

**ONLINE LEARN**

**Software Design Document**

– Hanoi, November 2021 –

**CLASS SE1509 TEAM 2**

**Team member:**

* Bui Hoang Phong
* Nguyen Duc Cuong
* Do Ngoc Hoang Anh
* Nguyen Thi Thanh Mai (Leader)
* Nguyen Cong Quan

**Table of Contents**

**Contents**

[I. Overview 9](#_heading=h.35nkun2)

[1. Code Packages/Namespaces 9](#_heading=h.1ksv4uv)

[2. Database Schema 10](#_heading=h.44sinio)

[II. Code Designs 12](#_heading=h.2jxsxqh)

[**1.**User Login 12](#_heading=h.z337ya)

[a. Class Diagram: 12](#_heading=h.3j2qqm3)

[b. Class Specifications 12](#_heading=h.3mj2wkv)

[c. Sequence Diagram(s) 13](#_heading=h.1y810tw)

[d. Database queries : 13](#_heading=h.4i7ojhp)

[**2.**User Register 13](#_heading=h.2xcytpi)

[a. Class Diagram 13](#_heading=h.1ci93xb)

[b. Class Specifications 13](#_heading=h.3whwml4)

[c. Sequence Diagram(s) 14](#_heading=h.2bn6wsx)

[d. Database queries 14](#_heading=h.qsh70q)

[**3.**Reset Password 15](#_heading=h.3as4poj)

[a. Class Diagram 15](#_heading=h.1pxezwc)

[b. Class Specifications 15](#_heading=h.49x2ik5)

[c. Sequence Diagram(s) 15](#_heading=h.2p2csry)

[d. Database queries 16](#_heading=h.147n2zr)

[**4.**User Authorization 17](#_heading=h.3o7alnk)

[a. Class Diagram 17](#_heading=h.23ckvvd)

[b. Class Specifications 17](#_heading=h.ihv636)

[c. Sequence Diagram(s) 17](#_heading=h.32hioqz)

[d. Database queries 17](#_heading=h.1hmsyys)

[**5.**User Profile 17](#_heading=h.41mghml)

[a. Class Diagram 17](#_heading=h.2grqrue)

[b. Class Specifications 17](#_heading=h.vx1227)

[c. Sequence Diagram(s) 17](#_heading=h.3fwokq0)

[d. Database queries 18](#_heading=h.1v1yuxt)

[**6.**Change Password 18](#_heading=h.4f1mdlm)

[a. Class Diagram 18](#_heading=h.2u6wntf)

[b. Class Specifications 18](#_heading=h.19c6y18)

[c. Sequence Diagram(s) 18](#_heading=h.3tbugp1)

[d. Database queries 18](#_heading=h.28h4qwu)

[**7.**Home 19](#_heading=h.nmf14n)

[a. Class Diagram 19](#_heading=h.37m2jsg)

[b. Class Specifications 19](#_heading=h.1mrcu09)

[b. Class Specifications 19](#_heading=h.xcbuftfao1qz)

[c. Sequence Diagram(s) 20](#_heading=h.46r0co2)

[d. Database queries 20](#_heading=h.2lwamvv)

[**8.**Blogs List 21](#_heading=h.111kx3o)

[a. Class Diagram 21](#_heading=h.3l18frh)

[b. Class Specifications 21](#_heading=h.206ipza)

[c. Sequence Diagram(s) 22](#_heading=h.4k668n3)

[d. Database queries 22](#_heading=h.2zbgiuw)

[**9.**Blog Details 23](#_heading=h.1egqt2p)

[a. Class Diagram 23](#_heading=h.3ygebqi)

[b. Class Specifications 23](#_heading=h.2dlolyb)

[c. Sequence Diagram(s) 23](#_heading=h.sqyw64)

[d. Database queries 23](#_heading=h.3cqmetx)

[**10.**Courses List 23](#_heading=h.1rvwp1q)

[a. Class Diagram 23](#_heading=h.4bvk7pj)

[b. Class Specifications 23](#_heading=h.2r0uhxc)

[c. Sequence Diagram(s) 23](#_heading=h.1664s55)

[d. Database queries 23](#_heading=h.3q5sasy)

[**11.**Course Details 23](#_heading=h.25b2l0r)

[a. Class Diagram 23](#_heading=h.kgcv8k)

[b. Class Specifications 23](#_heading=h.34g0dwd)

[c. Sequence Diagram(s) 23](#_heading=h.1jlao46)

[d. Database queries 23](#_heading=h.43ky6rz)

[**12.**Course Register 23](#_heading=h.2iq8gzs)

[a. Class Diagram 23](#_heading=h.xvir7l)

[b. Class Specifications 23](#_heading=h.3hv69ve)

[c. Sequence Diagram(s) 23](#_heading=h.1x0gk37)

[d. Database queries 23](#_heading=h.4h042r0)

[13.My Registrations 23](#_heading=h.2w5ecyt)

[a. Class Diagram 23](#_heading=h.1baon6m)

[b. Class Specifications 23](#_heading=h.3vac5uf)

[c. Sequence Diagram(s) 23](#_heading=h.2afmg28)

[d. Database queries 23](#_heading=h.pkwqa1)

[**14.**My Courses 24](#_heading=h.39kk8xu)

[a. Class Diagram 24](#_heading=h.1opuj5n)

[b. Class Specifications 24](#_heading=h.48pi1tg)

[c. Sequence Diagram(s) 24](#_heading=h.2nusc19)

[d. Database queries 24](#_heading=h.1302m92)

[**15.**Lesson View 24](#_heading=h.3mzq4wv)

[a. Class Diagram 24](#_heading=h.2250f4o)

[b. Class Specifications 24](#_heading=h.haapch)

[c. Sequence Diagram(s) 24](#_heading=h.319y80a)

[d. Database queries 24](#_heading=h.1gf8i83)

[**16.**Quiz Lesson 24](#_heading=h.40ew0vw)

[a. Class Diagram 24](#_heading=h.2fk6b3p)

[b. Class Specifications 24](#_heading=h.upglbi)

[c. Sequence Diagram(s) 24](#_heading=h.3ep43zb)

[d. Database queries 24](#_heading=h.1tuee74)

[**17.**Quiz Handle 24](#_heading=h.4du1wux)

[a. Class Diagram 24](#_heading=h.2szc72q)

[b. Class Specifications 24](#_heading=h.184mhaj)

[c. Sequence Diagram(s) 24](#_heading=h.3s49zyc)

[d. Database queries 24](#_heading=h.279ka65)

[**18.**Quiz Review 24](#_heading=h.meukdy)

[a. Class Diagram 24](#_heading=h.36ei31r)

[b. Class Specifications 24](#_heading=h.1ljsd9k)

[c. Sequence Diagram(s) 24](#_heading=h.45jfvxd)

[d. Database queries 24](#_heading=h.2koq656)

[**19.**Dashboard 24](#_heading=h.zu0gcz)

[a. Class Diagram 24](#_heading=h.3jtnz0s)

[b. Class Specifications 24](#_heading=h.1yyy98l)

[c. Sequence Diagram(s) 24](#_heading=h.4iylrwe)

[d. Database queries 24](#_heading=h.2y3w247)

[**20.**Post Details 25](#_heading=h.1d96cc0)

[a. Class Diagram 25](#_heading=h.3x8tuzt)

[b. Class Specifications 25](#_heading=h.2ce457m)

[c. Sequence Diagram(s) 25](#_heading=h.rjefff)

[d. Database queries 25](#_heading=h.3bj1y38)

[**21.**Sliders List 25](#_heading=h.1qoc8b1)

[a. Class Diagram 25](#_heading=h.4anzqyu)

[b. Class Specifications 25](#_heading=h.2pta16n)

[c. Sequence Diagram(s) 25](#_heading=h.14ykbeg)

[d. Database queries 25](#_heading=h.3oy7u29)

[**22.**Slider Details 25](#_heading=h.243i4a2)

[a. Class Diagram 25](#_heading=h.j8sehv)

[b. Class Specifications 25](#_heading=h.338fx5o)

[c. Sequence Diagram(s) 25](#_heading=h.1idq7dh)

[d. Database queries 25](#_heading=h.42ddq1a)

[**23.**Subjects List 25](#_heading=h.2hio093)

[a. Class Diagram 25](#_heading=h.wnyagw)

[b. Class Specifications 25](#_heading=h.3gnlt4p)

[c. Sequence Diagram(s) 26](#_heading=h.1vsw3ci)

[d. Database queries 26](#_heading=h.4fsjm0b)

[**24.**New Subject 27](#_heading=h.2uxtw84)

[a. Class Diagram 27](#_heading=h.1a346fx)

[b. Class Specifications 27](#_heading=h.3u2rp3q)

[c. Sequence Diagram(s) 27](#_heading=h.2981zbj)

[d. Database queries 27](#_heading=h.odc9jc)

[**25.**Subject Details 27](#_heading=h.38czs75)

[a. Class Diagram 27](#_heading=h.1nia2ey)

[b. Class Specifications 27](#_heading=h.47hxl2r)

[c. Sequence Diagram(s) 27](#_heading=h.2mn7vak)

[d. Database queries 27](#_heading=h.11si5id)

[**26.**Subject Dimension 27](#_heading=h.3ls5o66)

[a. Class Diagram 27](#_heading=h.20xfydz)

[b. Class Specifications 27](#_heading=h.4kx3h1s)

[c. Sequence Diagram(s) 27](#_heading=h.302dr9l)

[d. Database queries 27](#_heading=h.1f7o1he)

[**27.**Price Package 27](#_heading=h.3z7bk57)

[a. Class Diagram 27](#_heading=h.2eclud0)

[b. Class Specifications 27](#_heading=h.thw4kt)

[c. Sequence Diagram(s) 27](#_heading=h.3dhjn8m)

[d. Database queries 27](#_heading=h.1smtxgf)

[**28.**Subject Lessons 27](#_heading=h.4cmhg48)

[a. Class Diagram 27](#_heading=h.2rrrqc1)

[b. Class Specifications 27](#_heading=h.16x20ju)

[c. Sequence Diagram(s) 27](#_heading=h.3qwpj7n)

[d. Database queries 27](#_heading=h.261ztfg)

[**29.**Lesson Details 27](#_heading=h.l7a3n9)

[a. Class Diagram 27](#_heading=h.356xmb2)

[b. Class Specifications 27](#_heading=h.1kc7wiv)

[c. Sequence Diagram(s) 27](#_heading=h.44bvf6o)

[d. Database queries 27](#_heading=h.2jh5peh)

[**30.**Questions List 28](#_heading=h.ymfzma)

[a. Class Diagram 28](#_heading=h.3im3ia3)

[b. Class Specifications 28](#_heading=h.1xrdshw)

[c. Sequence Diagram(s) 28](#_heading=h.4hr1b5p)

[d. Database queries 28](#_heading=h.2wwbldi)

[**31.**Question Details 28](#_heading=h.1c1lvlb)

[a. Class Diagram 28](#_heading=h.3w19e94)

[b. Class Specifications 28](#_heading=h.2b6jogx)

[c. Sequence Diagram(s) 28](#_heading=h.qbtyoq)

[d. Database queries 28](#_heading=h.3abhhcj)

[**32.**Question Import 28](#_heading=h.1pgrrkc)

[a. Class Diagram 28](#_heading=h.49gfa85)

[b. Class Specifications 28](#_heading=h.2olpkfy)

[c. Sequence Diagram(s) 28](#_heading=h.13qzunr)

[d. Database queries 28](#_heading=h.3nqndbk)

[**33.**Quizzes List 28](#_heading=h.22vxnjd)

[a. Class Diagram 28](#_heading=h.i17xr6)

[b. Class Specifications 28](#_heading=h.320vgez)

[c. Sequence Diagram(s) 28](#_heading=h.1h65qms)

[d. Database queries 28](#_heading=h.415t9al)

[**34.**Quiz Details 28](#_heading=h.2gb3jie)

[a. Class Diagram 28](#_heading=h.vgdtq7)

[b. Class Specifications 28](#_heading=h.3fg1ce0)

[c. Sequence Diagram(s) 28](#_heading=h.1ulbmlt)

[d. Database queries 28](#_heading=h.4ekz59m)

[**35.**Registrations List 28](#_heading=h.2tq9fhf)

[a. Class Diagram 28](#_heading=h.18vjpp8)

[b. Class Specifications 28](#_heading=h.3sv78d1)

[c. Sequence Diagram(s) 29](#_heading=h.280hiku)

[d. Database queries 29](#_heading=h.n5rssn)

[**36.**Registration Details 30](#_heading=h.375fbgg)

[a. Class Diagram 30](#_heading=h.1maplo9)

[b. Class Specifications 30](#_heading=h.46ad4c2)

[c. Sequence Diagram(s) 30](#_heading=h.2lfnejv)

[d. Database queries 30](#_heading=h.10kxoro)

[**37.**Users List 30](#_heading=h.3kkl7fh)

[a. Class Diagram 30](#_heading=h.1zpvhna)

[b. Class Specifications 30](#_heading=h.4jpj0b3)

[c. Sequence Diagram(s) 30](#_heading=h.2yutaiw)

[d. Database queries 30](#_heading=h.21od6so)

[**38.**User Details 30](#_heading=h.1e03kqp)

[a. Class Diagram 30](#_heading=h.3xzr3ei)

[b. Class Specifications 30](#_heading=h.2d51dmb)

[c. Sequence Diagram(s) 30](#_heading=h.sabnu4)

[d. Database queries 30](#_heading=h.3c9z6hx)

[**39.**Settings List 30](#_heading=h.1rf9gpq)

[a. Class Diagram 30](#_heading=h.4bewzdj)

[b. Class Specifications 30](#_heading=h.2qk79lc)

[c. Sequence Diagram(s) 30](#_heading=h.15phjt5)

[d. Database queries 30](#_heading=h.3pp52gy)

[**40.**Setting Details 30](#_heading=h.24ufcor)

[a. Class Diagram 30](#_heading=h.jzpmwk)

[b. Class Specifications 30](#_heading=h.33zd5kd)

[c. Sequence Diagram(s) 30](#_heading=h.1j4nfs6)

[d. Database queries 30](#_heading=h.434ayfz)

[III. Database Tables 30](#_heading=h.2i9l8ns)

[1. <Question> 30](#_heading=h.xevivl)

[2. <Category > 31](#_heading=h.3hej1je)

[3. <Answer> 31](#_heading=h.1wjtbr7)

[4. <Blog> 32](#_heading=h.4gjguf0)

[5. <[Subject]> 32](#_heading=h.2vor4mt)

[6. <Calendar> 32](#_heading=h.gtnh0h)

[7. <Course> 33](#_heading=h.30tazoa)

[8. <Lesson> 33](#_heading=h.1au1eum)

[9. <PricePackage> 34](#_heading=h.3utoxif)

[10. <Quiz> 34](#_heading=h.29yz7q8)

[11. <Registration> 35](#_heading=h.p49hy1)

[12. <”Role”> 35](#_heading=h.393x0lu)

[13. <Score> 35](#_heading=h.1o97atn)

[14. <Slider> 36](#_heading=h.488uthg)

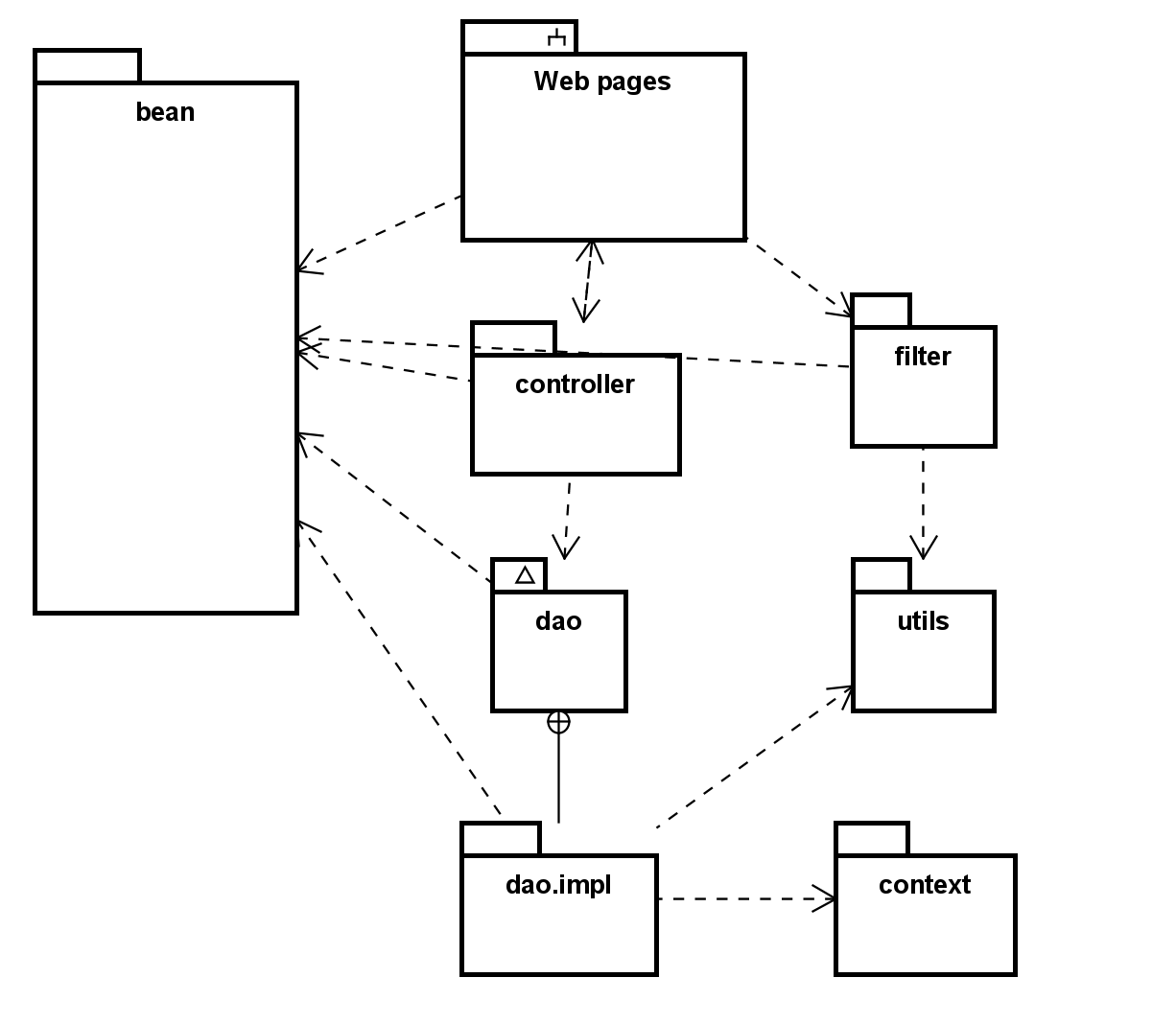
[15. <SubjectDimension> 36](#_heading=h.1fyl9w3)

