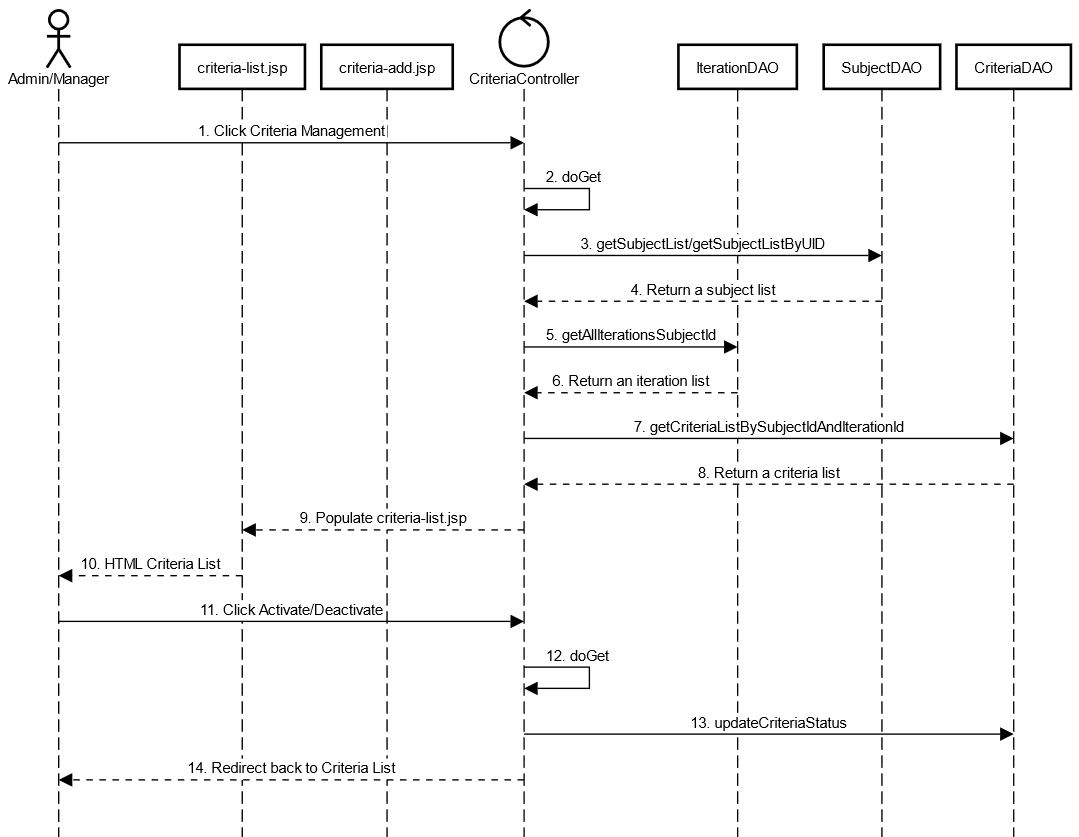
| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *ceiling* | *Input two integers, divide the first one with the second one and round up the result then return that result.* |

CriteriaController class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGet* | *Check current path value, then call functions based on that value.* |
| *02* | *doPost* | *Check current path value, then call functions based on that value.* |
| *03* | *criteriaList* | *Get filter value, pagination page value, and query then paginate data from the database based on those values, after that, display the list of iterations to criteria-list.jsp.* |
| *04* | *criteriaDetails* | *Get criteria informations from the database then display the details information to criteria-detail.jsp* |
| *05* | *updateCriteriaStatus* | *Update the criteria status with updateCriteriaStatus from CriteriaDAO then reload the page.* |
| *06* | *updateCriteriaDetails* | *Update the criteria details with information from the form by using updateCriteriaDetails from CriteriaDAO then reload the page.* |
| *07* | *toAddCriteria* | *Get current subject id, a list of all iterations with that subject id, the direct user to criteria-add.jsp* |
| *08* | *addNewCriteria* | *Get all information from current form and the id of the current user. Then call addNewCriteria function from CriteriaDAO to insert those information into the database.* |

### 13.1. Criteria List

#### 13.1.a. Sequence Diagram



#### 13.1.b. Database Queries

SELECT \* FROM subject WHERE subject\_id = ?

SELECT \* FROM subject WHERE manager\_id = ?

SELECT \* FROM iteration

SELECT \* FROM iteration WHERE subject\_id = ?

SELECT criteria.criteria\_id, criteria.iteration\_id, criteria.criteria\_name, criteria.is\_team\_eval, criteria.eval\_weight, criteria.max\_loc, criteria.status, criteria.description FROM evaluation\_criteria criteria JOIN iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE iteration.subject\_id = ? AND criteria.iteration\_id = ? LIMIT ? OFFSET ?

SELECT COUNT(\*) FROM evaluation\_criteria criteria JOIN iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE iteration.subject\_id = ? AND criteria.iteration\_id = ?

SELECT criteria.criteria\_id, criteria.iteration\_id, criteria.criteria\_name, criteria.is\_team\_eval, criteria.eval\_weight, criteria.max\_loc, criteria.status, criteria.description FROM evaluation\_criteria criteria JOIN iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE iteration.subject\_id = ? AND criteria.iteration\_id = ? AND criteria.status = ? LIMIT ? OFFSET ?

SELECT COUNT(\*) FROM evaluation\_criteria criteria JOIN iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE iteration.subject\_id = ? AND criteria.iteration\_id = ? AND criteria.status = ?

SELECT criteria.criteria\_id, criteria.iteration\_id, criteria.criteria\_name, criteria.is\_team\_eval, criteria.eval\_weight, criteria.max\_loc, criteria.status, criteria.description FROM evaluation\_criteria criteria JOIN iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE iteration.subject\_id = ? AND criteria.iteration\_id = ? AND criteria.criteria\_name LIKE ? LIMIT ? OFFSET ?

SELECT COUNT(\*) FROM evaluation\_criteria criteria JOIN iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE iteration.subject\_id = ? AND criteria.iteration\_id = ? AND criteria.criteria\_name LIKE ?

SELECT criteria.criteria\_id, criteria.iteration\_id, criteria.criteria\_name, criteria.is\_team\_eval, criteria.eval\_weight, criteria.max\_loc, criteria.status, criteria.description FROM evaluation\_criteria criteria JOIN iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE iteration.subject\_id = ? AND criteria.iteration\_id = ? ORDER BY criteria.eval\_weight ASC LIMIT ? OFFSET ?

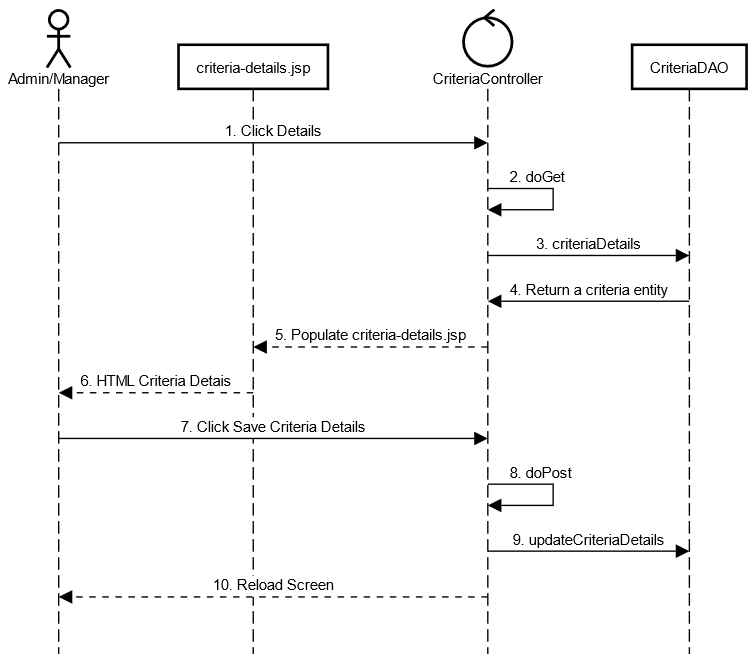
SELECT criteria.criteria\_id, criteria.iteration\_id, criteria.criteria\_name, criteria.is\_team\_eval, criteria.eval\_weight, criteria.max\_loc, criteria.status, criteria.description FROM evaluation\_criteria criteria JOIN iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE iteration.subject\_id = ? AND criteria.iteration\_id = ? ORDER BY criteria.eval\_weight DESC LIMIT ? OFFSET ?

