

**QUIZ PRACTICE SYSTEM**

**Software Design Document**

– Hanoi, January 2022 –

**Table of Contents**

[I. Overview](#_heading=h.324f0ii1dl1t) **5**

[1. Code Packages/Namespaces](#_heading=h.30j0zll) 5  
2. Database Schema

[II. Code Designs](#_heading=h.3znysh7)

[1. <Common/Login>](#_heading=h.2et92p0) 9

[a. Class Diagram 9  
 9  
b. Class Specifications](#_heading=h.tyjcwt) 5

[c. Sequence Diagram(s)](#_heading=h.1t3h5sf) 5

[2. <Common/ Register>](#_heading=h.n121qxqqk8d1) 6

[a. Class Diagram](#_heading=h.lcl4t0lb47si) 6

[b. Class Specifications](#_heading=h.z6g979iv8d3k) 6

[c. Sequence Diagram(s)](#_heading=h.7i81esk2em4y) 6

[3. <Common/Reset Password>](#_heading=h.lxjbkgy2lweg) 6

[a. Class Diagram](#_heading=h.jj5pjj26vveq) 6

[b. Class Specifications](#_heading=h.4s4qmh1romfp) 6

[c. Sequence Diagram(s)](#_heading=h.f3417ue9ef8y) 6

[4. <Common/Authorization>](#_heading=h.zbgihong077p) 6

[a. Class Diagram](#_heading=h.x1q1g0f025r3) 6

[b. Class Specifications](#_heading=h.k04uwqswvk36) 6

[c. Sequence Diagram(s)](#_heading=h.1yq2fmdzx1qk) 6

[5. <Common /User Profile >](#_heading=h.jx7s6fj6zo0z) 7

[a. Class Diagram](#_heading=h.ev199it52d4i) 7

[b. Class Specifications](#_heading=h.r1b1pg7ek9rl) 7

[c. Sequence Diagram(s)](#_heading=h.gkt2ynbrhm0w) 7

[6. <Common /'Change Password>](#_heading=h.iflxg6e79poo) 7

[a.Class diagram](#_heading=h.uj3vfwdhehs2) 7

[b.Class Specifications](#_heading=h.do7w5m7ljaam) 7

[c. Sequence Diagram(s)](#_heading=h.45fsenyhk2d7) 7

[7. <Public /Homet>](#_heading=h.8wo8xu5tfnaz) 8

[a. Class Diagram](#_heading=h.91o9q8wgpws5) 8

[b. Class Specifications](#_heading=h.c0p1gt7e92e0) 8

[c. Sequence Diagram(s)](#_heading=h.txcpst73we54) 8

[8. <Public /Quiz List>](#_heading=h.7rzbn9ybik14) 8

[a. Class Diagram](#_heading=h.4cebq4ggkl6c) 8

[b. Class Specifications:](#_heading=h.r2rydemdvd0j) 8

[c. Sequence Diagram(s)](#_heading=h.8nuvuvo4k5k4) 8

[9. <](#_heading=h.jlysr89jaosa)[Public](#_heading=h.7rzbn9ybik14) [/Quiz Detail>](#_heading=h.jlysr89jaosa) 8

[a. Class Diagram](#_heading=h.uunkr8eibijk) 8

[b. Class Specifications:](#_heading=h.vklu7xgnc5l7) 8

[c. Sequence Diagram(s)](#_heading=h.ky5jr43ffdap) 8

[10.<Public/ Subjects List>](#_heading=h.ictt7xs4smyx) 8

[a. Class Diagram](#_heading=h.kltbvro2ogg) 8

[b. Class Specifications](#_heading=h.6lli4qwr3awm) 8

[c. Sequence Diagram(s)](#_heading=h.ryethvika3pn) 8

[11.<](#_heading=h.ah6pik31nyv1)[Public](#_heading=h.ictt7xs4smyx)[/Subject Details>](#_heading=h.ah6pik31nyv1) 9

[a. Class Diagram](#_heading=h.jglfrghnvk7r) 9

[b. Class Specifications](#_heading=h.n3o040lmrv25) 9

[c. Sequence Diagram(s)](#_heading=h.2l3rv239fft1) 9

[12.<Student/Quiz Handle>](#_heading=h.tb3rmetqwglm) 9

[a. Class Diagram](#_heading=h.dlgihm4cli9t) 9

[b. Class Specifications](#_heading=h.yympx7fp6ygj) 9

[c. Sequence Diagram(s)](#_heading=h.87rof7b8ebcb) 9

[13.<Student/Quiz Review>](#_heading=h.ux46p7i6jmi2) 9

[a. Class Diagram](#_heading=h.eyt0fy1oy6kg) 9

[b. Class Specifications](#_heading=h.2mkwg1xpt9v9) 9

[c. Sequence Diagram(s)](#_heading=h.i8hk4u6r7uzg) 9

[14.<Expert/](#_heading=h.ga6mii9fe26a)Quiz List> 10

[a. Class Diagram](#_heading=h.xzf1d66zssho) 10

[b. Class Specifications](#_heading=h.vw4h8rc644s9) 10

[c. Sequence Diagram(s)](#_heading=h.g6ux7ygkx2b1) 10

[15.<Expert/Management Questions>](#_heading=h.jb2gkefov1c3) 10

[a. Class Diagram](#_heading=h.il8zok59bc30) 10

[b. Class Specifications](#_heading=h.copvgbm61taf) 10

[c. Sequence Diagram(s)](#_heading=h.opwrwwst7fge) 10

[16.<Student/Practices List>](#_heading=h.ttj9rbtvjfeb) 10

[a. Class Diagram](#_heading=h.k0ge7zn9kolm) 10

[b. Class Specifications](#_heading=h.7m5kd5lz4fyn) 10

[c. Sequence Diagram(s)](#_heading=h.tt6ru32ouv0h) 10

[a. Class Diagram](#_heading=h.csvdjnnqxqup) 10

[b. Class Specifications](#_heading=h.duzqqze3nopd) 10

[c. Sequence Diagram(s)](#_heading=h.cl3nuxdhumss) 11

[17.<Expert/Subject List>](#_heading=h.bthxq4bt7y26) 11

[a. Class Diagram](#_heading=h.sfca34nytcbe) 11

[b. Class Specifications](#_heading=h.zi8jpjws4hau) 11

[c. Sequence Diagram(s)](#_heading=h.j04p5dud82ky) 11

[18.<Admin/Management Question>](#_heading=h.i270rikkheue) 11

[a. Class Diagram](#_heading=h.wxez1ltixk09) 11

[b. Class Specifications](#_heading=h.llmswi4zd9eo) 11

[c. Sequence Diagram(s)](#_heading=h.p4w24233ijmd) 11

[20.<Admin/](#_heading=h.3r6z0k65an0y)[Management](#_heading=h.ys2pt6a6r39g) [Quiz>](#_heading=h.3r6z0k65an0y) 11

[a. Class Diagram](#_heading=h.igndznd4whhf) 11

[b. Class Specifications](#_heading=h.hw7plwhyvwlb) 11

[c. Sequence Diagram(s)](#_heading=h.hwxv34x109u5) 11

[21.<Admin/](#_heading=h.92wv6fpavmtu)[Management](#_heading=h.ys2pt6a6r39g) [Subject>](#_heading=h.92wv6fpavmtu) 12

[a. Class Diagram](#_heading=h.w42nudvroa1j) 12

[b. Class Specifications](#_heading=h.71auoa8l9dyx) 12

[c. Sequence Diagram(s)](#_heading=h.st7j613fcvzz) 12

[22.<Admin/Management User>](#_heading=h.ys2pt6a6r39g) 13

[a. Class Diagram](#_heading=h.b5zfjamziwqz) 13

[b. Class Specifications](#_heading=h.ao3xg8trlun) 13

[c. Sequence Diagram(s)](#_heading=h.q4ee2lyco1x9) 13

# I. Overview

## 1. Code Packages/Namespaces

| **No** | **Package** | **Description** |
| --- | --- | --- |
| *01* | *model* | *This package contains a lot of java model class, which use to storage data from query database:* |
| *02* | *view* | *This package contains more sub-package concern with frontend of this website: jsp package, css package, js package* |
| *03* | *controller* | *This is a package containing many servlets to help handle business logic and page redirects* |
| *04* | *filter* | *This package contains filter classes that are responsible for intercepting requests from the client before they are passed to the backend and for manipulating responses from the servlet before they are sent back to the client.* |
| *05* | *dao* | *This package contains DAO classes that is responsible for keeping the connection to the database, from which servlets can declare through these classes in this package.* |

## 2. Database Schema

| **No** | **Table** | **Description** |
| --- | --- | --- |
| *01* | *Option* | *PK: optionId*  *FK: questionId foreign key to Question [questionId]* |
| *02* | *Question* | *PK: questionId*  *FK: quizId foreign key to Quiz[quizId]* |
| *03* | *Quiz* | *PK: quizId*  *FK: ownerId foreign key to User [userId]  subjectId foreign key to Subject [subjectId ]*  *quizLevelId foreign key to QuizLevel [quizLevelId]* |
| *04* | *QuizLevel* | *PK: quizLevelId* |
| *05* | *StudentWork* | *PK: studentWorkId FK: userId foreign key to User [userId]*  *quizResultId foreign key to QuizResult [quizResultId]*  *questionId foreign key to Question [questionId]  optionId foreign key to Option [OptionId]* |
| *06* | *QuizResult* | *PK: quizResultId*  *FK: userId foreign key to User [userId]*  *quizId foreign key to Quiz [quizId]* |
| *07* | *User* | *PK: userId*  *FK: roleId foreign key to UserRole [roleId] FK: status foreign key to Status [statusId]* |
| *08* | *UserRole* | *PK: roleId* |
| *09* | *Subject* | *PK: subjectId* |
| *10* | *Status* | *PK: StatusId* |

# II. Code Designs

## 1. Common/Login

### a. Class Diagram

### b. Class Specifications **- LoginController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void doGet(HttpServletRequest request, HttpServletResponse response)* | *Forward user to login screen to input email and password.* |
| *02* | *protected void doPost(HttpServletRequest request, HttpServletResponse response)* | *User inputs email and password, use method from UserDAO to check. If success then creates a session to hold the user, if not success then redirect to the login screen to input again.* |

**-UserDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *User getUserByEmail(String email)* | *The input of this method is email with String type  Return User Object in table in sql by email parameter* |
| *02* | *User checkLogin(String email, String password)* | *The input of this method are email and password with String type  Return User Object in table in sql by email and password parameter* |

**-User:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes userId , userName , email , password , avatar ,gender, dob, roleId, status, code are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

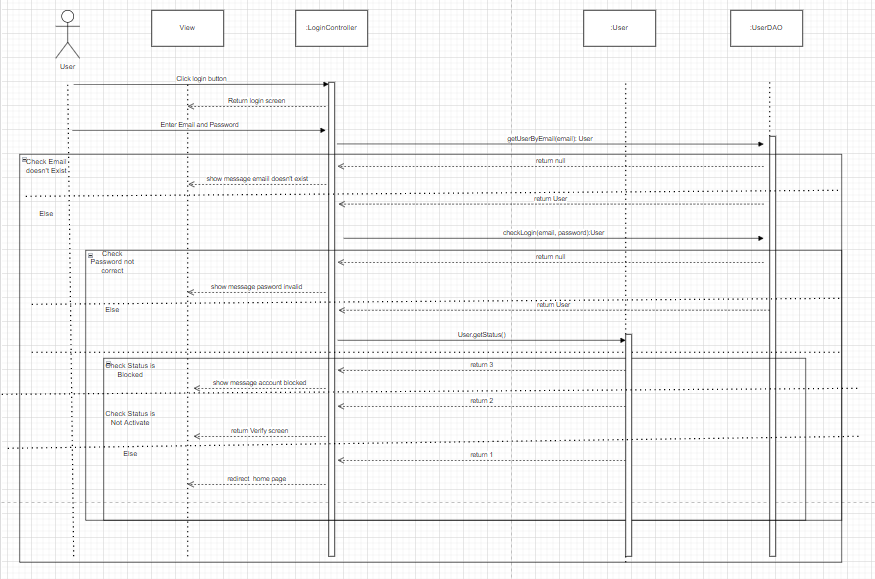
#### 

**- SecurityFilter Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *LoginFilter* | *Check if the user is logged in, if logged in, they cannot login* |

### c. Sequence Diagram

### 



## 2. Common/ Register

### a. Class Diagram

### b. Class Specifications - RegisterController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void doGet(HttpServletRequest request, HttpServletResponse response)* | *Forward user to register screen to input username, email, password, confirm password, gender, dob.* |
| *02* | *protected void doPost(HttpServletRequest request, HttpServletResponse response)* | *User input username, email, password, confirm password, gender, dob from screen, use method from UserDAO to check username, email exist. If success then redirect to login screen, if not success then redirect to register screen to input again.* |

### -UserDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *User getUserByEmail(String email)* | *The input of this method is email with String type  Return User Object in table in sql by email parameter* |
| *02* | *User checkLogin(String email, String password)* | *The input of this method are email and password with String type  Return User Object in table in sql by username parameter* |
| *03* | *void registerAccount(String username, String email, String password, boolean gender, Date dob)* | *The input of this method are username , email, password with String type, gender with boolean type, dob with Date type  This method add a row to sql by username , email, password, gender, dob parameter* |
| *04* | *void updateCode(String code,String email)* | *The input of this method are email and code type.  This method update code of user in sql by code, email parameter* |

### -User:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attribute userId , userName , email , password , avatar ,gender, dob, roleId, status, code are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

### -Email:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *String getRandomNumber()* | *This method to generate A string of numbers is 8 length.* |
| *02* | *void sendEmail(String toEmailAddress, String subject, String message, String verificationCode)* | *This method to create a form and send email information to an email address.* |

### 

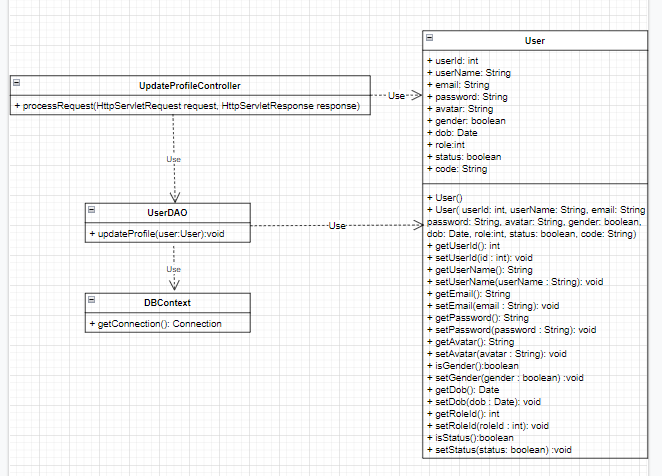
#### - SecurityFilter Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *RegisterFilter* | *Check if the user is logged in, if logged in, they cannot register* |

### c. Sequence Diagram

## 3. Common/User Profile

### a. Class Diagram



### b. Class Specifications

### UpdateProfileController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward user to User Profile page to update data* |

### userDAO:

### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateProfile(User user)* | *Update information of user’s profile in the database* |

### 

* **User:**

## 

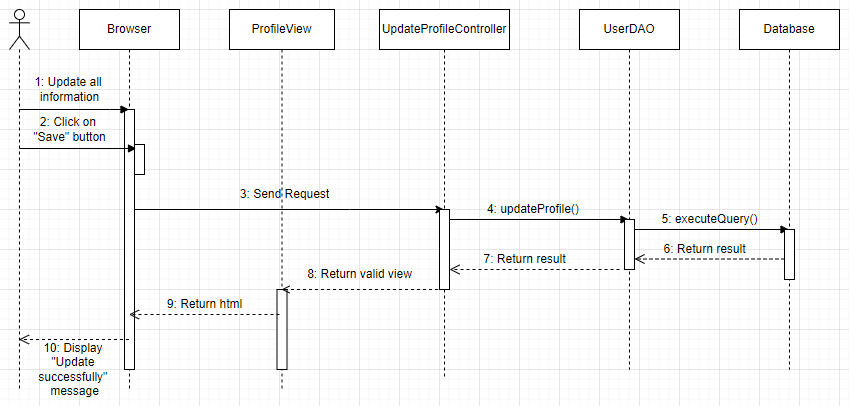
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes userId, userName, email, password, avatar, gender, dob, roleId, status are private, so these methods are used to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

## 

* **DBContext:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect to the database* |

### c. Sequence Diagram(s)



## 4. Common/Change Password

### a. Class Diagram

### b. Class Specifications

### ChangePassController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward user to Change Password screen to input data* |

### userDAO:

### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *changePassword (User user)* | *Update password which has changed to the database* |

### 

* **User:**

## 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes userId, userName, email, password, avatar, gender, dob, roleId, status are private, so these methods are used to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

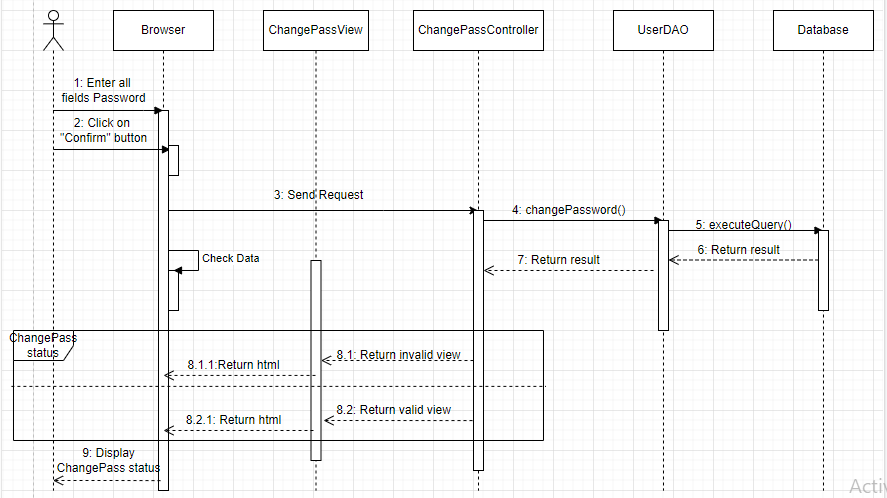
## 

* **DBContext:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect to the database* |