[16. < "User" > 36](#_heading=h.2ne53p9)

[17. < Setting > 37](#_heading=h.12jfdx2)

# I. Overview

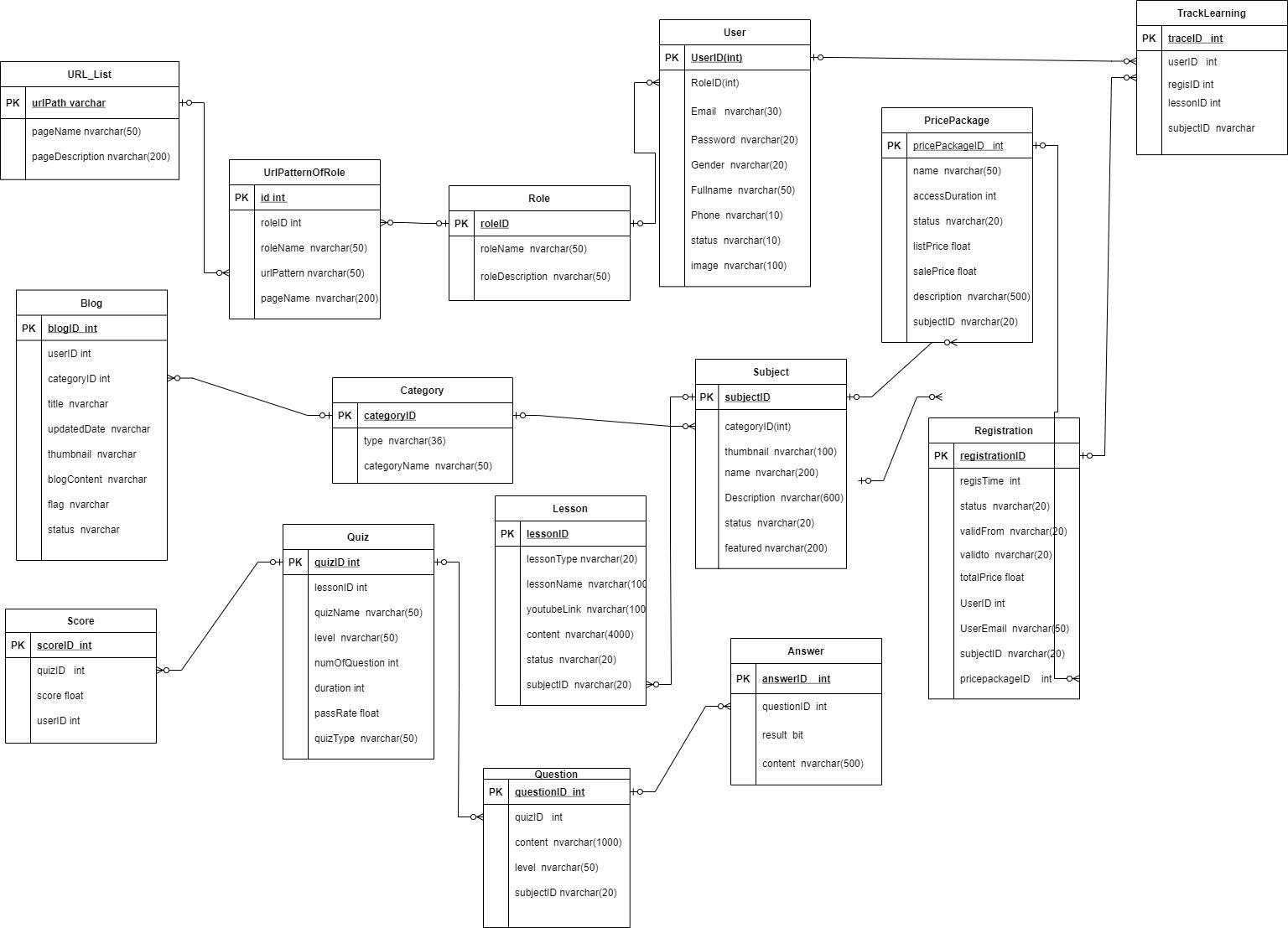
## 1. Code Packages/Namespaces



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

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | bean | The content of the package bean includes all of Objects containing the necessary information of that object in database  Class naming convention followed by the name of that object. Capitalize the first letter, no accents.  e.g: Answer, Course, blog … |
| 02 | context | The content of the package context to connect to Database  Class naming convention will be set according to the function. Capitalize the first letter, no accents.  Name class = DB+ function.java  e.g : DBContext.java> |
| 03 | controller | The content of the package controller include servlets  Class naming convention following by :  Class name = (Role) + function + Object +Controllerjava  e.g: LoginController.java, AddSubjectController.java …> |
| 04 | dao | The content of the package dao includes sql statements to get data from SQLSever  Class naming convention will be set according to the function. Capitalize the first letter, no accents.  include 2 sub package  dao.itf : has Interfaces. Interface name = Object+DAO  dao.impl: Class name = Function + Object java |
| 05 | filter | The content of the package filter includes the filter configuration we are associated with  Class naming convention following by :  Class name = function+Filter+Object.java  Capitalize the first letter, no accents.  e.g: SecurityFilterAll.java… |

## 2. Database Schema



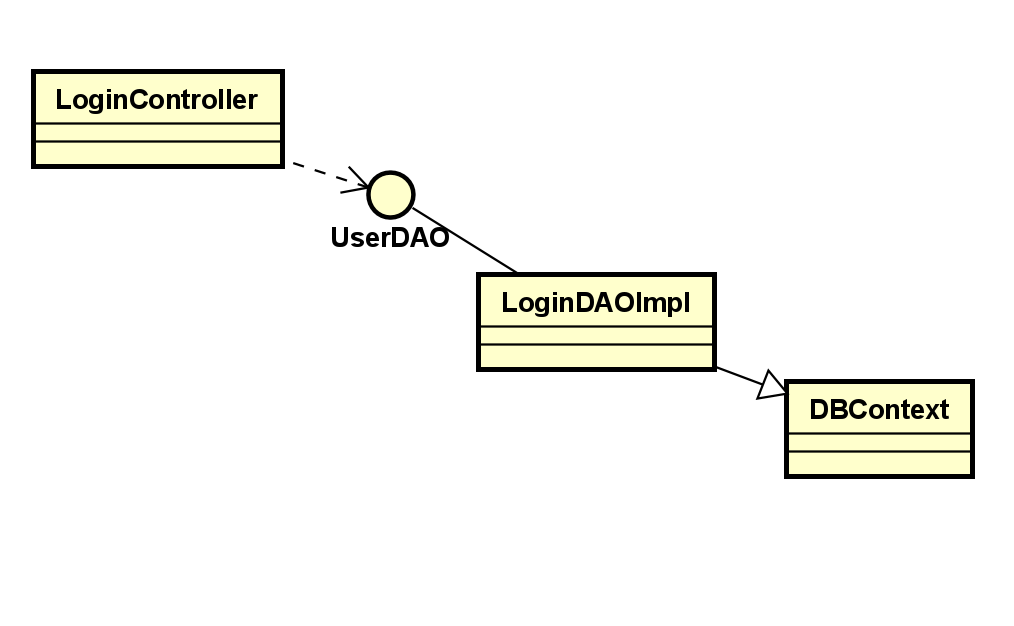
***Table descriptions & package class naming conventions are as below***

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 1 | <Answer> | - Primary keys: < Answer ID>  - Foreign keys: <questionID> |
| 2 | <Blog> | - Primary keys: <BlogID>  - Foreign keys: <categoryID>,<userID> |
| 3 | <Category> | - Primary keys: < Category ID>  - Foreign keys: <none> |
| 4 | <Lesson> | - Primary keys: <lessonID>  - Foreign keys: <subjectID> |
| 5 | <PricePackage> | - Primary keys: < PricePackage ID>  - Foreign keys: <subjectID> |
| 6 | <Question> | - Primary keys: < QuestionID>  - Foreign keys: <quizID>,<dimensionID>,<subjectID>,<lessoniD> |
| 7 | <Quiz> | - Primary keys: <QuizID>  - Foreign keys: <lessonID>,<subjectID> |
| 8 | <Registration> | - Primary keys: < Registration ID>  - Foreign keys: <UserID> |
| 9 | <Role> | <Description of the table>  - Primary keys: <roleID>  - Foreign keys: <none> |
| 10 | <Score> | - Primary keys: <scoreID >  - Foreign keys: <quizID> |
| 11 | <Slider> | - Primary keys: <SliderID>  - Foreign keys: <none> |
| 12 | <Subject> | - Primary keys: <subjectID>  - Foreign keys: <CategoryID> |
| 13 | <TrackLearning> | -Primary key: <traceID>  -Foreign key: <userID>,<regisID> |
| 14 | <URL\_List> | -Primary key: <urlPath>  -Foreign key: none |
| 15 | <UrlPatternOfRole> | -Primary key: <id>  -Foreign key: <urlPattern>,<roleID> |
| 16 | <User> | - Primary keys: <UserID>  - Foreign keys: <RoleID> |

# II. Code Designs

# **1.**User Login

### a. Class Diagram:



b. Class Specifications

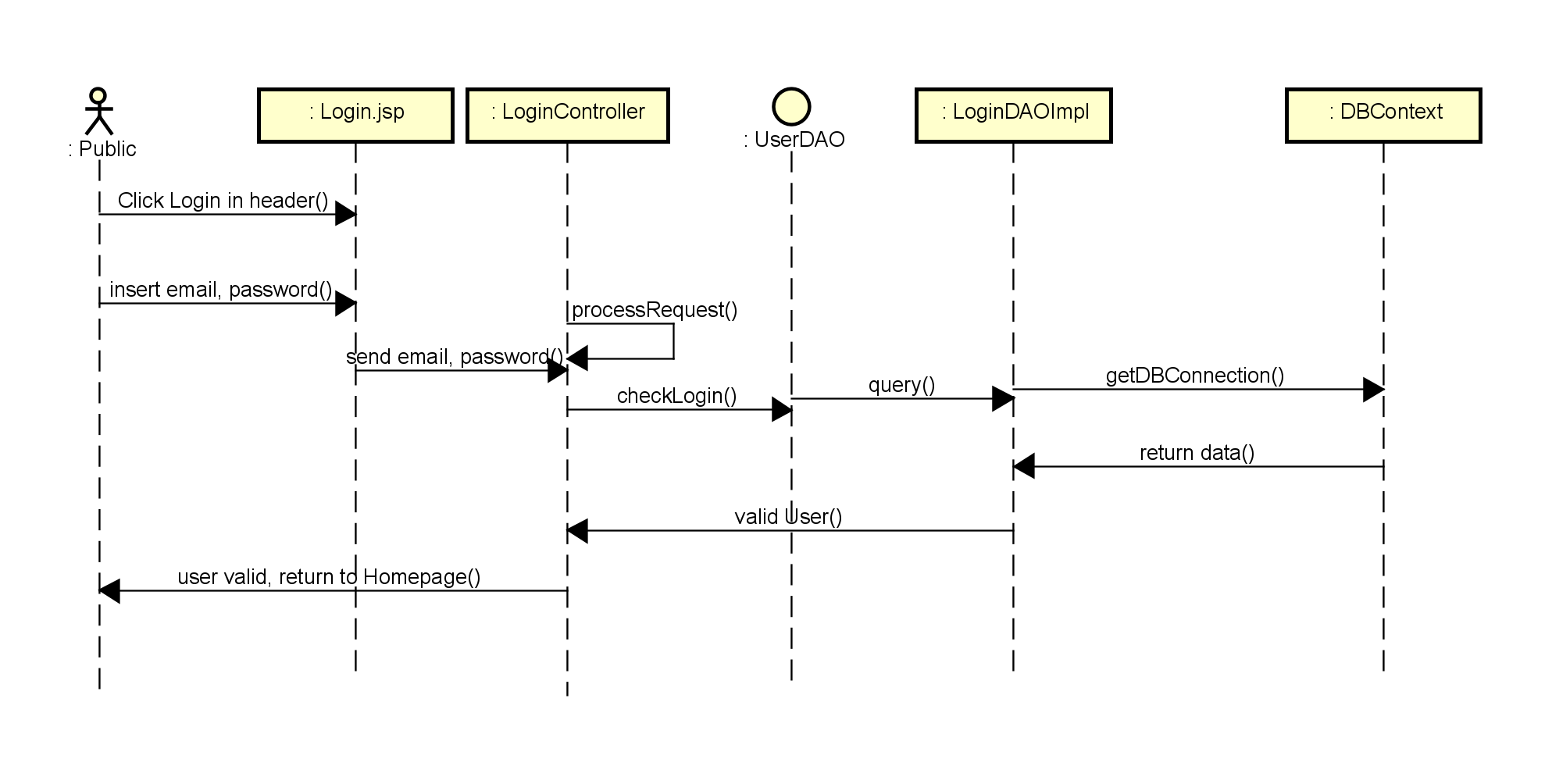
***Class Login.java***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | checkLogin | * this function is used for check if user exist in the db * input: email, password * output: all of informations of user with that email * use: DBContext, getConnection, PrepareStatement |

*Class LoginController*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | processRequest | - input: request, response  - output: check valid user?  - use: requestDispatcher |

### c. Sequence Diagram(s)

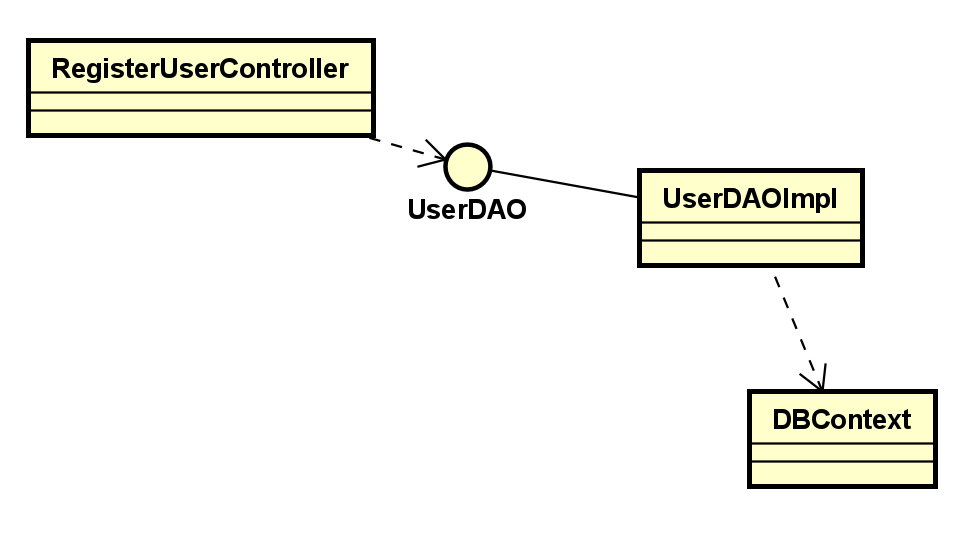


### d. Database queries :

- Get email and password of user from User table : select \* FROM [SWP].[dbo].[User] where email=? AND password=?