UPDATE evaluation\_criteria SET status = ?, updater\_id = ?, updated\_time = ? WHERE criteria\_id = ?

INSERT INTO evaluation\_criteria (iteration\_id, criteria\_name, is\_team\_eval, eval\_weight, max\_loc, status, description, creator\_id, created\_time) VALUES (?,?,?,?,?,?,?,?,?)

### 13.2 Criteria Details

#### 13.2.a. Sequence Diagram



#### 13.2.b. Database Queries

SELECT \* FROM evaluation\_criteria WHERE criteria\_id = ?

SELECT subject.subject\_name FROM evaluation\_criteria criteria JOIN iteration iteration ON criteria.iteration\_id = iteration.iteration\_id JOIN subject subject ON iteration.subject\_id = subject.subject\_id WHERE criteria.criteria\_id = ?

SELECT iteration.iteration\_name FROM evaluation\_criteria criteria JOIN iteration iteration ON criteria.iteration\_id = iteration.iteration\_id WHERE criteria.criteria\_id = ?

UPDATE evaluation\_criteria SET criteria\_name = ?, is\_team\_eval = ?, eval\_weight = ?, max\_loc = ?, status = ?, description = ?, updater\_id = ?, updated\_time = ? WHERE criteria\_id = ?

## 14. Class Management

#### Class Diagram

#### Class Specification

MySQLDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getConnection* | *Connect to Database* |

SubjectDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getAllSubjects* | *Query all subject from subject table in database and return a list of subject.* |

UserDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getTrainers* | *Query information of users with role is trainer* |

SubjectDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getAllSettings* | *Query all setting from setting table in database and return a list of setting.* |

ClassDAO class

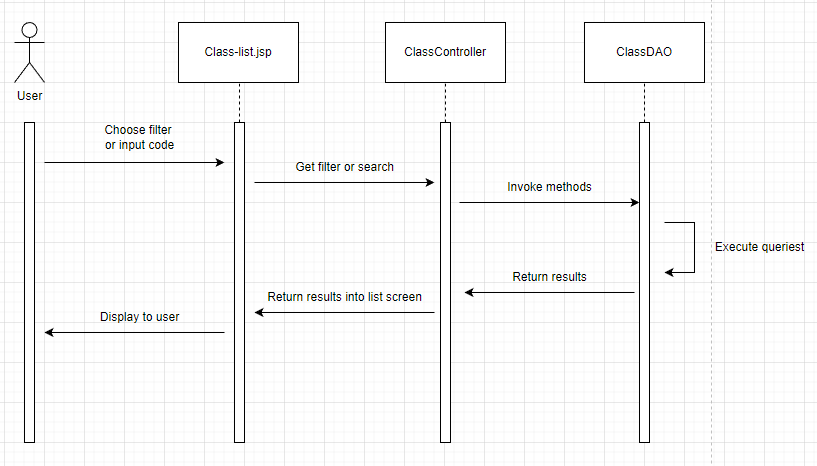
| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getClasses* | *Query all classes from class table with given query and return list of classes corresponding to query* |
| *02* | *getTermList* | *Query all term from class table with given query and return list of terms corresponding to query* |
| *03* | *updateClassStatus* | *Input the table id, the id of the updater as well as the status, then get current time. After that, update the status, updater id and update time of the table based on the input value.* |
| *04* | *getClassDetails* | *Input the id of the table , then query from the database with that table id. After that, return an table object with queried values.* |
| *05* | *updateClassDetails* | *Input an table object and user id, then get current time. After that, update the details, updater id and update time of the table based on the values of the input table object and the 2 remaining values.* |
| *06* | *addNewClass* | *Input an table object and user id, then get current time. After that, insert into the database with the inputted data.* |

ClassController class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGet* | *Check current path value, then call functions based on that value.* |
| *02* | *doPost* | *Check current path value, then call functions based on that value.* |
| *03* | *classList* | *Get filter value, pagination page value, and query then paginate data from the database based on those values, after that, display the list of classes to class-list.jsp.* |
| *04* | *updateClassStatus* | *Get current user id and status value, then call updateClassStatus. After that redirect the user to class list.* |
| *05* | *classDetails* | *Get the requested class id and query the details from database with that id. Then populated the class-detail.jsp with the queried information.* |
| *06* | *updateClassDetails* | *Get all information sent by class details form, then update the database with the received information.* |

### 14.1 Class List

#### 14.1.a. Sequence Diagram



#### 14.1.b. Database queries

SELECT c.class\_id, s.subject\_name, c.class\_code, c.term\_name, c.is\_block5, (select u.full\_name from spm.user u where u.user\_id = c.trainer\_id) as trainer\_name, c.status FROM spm.class c, spm.user u, spm.subject s WHERE c.subject\_id = s.subject\_id AND u.user\_id = c.trainer\_id

SELECT \* FROM spm.subject

SELECT term\_name FROM spm.class

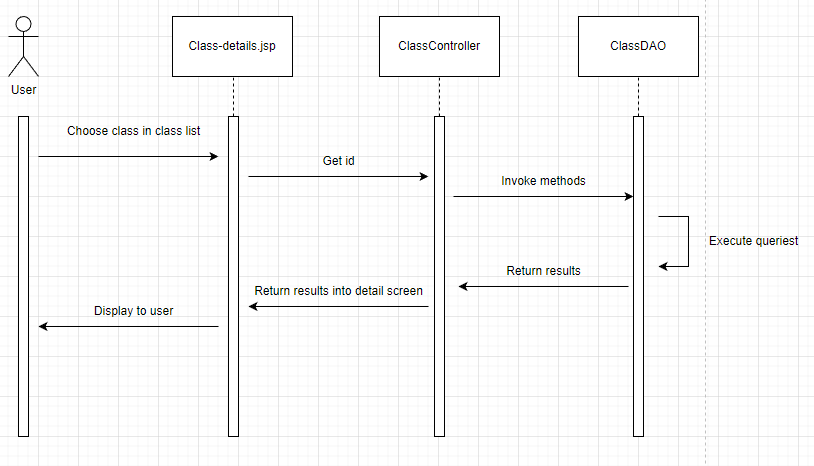
SELECT user\_id, full\_name FROM spm.user where role\_id = 4

UPDATE spm.class SET status = ?, updater\_id = ?, updated\_time = ? WHERE class\_id = ?

INSERT INTO class (subject\_id, class\_code, term\_name, trainer\_id, is\_block5, status, creator\_id, created\_time) VALUES (?,?,?,?,?,?,?,?)