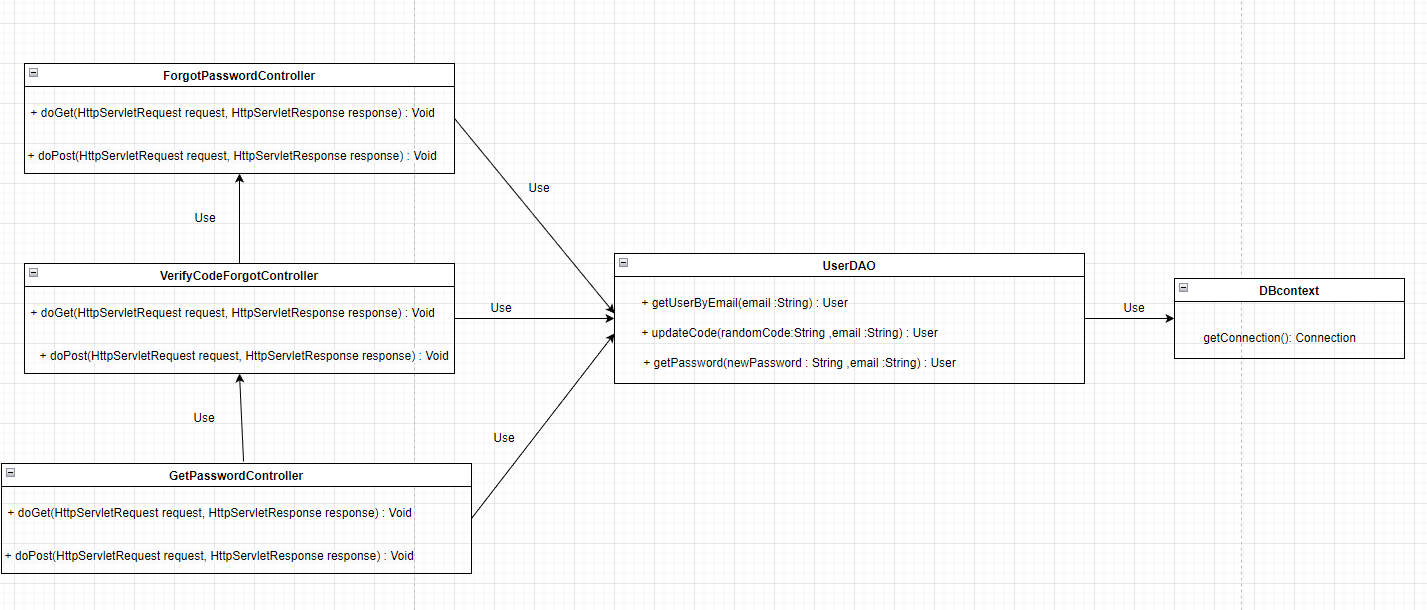
### 

### c. Sequence Diagram(s)



## 5. Common /Forgot Password

### **a.** **Class Diagram**



### b. Class Specifications

### - ForgotPasswordController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void doGet(HttpServletRequest request, HttpServletResponse response)* | *Forward user to forgot password screen to input email .* |
| *02* | *protected void doPost(HttpServletRequest request, HttpServletResponse response)* | *User enters email from screen, uses method from UserDAO to check. If successful, send the code to the email that the user entered and then switch to the verify code page. If it fails, the forgot password page remains the same* |

### 

### - UserDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserByEmail(String email* | *Check if the email from the user entered exists in the database* |
| *02* | *updateCode(String codeRandom,String email)* | *Update code to database* |
| *03* | *getPassword(String newPassword , String email)* | *Update new password to database* |

### 

### 

### -VerifyCodeForgotController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void doGet(HttpServletRequest request, HttpServletResponse response)* | *Forward user to Verify Code screen to input code.* |
| *02* | *protected void doGet(HttpServletRequest request, HttpServletResponse response)* | *Check that the user code entered is correct with the code sent via email, if successful, then switch to the Get password page. If not, keep the page as it is* |

### 

### 

### -GetPasswordController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void doGet(HttpServletRequest request, HttpServletResponse response)* | *Forward user to Get Password screen to input new password and confirm password* |
| *02* | *protected void doGet(HttpServletRequest request, HttpServletResponse response)* | *Check if the user has entered the correct confirmation password, if so, update the password in the database and go to the home page. If not, the page will stay the same* |

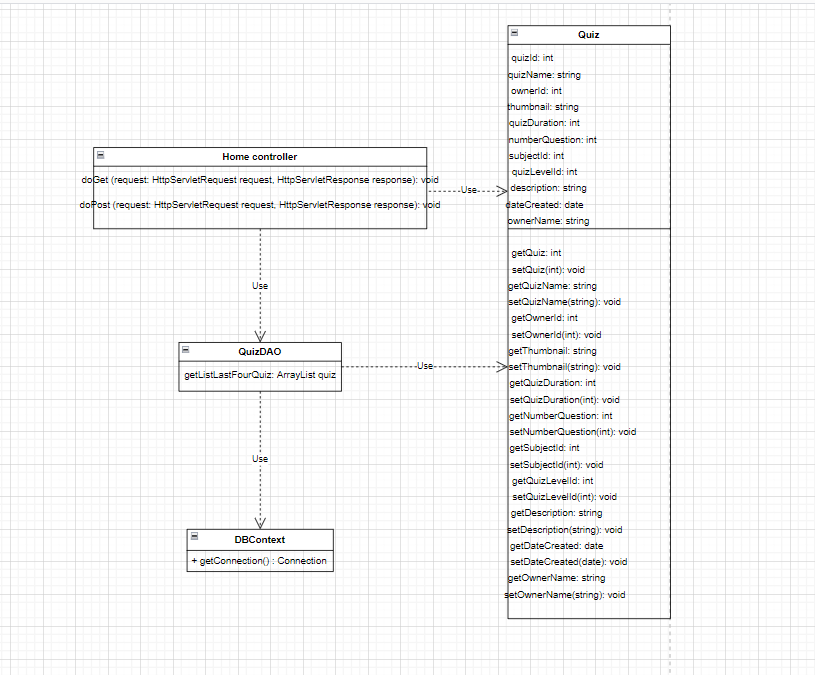
### 

### c. Sequence Diagram(s)

### 

## 6. Common /Home

### a. Class Diagram



### b. Class Specifications

### c. Sequence Diagram(s)

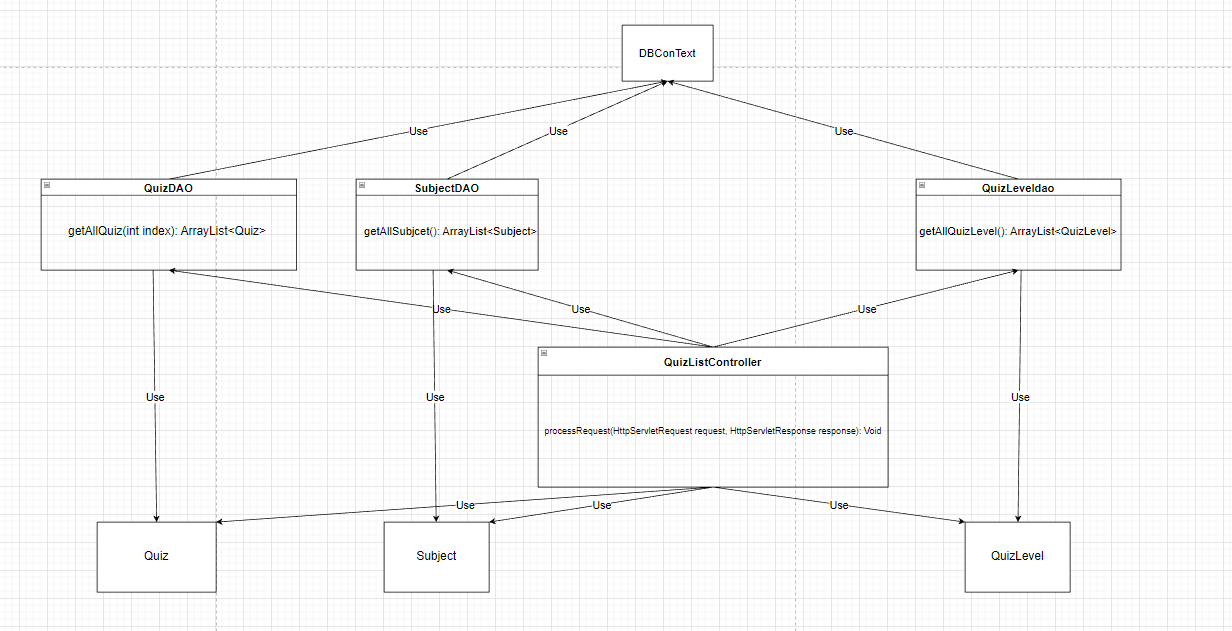
### 

# 

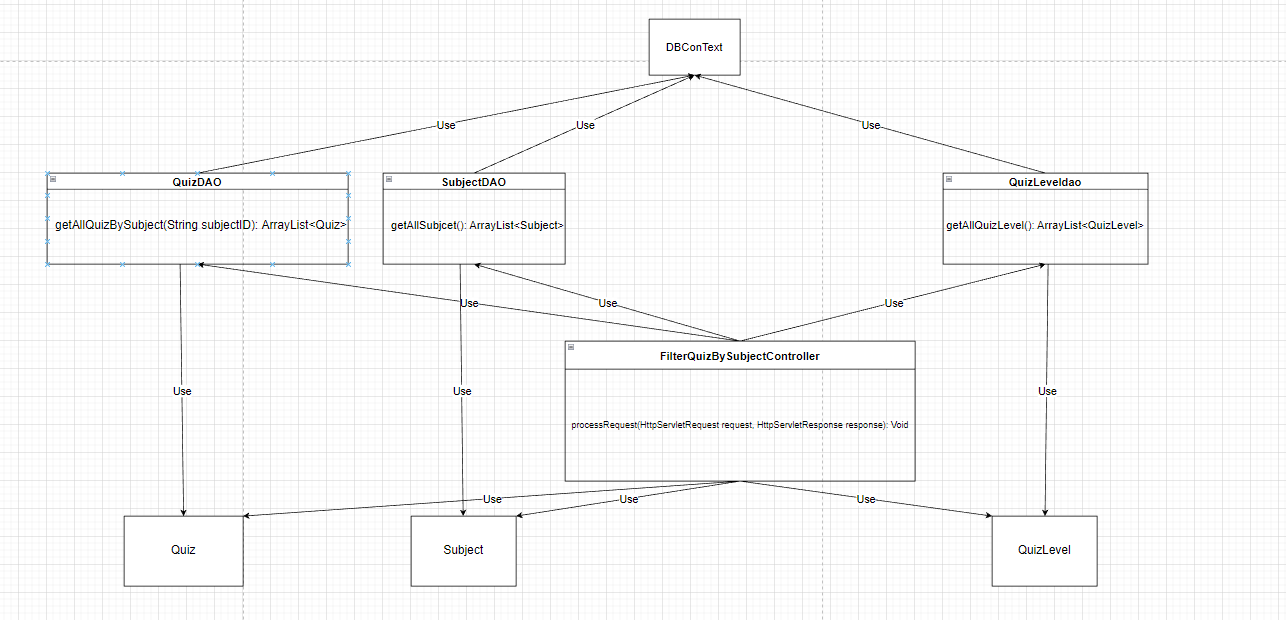
## 7. Common /Views Quiz List

### a. Class Diagram

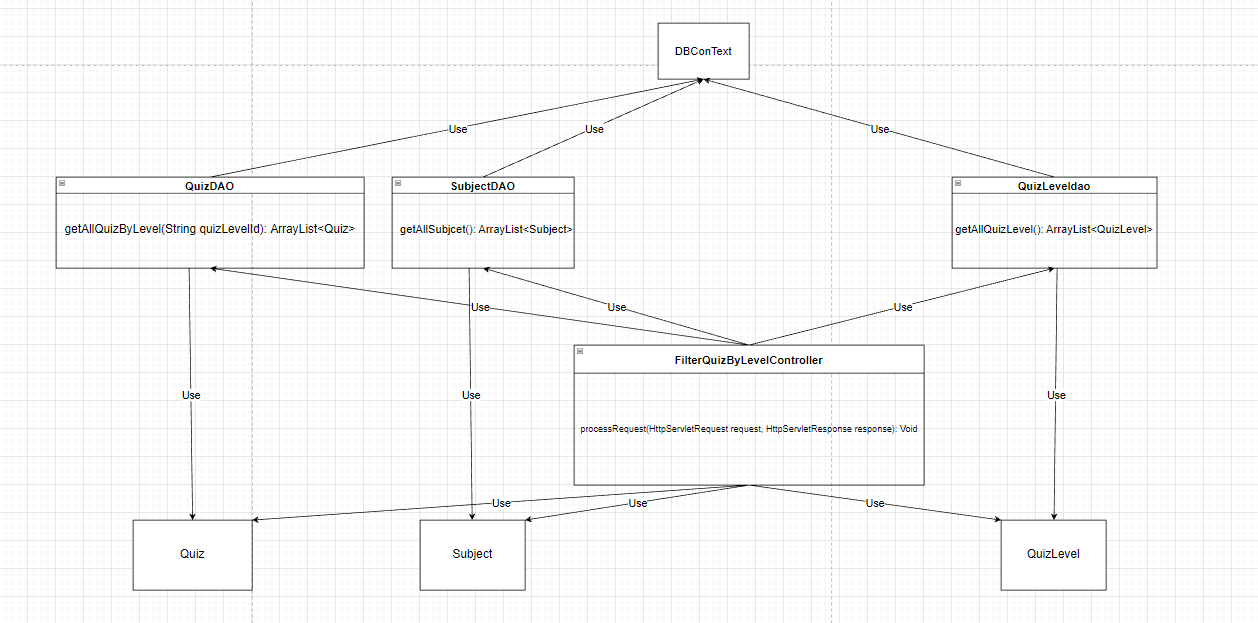
* **Class Diagram for View QuizList**



* **Class Diagram for FilterQuizBySubject**



* **Class Diagram for FilterQuizByLevel**



### b. Class Specifications

**- QuizListController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | * *call getAllQuiz() to get quiz list* * *call getAllSubject() to get subject list* * *call getAllQuizLevel() to get quizLevel list* |

### 

**- FilterQuizBySubjectController**

### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | * *call getAllQuizBySubject() to get quiz list filter by that subject* * *call getAllSubject() to get subject list* * *call getAllQuizLevel() to get quizLevel list* |

### 

**- FilterQuizByLevelController**

### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | * *call getAllQuizBySubject() to get quiz list filter by that quizlevel* * *call getAllSubject() to get subject list* * *call getAllQuizLevel() to get quizLevel list* |

### 

**- QuizDAO**

### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllQuiz(int index)* | *Get quiz list from database* |
| *02* | *getAllQuizBySubject(String subjectID)* | *Get quiz list filter by subject from database* |
| *03* | *getAllQuizByLevel(String quizLevelId)* | *Get quiz list fitter by quiz level from database* |

### 

**- SubjectDAO**

### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllSubjcet()* | *Get all subject list from database* |

### 

**- QuizLevelDAO**

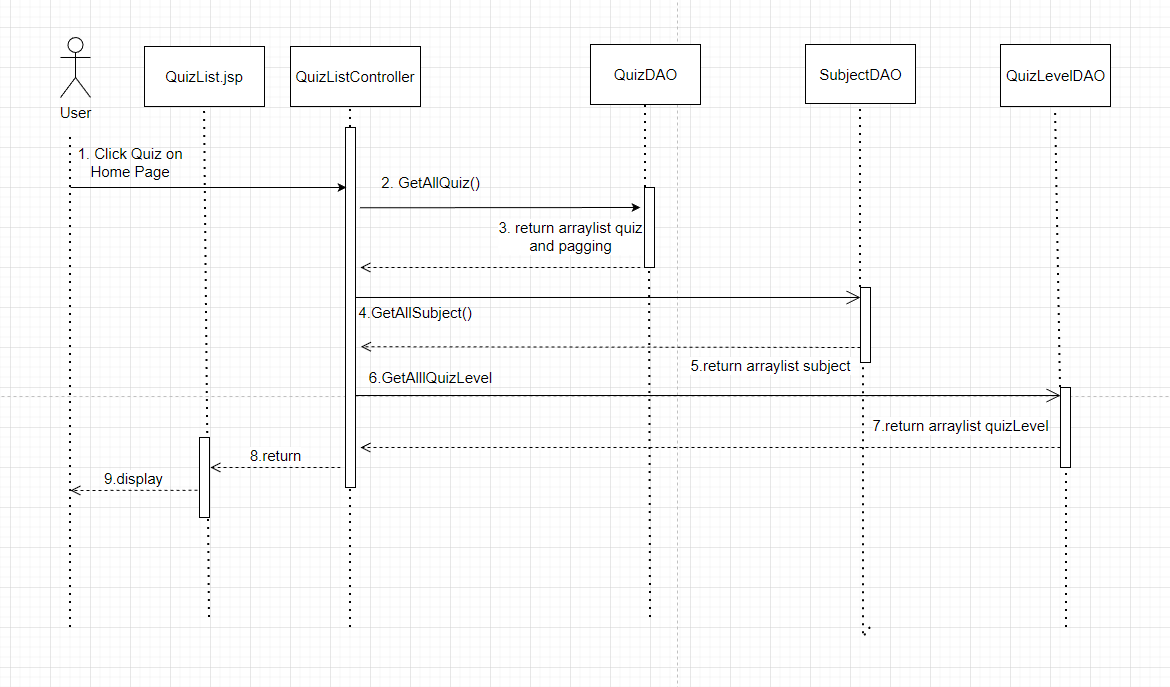
### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllQuizLevel()* | *Get all quizLevel list from database* |