# **2.**User Register

### a. Class Diagram



### b. Class Specifications

**RegisterUserServlet Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |
| 02 | doPost | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  - use: processRequest |
| 03 | doGet | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  use: processRequest |
| 04 | addAttributeWithExceptParam | - input:  + Map<String, String[]> params  + String except  + HttpServletRequest request  - output: None  - use: N.A |

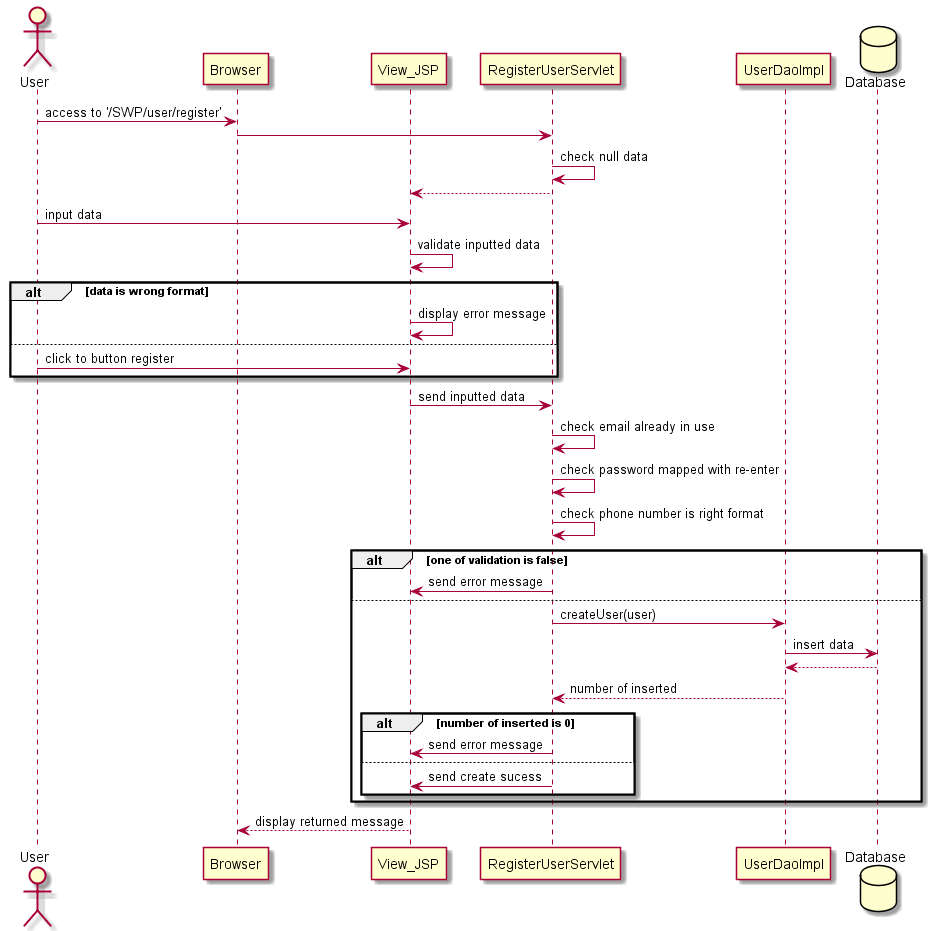
**StringValidation Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isNullOrEmpty | - input:  + String data  - output: true if data inputted is null or empty and false if it’s not  - use: N.A |

**UserDaoImpl Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createUser | - input:  + User user  - output: number of created user  - use: getConnection, PreparedStatement, Connection |
| 02 | countExistedUser | - input:  + User user  - output: number of existed user in database  use: getConnection, PreparedStatement, Connection, ResultSet |

### c. Sequence Diagram(s)

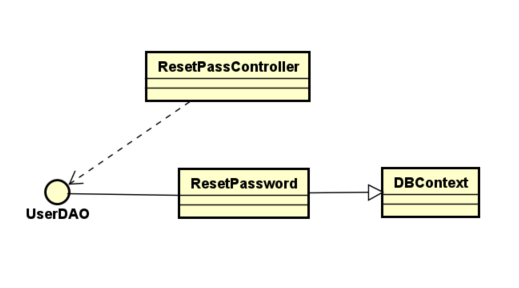


### d. Database queries

- Insert new user to database: "INSERT INTO dbo.[User](RoleID, SettingID,Email,Password,Gender,Fullname,Phone,enabled) VALUES(?,?,?,?,?,?,?,?)"

# **3.**Reset Password

### a. Class Diagram



### b. Class Specifications

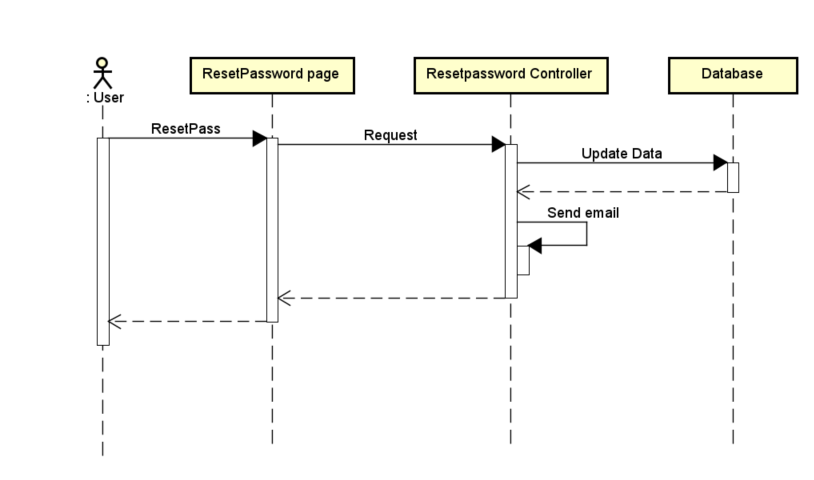
**Class ResetPword**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | checkAccountExist | * input: email, password * output:true if Account Exist else false * use: DBContext, getConnection, PrepareStatement |
| 02 | resetPassword | * input: email * output:reset password to null * use: DBContext, getConnection, PrepareStatement |

**Class ResetPasswordServlet**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: password reset  - use:checkAccountExist,resetPassword |
| 02 | doPost | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  - use: processRequest |
| 03 | doGet | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  use: processRequest |

### c. Sequence Diagram(s)

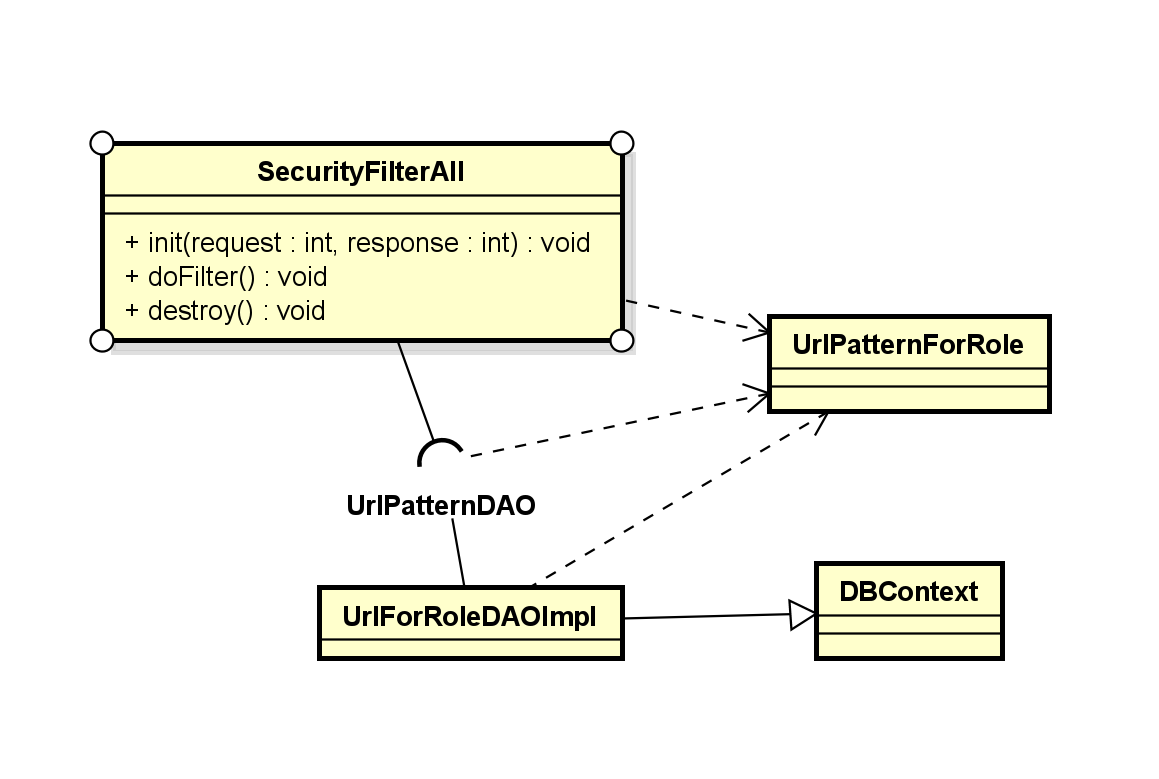


### d. Database queries

- update [User] set password = null where Email= ? [Cập nhật mật khẩu của Email = user pnull]

# **4.**User Authorization

### a. Class Diagram



### b. Class Specifications

**UrlPatternDAO**

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest | - input:+ HttpServletRequest request+ HttpServletResponse response- output: Redirect- use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |

### 

**UrlFilterDAOImpl**

### 

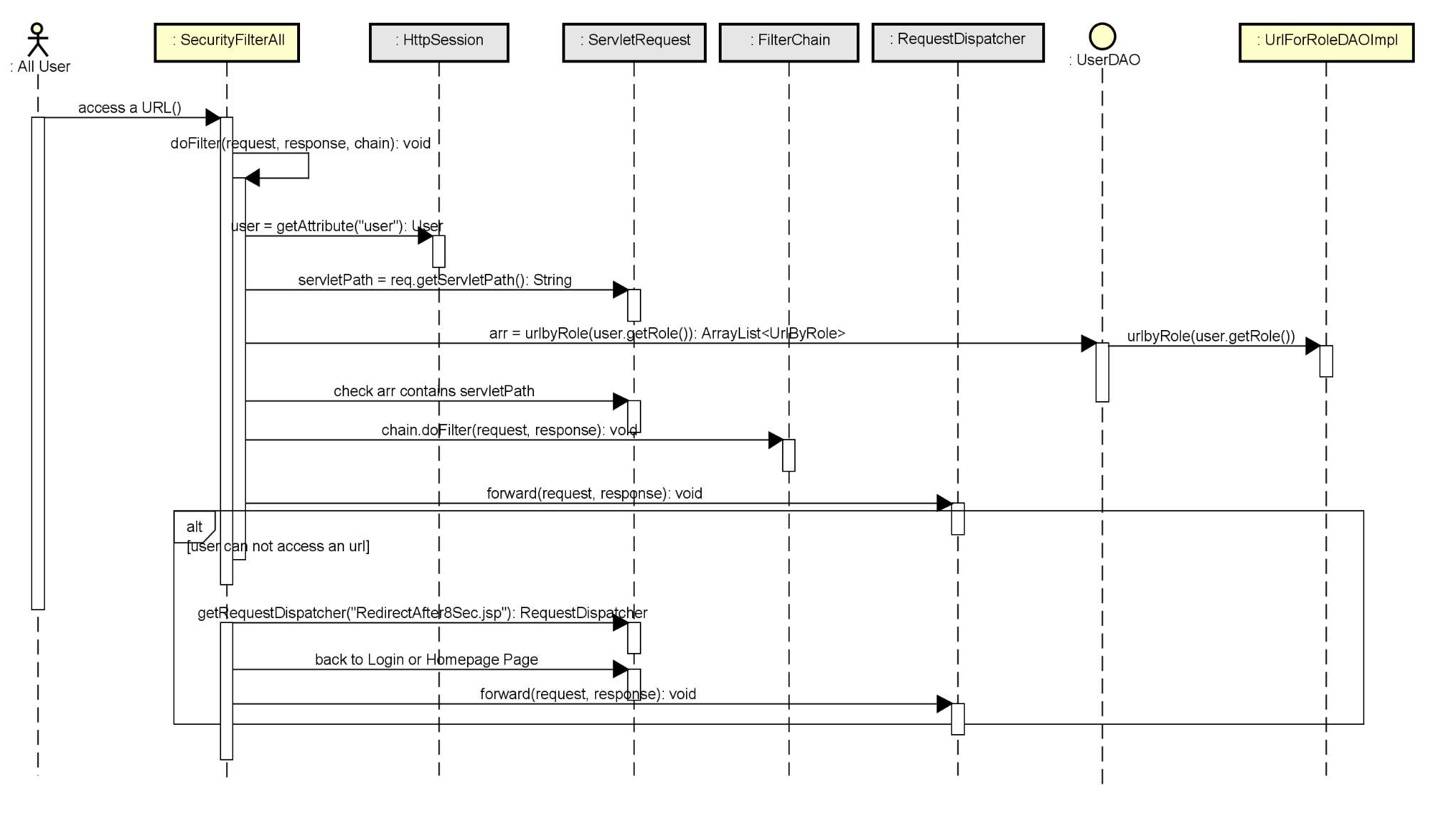
| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest | - input:+ HttpServletRequest request+ HttpServletResponse response- output: N.A- use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |

**SecurityFilterAll**

| No | Method | Description |
| --- | --- | --- |
| 01 | init | start filter  * input FilterConfig filterConfig * throws Servlet exception |
| 02 | doFilter | input:  * ServletRequest request, ServletResponse response, FilterChain chain * throws IOException, ServletException   use: to handle request, response of every url pattern, to direct user to authorized pages |
| 03 | destroy() | use: when filter get cancelled |

### 

### c. Sequence Diagram(s)

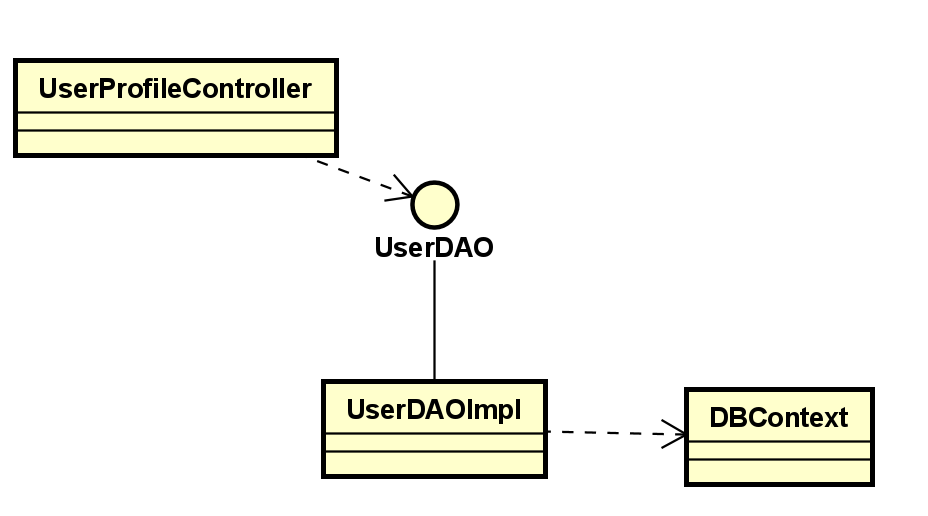


### d. Database queries

* SELECT \* from UrlPatternOfRole
* SELECT \* from UrlPatternOfRole where roleID = ?

# **5.**User Profile

### a. Class Diagram:



### b. Class Specifications

#### UserProfileServlet Class

### 

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest | - input:+ HttpServletRequest request+ HttpServletResponse response- output: N.A- use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |

### 

#### UserDaoImpl Class

| No | Method | Description |
| --- | --- | --- |
| 01 | findUserById | - input:+ String userId- output: User with specified ID- use: getConnection, PreparedStatement, Connection |

### 

### c. Sequence Diagram(s)

### d. Database queries :

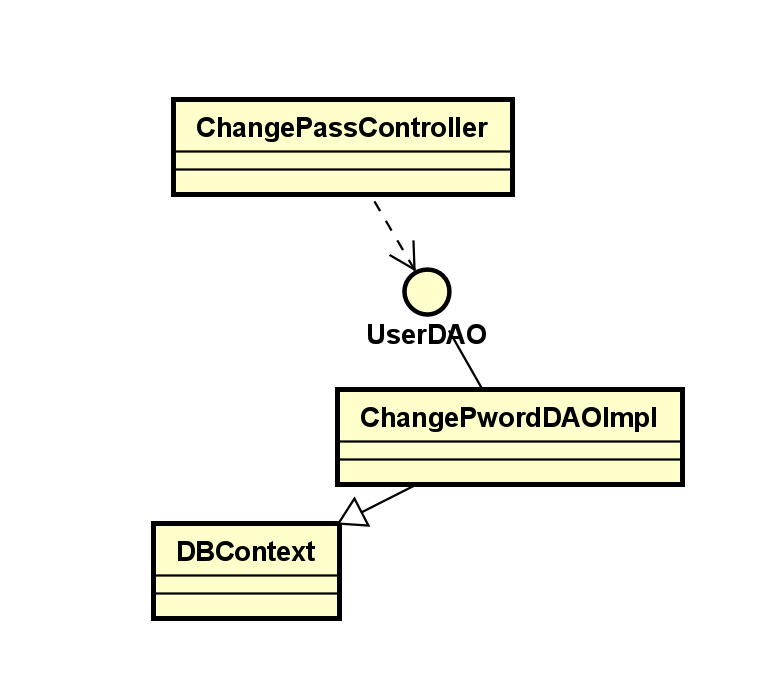
"SELECT \* FROM dbo.[User]"

+ "WHERE email = ? AND password = ? AND [enabled] = 'True' "