### 14.2 Class Details

#### 14.2.a. Sequence Diagram



#### 14.2.b. Database queries

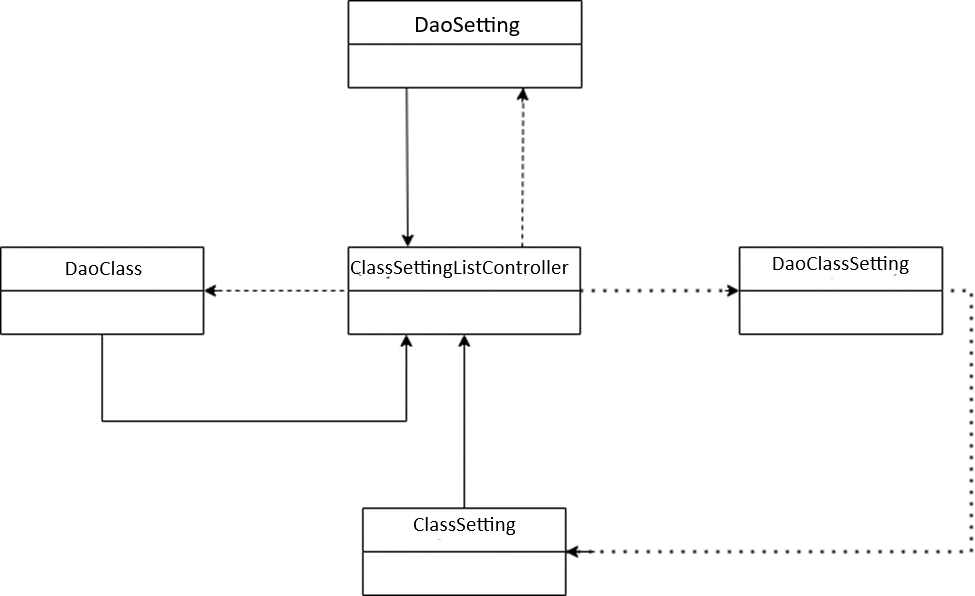
SELECT \* FROM spm.class WHERE class\_id =

UPDATE spm.class SET subject\_id = ?, class\_code = ?, term\_name = ?, is\_block5 = ?, trainer\_id = ?, status = ?, updater\_id = ?, updated\_time = ? WHERE class\_id = ?

## 15. Class Setting Management

### 15.1 Class Setting List

#### 15.1.a Class diagram



#### 15.1.b Table Description

***ClassSettingListController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest* | *Get data, update data, Delete data to database* |

***DaoClassSetting***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *viewAllSubjectSetting* | *Get data, update data, Delete data to database* |
| *02* | *getTotal* | *Get the number of Subject Setting in the database* |
| *03* | *changeStatus* | *Update the status of Subject Setting in the database* |

***DaoSetting***

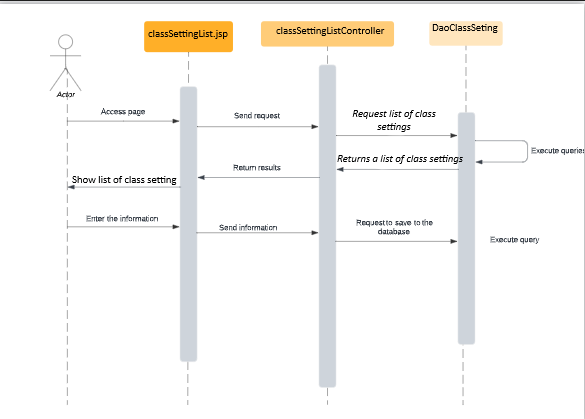
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSettingToFilter* | *Get a list of Setting from the database* |

***DaoClassSetting***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllSubject* | *Get a list of all Subject from the database* |

### 

#### 15.1.c Sequence Diagram(s)



#### 15.1.d Database queries

SELECT \* FROM spm.class\_setting;

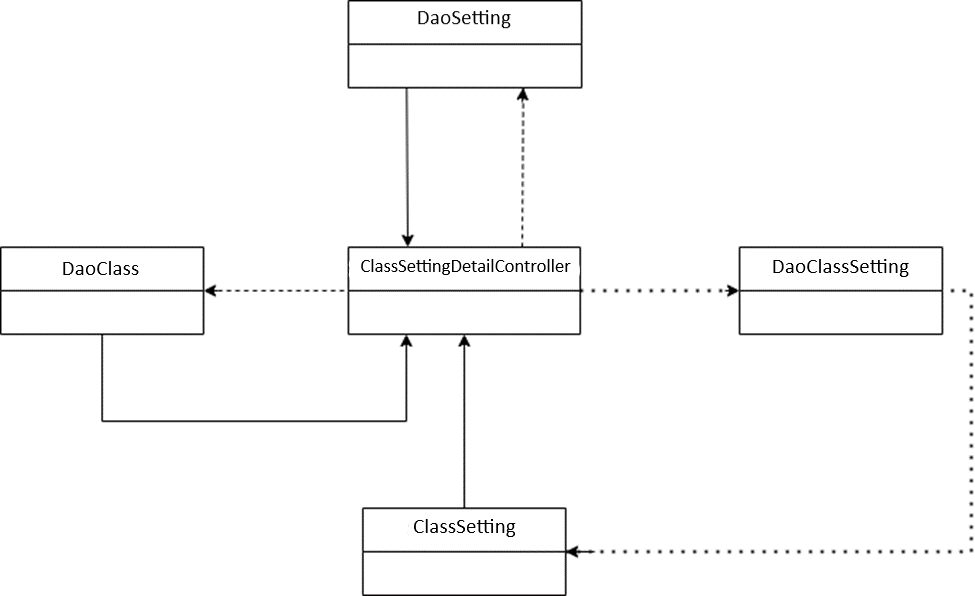
SELECT \* FROM spm.class\_setting WHERE setting\_id = ?

UPDATE `spm`.`class\_setting` SET `status` = ? WHERE (`setting\_id` = ?);

INSERT INTO `spm`.`class\_setting` (`setting\_title`, `setting\_value`, `display\_order`, `description`, `status`, `type\_id`) VALUES (?, ?, ?, ?, b'1' , ?);

### 15.2 Class Setting Details

#### 15.2.a Class diagram



#### 15.2.b Table Description

***ClassSettingDetailController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest* | *Get data, update data, Delete data to database* |

***DaoClassSetting***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getClassSettingById* | *Find subject setting in database by class setting id* |
| *02* | *addClassSetting* | *Add Class Setting to the database* |
| *03* | *updateClassSetting* | *Update Class Setting to the database* |

***DaoSetting***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSettingToFilter* | *Get a list of Setting from the database* |

***DaoSubject***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllClass* | *Get a list of all Class from the database* |

#### 15.2.c Sequence Diagram(s)

#### 15.2.d Database queries

SELECT \* FROM spm.class\_setting;

SELECT \* FROM spm.class\_setting WHERE setting\_id = ?