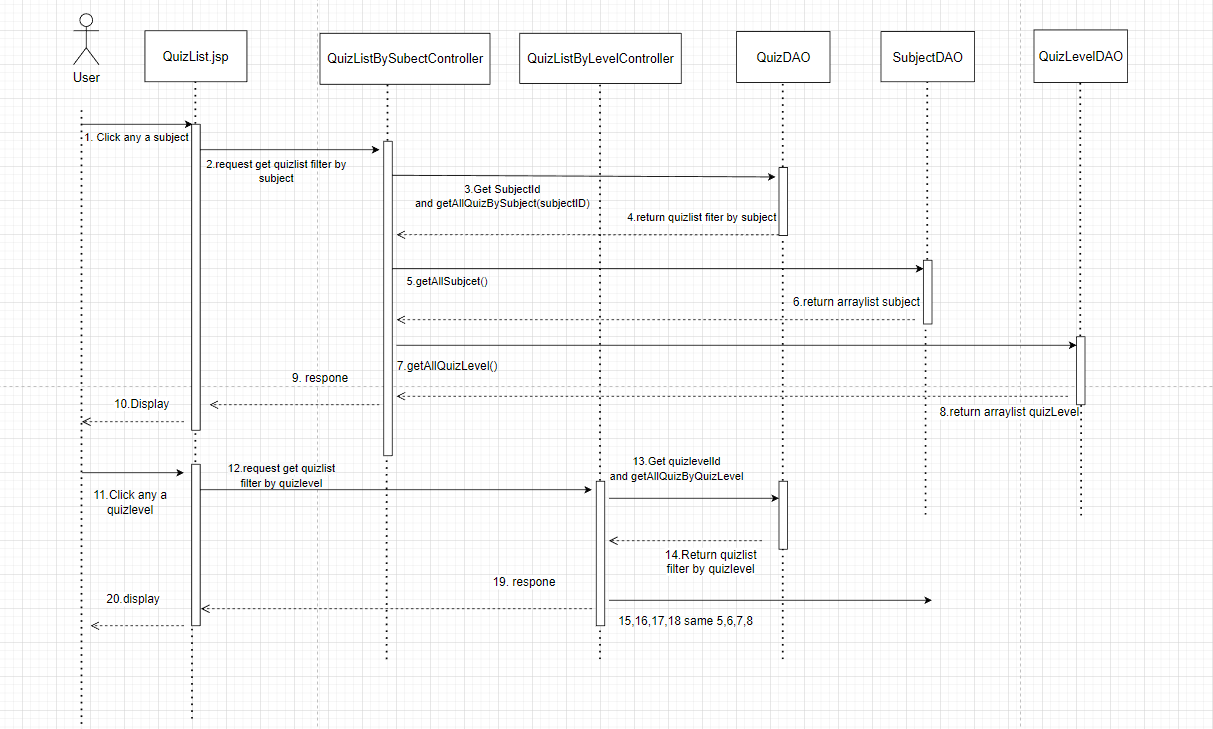
### 

### c. Sequence Diagram(s)

* **Sequence Diagram for View QuizList**



* **Sequence Diagram for FilterQuizBySubject and FilterQuizByQuizLevel**

****

## 

## 8.Common /View Quiz Detail

## a. Class Diagram

### 

### 

### 

### 

### 

### 

### b. Class Specifications

**- QuizDetailController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | * *call getQuizByID(String id) to get information quiz by Quiz id* |

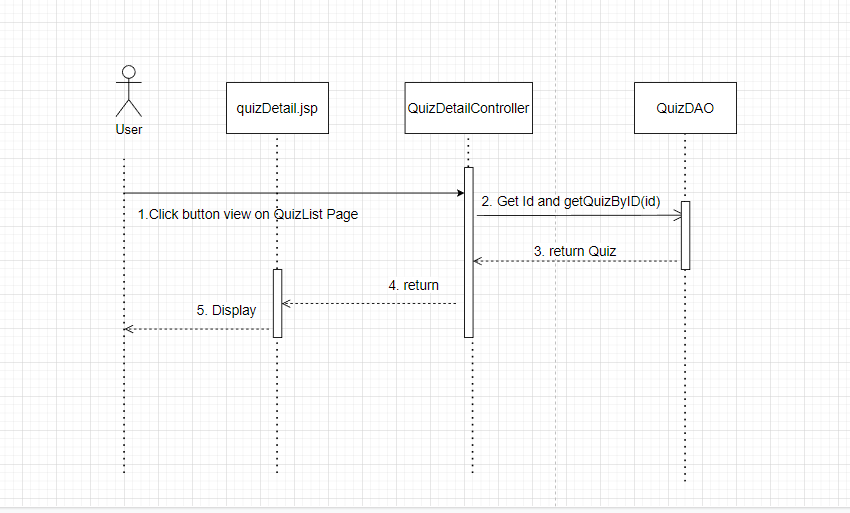
**- QuizLevelDAO**

### 

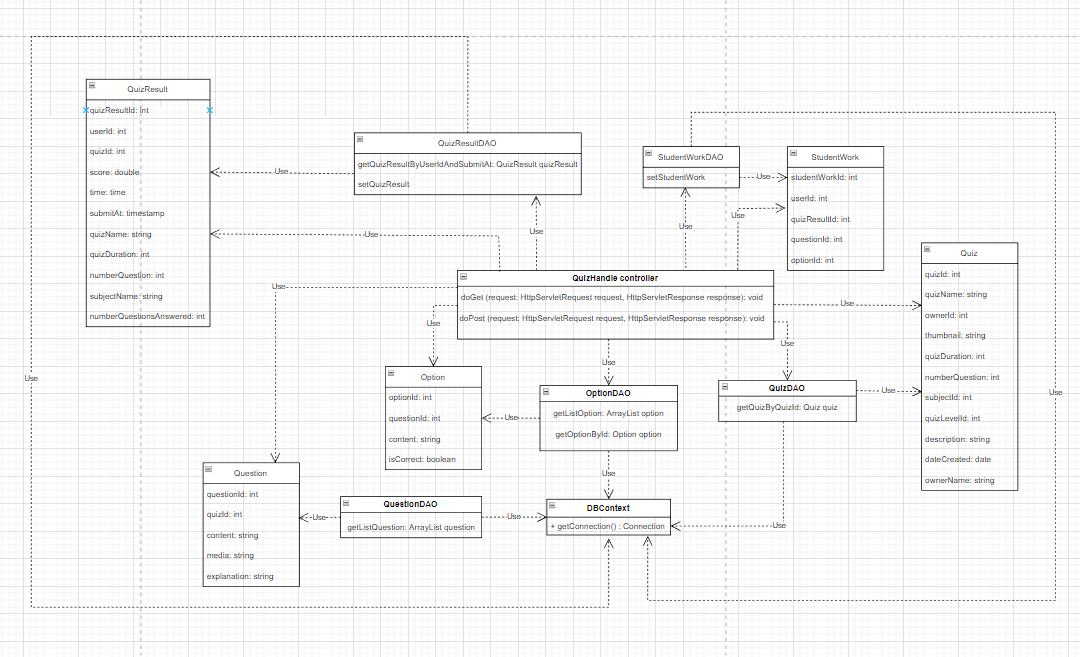
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getQuizByID(String id)* | *Get information quiz by quizId* |

### 

### c. Sequence Diagram(s)

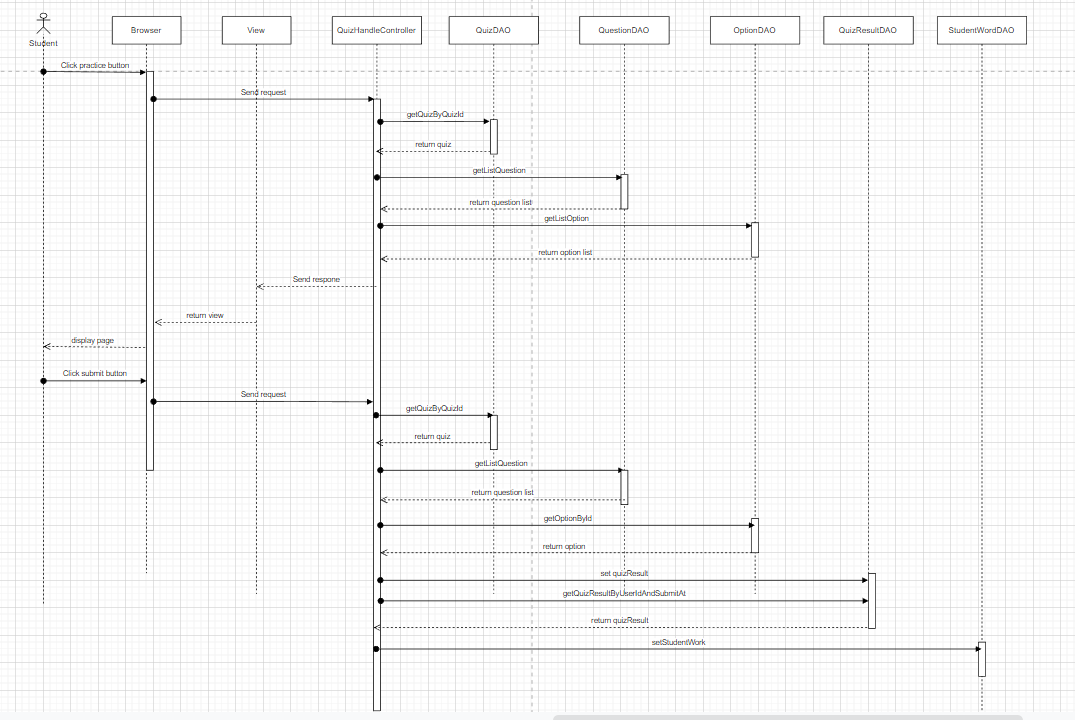


## 9.Student /Quiz Handle a. Class Diagram



### b. Class Specifications

### c. Sequence Diagram(s)



## 10. Student /Quiz Review

### a. Class Diagram

### b. Class Specifications

**- QuizReviewController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward user to quiz review screen after they done a quiz.* |

**-QuizResultDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *QuizResult getQuizResultById(*  *int quizResultId)* | *The input of this method is quizResultId with int type  Return QuizResult Object in table in sql by quizResultId parameter* |

**-QuestionDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *ArrayList<Question> getAllQuestionsByQuizId(int quizId)* | *The input of this method is quizId with int type  Return List Question in table in sql by quizId parameter* |

**-OptionDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *ArrayList<Option> getAllOptionsByQuestionId( int questionId)* | *The input of this method is questionId with int type  Return List Option in table in sql by questionId parameter* |

**-StudentWorkDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *ArrayList<StudentWork> getAllStudentWorkByQuizResultId( int quizResultId )* | *The input of this method is quizResultId with int type  Return List StudentWork in table in sql by email parameter* |

**-QuizResult:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes quizResultId, userId, quizId, score, time, submitAt are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

**-Question:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes questionId, quizId, content, media, explanation are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

**-Option:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes optionId, questionId, content, isCorrect are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

**-StudentWork:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes studentWorkId, userId ,quizResultId , questionId, optionId are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

## 11. Student /Practice List

### a. Class Diagram

### b. Class Specifications

- PracticeListController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward user to practice list.* |

**-QuizResultDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *ArrayList<QuizResult>*  *getAllQuizResultByUserId(int userId, int subjectId)* | *The input of this method are userId and subjectId with int type  Return List QuizResult table in sql by userId and subjectId parameter* |

**-SubjectDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *ArrayList<Subject>*  *getAllQuizSubjects()* | *Return List Subject table in sql* |

**-QuizResult:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes quizResultId, userId, quizId, score, time, submitAt, quizName, quizDuration, numberQuestion, subjectName, numberQuestionAnswerd, numberCorrectAnswer are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

-Subject:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes subjectId, subjectName, thumbnail, description are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

## 

## 12. Common/Subject List

### a. Class Diagram

### b. Class Specifications

**- SubjectListController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward user to list of subjects with pagination* |

* **SubjectDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *ArrayList<Subject> getAllSubject(int pageIndex, int pageSize)* | *Get all of subjects in each page from database* |
| *02* | *int getTotalSubject()* | *Get total of subjects in the database* |

* **Subject:**

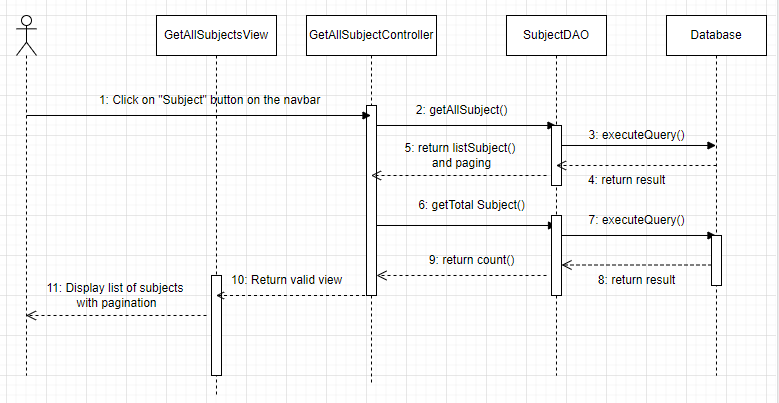
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes subjectId, subjectName, description and image are private, so these methods are used to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

* **DBContext:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect to the database* |

### 

### c. Sequence Diagram:



## 12.Common/Subject Details

### a. Class Diagram

### b. Class Specifications

**- SubjectDetailController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward user to list of subjects with pagination* |

**-SubjectDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSubjectById(int id)* | *Get all of Subjects by ID from database* |

**-Subject:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes subjectId, subjectName, description and image are private, so these methods are used to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

## 

## 

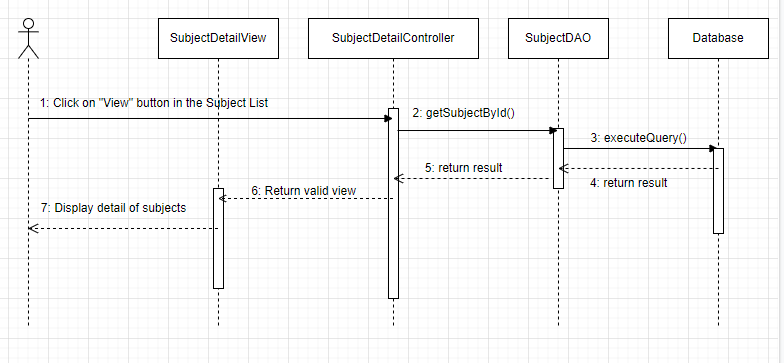
* **DBContext:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect to the database* |

### 

## 

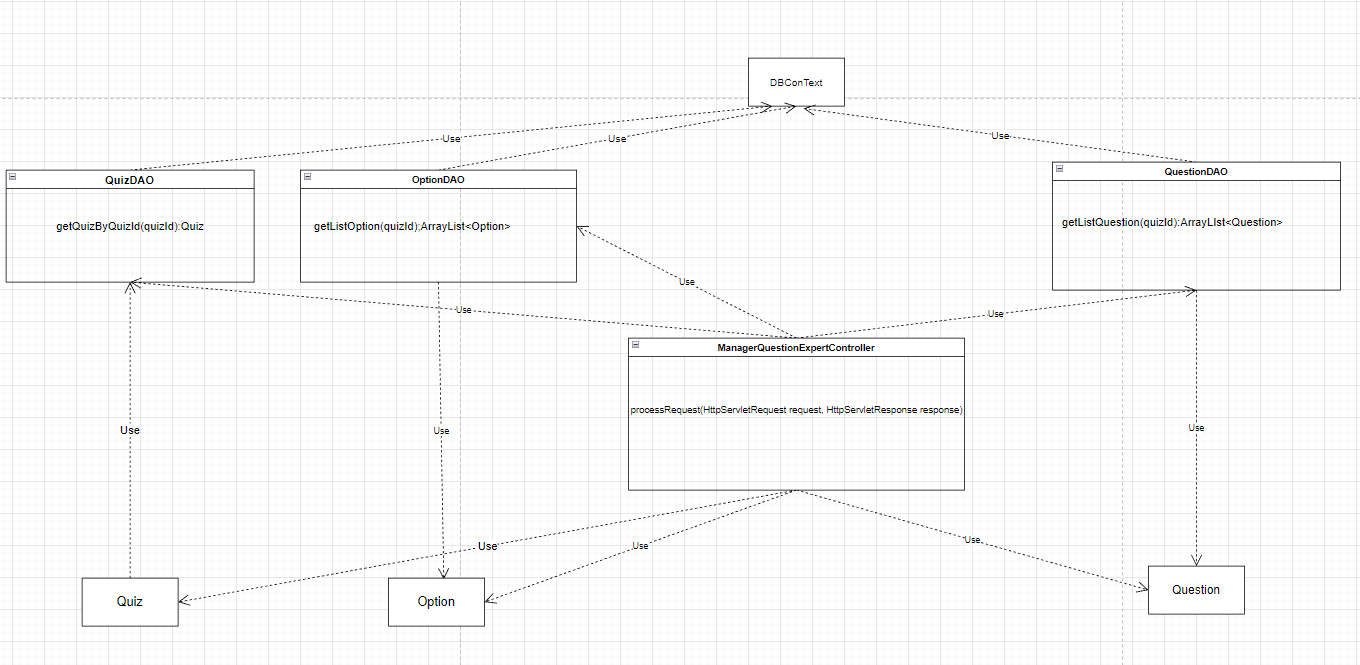
### c. Sequence Diagram:



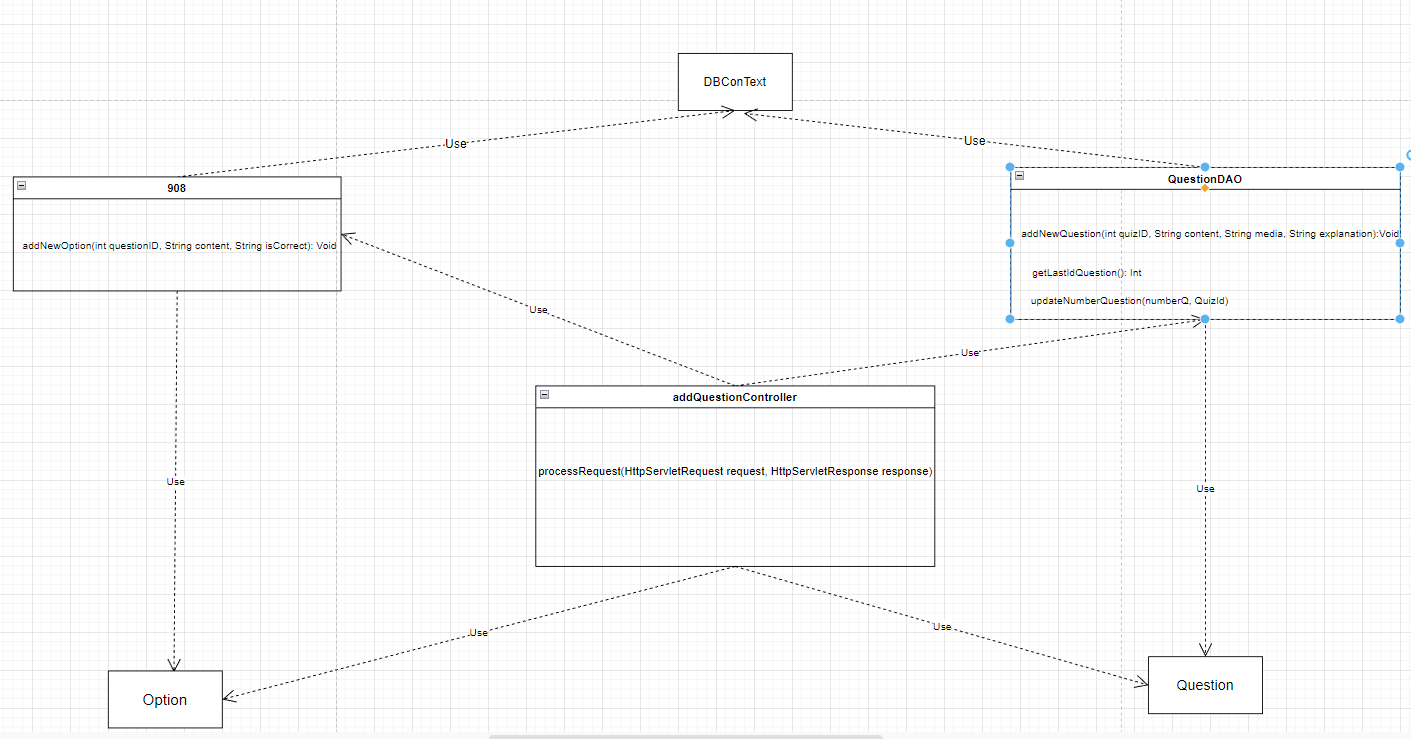
## 13. <Expert/Management Question >

### a. Class Diagram

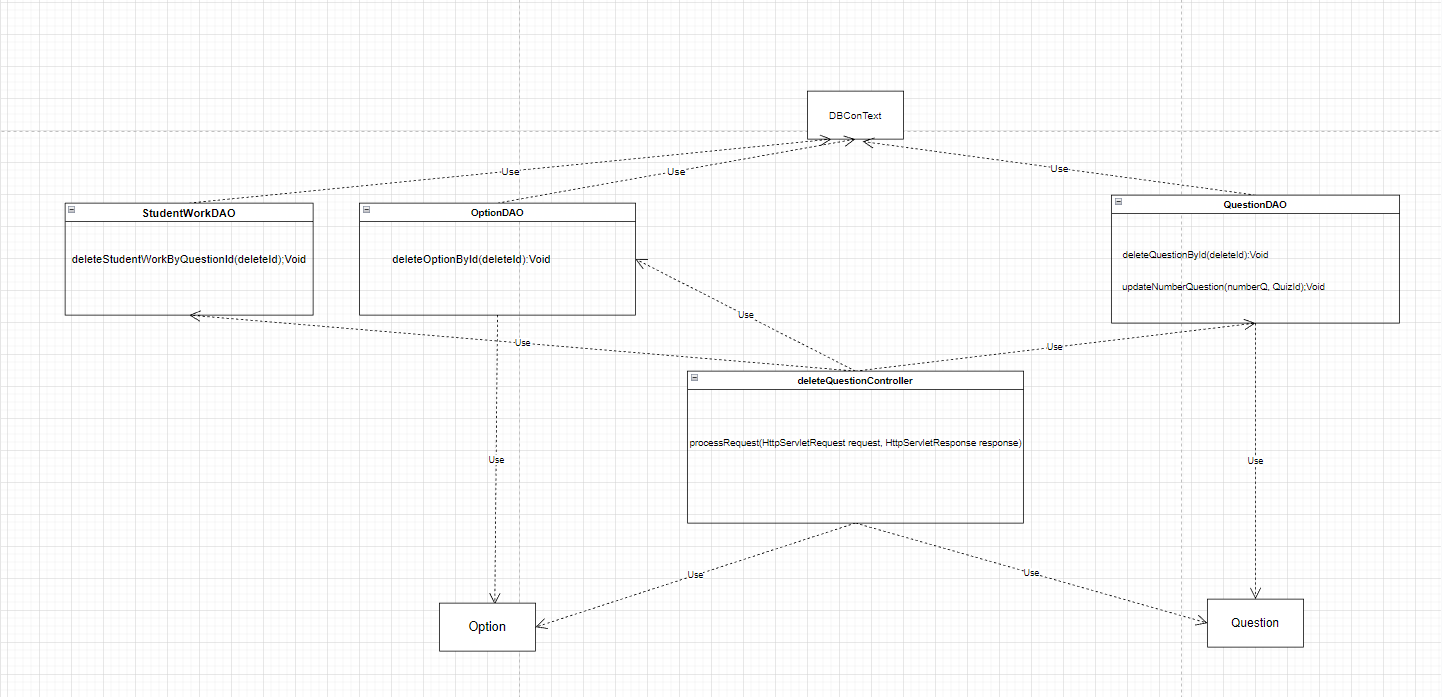
* **Class Diagram for View Question List**

****

* **Class Diagram for Add Question**

****

* **Class Diagram for Delete Question**

****

* **Class Diagram for Edit Question**

****

### b. Class Specifications

**- ManagementQuestionExpertController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward the user to the user management Question with the Question list information in the system.* |

### 

**-AddQuestionController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Get the value of input fields Expert entered and add new question to Quiz* |

### 

**UpdateQuestionController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Get the value of input fields Expert entered and update question to Quiz* |

### 

**DeleteQuestionController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Delete question to Quiz* |

**-informationQuestionToEditController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Get the information of the question you want to edit and send the expert to the question edit page* |

## 

**-QuizDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getQuizByQuizId(quizId)* | *get Quiz from database* |

## 

**-QuestionDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListQuestion(quizId)* | *get List Question from database* |
| *02* | *getQuestion(quizId,questionId)* | *Get Question from database to Expert* |
| *03* | *updateQuestion(questionId, nameQuesttion,explanation)* | *Update question to database* |
| *04* | *deleteQuestionById(deleteId)* | *delete question to database* |
| *05* | *updateNumberQuestion(numberQ, QuizId);* | *Update amount to database* |

## 

**-OptionDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListOption(quizId)* | *get List Option from database to Expert* |
| *02* | *deleteOptionById(deleteId)* | *delete option to database* |
| *03* | *addNewOption()* | *add option to database* |

## 

**- StudentWorkDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteStudentWorkByQuestionId(deleteId)* | *delta studentwork from database* |

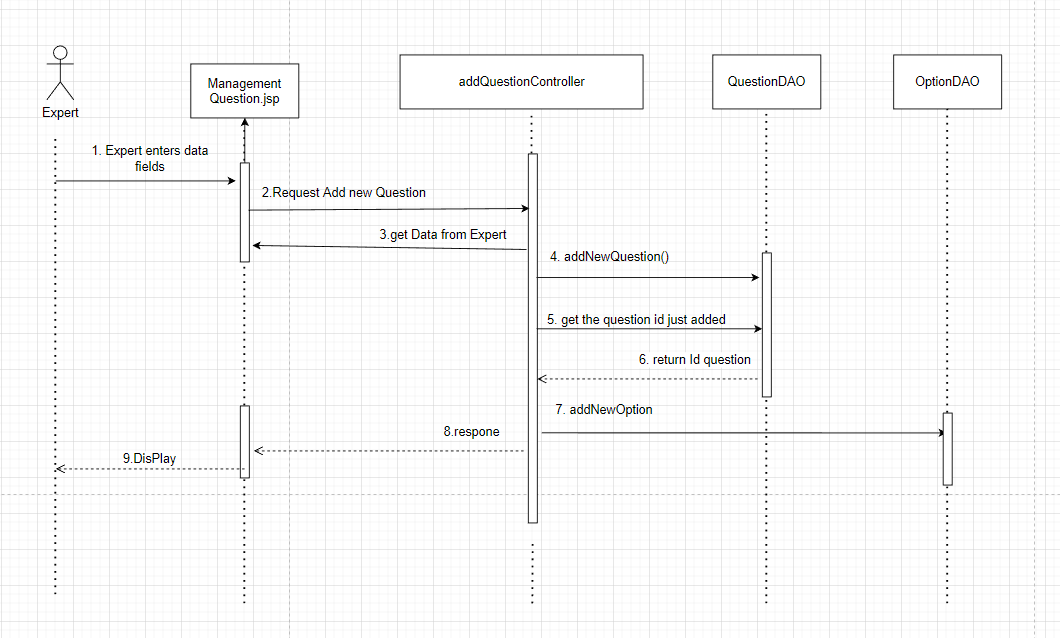
## 

### c. Sequence Diagram(s)

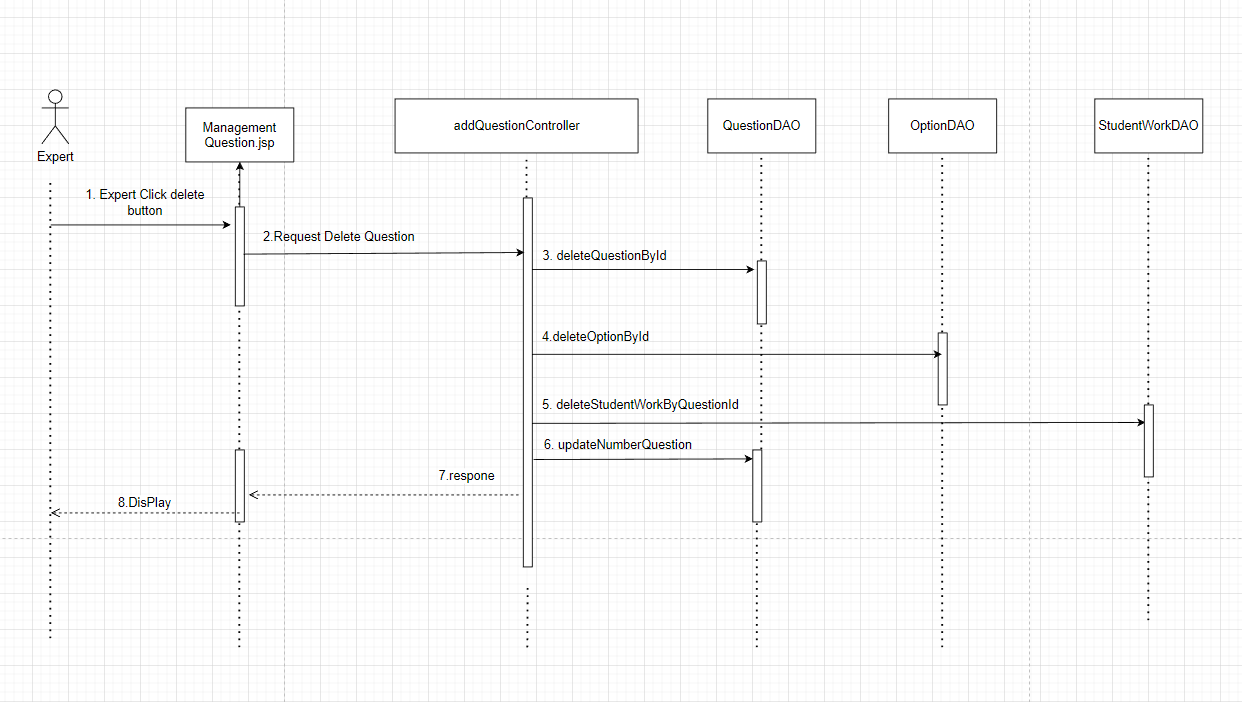
* **Sequence Diagram for View QuestionList**