### 

# **6.**Change Password

### a. Class Diagram



### b. Class Specifications

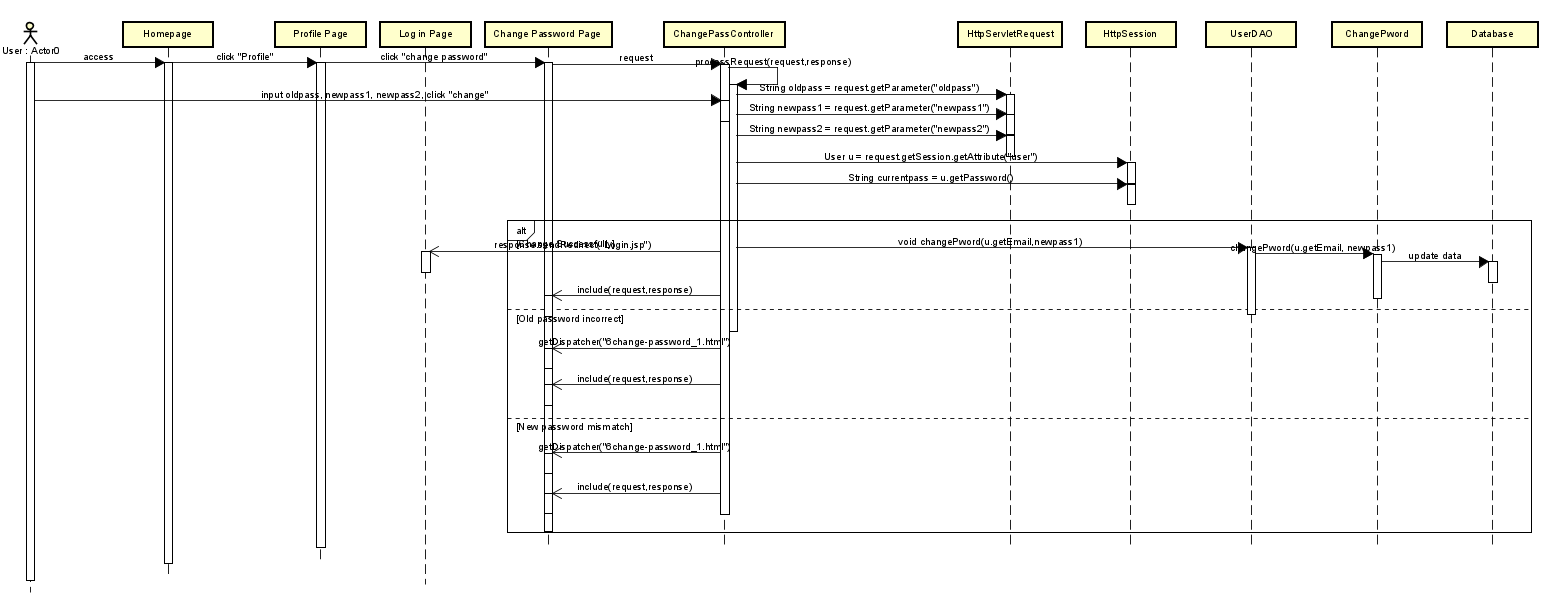
**class changePword**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | changePassword | * Method type: Public * Return type: void * Parameters: (String email, String password)   This method changes the password by using SQL Query in the database |

**Servlet class ChangePassServlet**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doPost | * Method type: Public * Return type: void * Role: Public   This method gets the inputs of User including Old password, New Password and Confirm New Password. Password only change if The Old password is correct and the new password matches the confirm new password |

### c. Sequence Diagram(s)

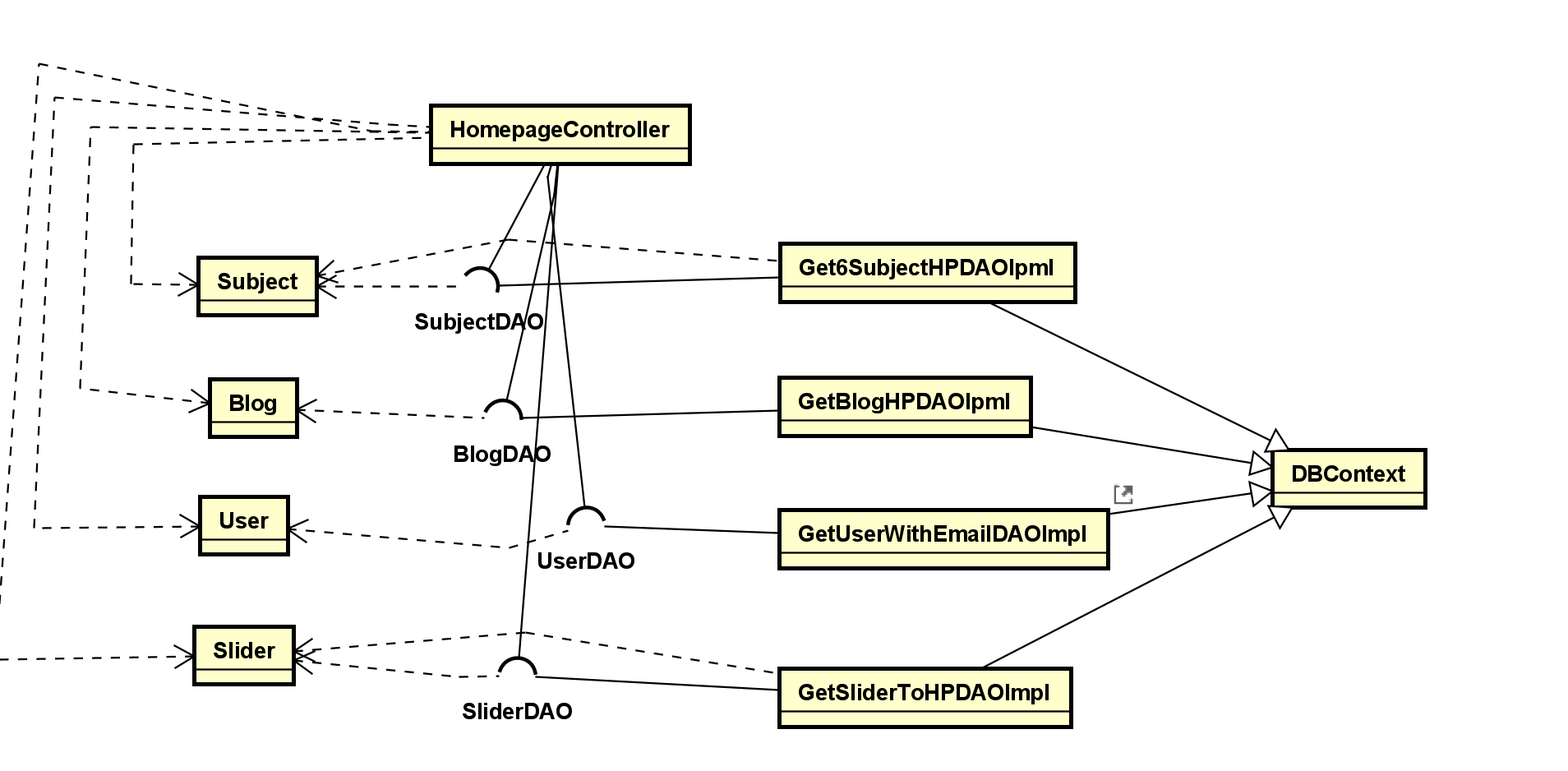


### d. Database queries

* Update [User] set Password = ? where Email = ?

# **7.**Home

### a. Class Diagram



### b. Class Specifications

### b. Class Specifications

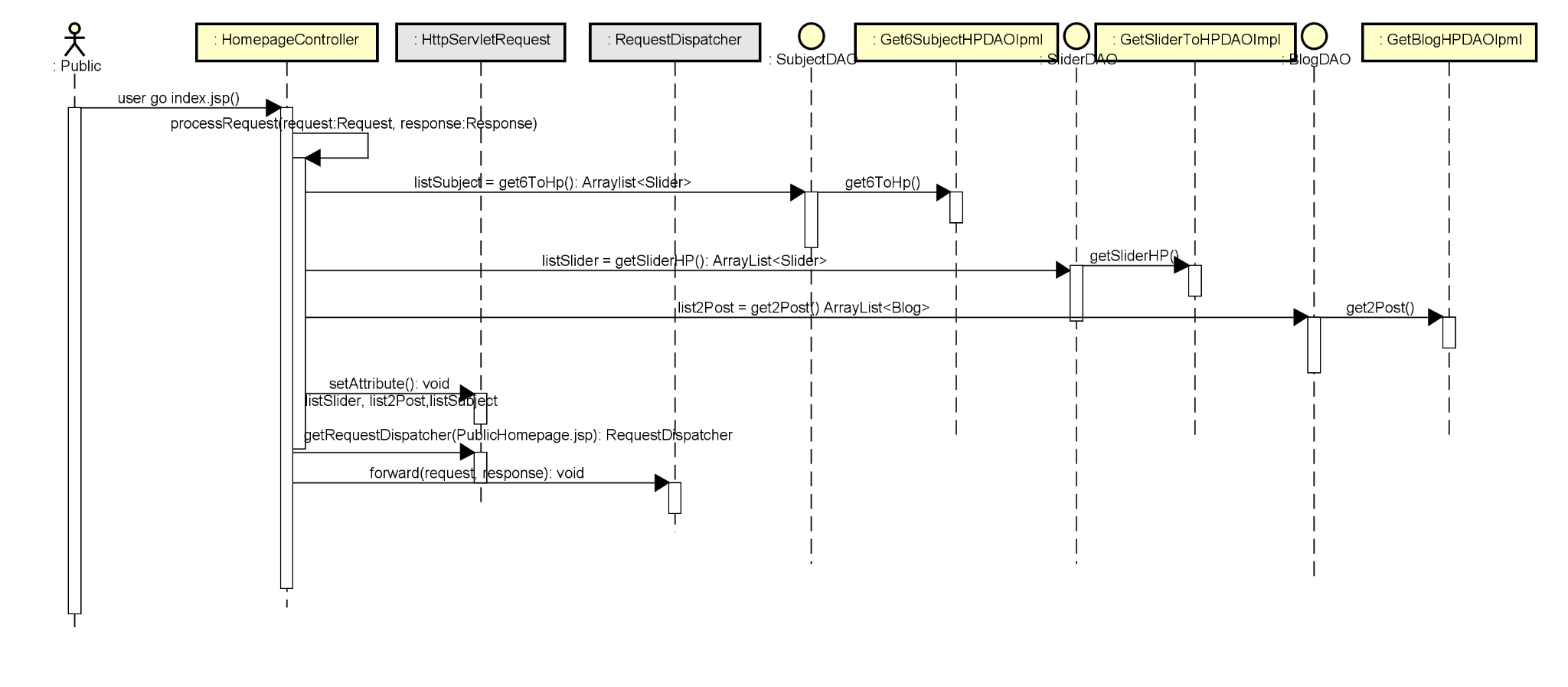
#### Get6SubjectHPDAOImpl Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get6ToHP | * input: none * output: 4 Course from database * use: DBContext, Connection, PrepareStatement |

#### Get2PostToHP Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get2PostToHP | * input: none * output: 2 Blog from database * use: DBContext, Connection, PrepareStatement |

### c. Sequence Diagram(s)

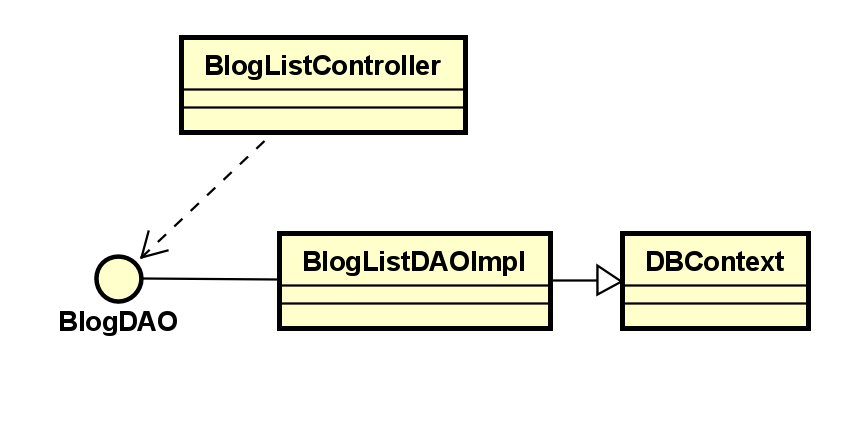


### d. Database queries

* get 2 post to homepage: “select top 2 from Post”
* show 6 newest course to homepage: “select top 6 from Subject ”

# **8.**Blogs List

### a. Class Diagram



### b. Class Specifications

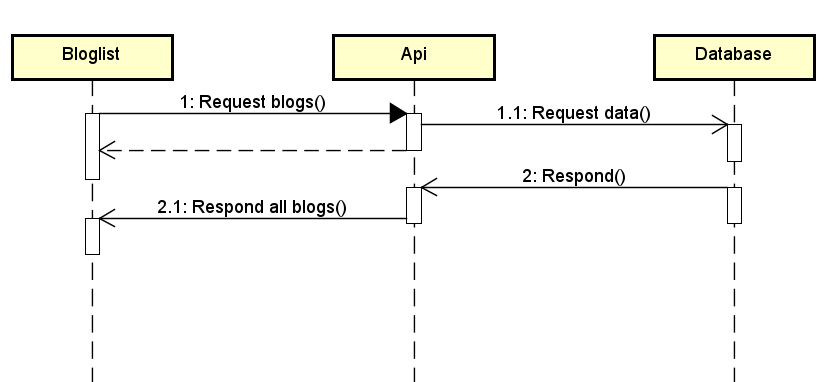
Class BlogList

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllBlog() | * input: [SWP].[dbo].[Blog] * output:get all blog list * use: DBContext, getConnection, PrepareStatement |

**Class Bloglistcontroller**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: BlogList  - use: get all BlogList |
| 02 | doPost | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  - use: processRequest |
| 03 | doGet | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  use: processRequest |

### c. Sequence Diagram(s)

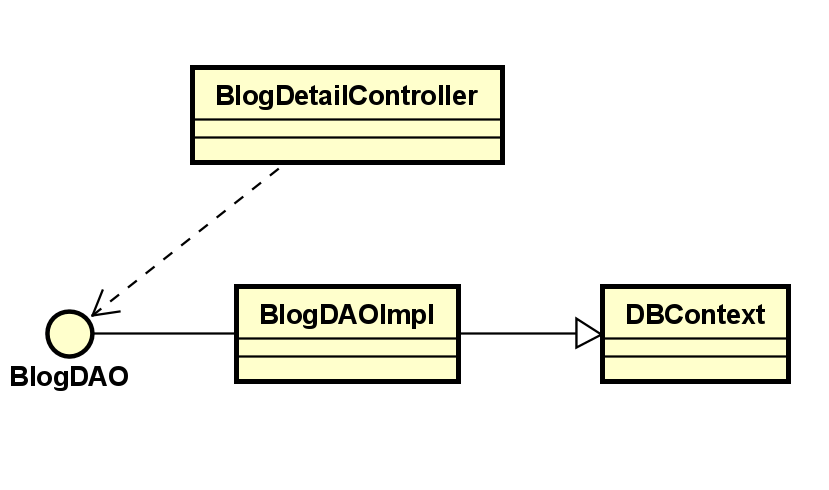


### d. Database queries

- Select \* from Blog //Get all Blog from DB  
- select \* from (select b.\*, u.fullname from blog b, user u where u.userID = b.userID order by updatedDate desc) as T limit 5 // Get first 5 Blog From DB

# **9.**Blog Details

### a. Class Diagram



### b. Class Specifications

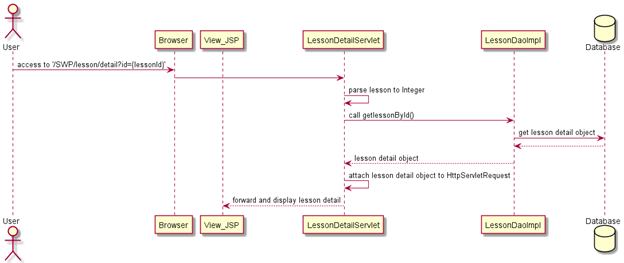
#### LessonDetailServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |

#### LessonDaoImpl Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findLessonById | - input:  + String lessonId  - output: Lesson with specified ID  - use: getConnection, PreparedStatement, Connection |

### c. Sequence Diagram(s)



### d. Database queries

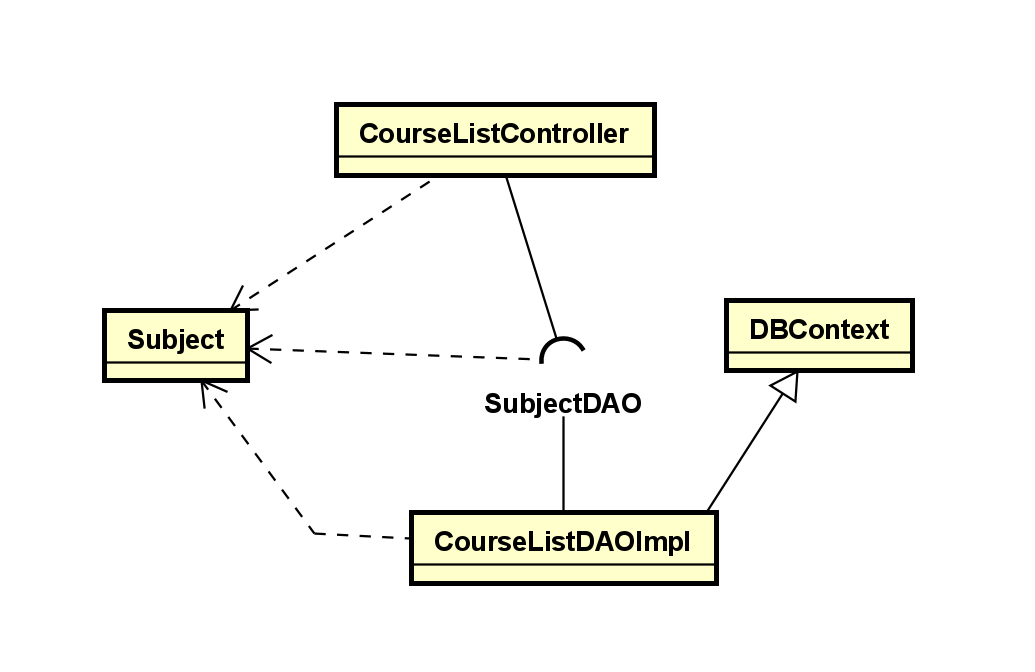
"UPDATE [dbo].[Blog]\n"