UPDATE `spm`.`class\_setting` SET `status` = ? WHERE (`setting\_id` = ?);

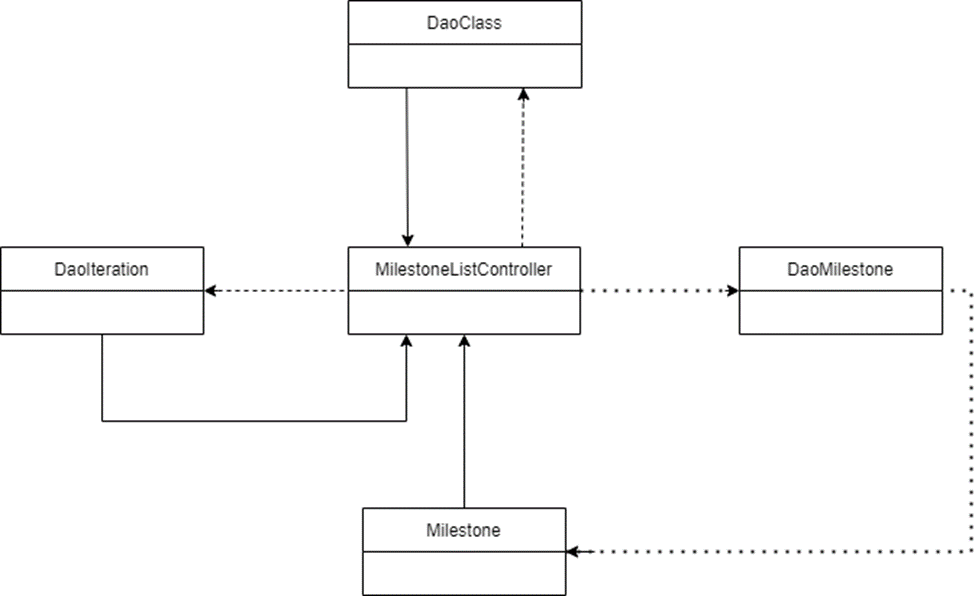
SELECT \* FROM spm.class\_setting WHERE setting\_id = ?

UPDATE `spm`.`class\_setting` SET `setting\_title` = ?, `setting\_value` = ?, `display\_order` = ?, `class\_id` = ?, `description` = ?, `type\_id` =? WHERE (`setting\_id` = ?);

## 16. Milestone Management

### 16.1. Milestone List

#### 16.1.a. Class Diagram

******

#### 16.1.b. Table Description

***MilestoneListController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest* | *Get data, update data, Delete data to database* |

***DaoMilestone***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *viewAllMilestones* | *Get the list of Milestone in the database* |
| *02* | *getTotal* | *Get the number of Milestone in the database* |
| *03* | *changeStatus* | *Update the status of Milestone in the database* |

***DaoClass***

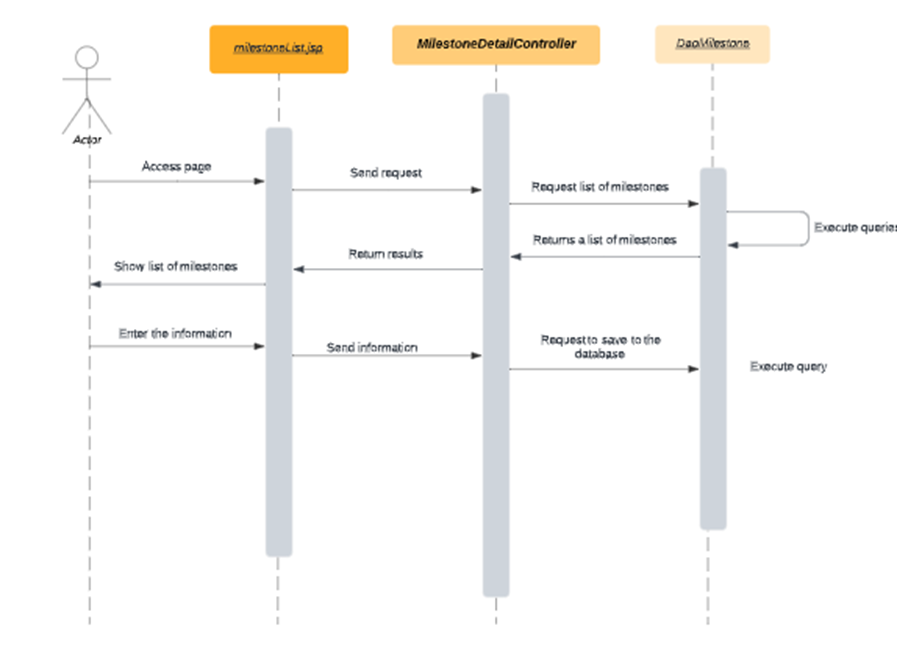
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListClassToFilter* | *Get the list of Class in the database* |

***DaoIteration***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllIteration* | *Get the list of all Iteration in the database* |

### 

#### 16.1.c. Sequence Diagram(s)

******

#### 16.1.d. Database queries

SELECT \* FROM spm.milestone order by from\_date;

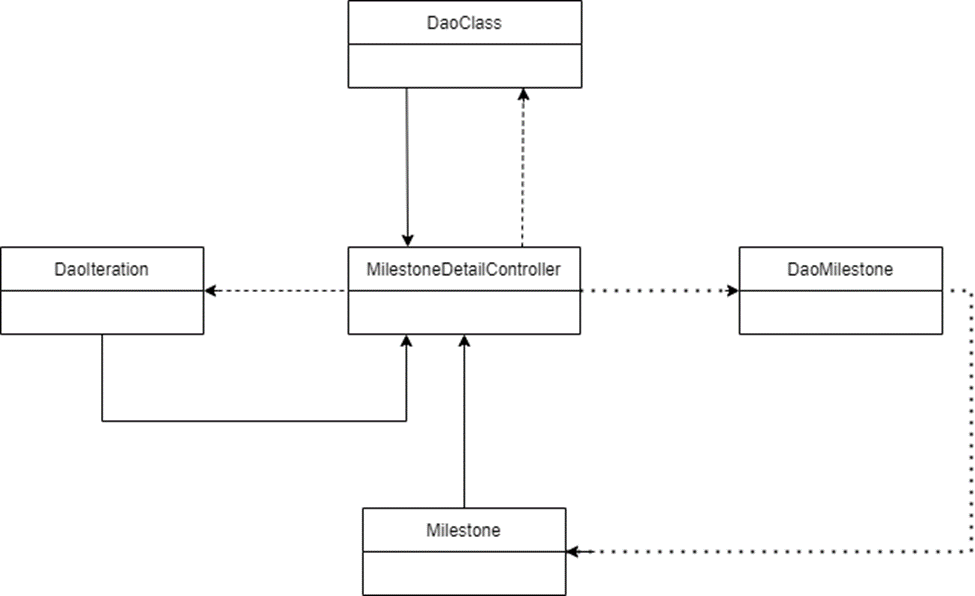
UPDATE `spm`.`milestone` SET `status` = ? WHERE (`milestone\_id` = ?);

SELECT \* FROM spm.milestone WHERE milestone\_id = ?;

INSERT INTO `spm`.`milestone` (`iteration\_id`, `class\_id`, `from\_date`, `to\_date`, `description`, `status`) VALUES (?, ?, ?, ?, ?, '1');

### 16.2.b. Milestone Details

#### 16.2.a. Class Diagram

******

#### 16.2.b. Table Description

***MilestoneDetailController***

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| ***01*** | ***processRequest*** | ***Get data, update data, Delete data to database*** |

***DaoMilestone***

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| ***01*** | ***addMilestone*** | ***Add Milestone to the database*** |
| ***02*** | ***getMilestoneById*** | ***Find Milestone in database by Milestone id*** |
| ***03*** | ***updateMilestone*** | ***Update Milestone to the database*** |