****

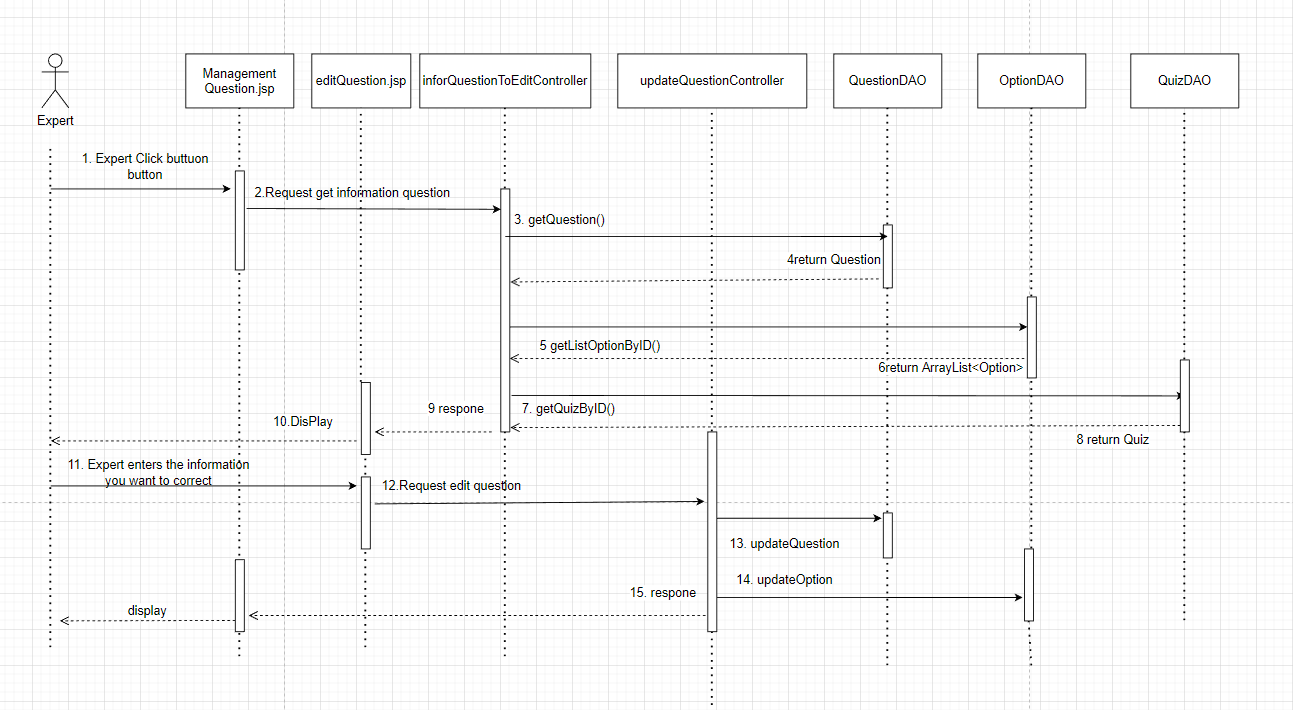
* **Sequence Diagram for Add Question**

****

* **Sequence Diagram for Delete Question**

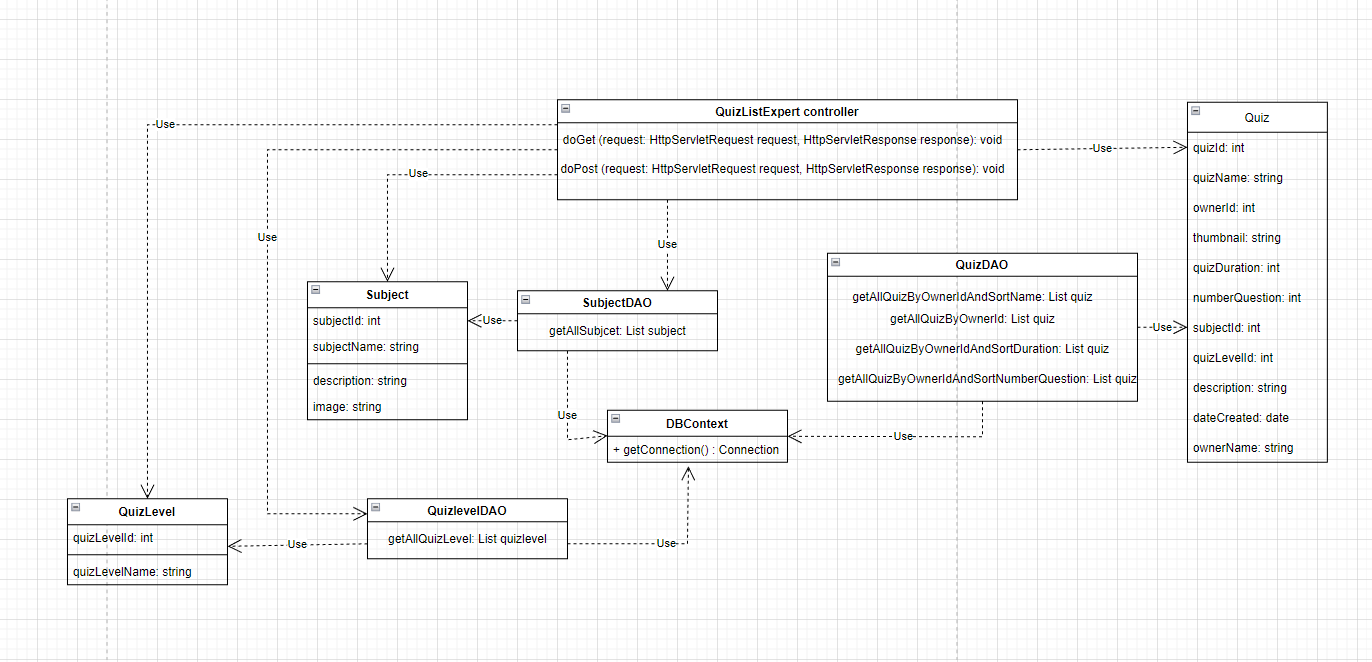
****

* **Sequence Diagram for Edit Question**

****

## 14. Expert/Management Quiz

### a. Class Diagram - View quiz List



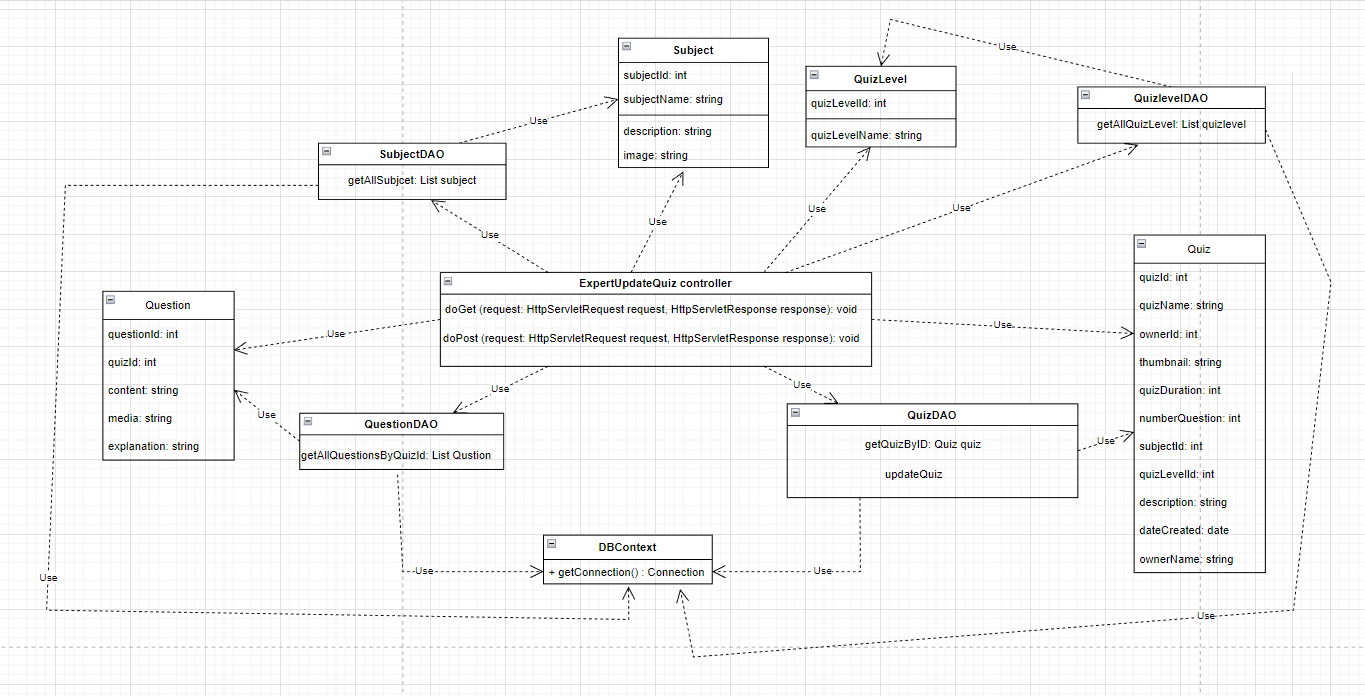
- **Add quiz List**



- **Delete quiz List**

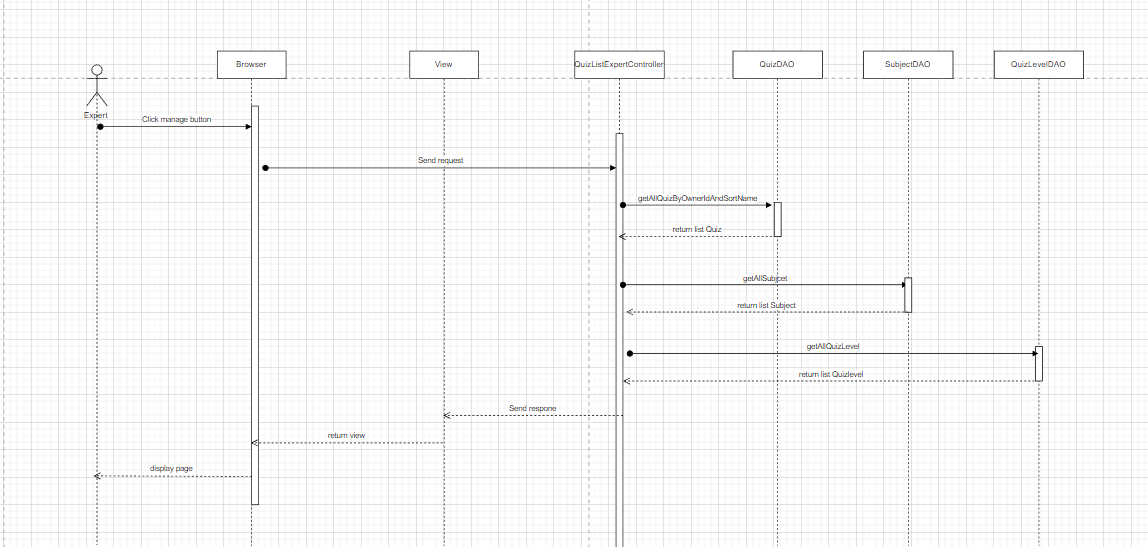
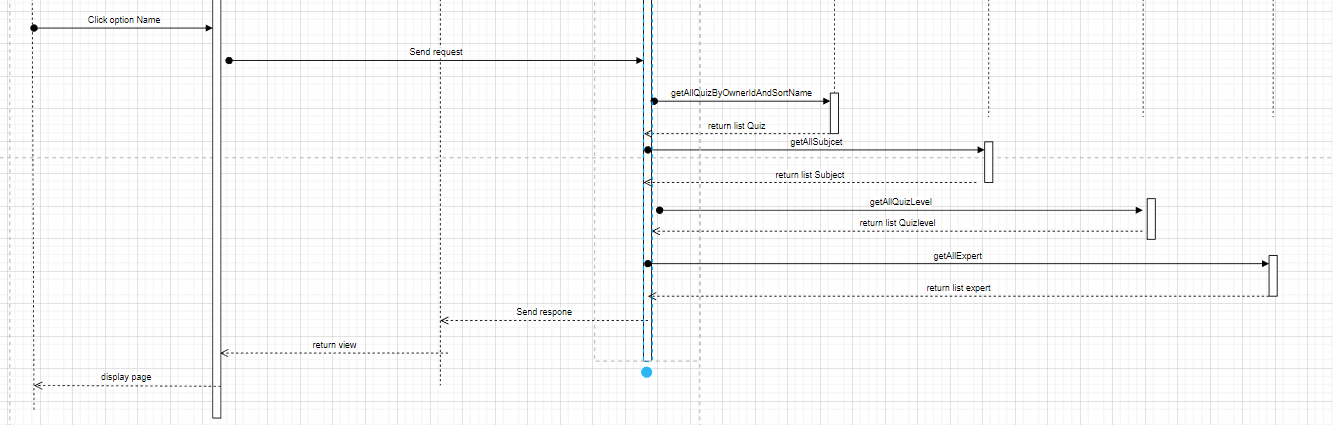
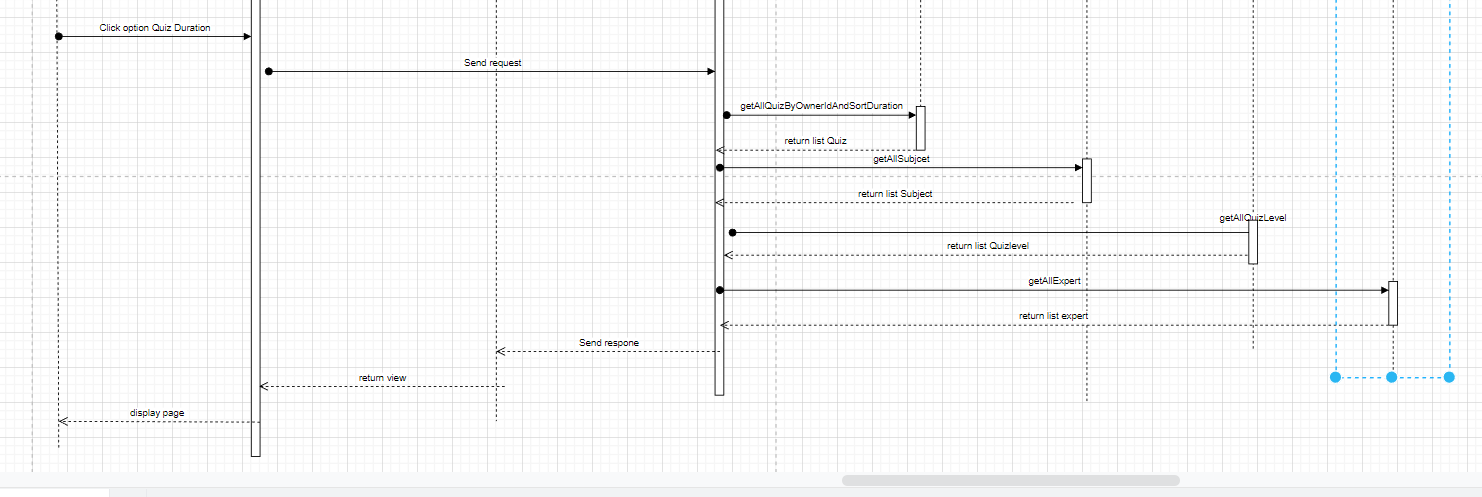
****

- **Update quiz List**

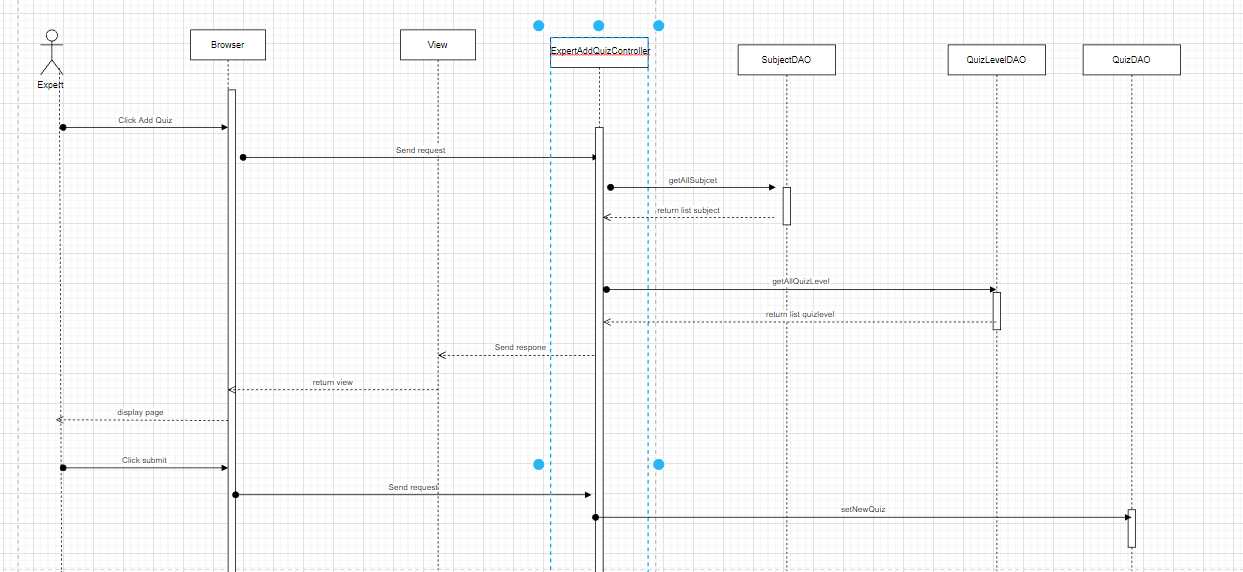
****

### c. Sequence Diagram(s)

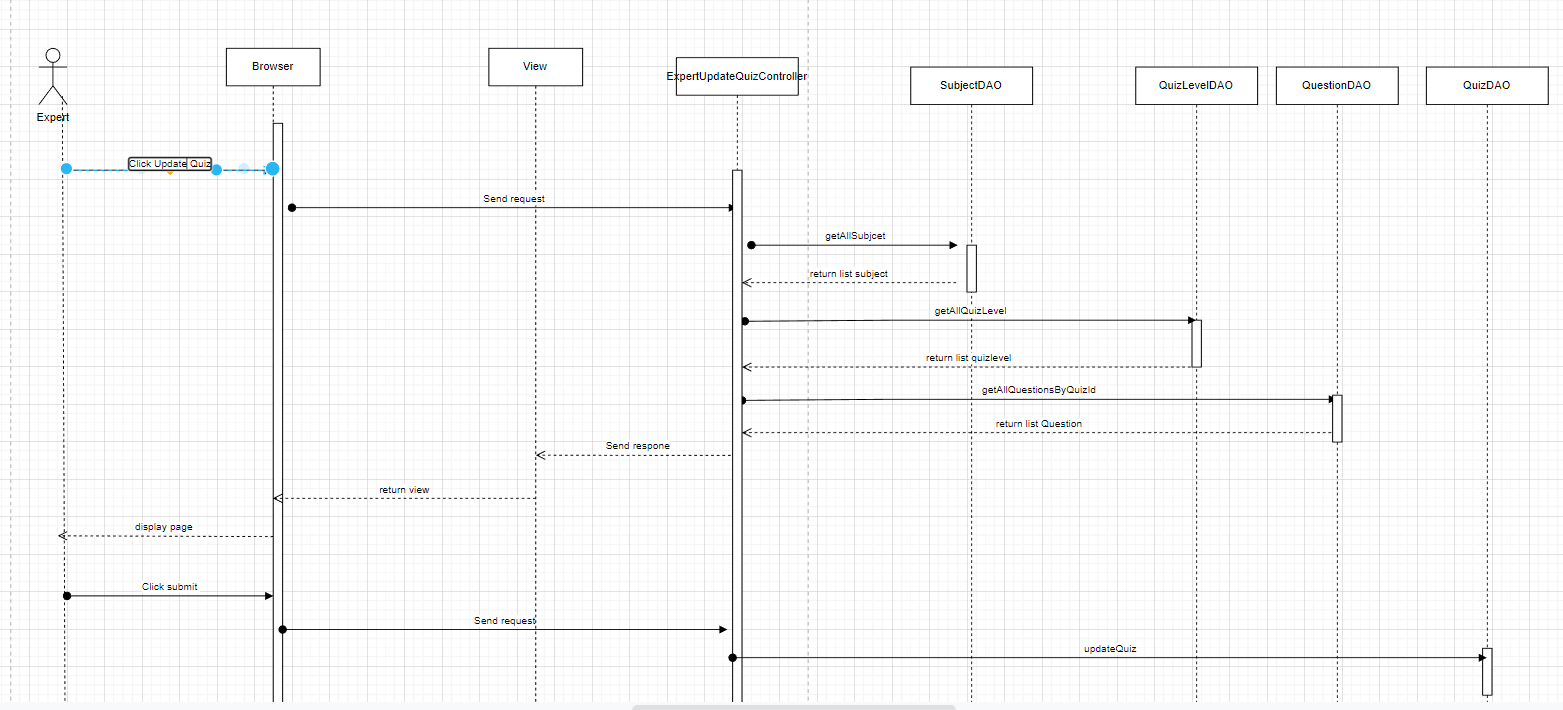
### - View quiz List

* 
* 
* 

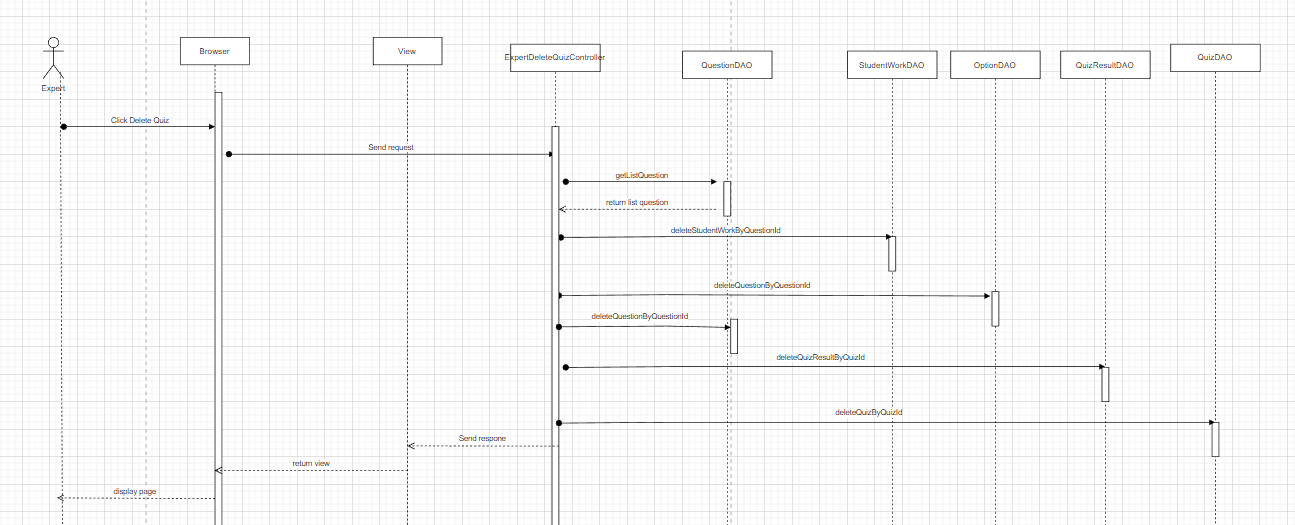
- **Add quiz List**

****

- **Update quiz List**

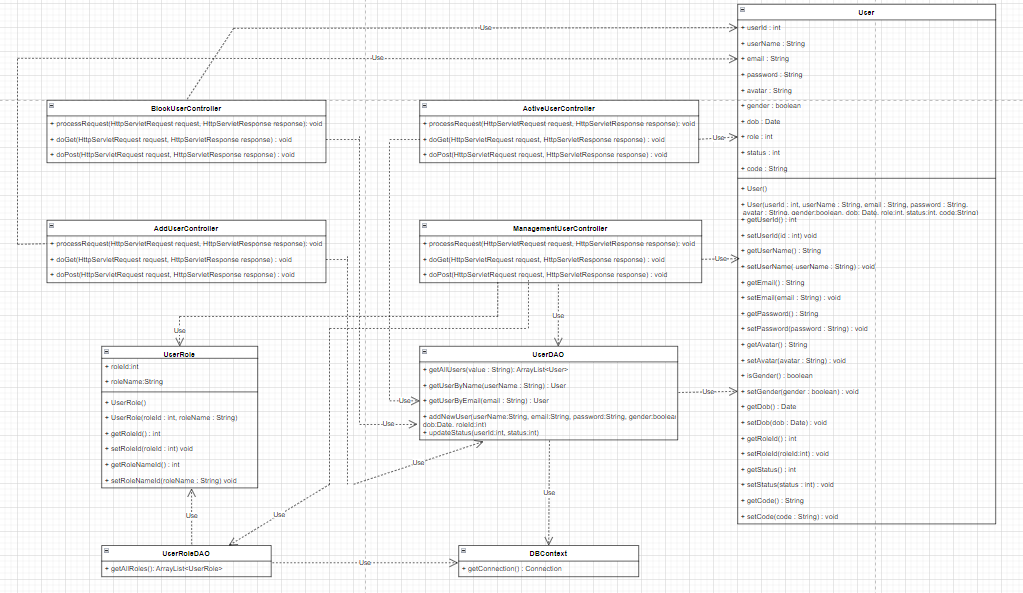
****

- **Delete quiz List**

****

## 15.Admin/Management User

### a. Class Diagram

****  
  
  
-

### b. Class Specifications - ManagementUserController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward user to management user page with list information of user in system .* |

**- AddUserController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Get the value of input fields admin entered and add new user to the system* |

**- ActiveUserController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Change user status from block/not active to active* |

**- BlockUserController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Change user status from active to blocked* |

**-UserDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *QuizResult getQuizResultById(*  *int quizResultId)* | *The input of this method is quizResultId with int type  Return QuizResult Object in table in sql by quizResultId parameter* |