+ " SET title = ? , blogContent = ? , updatedDate = ?\n"

+ "WHERE blogID = ?";

# **10.**Courses List

### a. Class Diagram

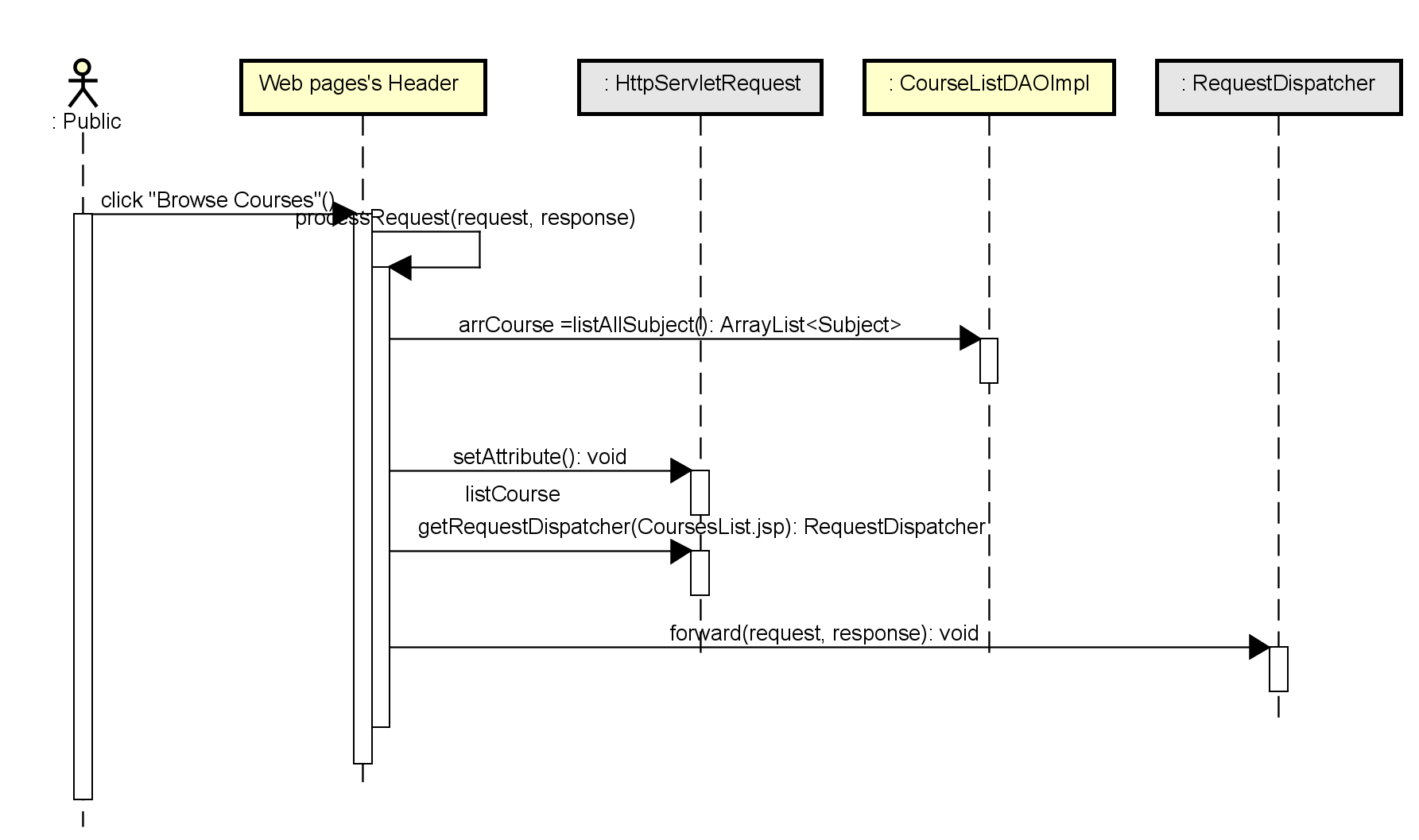


### b. Class Specifications

*CourseListImple*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCourse* | * *input: none* * *output: all Course from database* * *use: DBContext, Connection, PrepareStatement* |
|  |  |  |

### c. Sequence Diagram(s)

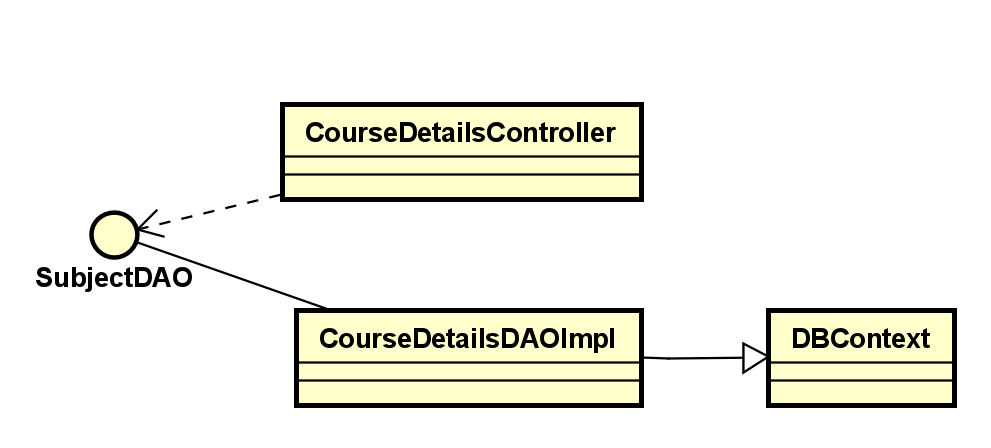


### d. Database queries

 -SELECT \*FROM [SWP].[dbo].[Subject] : to get all data from table Subject

# **11.**Course Details

### a. Class Diagram



### b. Class Specifications

*Class CourseDetailsDAOImpl*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *subjectBySubjectID()* | * *input: [SWP].[dbo].[Subject]* * *output:get all information from Subject bye SubjectID* * *use: DBContext, getConnection, PrepareStatement* |
|  |  |  |

### 

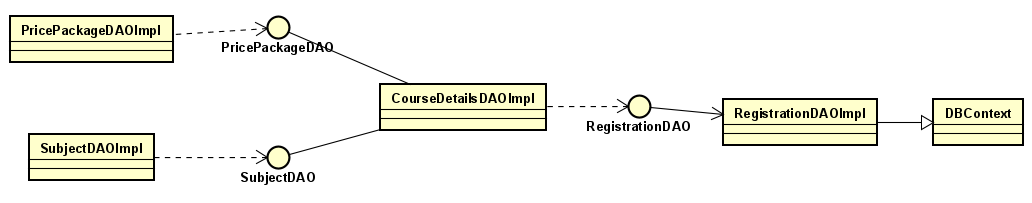
### c. Sequence Diagram(s)

### d. Database queries

SELECT \* FROM [SWP].[dbo].[Subject] where subjectID=?

# **12.**Course Register

### a. Class Diagram



### b. Class Specifications

Class CourseRegister

### 

| No | Method | Description |
| --- | --- | --- |
| *01* | *addRegistration* | *- Package: Dao.impl**- Return type: void**- Type: Public**As customer buy a course, this method used to add a registration information into database* |

### 

### c. Sequence Diagram(s)

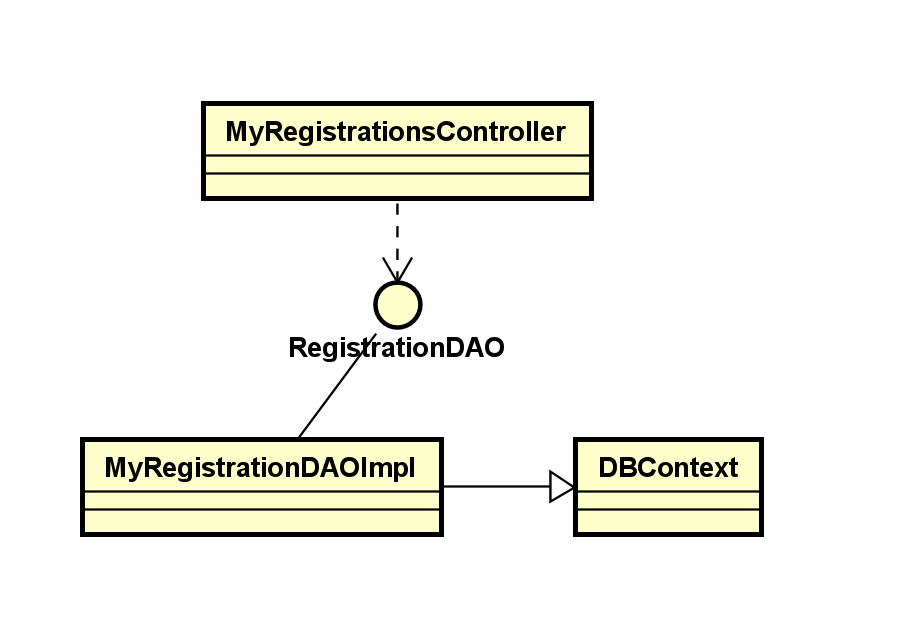
### d. Database queries

### 

insert into Registration values (?,?,?,?,?,?,?,?,?)

### 13.My Registrations

### a. Class Diagram



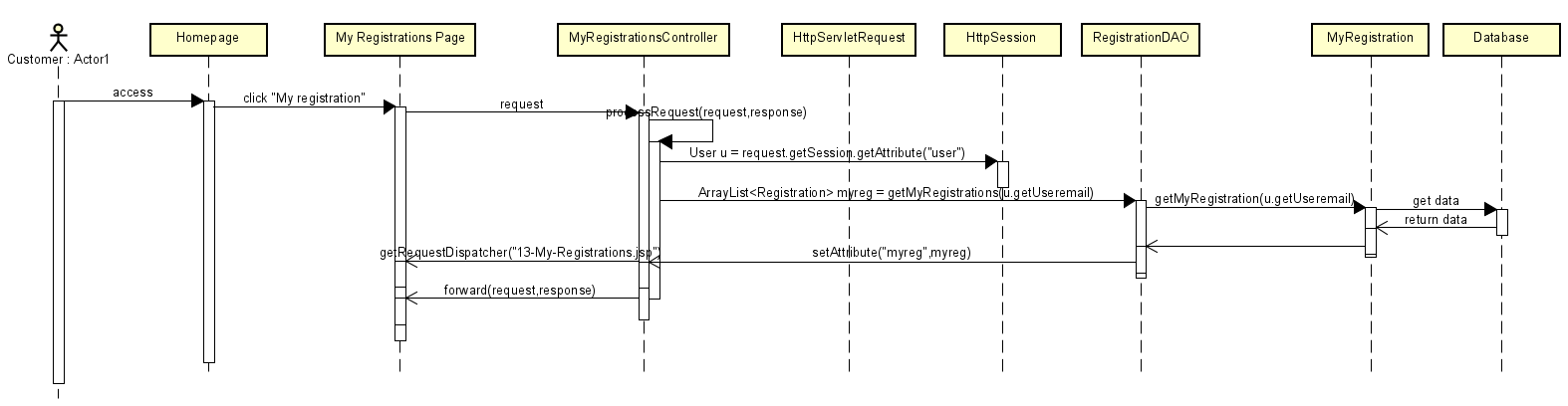
### b. Class Specifications

| No | Method | Description |
| --- | --- | --- |
| *01* | *getRegisByEmail* | *- Package: Dao.impl**- Return type: List<Registration>**- Type: Public**This method gets all the Registrations of the current User by Email using SQL Query.* |

### Servlet class MyRegistrationsServlet

| No | Method | Description |
| --- | --- | --- |
| *01* | *doGet* | *- Package: Controller**- Type: Public**- Role: Customer**This method get all the Registration using the getRegisByEmail method and display them on the jsp page.* |
|  |  |  |

### c. Sequence Diagram(s)

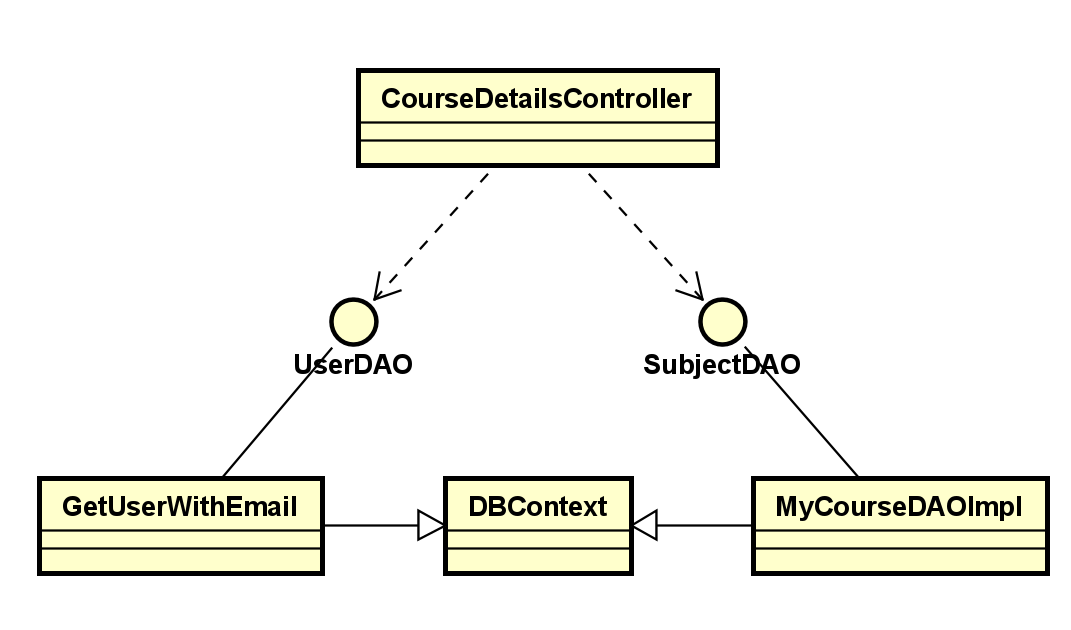


### d. Database queries

* select \* from Registration where Email = ?

# **14.**My Courses

### a. Class Diagram



### b. Class Specifications

*Class MyCourseDAOImpl*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSubjectOfUser* | * *input: [SWP].[dbo].[Subject] JOIN [Registration]* * *output: get all information of all subject user registrationed* * *use: DBContext, getConnection, PrepareStatement* |
|  |  |  |

### 

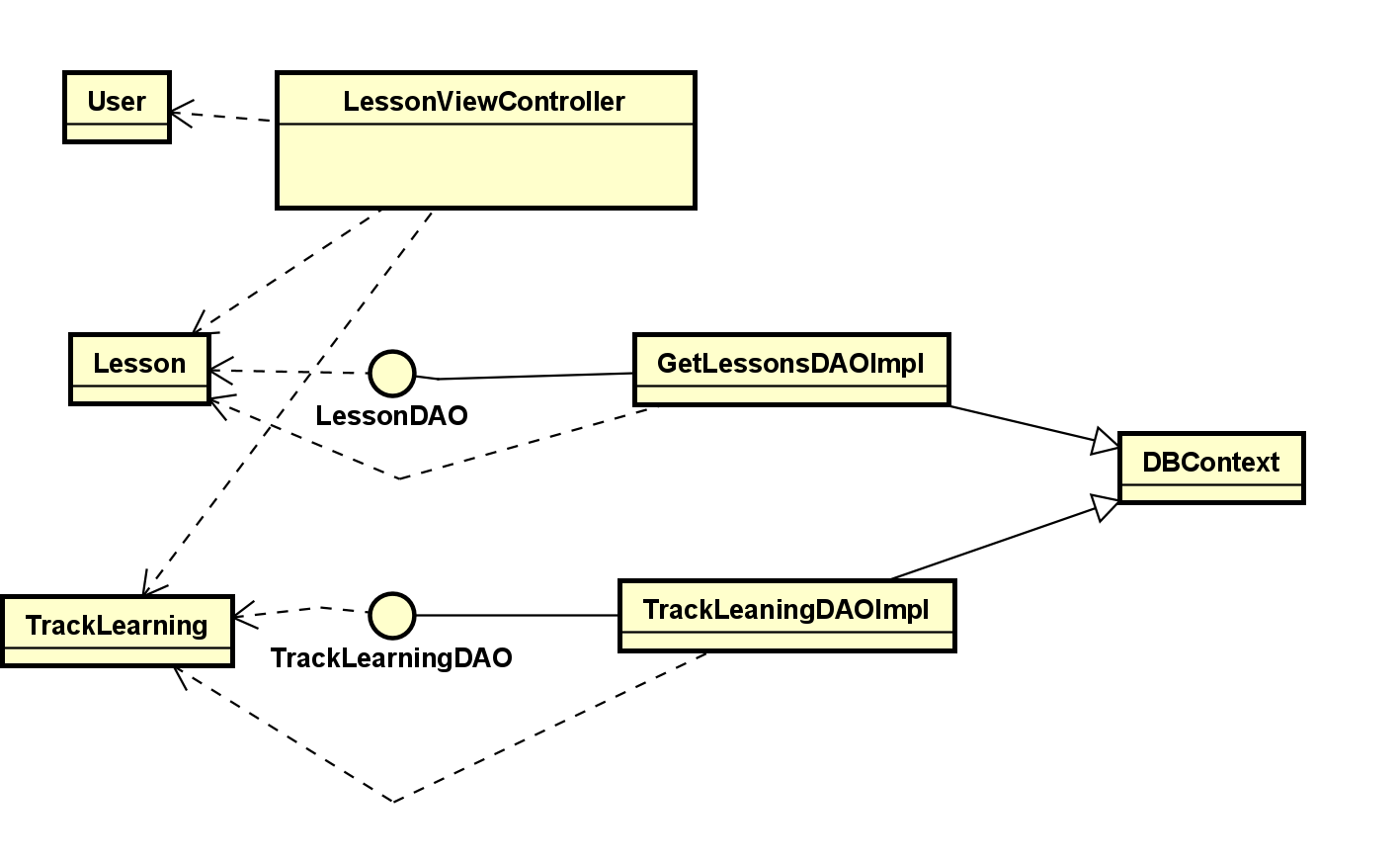
### c. Sequence Diagram(s)

### d. Database queries

select S.\* from Subject as S inner join Registration as R on s.subjectID =r.subjectID where r.UserID =?