***DaoClass***

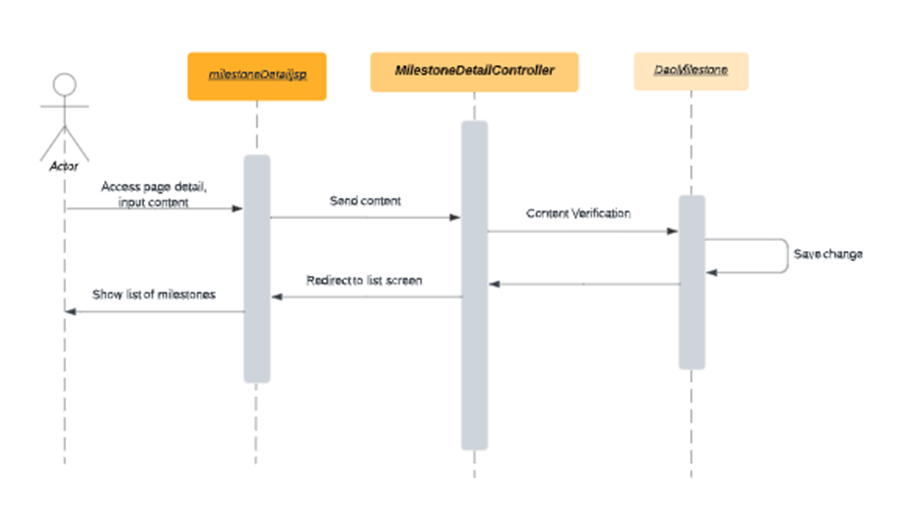
| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| ***01*** | ***getAllClass*** | ***Get a list of all Class from the database*** |

***DaoIteration***

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| ***01*** | ***getAllIteration*** | ***Get a list of all Iterations from the database*** |

### 

#### 16.2.c. Sequence Diagram(s)



#### 16.2.d. Database queries

SELECT \* FROM spm.milestone order by from\_date;

UPDATE `spm`.`milestone` SET `status` = ? WHERE (`milestone\_id` = ?);

SELECT \* FROM spm.milestone WHERE milestone\_id = ?;

UPDATE `spm`.`milestone` SET `iteration\_id` = ?, `class\_id` = ?, `from\_date` = ?, `to\_date` = ?, `description` = ? "WHERE (`milestone\_id` = ?);

SELECT \* FROM spm.milestone

### 16.3. Milestone Submits

#### 16.3.a Class Diagram

#### 

#### 16.3.b Table Description

***MilestoneSubmitController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest* | *Get data, update data, Delete data to database* |
| *022* | *doGet* | *Get data and push data to view* |
| *033* | *doPost* | *Get data, process and execute updates, send redirect* |

***DaoMilestone***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *addMilestone* | *Add Milestone to the database* |
| *02* | *getMilestoneById* | *Find Milestone in database by Milestone id* |
| *03* | *updateMilestone* | *Update Milestone to the database* |

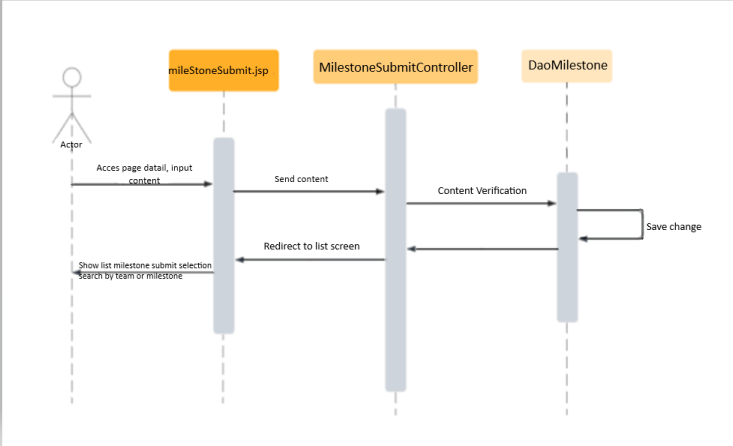
***DaoClass***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllClass* | *Get a list of all Class from the database* |

***DaoIteration***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllIteration* | *Get a list of all Iterations from the database* |

#### 16.3.c Sequence Diagram



#### 16.3.d Database queries

SELECT \* FROM spm.team;

SELECT \* FROM spm.evaluation\_criteria;

UPDATE `spm`.`milestone` SET `status` = ? WHERE (`milestone\_id` = ?);

SELECT \* FROM spm.milestone WHERE milestone\_id = ?;

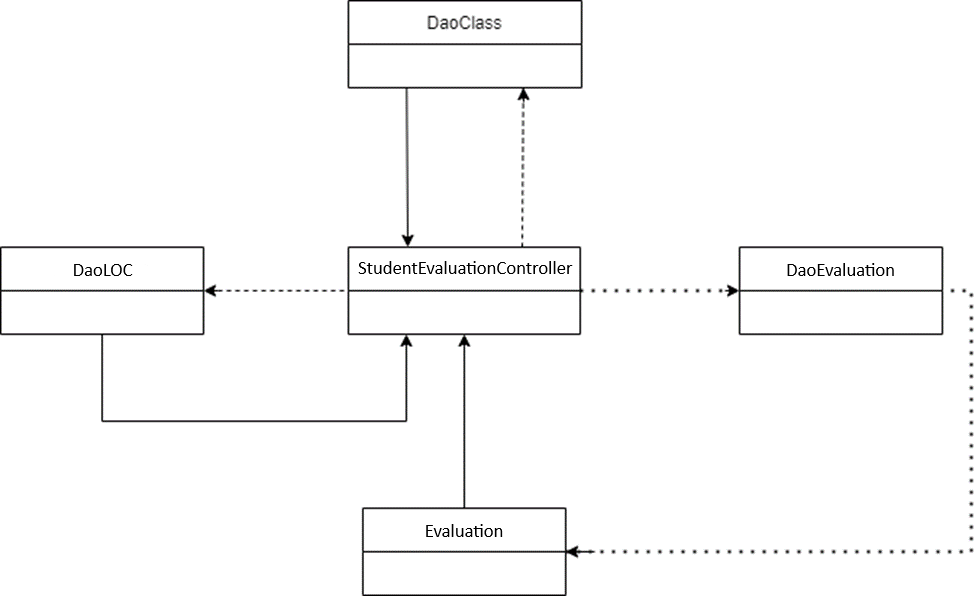
"UPDATE `spm`.`evaluation\_criteria` SET `is\_team\_eval` = ?, `eval\_weight` = ?, `description` = ? WHERE (`criteria\_id` = ?);";

SELECT \* FROM spm.team\_evaluation

UPDATE `spm`.`team\_evaluation` SET `grade` = ?, `comment` = ? WHERE (`id` = ?);

## 17. Student Evaluation

### 17.a Class Diagram



### 17.b Table Description

***StudentEvaluationController***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest* | *Get data, update data, Delete data to database* |
| *022* | *doGet* | *Get data and push data to view* |
| *033* | *doPost* | *Get data, process and execute updates, send redirect* |

***StudentEvaluationDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getTeams* | *get all Teams in the database* |
| *02* | *getEvas* | *get all Evaluation in database* |
| *03* | *updateEva* | *Update EvaluationEvaluation to the database* |