**-UserRoleDAO:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *ArrayList<User> getAllUsers(String value)* | *Return List Role in table in sql* |
| *02* | *User getUserByName( String userName)* | *The input of this method is username with String type  Return User table in sql by username parameter* |
| *03* | *User getUserByEmail( String email)* | *The input of this method is email with String type  Return User table in sql by email parameter* |
| *04* | *void addNewUser(String userName,String email, String password, boolean gender, Date*  *dob, int roleId)* | *Insert new user with parameter are username, email, password with String type, gender type boolean, date type Date, roleId type int and to database* |
| *05* | *void updateStatus(int userId, int status)* | *Update status of user with 2 parameter are userId, status with int type* |

**-User:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes userId, username, email, password , avatar, gender, dob, role, status , code are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

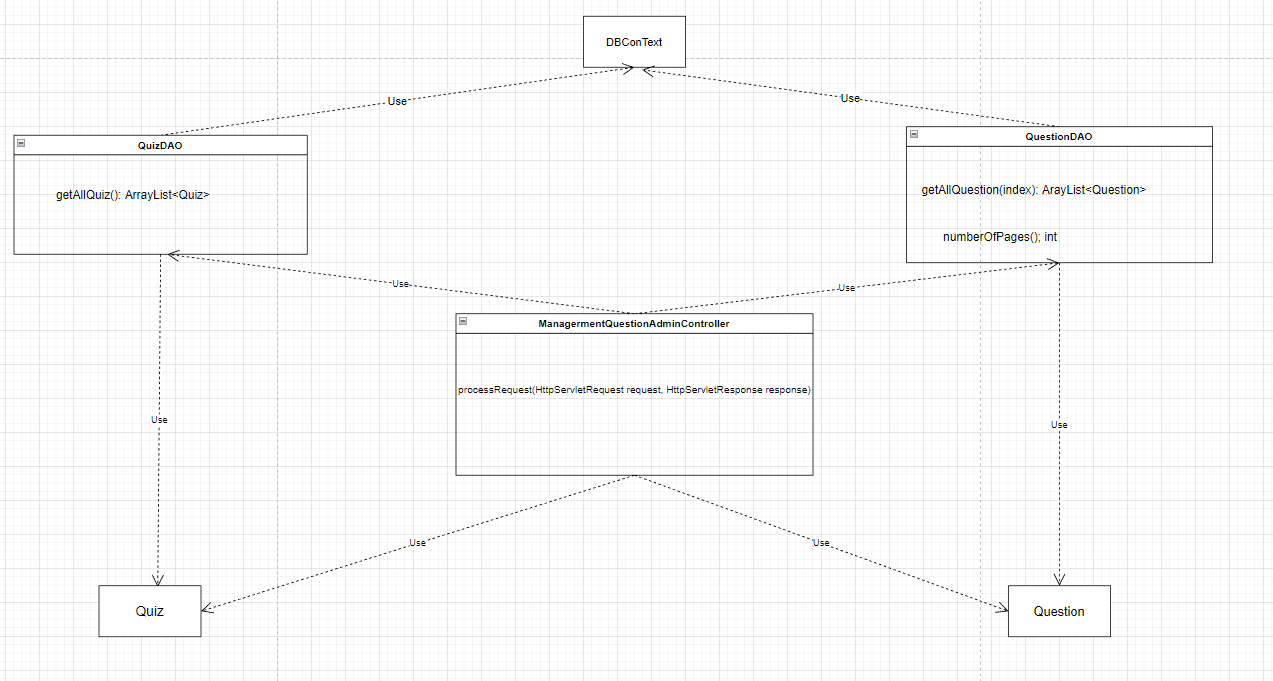
**-UserRole:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *All public Get() method* | *Because all attributes roleId, roleName are private, so these methods use to get the value of each attribute.* |
| *02* | *All public void Set() method* | *We need these methods to set the value of the attribute we want to change.* |

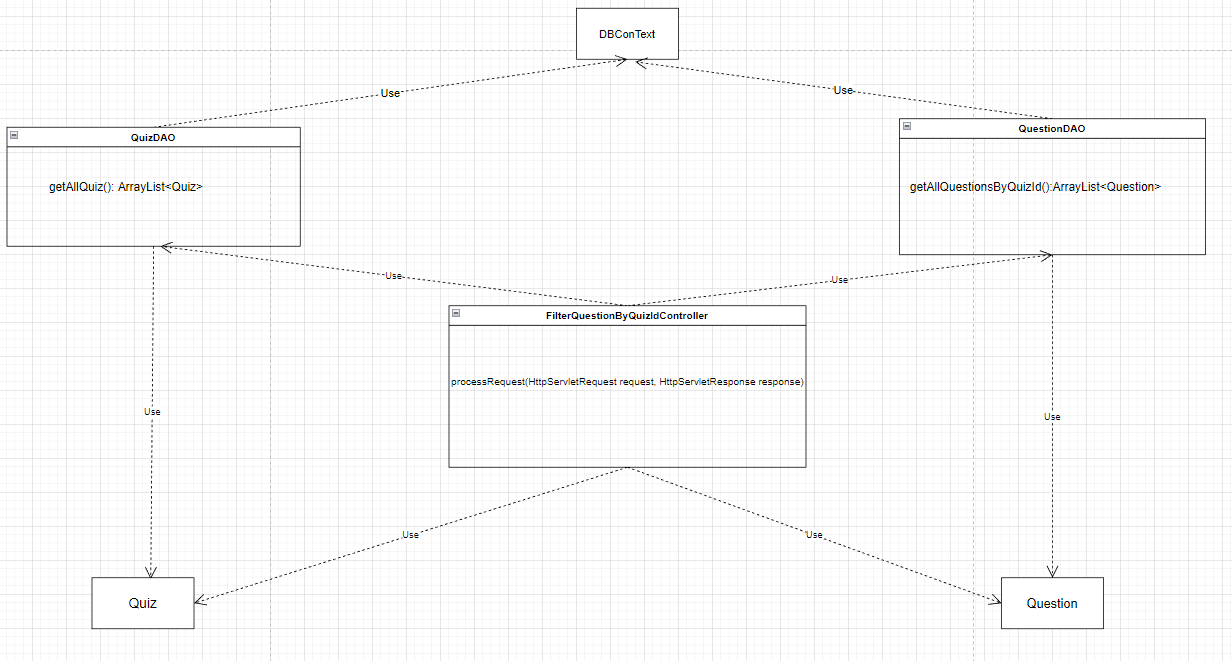
## 16. Admin/Management Question

### a. Class Diagram

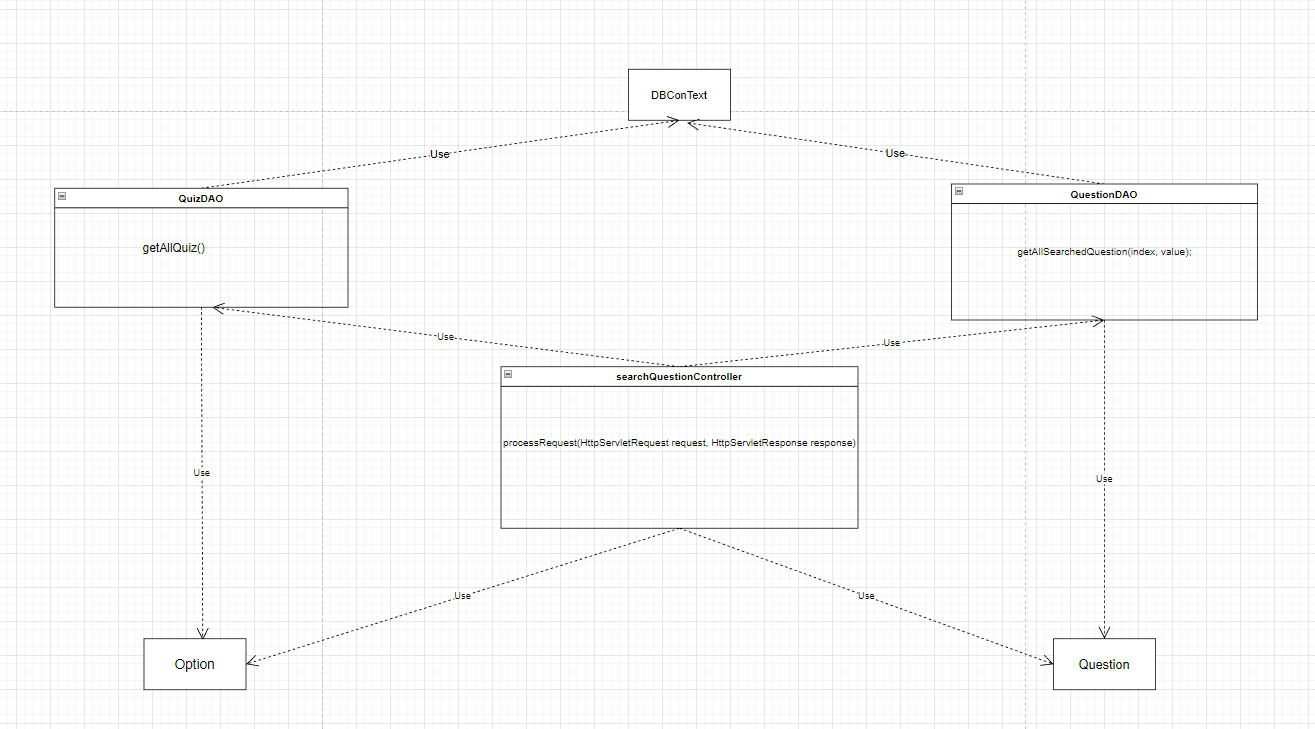
* **Class diagram for Question List**

****

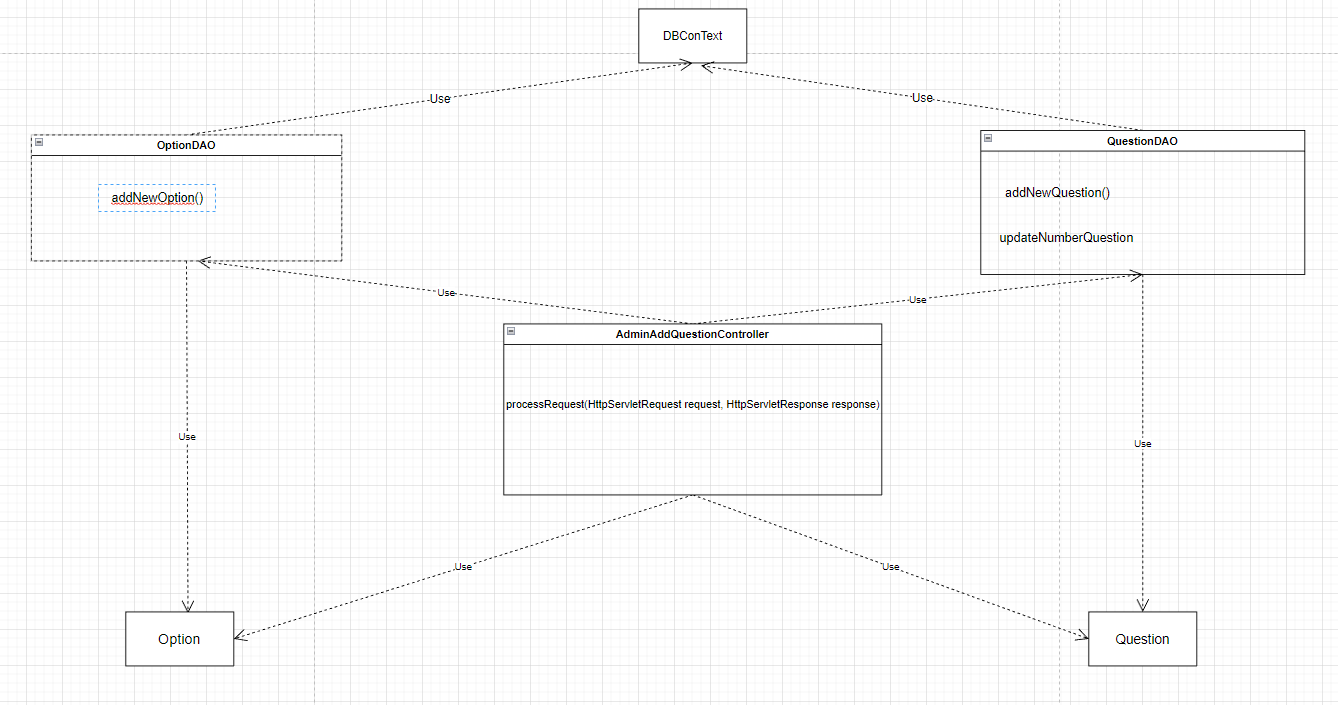
* **Class diagram for Filter question by quiz**

****

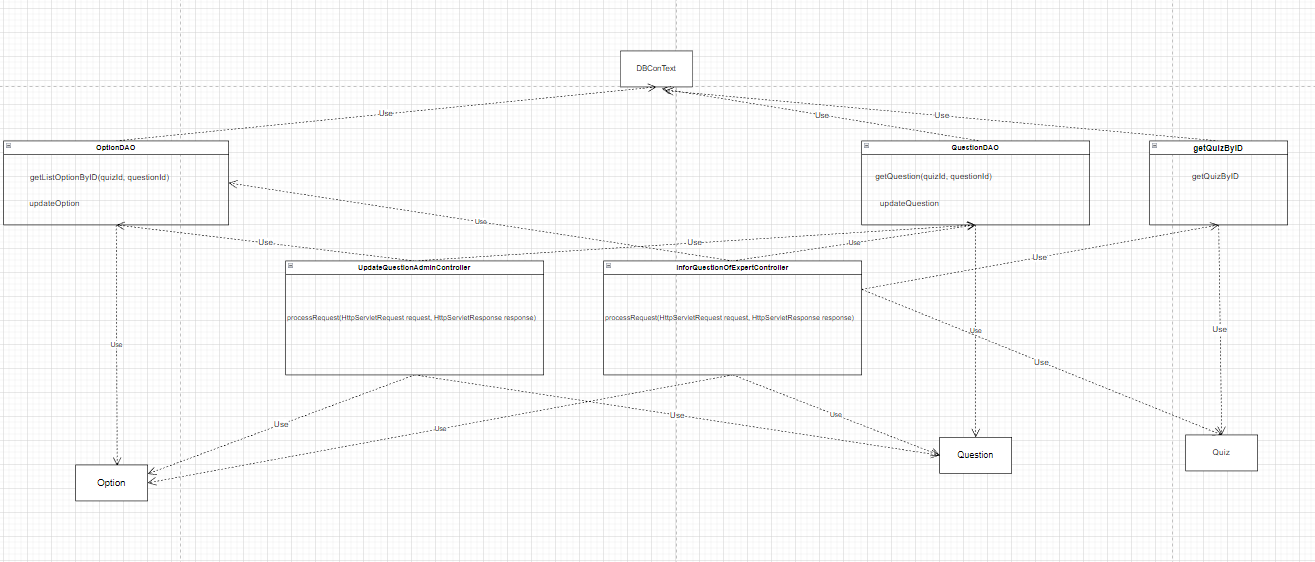
* **Class diagram for Search question by QuestionName**

****

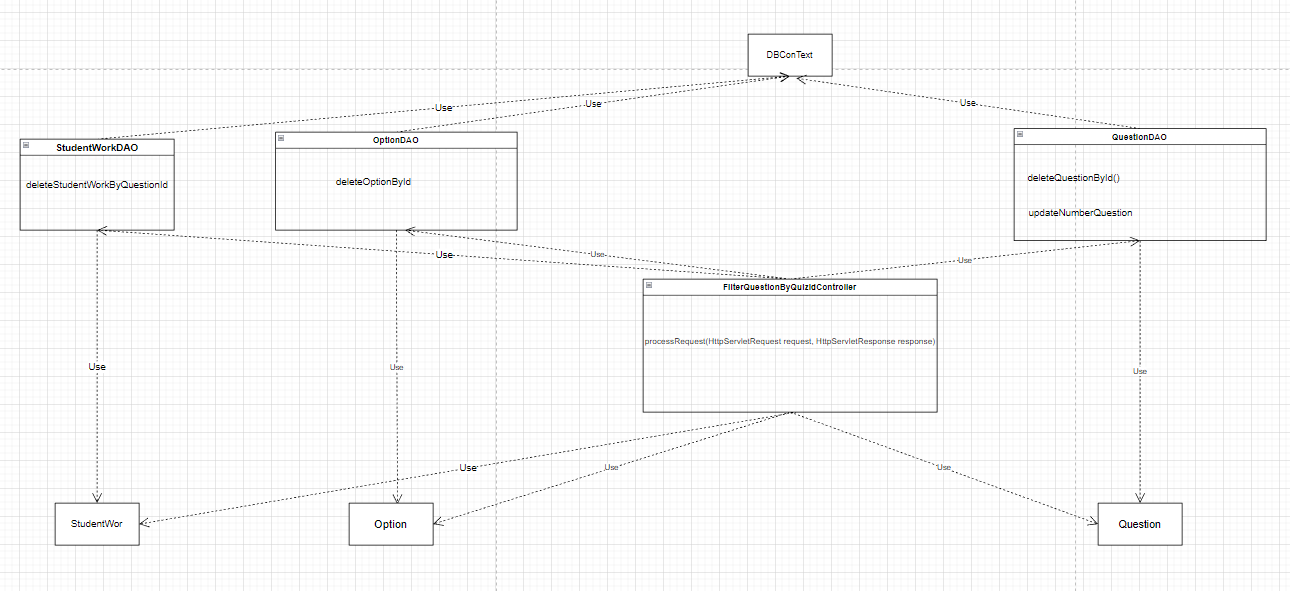
* **Class diagram for Add new question**

****

* **Class diagram for Edit Question**

****

* **class diagram for delete Question**

****

### b. Class Specifications

**- ManagermentQuestionAdminController:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Forward user to management Question page with list information of Question in system .* |

### 

**-AdminAddQuestionController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Get the value of input fields admin entered and add new question to Quiz* |

### 

**UpdateQuestionAdminController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Get the value of input fields admin entered and update question to Quiz* |

### 

**DeleteQuestionAdminController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Delete question to Quiz* |

**-informationQuestionOfExpertController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *Get the information of the question you want to edit and send the admin to the question edit page* |

## 

**-QuizDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getQuizByQuizId(quizId)* | *get Quiz from database* |

## 

**-QuestionDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListQuestion(quizId)* | *get List Question from database* |
| *02* | *getQuestion(quizId,questionId)* | *Get Question from database to Expert* |
| *03* | *updateQuestion(questionId, nameQuesttion,explanation)* | *Update question to database* |
| *04* | *deleteQuestionById(deleteId)* | *delete question to database* |
| *05* | *updateNumberQuestion(numberQ, QuizId);* | *Update amount to database* |