# **15.**Lesson View

### a. Class Diagram



### b. Class Specifications

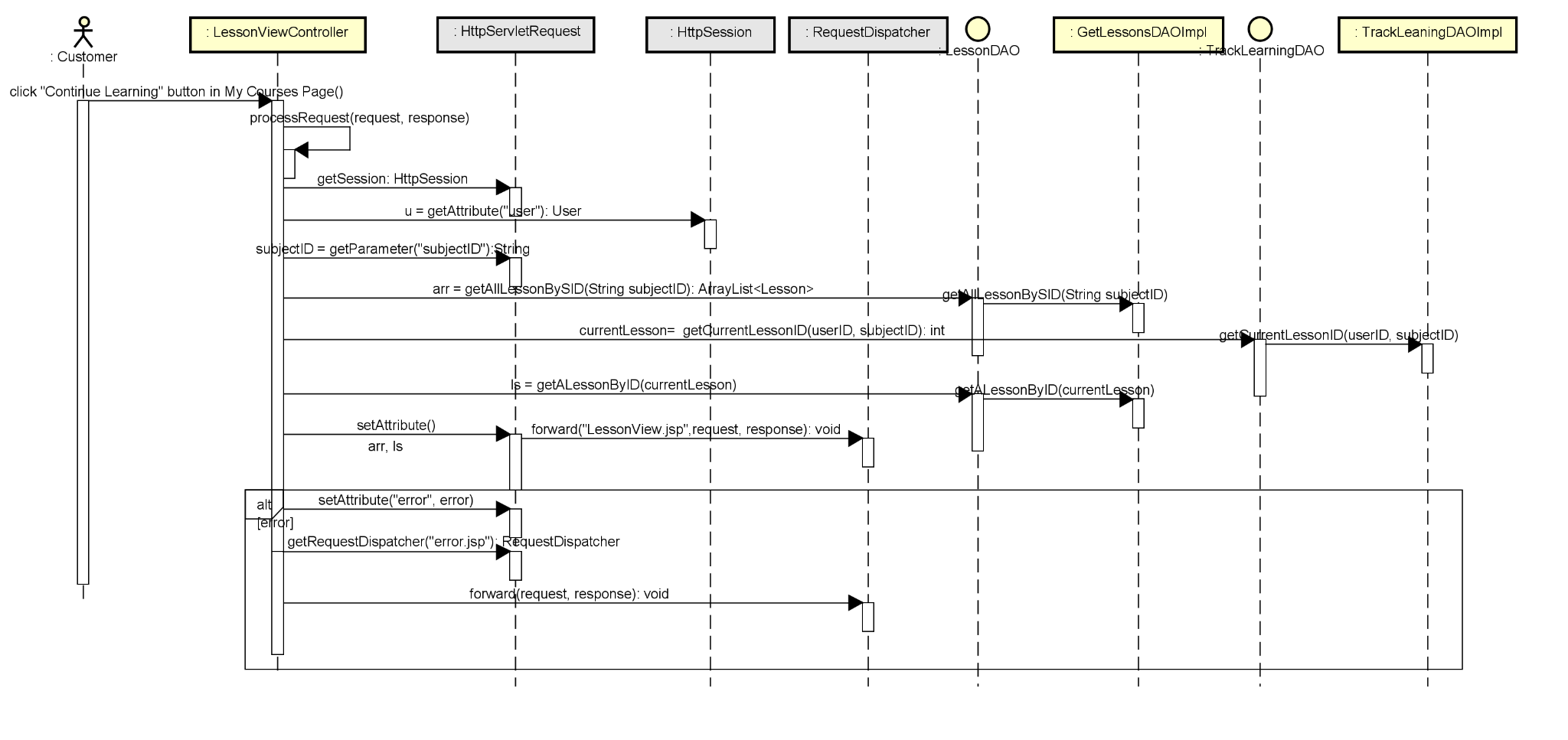
*Class LessonDAO*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllLessonBySID | - input: quizID  - output: arraylist get all information of all Question that include QuizID  - use: DBContext, getConnection, PrepareStatement |
| 02 | getQuestionType | - input: questionID  - output: type of question(Lesson or quiz)  - use: Question |

*Class GetListQuestionDAOImpl*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllQuestionByQuizID | - input: quizID  - output: arraylist get all information of all Question that include QuizID  - use: DBContext, getConnection, PrepareStatement |

### c. Sequence Diagram(s)



* quiz
* d. Database queries
* select \* from Lesson where subjectID=?
* select \* from Quiz where lessionID= ?
* select \* from Question where quizID =?
* select \* from Answer where questionID =?

# **16.**Quiz Lesson

### a. Class Diagram

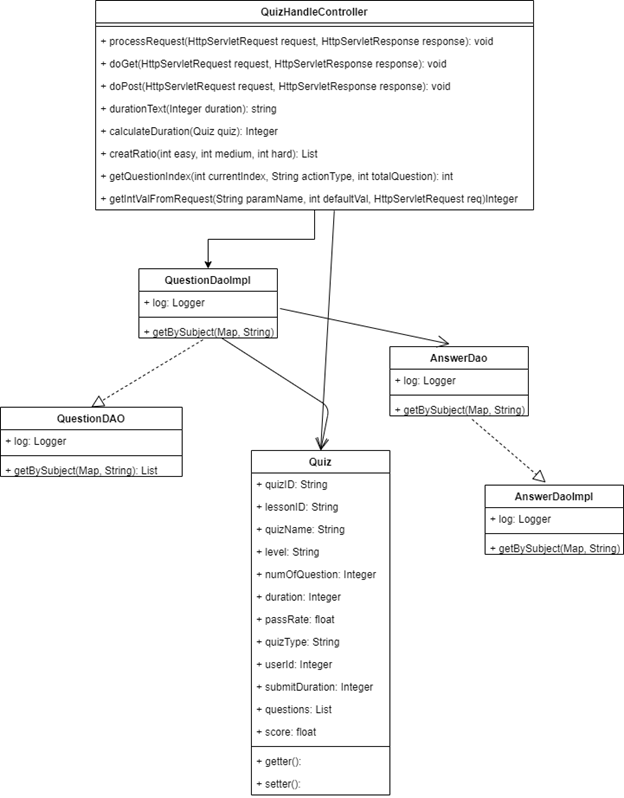
### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

# **17.**Quiz Handle

### a. Class Diagram



### b. Class Specifications

#### QuizHandlerServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - Description: process for both POST and GET method  - input:  + Integer easy  + Integer medium  + Integer hard  - output: Map<String,Integer>  - use: createRatio, getDurationText, getQuestionIndex, getIntFromRequest |
|  | doGet | - Description: handle GET method  - input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: processRequest |
|  | doPost | - Description: handle POST method  - input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: processRequest |
|  | createRatio | - Description: calculate percent of each level of question  - input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: ramdom |
|  | getDurationText | - Description: get duration in text with format ‘XXhXXmXXs’  - input:  + Integer duration  - output: String  - use: N/A |
|  | getQuestionIndex | - Description: get question index with specified button type  - input:  + Integer totalquestion  + String buttonType  - output: Integer  - use: N/A |
|  | getIntFromRequest | - Description: get integer value of parameter attached in request  - input:  + HttpServletRequest request  + String param  - output: Integer  - use: N/A |

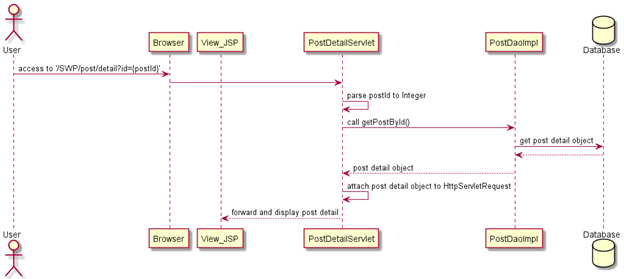
#### QuestionDaoImpl Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getBySubject | - Description: get question by subject with level  - input:  + Map ratio  + String subjectName  - output: List question  - use: getConnection, PreparedStatement, Connection |

#### AnswerDaoImpl Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getByQuestionId | - Description: get answers by question  - input:  + Integer questionId  - output: List answer  - use: getConnection, PreparedStatement, Connection |

### c. Sequence Diagram(s)



### d. Database queries

-"INSERT INTO "

+ "[dbo].[Quiz](quizID, lessonID, level, numOfQuestion, duration, passRate, userID, submit\_duration, score)"

+ " VALUES(?,?,?,?,?,?,?,?,?)";

-"INSERT INTO [dbo].[quiz.question](quiz\_id, question\_id,user\_answ)"

+ " VALUES(?,? ,?)";

# **18.**Quiz Review

### a. Class Diagram

### b. Class Specifications

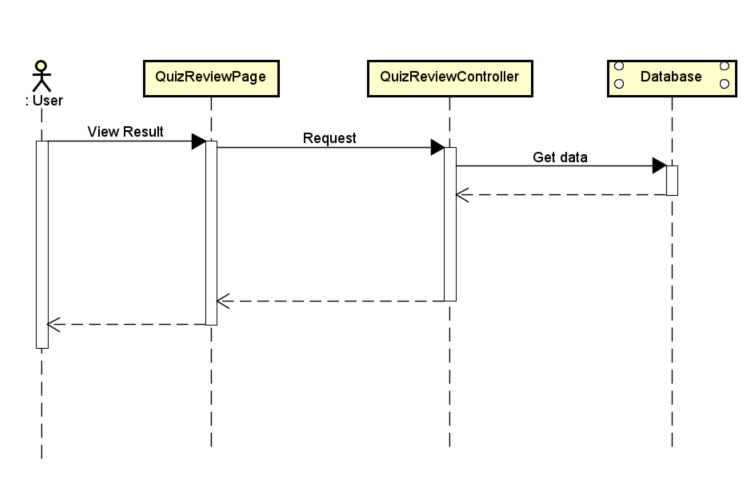
#### QuizReviewController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - Description: process for both POST and GET method  - input:  + Integer easy  + Integer medium  + Integer hard  - output: Map<String,Integer>  - use: createRatio, getDurationText, getQuestionIndex, getIntFromRequest |
|  | doGet | - Description: handle GET method  - input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: processRequest |
|  | doPost | - Description: handle POST method  - input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: processRequest |

#### QuizReview Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getQuestionbyQuizID | - Description: get the Question  - input:  + String subjectName  - output: List question  - use: getConnection, PreparedStatement, Connection |
| 2 | getAllAnswer | - Description: get answer  - input:  + String subjectName  - output: List answer  - use: getConnection, PreparedStatement, Connection |
| 3 | getAllCorrectAnswer | - Description: get all correct answer  - input:  + String subjectName  - output: List Correct answer  - use: getConnection, PreparedStatement, Connection |

### c. Sequence Diagram(s)



### d. Database queries

select question.questionID, question.quizID, question.content from question inner join quiz on question.quizID=quiz.quizID where quiz.quizID=? <get question by quiz ID >

select answerID, answer.questionID, result, answer.content from answer <Get answer>

select answerID, answer.questionID, result, answer.content from answer where result =1 <Get Correct Answer>

# **19.**Dashboard

### a. Class Diagram

### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

# **20.**Post Details

### a. Class D

### b. Class Specifications

### 

#### PostDetailServlet Class

### 

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest | - input:+ HttpServletRequest request+ HttpServletResponse response- output: N.A- use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |

### 

#### PostDaoImpl Class

| No | Method | Description |
| --- | --- | --- |
| 01 | findPostById | - input:+ String postId- output: Post with specified ID- use: getConnection, PreparedStatement, Connection |

### 

### c. Sequence Diagram

### 

### d. Database queries

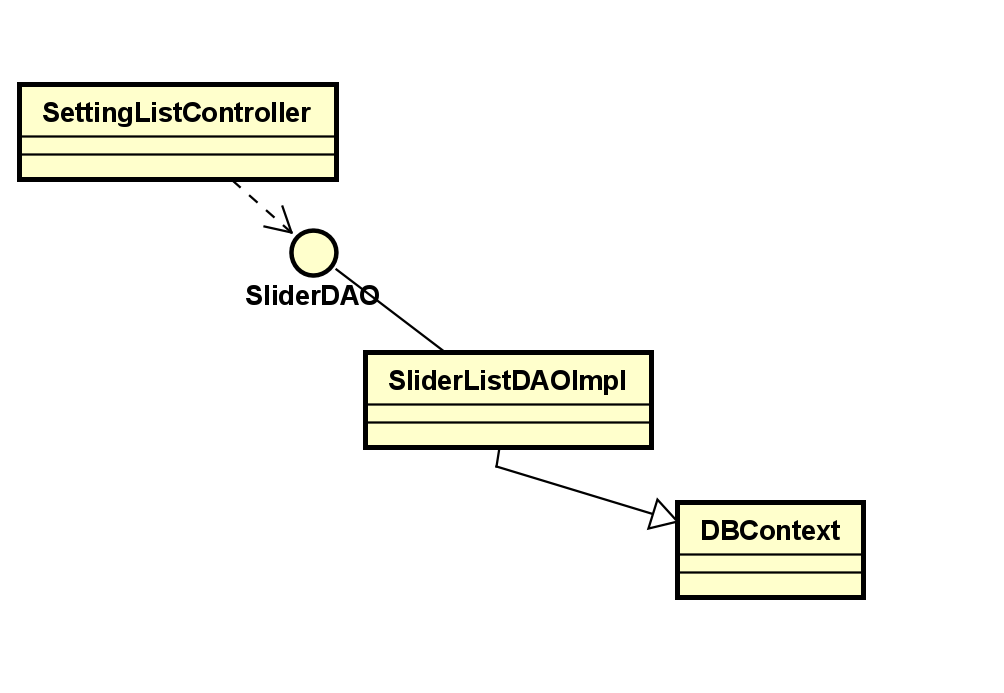
"UPDATE [dbo].[Blog]\n"

+ " SET title = ? , blogContent = ? , updatedDate = ?\n"

+ "WHERE blogID = ?";

**21.**Sliders List

### a. Class Diagram



### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

# **22.**Slider Details

### a. Class Diagram

### b. Class Specifications

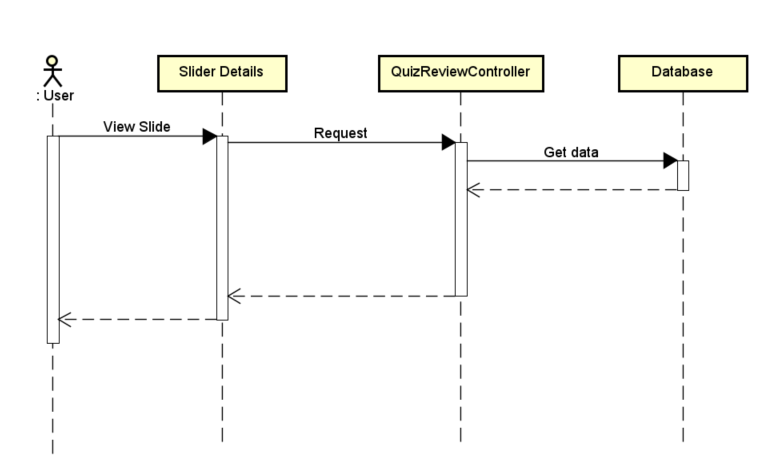
***Class Slider Details***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getSliderByID | * input: [SWP].[dbo].[Slider] * output:Get Slider Details * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

**Class getSlider**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: Slider details  - use:Get Slider Details |
| 02 | doPost | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  - use: processRequest |
| 03 | doGet | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  use: processRequest |