***DaoClass***

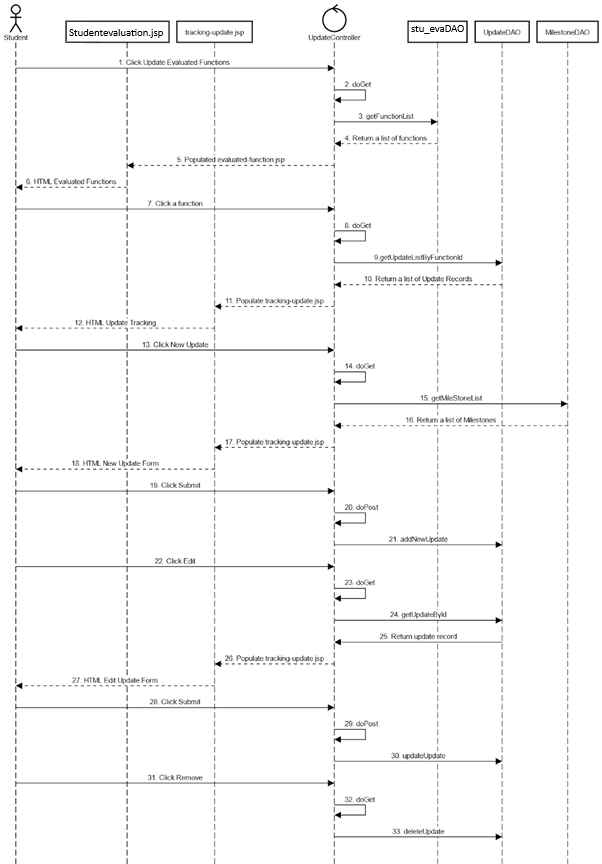
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllClass* | *Get a list of all Class from the database* |

***DaoLOC***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllLOC* | *Get a list of all Iterations from the database* |

#### 

### 17.c Sequence Diagram



### 17.d Database queries

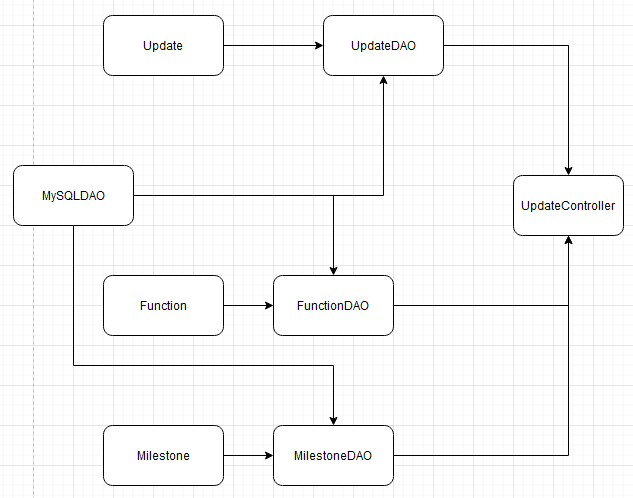
SELECT \* FROM spm.iteration\_evaluation;

SELECT \* FROM spm.loc\_evaluation;

SELECT \* FROM spm.member\_evaluation;

## 18. Tracking Update

### Class Diagram



### Table Description

MySQLDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getConnection* | *Connect to Database* |

FunctionDAO Class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getFunctionList* | *Get a list of functions from the database.* |

MilestoneDAO Class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getMileStoneList* | *Get a list of milestones from the database.* |

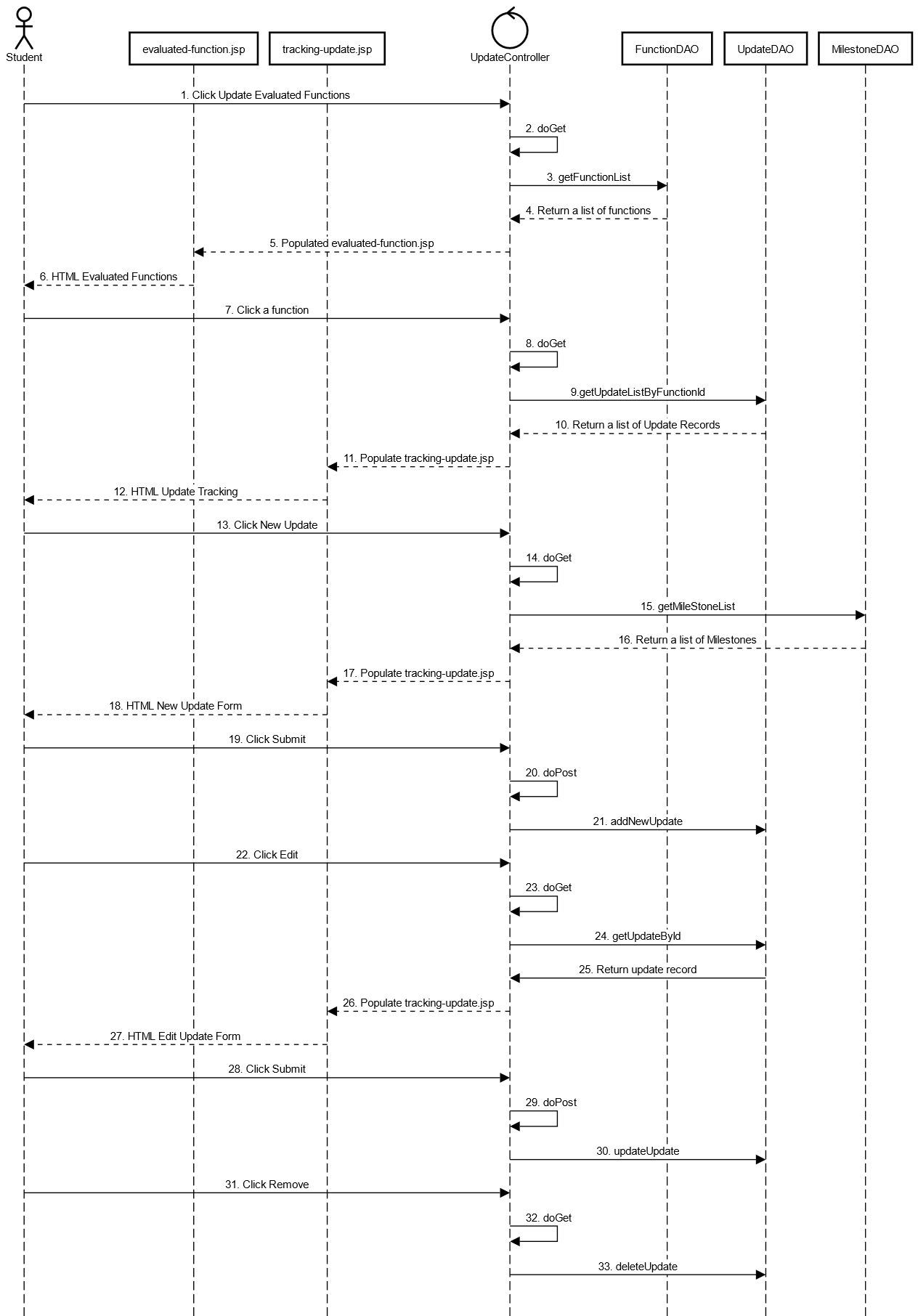
UpdateDAO Class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getUpdateListByFunctionId* | *Input function id as an integer, then return a list of update records based on that id.* |
| *02* | *addNewUpdate* | *Input an update entity and a user id, add those information into the database.* |
| *03* | *getUpdateById* | *Input an integer, get an update record from the update table based on that number as an id.* |
| *04* | *deleteUpdate* | *Input an integer, delete a record from the update table based on that number as an id.* |
| *05* | *updateUpdate* | *Input an update entity and a user id, update records in the update table based on the information from the update entity and the user id.* |