## 

**-OptionDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListOption(quizId)* | *get List Option from database to Expert* |
| *02* | *deleteOptionById(deleteId)* | *delete option to database* |
| *03* | *addNewOption()* | *add option to database* |

## 

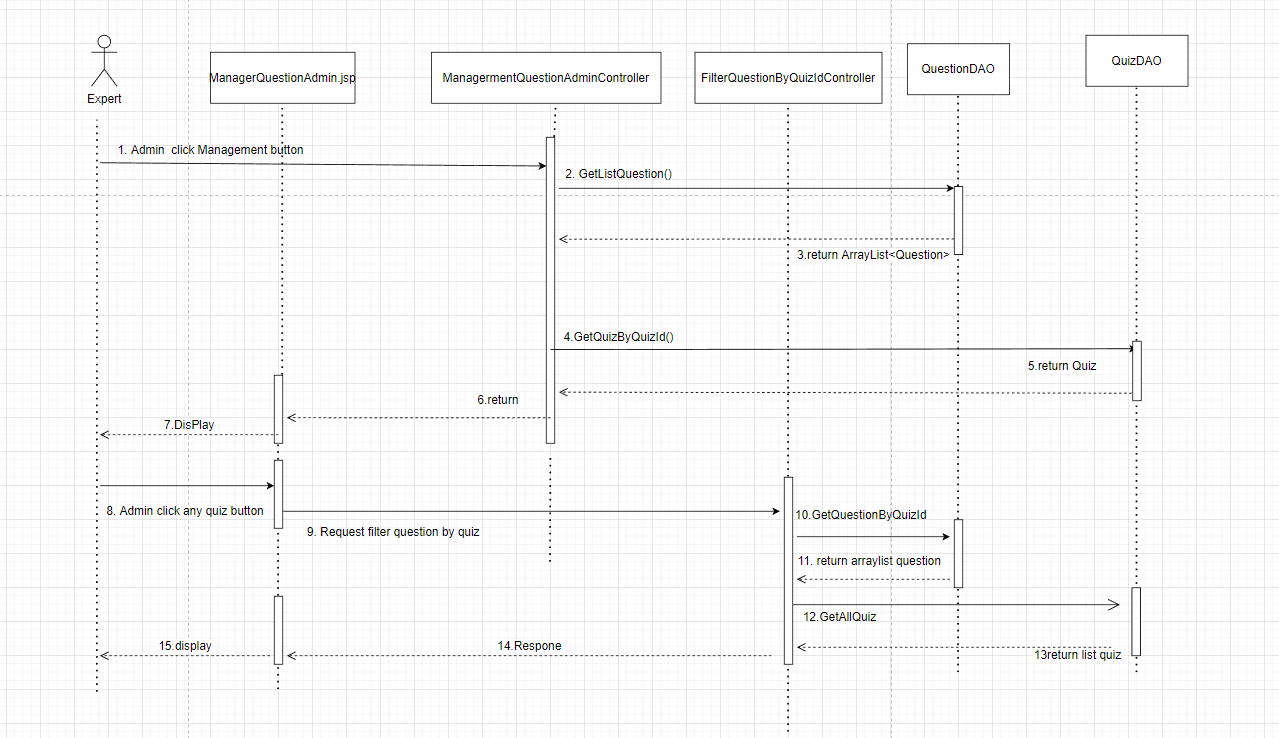
**- StudentWorkDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deleteStudentWorkByQuestionId(deleteId)* | *delta studentwork from database* |

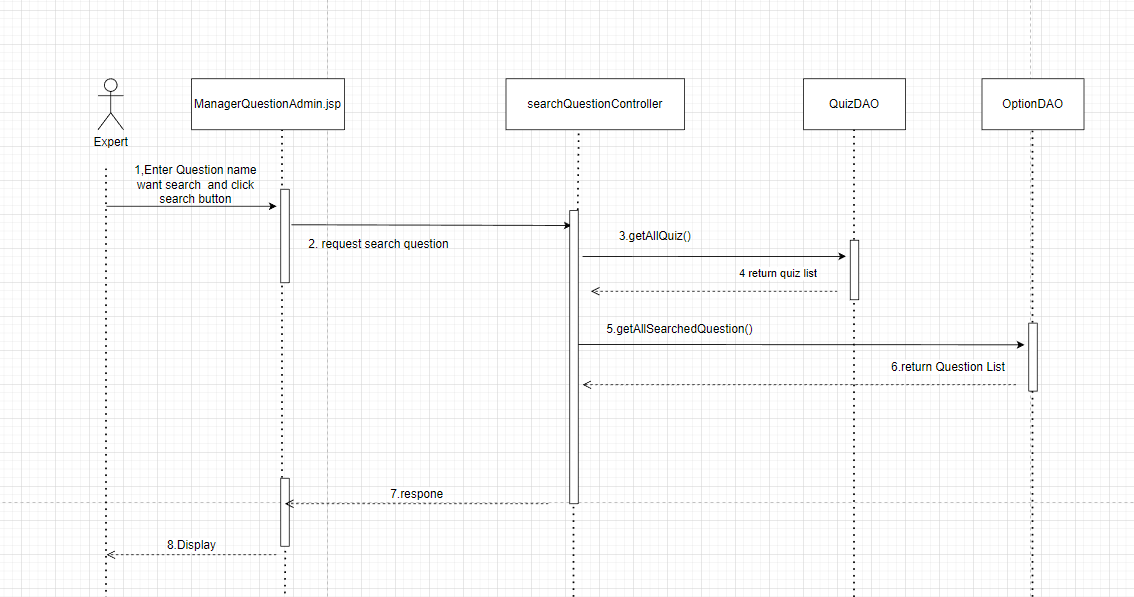
## 

### c. Sequence Diagram(s)

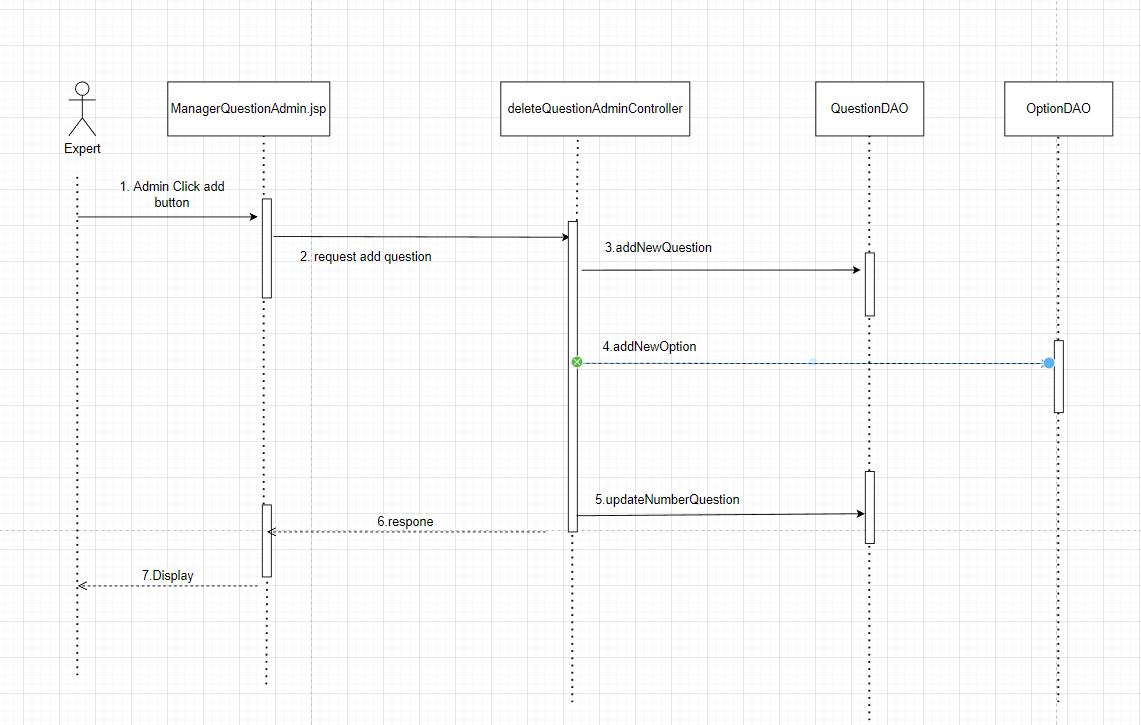
* **Sequence diagram for question list and filter question by quiz**

****

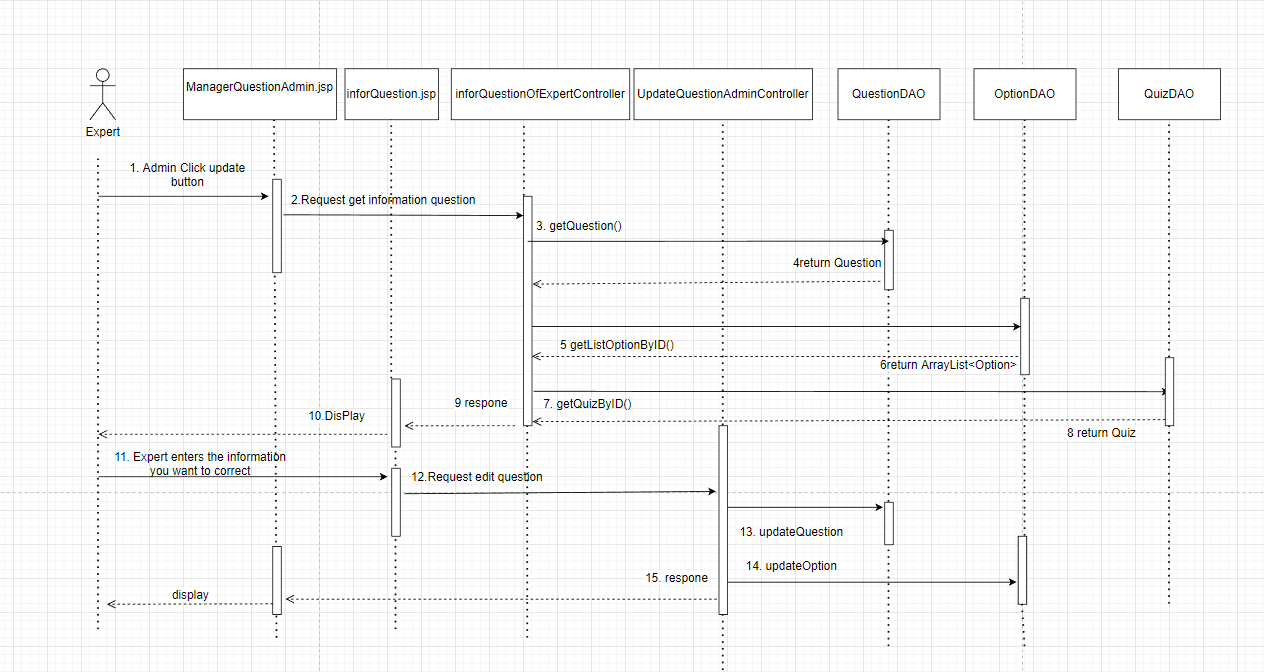
* **sequence diagram for search question**

****

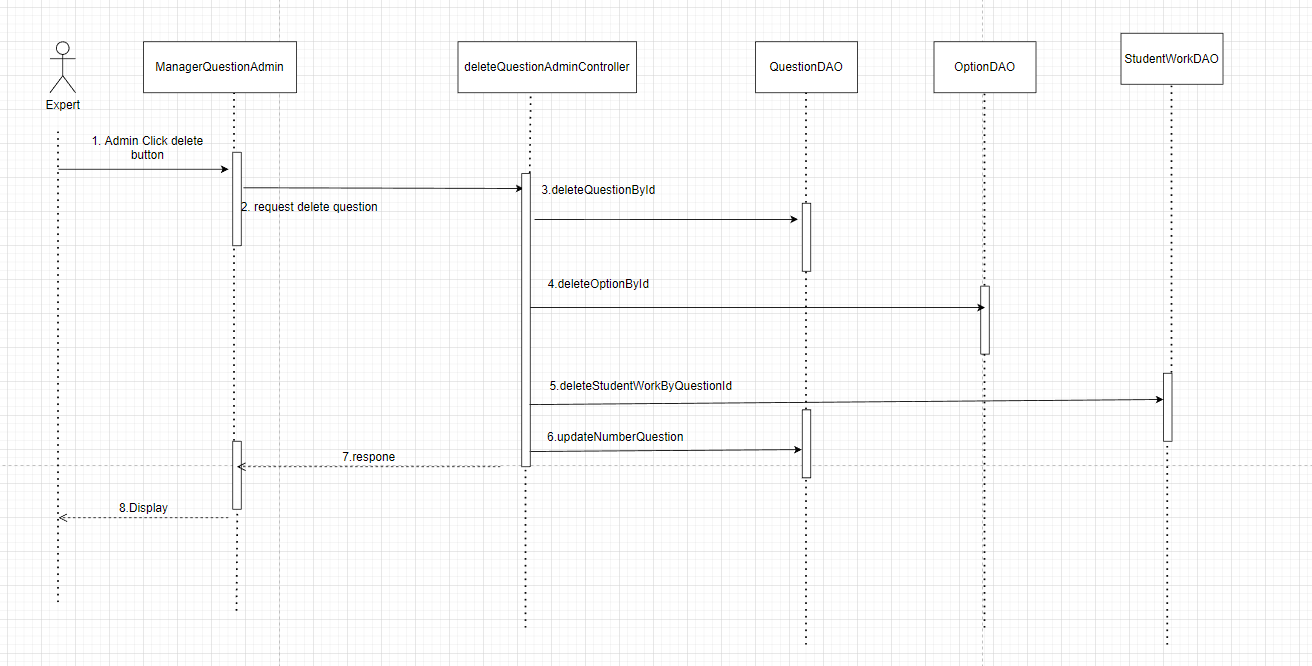
* **sequence diagram for add new question**