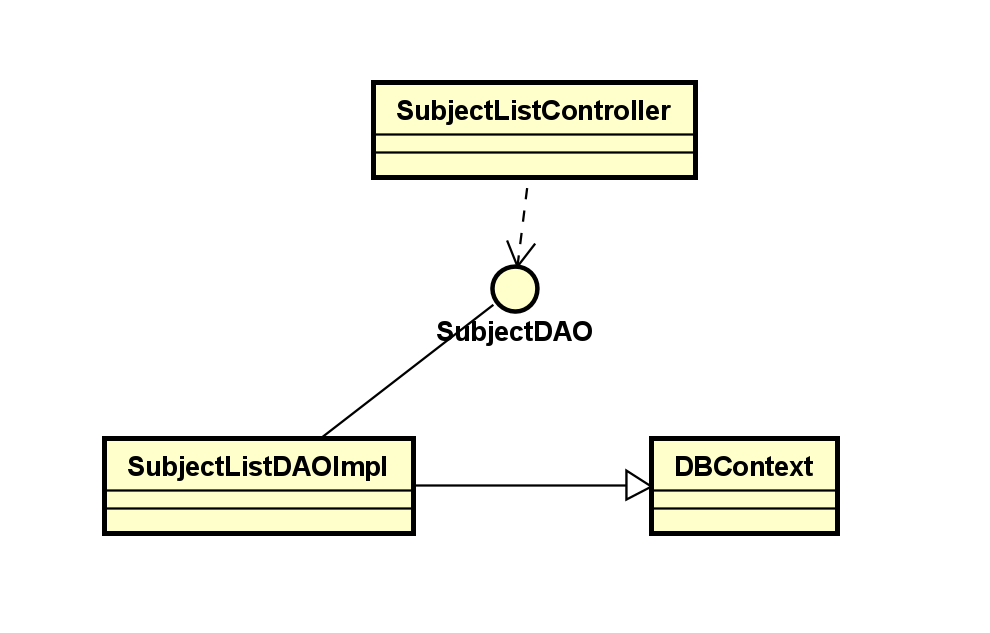
### c. Sequence Diagram(s)



### d. Database queries

# **23.**Subjects List

### a. Class Diagram



### b. Class Specifications

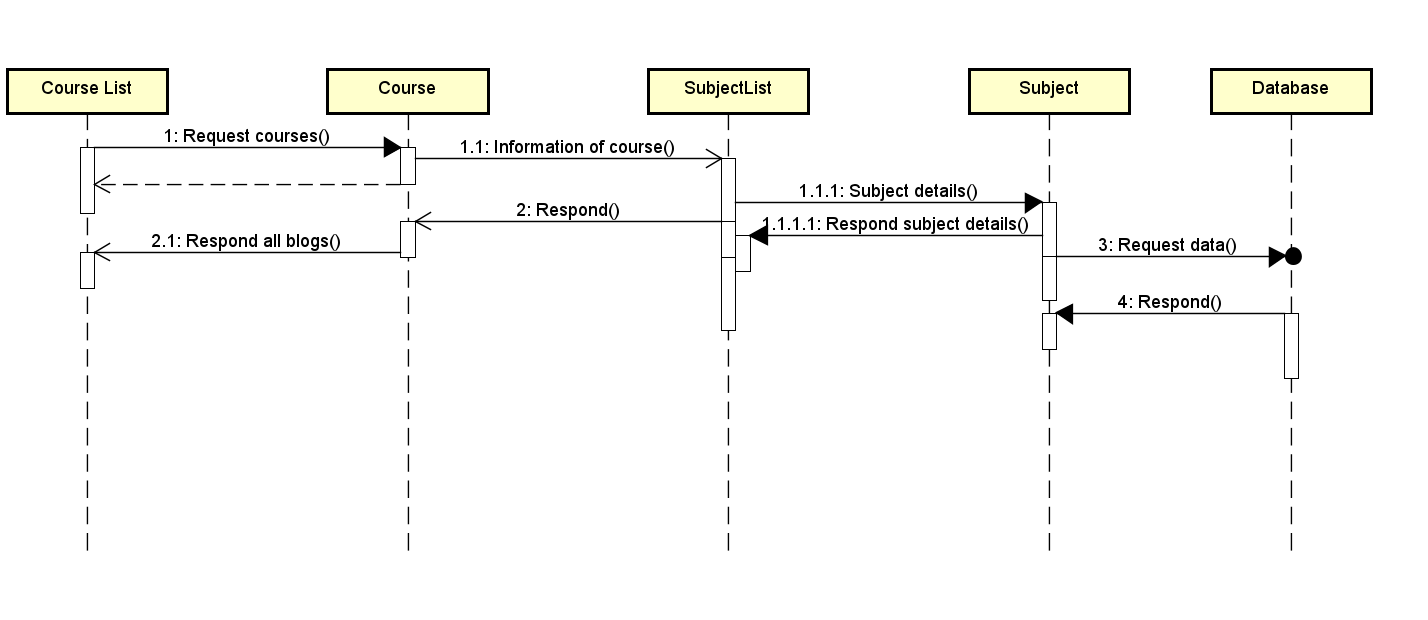
***Class SubjectList***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllSubject | * input: [SWP].[dbo].[Subject] * output:get all Subject * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

**Class Subjectlistcontroller**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: all subject  - use:Get all subject |
| 02 | doPost | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  - use: processRequest |
| 03 | doGet | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  use: processRequest |

### c. Sequence Diagram(s)

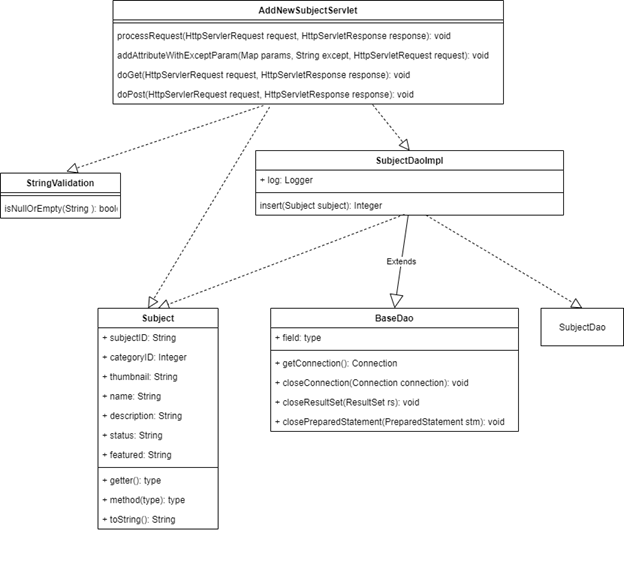


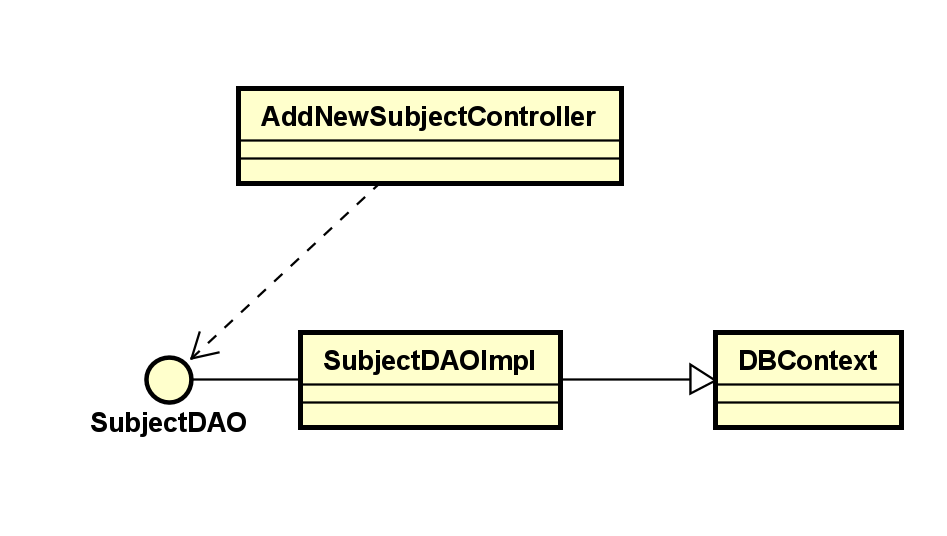
### d. Database queries

select \* from [Subject]// Get all Subject from DB

# **24.**New Subject

### a. Class Diagram





### b. Class Specifications

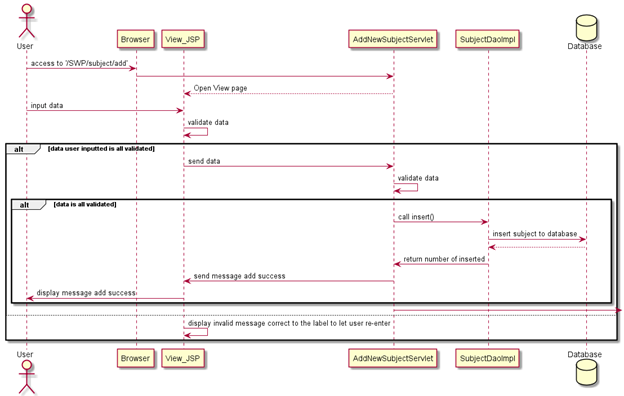
#### AddNewSubjectServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |
| 02 | addAttributeWithExceptParam | - input:  + Map<String, String[]> params  + String except  + HttpServletRequest request  - output: None  - use: N.A |
| 03 | addAttribute | - input:  + Map<String, Object> data  + HttpServletRequest request  - output: None  - use: N.A |

#### SubjectDaoImpl Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insert | - input:  + Subject subject  - output: number of subject inserted to database  - use: getConnection, PreparedStatement, Connection |

### c. Sequence Diagram(s)



### d. Database queries

"INSERT INTO [dbo].[Subject] (subjectID, categoryID, thumbnail, name, Description, "

+ "status, featured, owner)\n"

+ "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

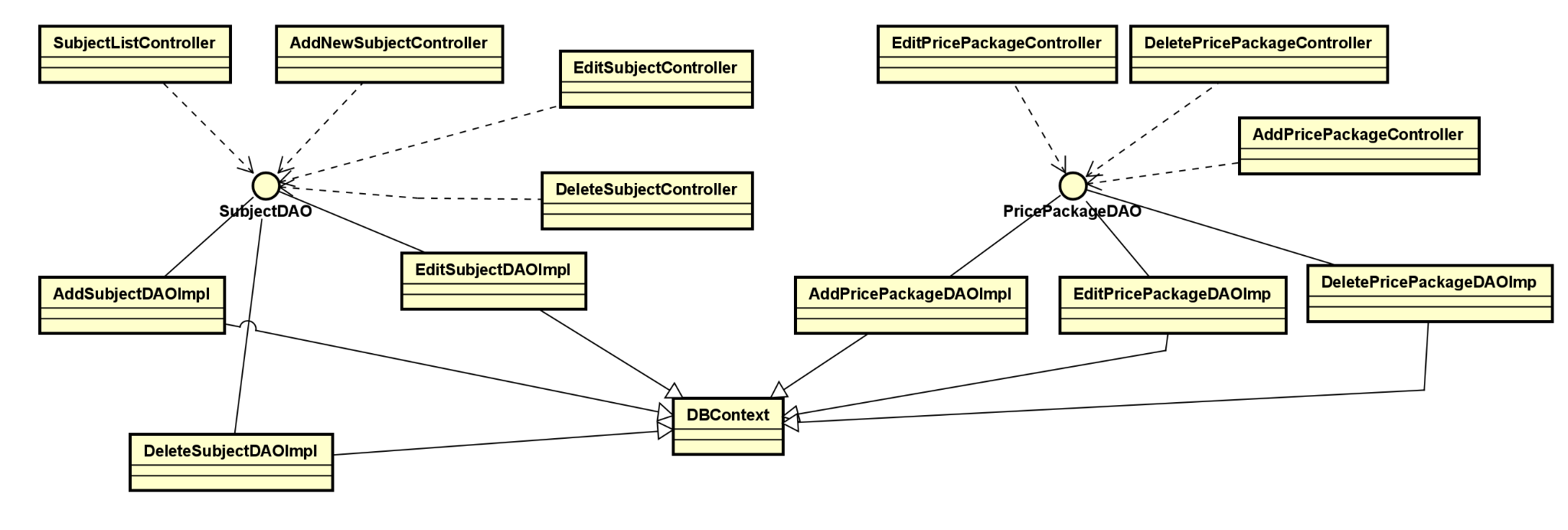
checkExist

SELECT COUNT(\*) as total FROM [dbo].[Subject]"

+ " WHERE subjectID = ?";

# **25.**Subject Details

### a. Class Diagram



### b. Class Specifications

class UpdateSubject

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | editSubject | * input: [SWP].[dbo].[Subject] * output:edit Subject * use: DBContext, getConnection, PrepareStatement * This class is used to edit a subject |
|  |  |  |

class DeleteSubject

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteSubject | * input: [SWP].[dbo].[Subject] * output:delete a Subject * use: DBContext, getConnection, PrepareStatement * This class is used to delete a subject |
|  |  |  |

class AddPricePackage

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addPricePackage | * input: [SWP].[dbo].[PricePackage] * output: add a price package * use: DBContext, getConnection, PrepareStatement * this class is used to add a price package |
|  |  |  |

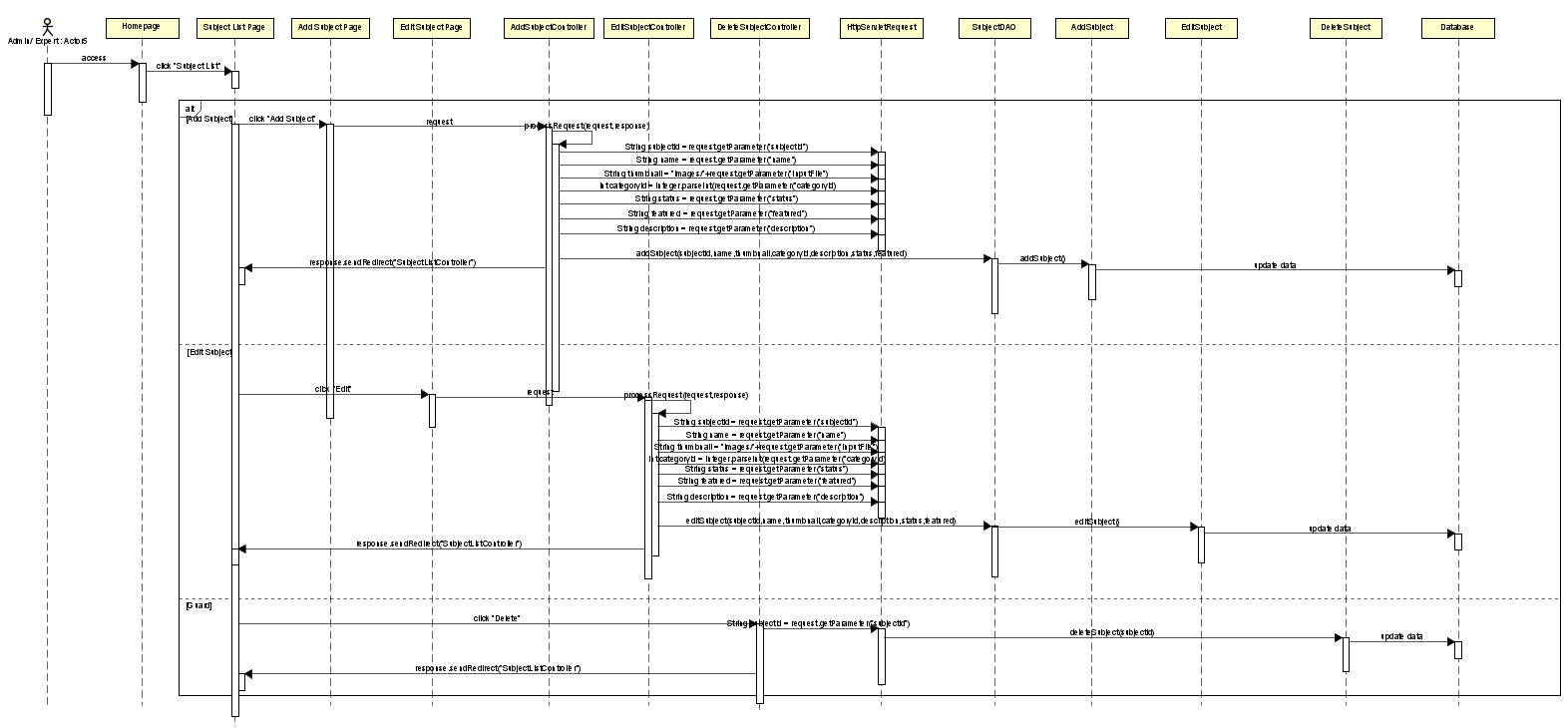
class EditPricePackage

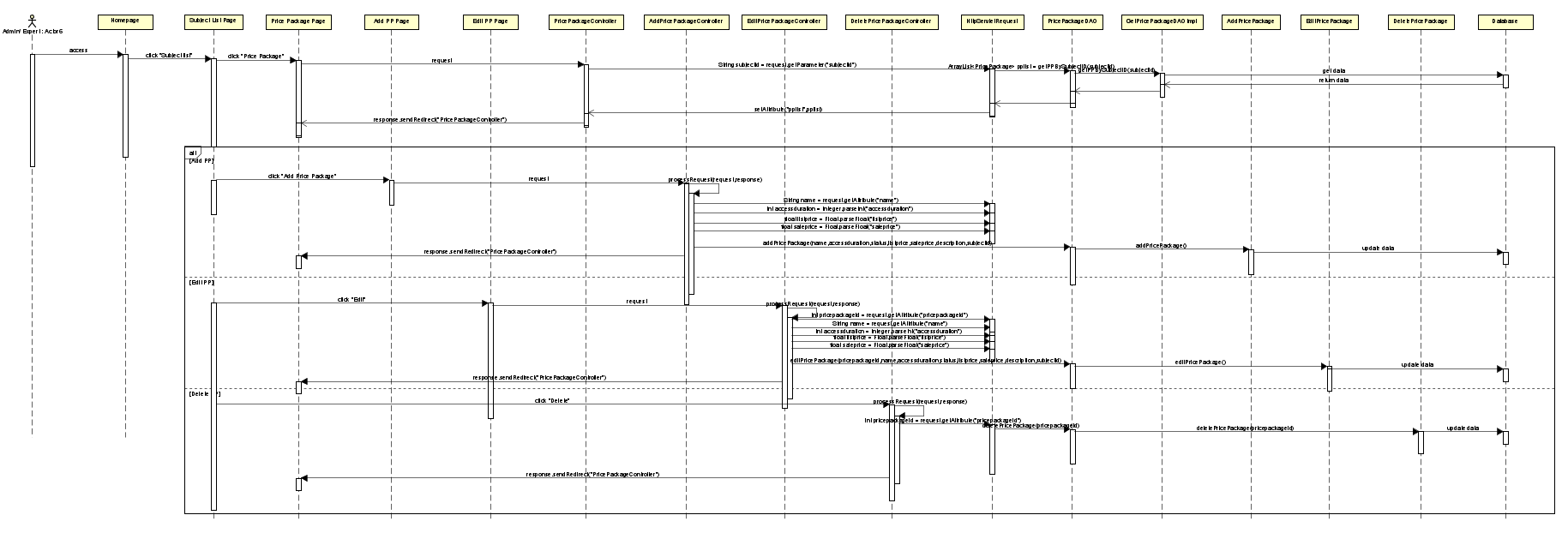
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | editPricePackage | * input: [SWP].[dbo].[PricePackage] * output:edit a Price Package * use: DBContext, getConnection, PrepareStatement * This class is used to edit a price package |
|  |  |  |

class DeletePricePackage

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deletePricePackage | * input: [SWP].[dbo].[PricePackage] * output: delete a price package * use: DBContext, getConnection, PrepareStatement * this class is used to delete a price package |
|  |  |  |