UpdateController Class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGet* | *Check current path value, then call functions based on that value.* |
| *02* | *doPost* | *Check current path value, then call functions based on that value.* |
| *03* | *updatesList* | *Get an display a list of functions you can add/edit/delete the update records of.* |
| *04* | *updatesListDetails* | *Get all update records of the selected function.* |
| *05* | *newUpdate* | *Show a form to allow user to add new update to current function.* |
| *06* | *addNewUpdate* | *Use data from the form mentioned above to add into the update table.* |
| *07* | *getUpdateDetails* | *Get details of a specific update record so user can view/edit those informations.* |
| *08* | *removeUpdate* | *Remove the selected record from the database.* |
| *09* | *updateUpdate* | *Update details of the selected record.* |

### Sequence Diagram



### Database Queries

SELECT \* FROM `function`;

SELECT \* FROM milestone;

SELECT \* FROM `update` where function\_id = ?;

SELECT \* FROM `update` where id = ?;

INSERT INTO `update` (title, description, update\_date, function\_id, milestone\_id, creator\_id, created\_time) VALUES (?,?,?,?,?,?,?);

UPDATE `update` SET title = ?, description = ?, update\_date = ?, milestone\_id = ?, updater\_id = ?, updated\_time = ? WHERE id = ?;

DELETE FROM `update` WHERE id = ?;

## 19. Tracking Submit

### Class Diagram

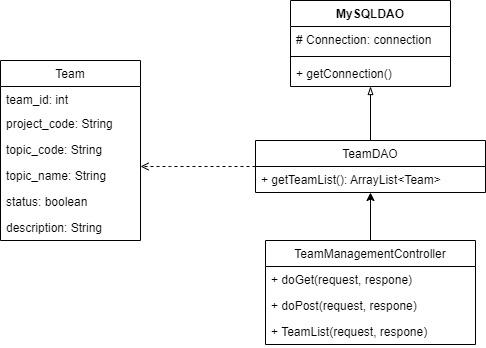
### Table Description

### Sequence Diagram

### Database Queries

## 20. Team List

### Class Diagram



### Table Description

MySQLDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection | Connect to Database |

TeamDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getTeamList() | get list of team from database |

TeamManagementController class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(request, response) | call function teamList(request, respone) |
| 02 | teamList(request, response) | load data of the user from database to Team List screen. |

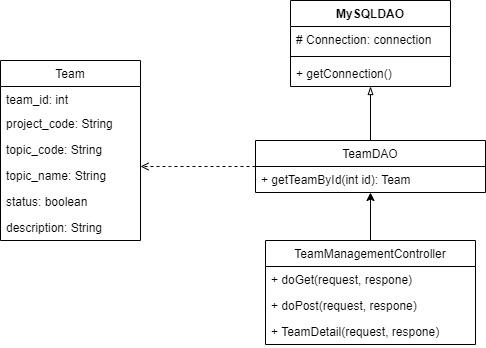
### Sequence Diagram

### Database Queries

select \* from team

## 21. Team Detail

### Class Diagram



### Table Description

MySQLDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection | Connect to Database |

TeamDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getTeamById(int id) | get information of a team from database following team\_id |

TeamManagementController class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(request, response) | call function teamList(request, respone) |
| 02 | teamList(request, response) | load data of a team from database to Team Detail screen. |
| 03 | doPost(request, response) |  |

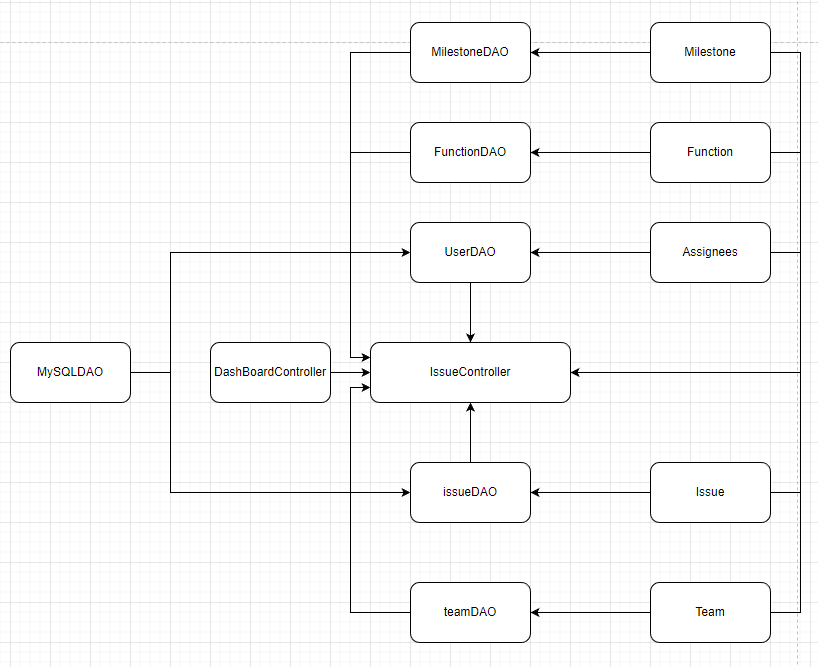
### Sequence Diagram

### Database Queries

SELECT \* FROM team WHERE team\_id = ?

## 22. Issue Management

#### Class Diagram



#### Class Specification

MySQLDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getConnection* | *Connect to Database* |

IssueDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getIssues* | *Query all issues from issue table with given query and return list of issues corresponding to query* |
| *02* | *getTypeList* | *Query all types from types table with given query and return list of types corresponding to query* |

TeamDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getTeamList* | *Query all teams from team table with given query and return list of teams corresponding to query* |

FunctionDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getFunctionList* | *Query all functions from function table with given query and return list of functions corresponding to query* |

MilestoneDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getMileStoneList* | *Query all milestones from milestone table with given query and return list of milestones corresponding to query* |

UserDAO class

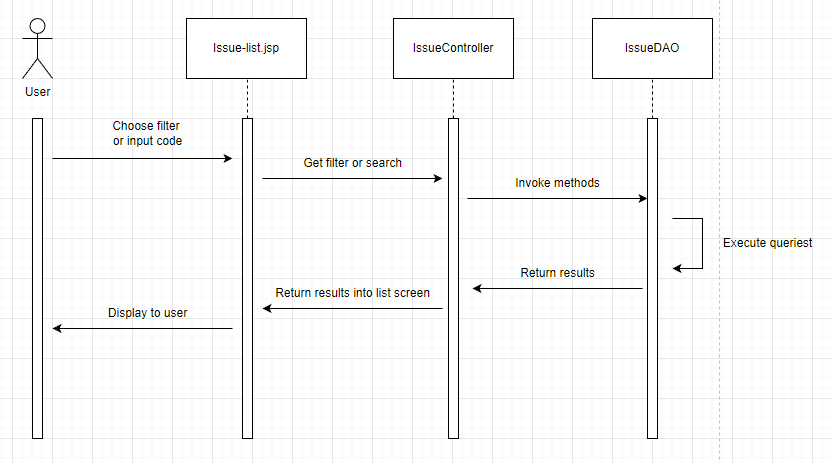
| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getAssignees* | *Query all assignees from assignee table with given query and return list of assignees corresponding to query* |