****

* **sequence diagram for edit question**

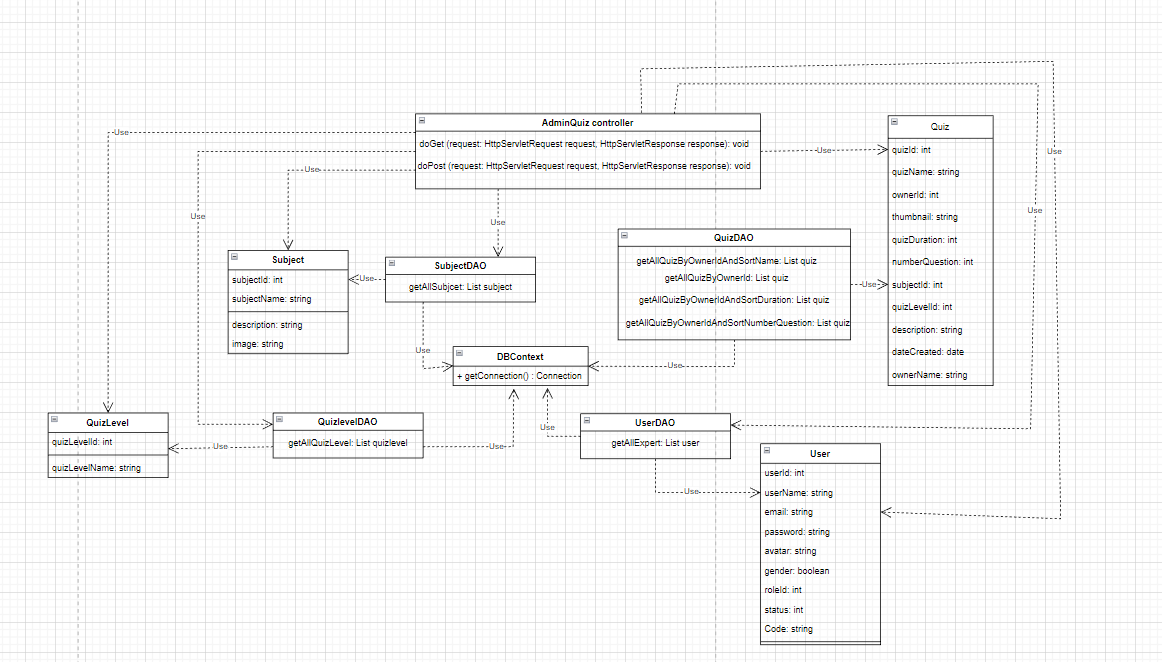
****

* **sequence diagram for delete question**

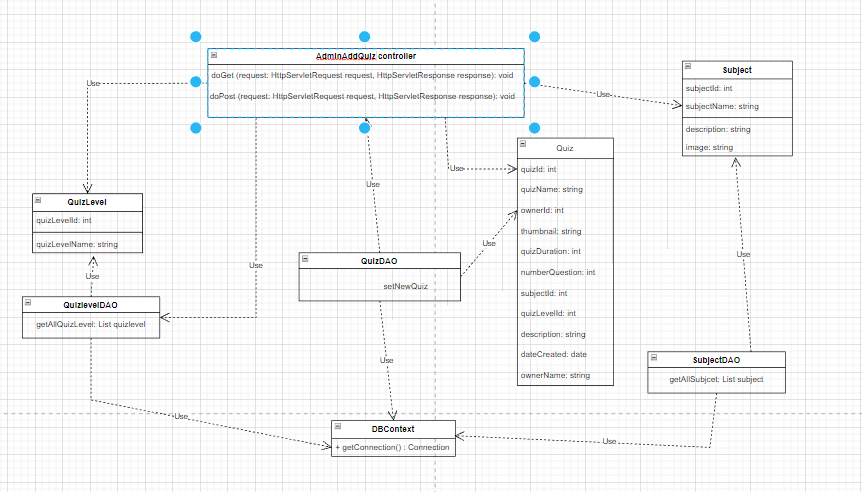
****

## 17. Admin/Management Quiz

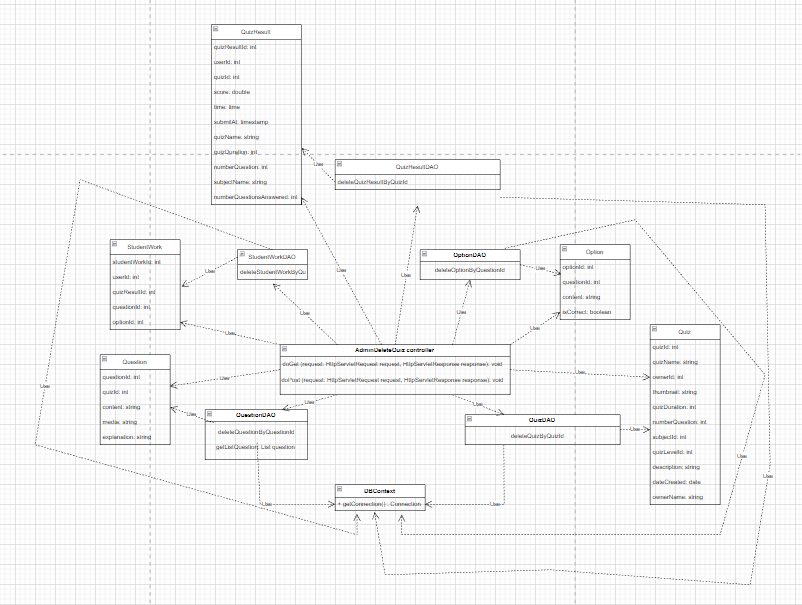
### a. Class Diagram - View quiz List



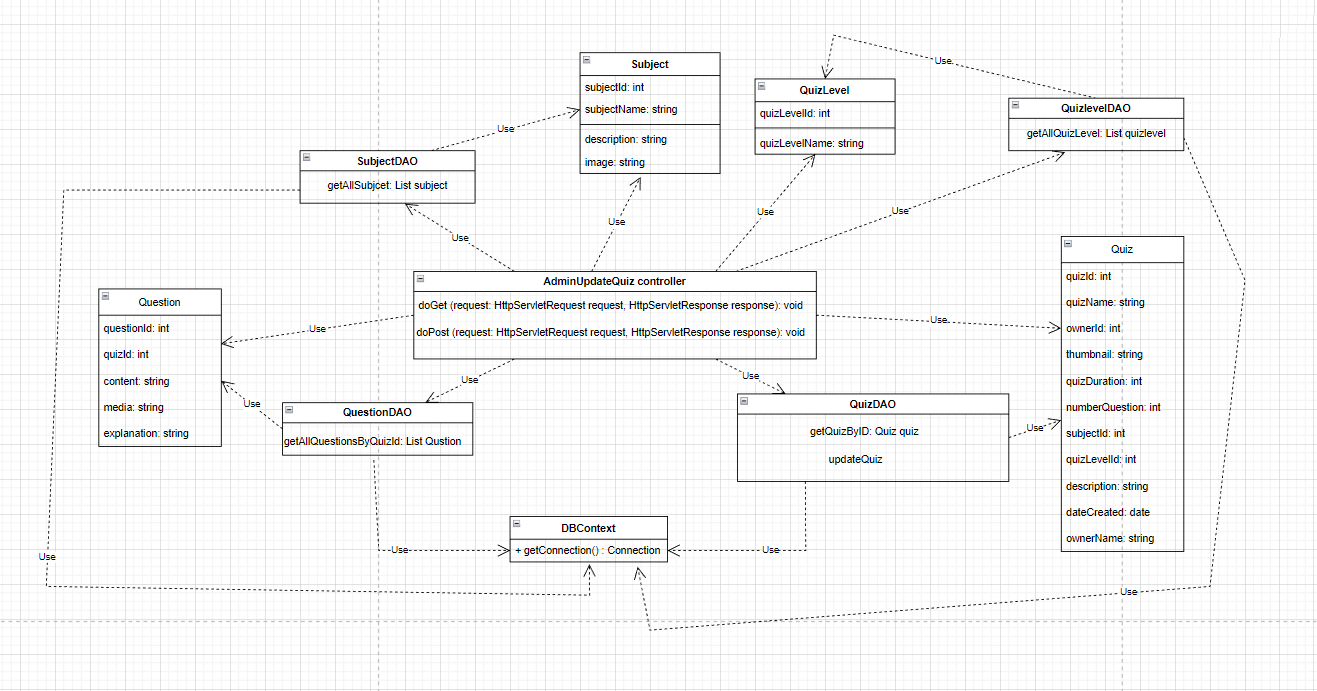
### - Add quiz List



### - Delete quiz List

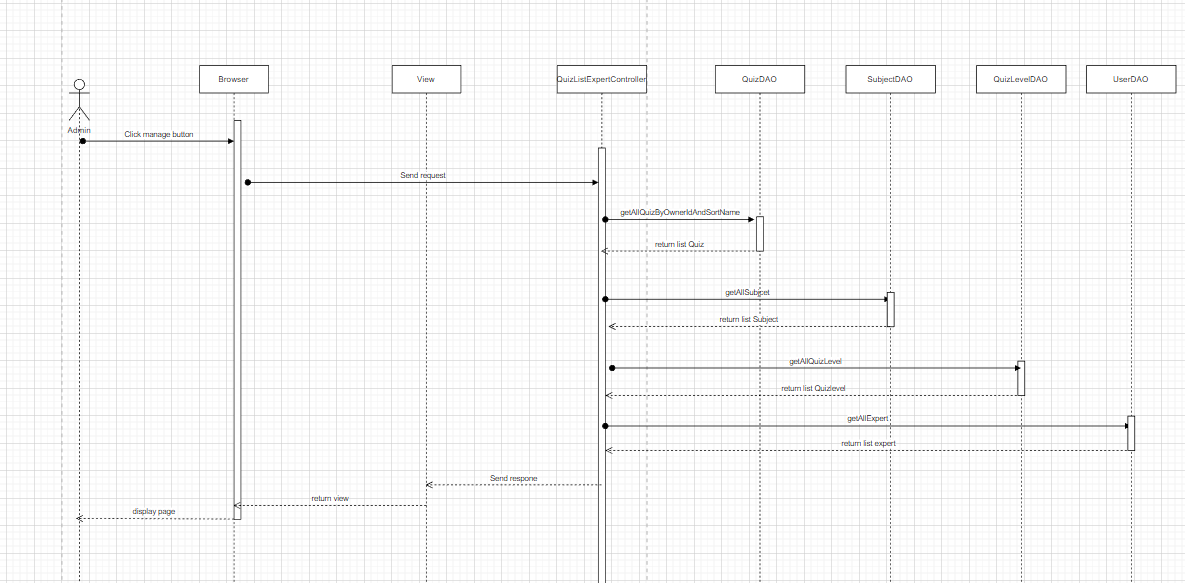


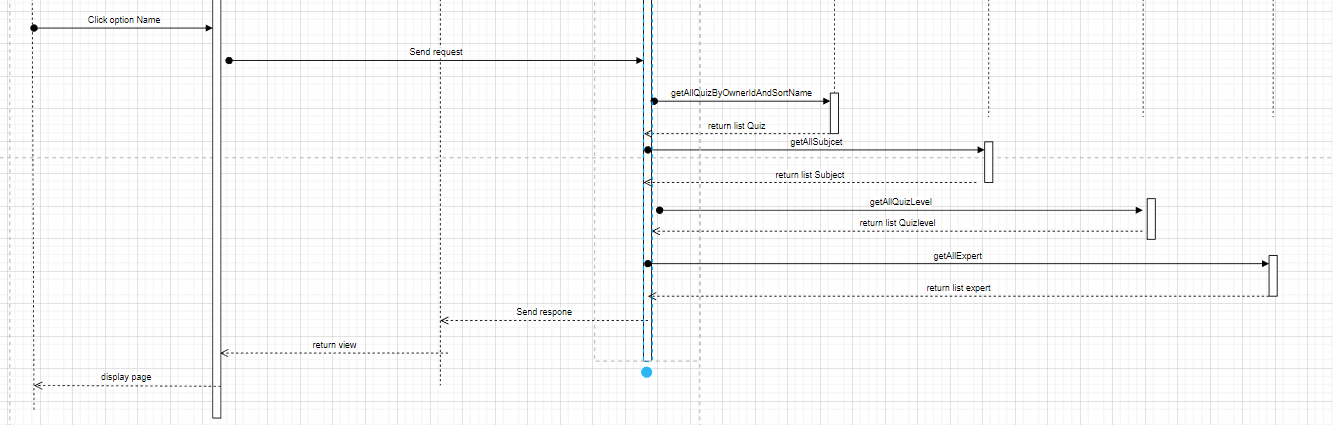
### - Update quiz List

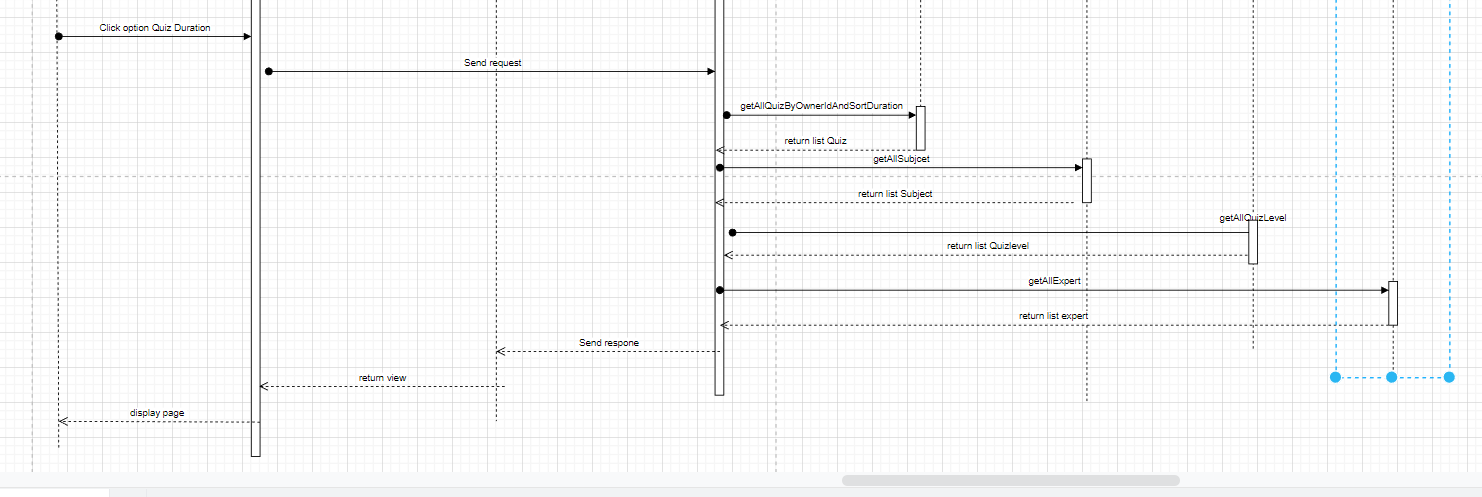


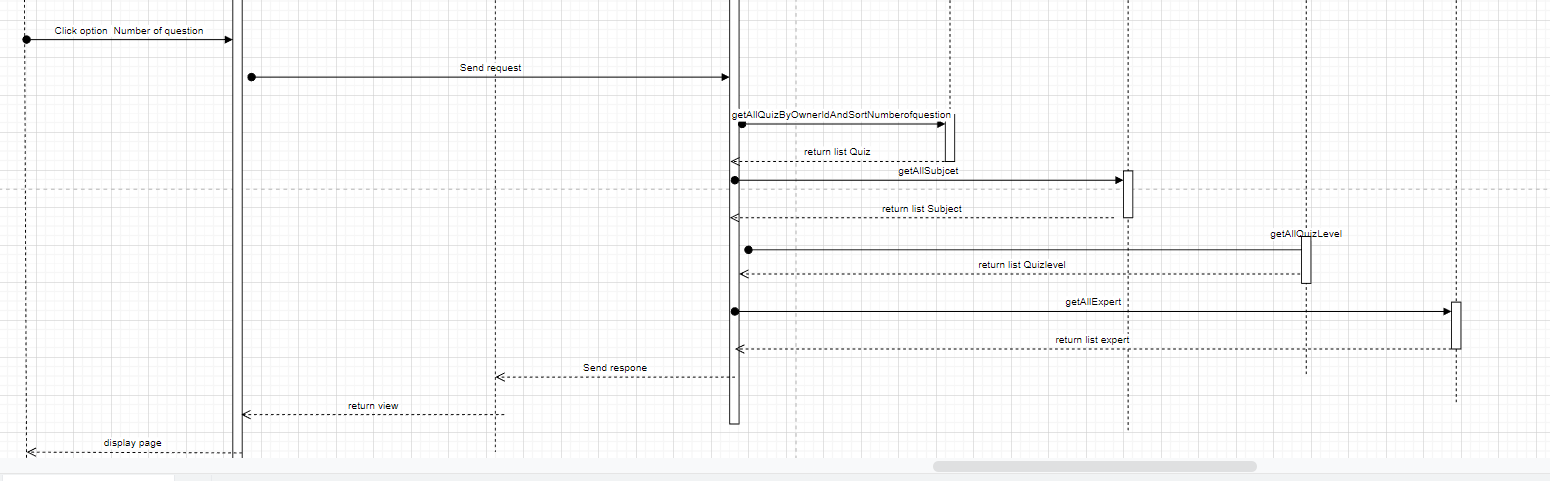
### c. Sequence Diagram(s)

### - View quiz List

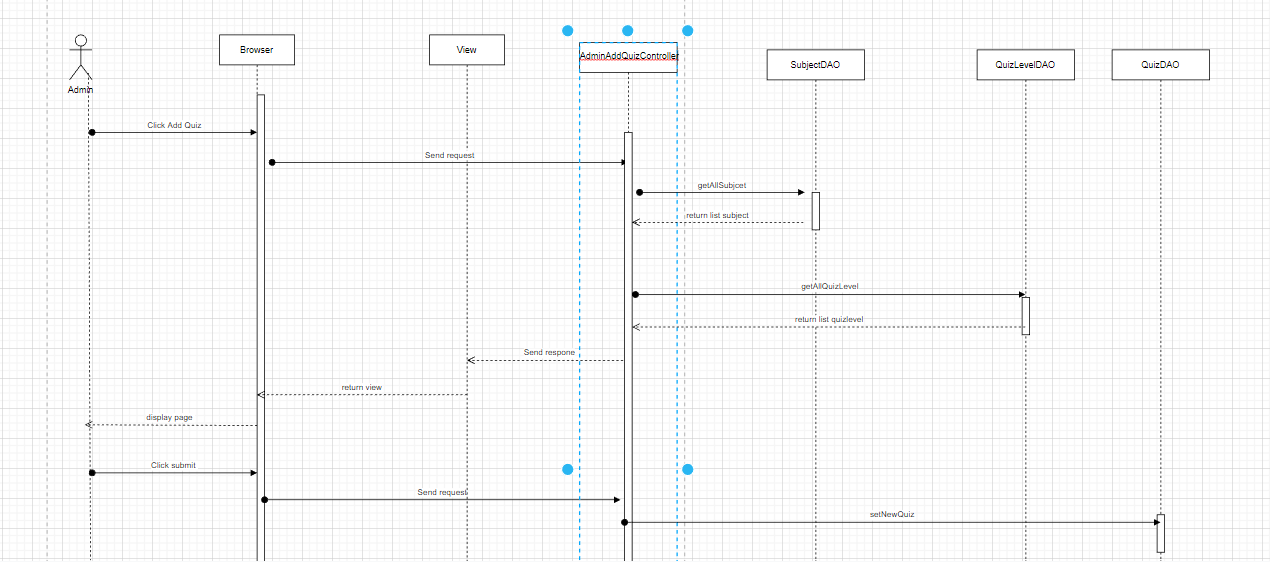




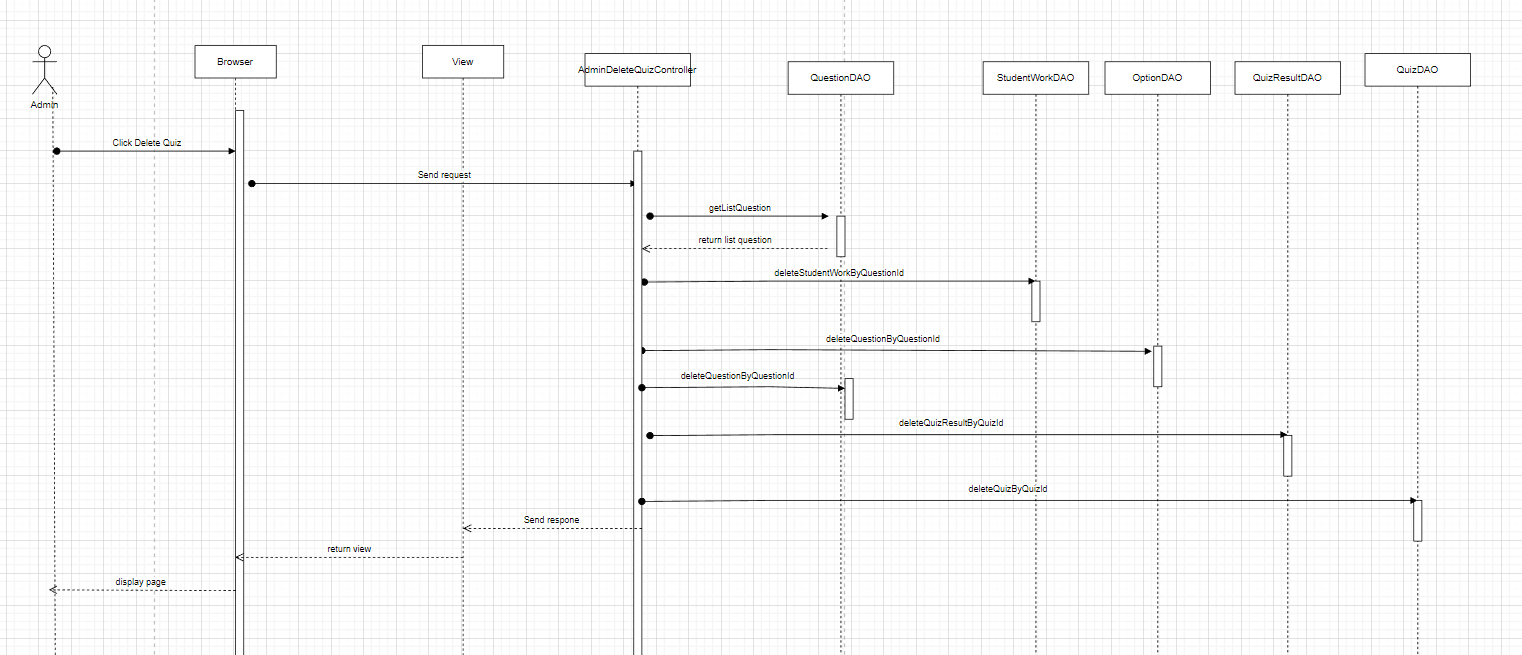




- **Add quiz List**

****

- **Delete quiz List**

****

- **Update quiz List**