IssueController class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGet* | *Check current path value, then call functions based on that value.* |
| *02* | *doPost* | *Check current path value, then call functions based on that value.* |
| *03* | *issueList2* | *Get filter value, pagination page value, and query then paginate data from the database based on those values, after that, display the issue of subjects to issue-list.jsp.* |

### 22.1 Issue List

#### 22.1.a. Sequence Diagram



#### 22.1.b. Database queries

SELECT i.issue\_id, i.issue\_title, i.issue\_type, f.title, m.milestone\_name, (select u.full\_name from spm.user u where u.user\_id = i.assigner\_id) as assigner\_name, (select u.full\_name from spm.user u where u.user\_id = i.assignee\_id) as assignee\_name, t.team\_name, i.status FROM spm.issue i, spm.function f, spm.milestone m, spm.team t WHERE i.function\_id = f.function\_id AND i.milestone\_id = m.milestone\_id AND i.team\_id = t.team\_id

SELECT issue\_type FROM spm.issue

select \* from spm.team

SELECT \* FROM spm.`function’

ELECT \* FROM spm.milestone

SELECT user\_id, full\_name FROM spm.user where role\_id = 5

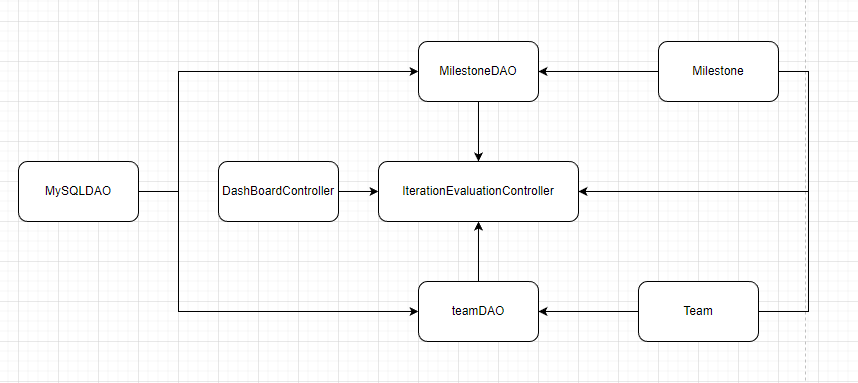
### 22.2 Issue Sync

#### 22.2.a. Sequence Diagram

#### 22.2.b. Database queries

## 23. Iteration Evaluation

### Class Diagram



### Class Diagram

MySQLDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getConnection* | *Connect to Database* |

IterationEvaluationDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getIterValuas* | *Query all iteration evaluations from this table with given query and return list of iteration evaluations corresponding to query* |
| *02* | *importEvaluation* | *Add comments to Team Eval or Individual Eval* |

TeamDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getTeamList* | *Query all teams from team table with given query and return list of teams corresponding to query* |

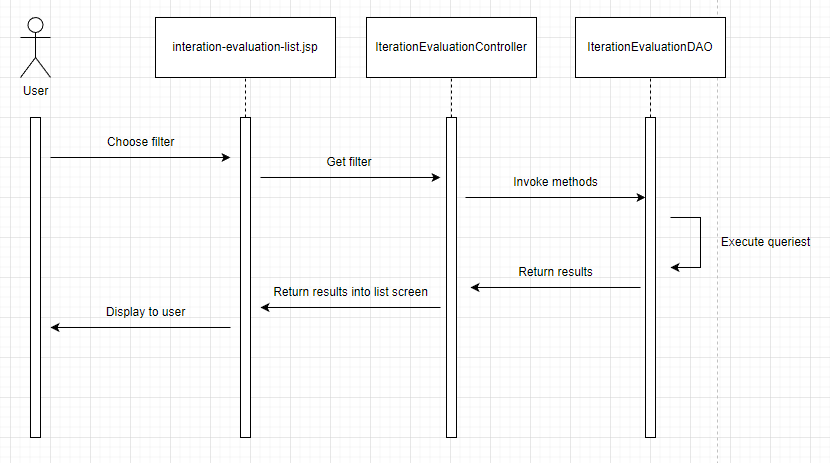
MilestoneDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getMileStoneList* | *Query all milestones from milestone table with given query and return list of milestones corresponding to query* |

IterarionEvaluationController class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doGet* | *Check current path value, then call functions based on that value.* |
| *02* | *doPost* | *Check current path value, then call functions based on that value.* |
| *03* | *itervaluaList* | *Get filter value, pagination page value, and query then paginate data from the database based on those values, after that, display the issue of student to iterarion-evaluation-list.jsp.* |

#### Sequence Diagram

1. 

#### Database queries

SELECT ie.evaluation\_id, u.full\_name, t.team\_name, ie.total\_grade, ie.comment, ie.indi\_comment, ie.loc, ie.bonus FROM spm.iteration\_evaluation ie, spm.team t, spm.user u WHERE ie.team\_id = t.team\_id AND ie.user\_id = u.user\_id

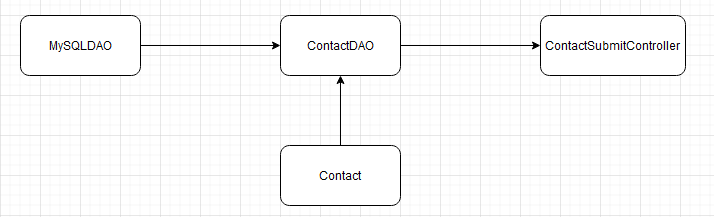
INSERT INTO spm.iteration\_evaluation (comment, indi\_comment, creator\_id, created\_time) VALUES (?,?,?,?)

select \* from spm.team

SELECT \* FROM spm.milestone

## 24. Web Contact

### Class Diagram



### Class Specification

MySQLDAO class

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *getConnection* | *Connect to Database* |

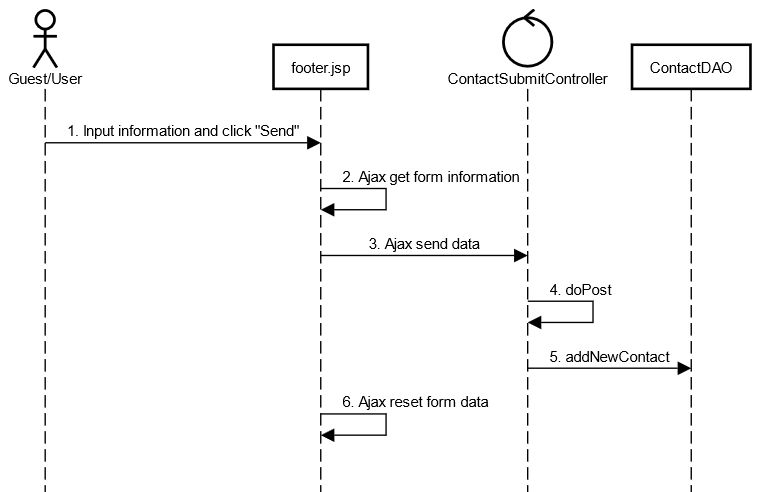
ContactDAO

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *addNewContact* | *Input a contact entity, then insert into the database information of that entity.* |

ContactSubmitController

| ***No*** | ***Method*** | ***Description*** |
| --- | --- | --- |
| *01* | *doPost* | *Receive information from the form in web footer, then put that information into a contact entity, then send that entity into addNewContact as an input.* |

### Sequence Diagram



### Database Queries

INSERT INTO web\_contact(full\_name, email, mobile, category\_id, message, status, created\_time) VALUES (?,?,?,?,?,?,?)