### c. Sequence Diagram(s)





### d. Database queries

- update [Subject] set categoryID = ?, thumbnail = ?, name = ?, Description = ?, status = ?, featured = ? where subjectID = ? (Update subject)

- delete from Subject where subjectID = ? (Delete subject)

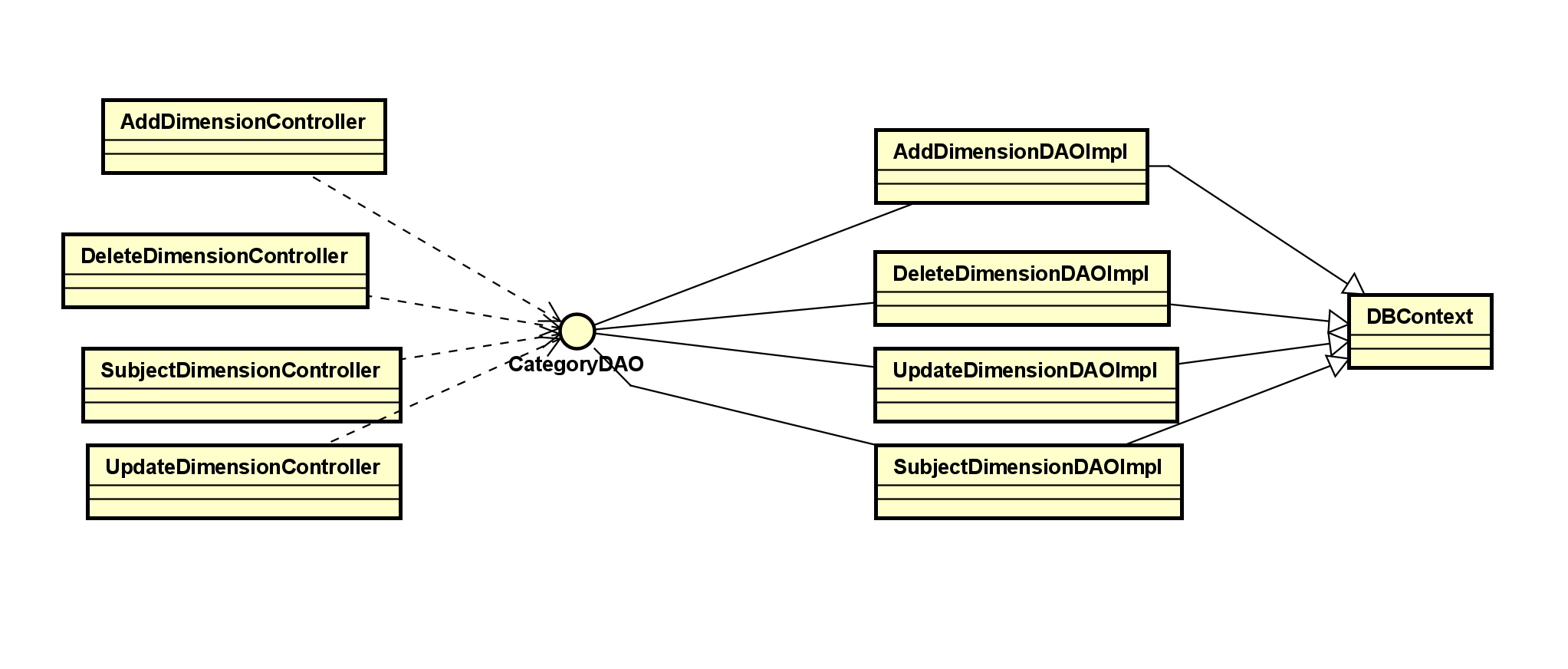
- insert into PricePackage values (?,?,?,?,?,?,?) (Add a price package)

- update PricePackage set [name] = ?, accessDuration = ?, [status] = ?, listPrice = ?, salePrice = ?, description = ?, subjectID = ? where pricePackageID = ? (Edit a price package)

- delete from PricePackage where pricePackageID = ? (Delete a price package)

# **26.**Subject Dimension

### a. Class Diagram



### b. Class Specifications

**class SubjectDimension**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllCategories | * input: [SWP].[dbo].[Category] * output:get all Category * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

**class AddDimension**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addCategory | * input: [SWP].[dbo].[Category] * output:add a category into database * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

**class GetDimensionById**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllCategories | * input: [SWP].[dbo].[Category] * output:get a dimension by its id * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

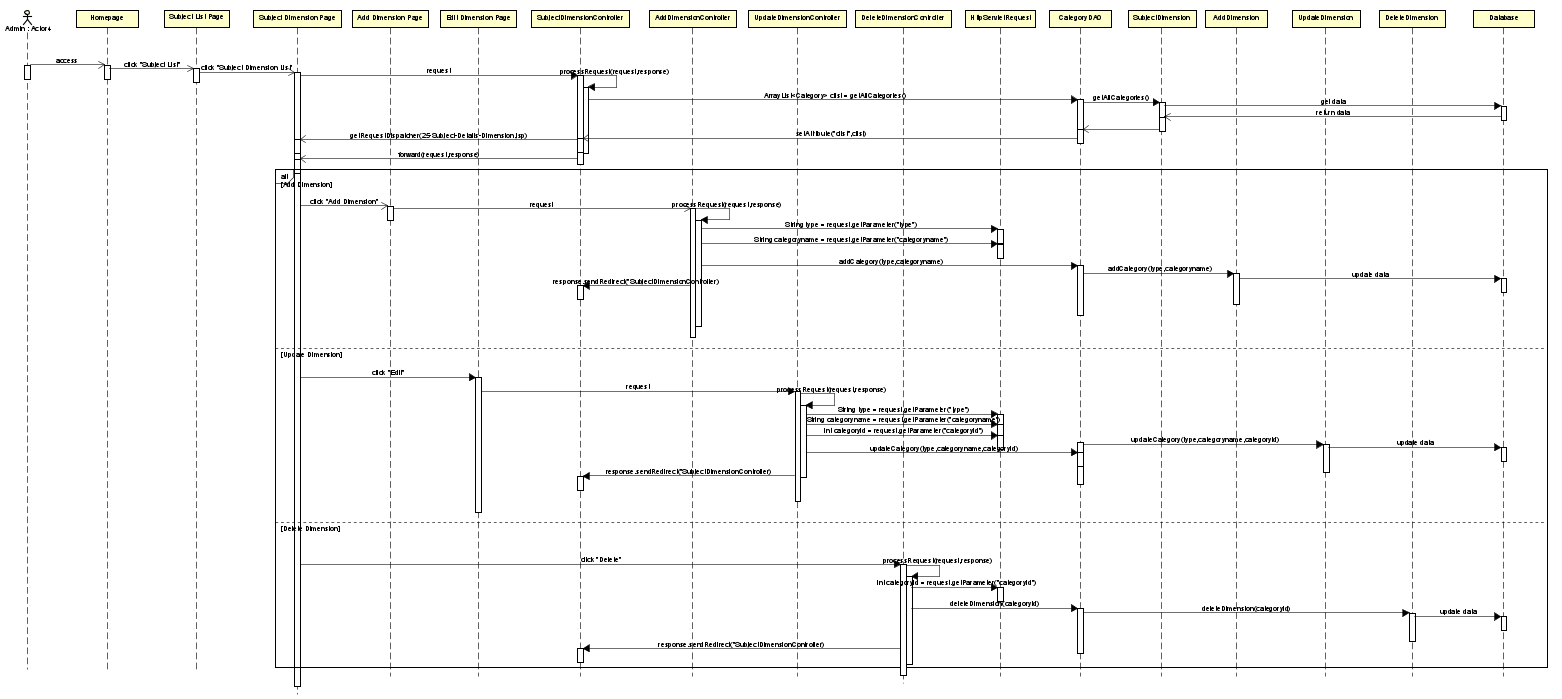
**class UpdateDimension**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateCategory | * input: [SWP].[dbo].[Category] * output:update a dimension and save it into database * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

**class DeleteDimension**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteDimension | * input: [SWP].[dbo].[Category] * output:get a Category from database * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

### c. Sequence Diagram(s)



### d. Database queries

SELECT \* FROM [Category] (This is used to get all Dimensions)

SELECT \* FROM [Category] WHERE categoryId = ? (Get dimension using the given id)

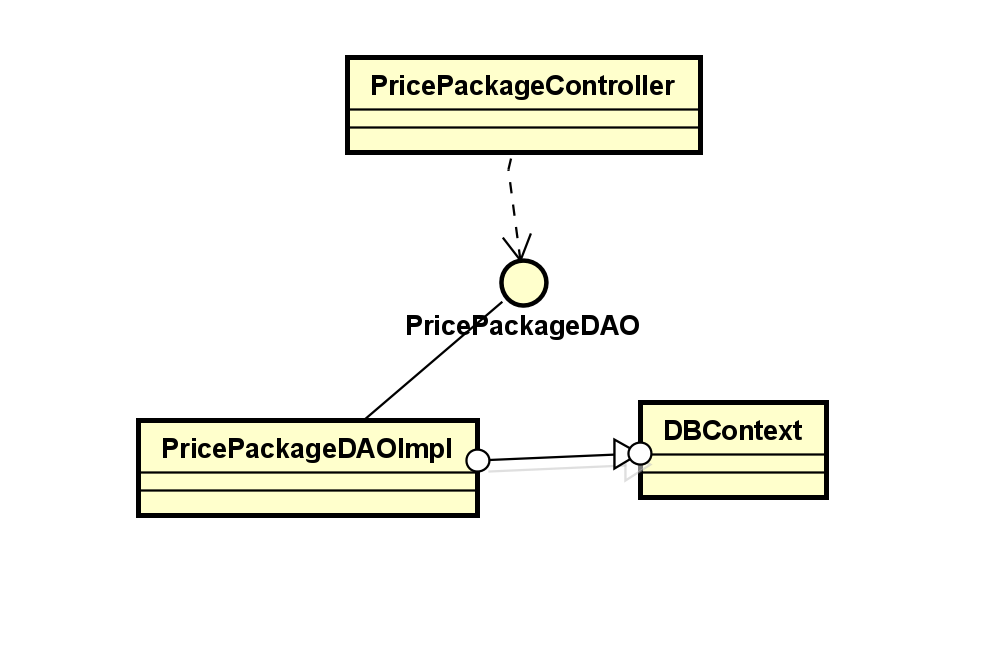
UPDATE Category SET [type] = ?, categoryName = ? WHERE categoryId = ? (Edit a dimension)

INSERT INTO Category VALUES (?,?) (Add dimension into database)

DELETE FROM Category WHERE categoryId = ? (Delete a dimension)

# **27.**Price Package

### a. Class Diagram



### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

# **28.**Subject Lessons

### a. Class Diagram

### b. Class Specifications

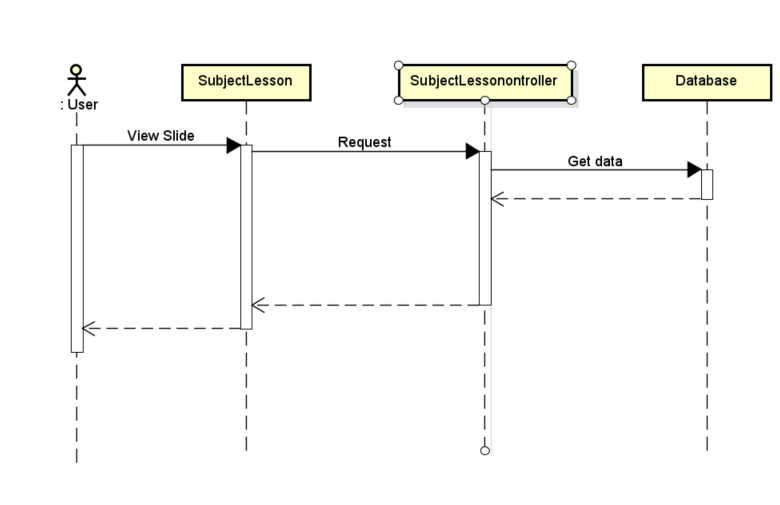
***Class SubjectLeson***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllSubject | * input: [SWP].[dbo].[Subject] * output:get all SubjectLesson * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

**Class SubjectLessonController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: all subject  - use:Get all subject |
| 02 | doPost | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  - use: processRequest |
| 03 | doGet | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  use: processRequest |

### c. Sequence Diagram(s)



### d. Database queries

"SELECT \* FROM [dbo].[PricePackage]"

+ " WHERE pricePackageID = ?";

"UPDATE [dbo].[PricePackage]"

+ " SET name = ?, "

+ " accessDuration = ?,"

+ " status = ?,"

+ " listPrice = ?,"

+ " salePrice = ?,"

+ " description = ?";

**29.**Lesson Details

### a. Class Diagram

### b. Class Specifications

#### LessonDetailServlet Class

### 

| No | Method | Description |
| --- | --- | --- |
| 01 | processRequest | - input:+ HttpServletRequest request+ HttpServletResponse response- output: N.A- use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |

### 

#### LessonDaoImpl Class

### 

| No | Method | Description |
| --- | --- | --- |
| 01 | findLessonById | - input:+ String lessonId- output: Lesson with specified ID- use: getConnection, PreparedStatement, Connection |

### 

### c. Sequence Diagrams

### 

### d. Database queries

### 

### 

# **30.**Questions List

### a. Class Diagram

### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

# **31.**Question Details

### a. Class Diagram

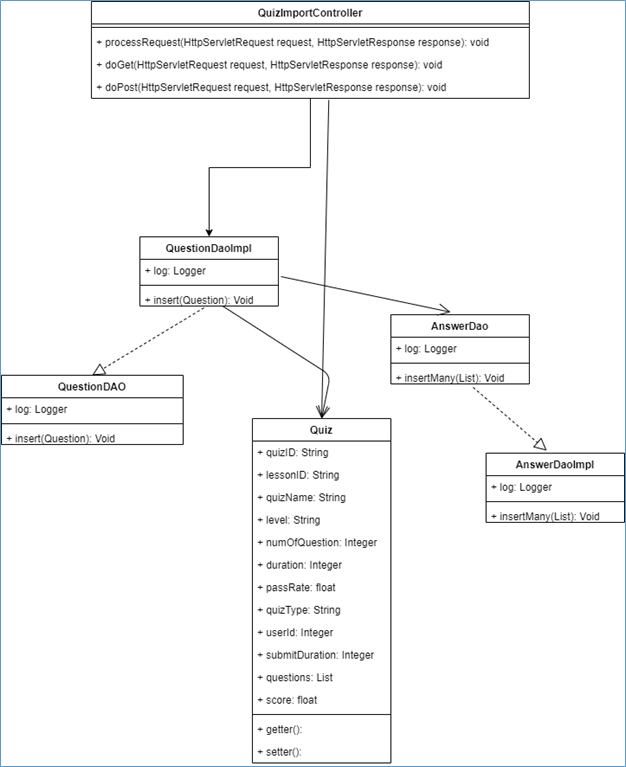
### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

# **32.**Question Import

### a. Class Diagram



### b. Class Specifications

#### QuizImportServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - Description: process common for both POST and GET method  - input:  + HttpServletRequest request  + HttpServletResponse response  - output: N.A  - use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |

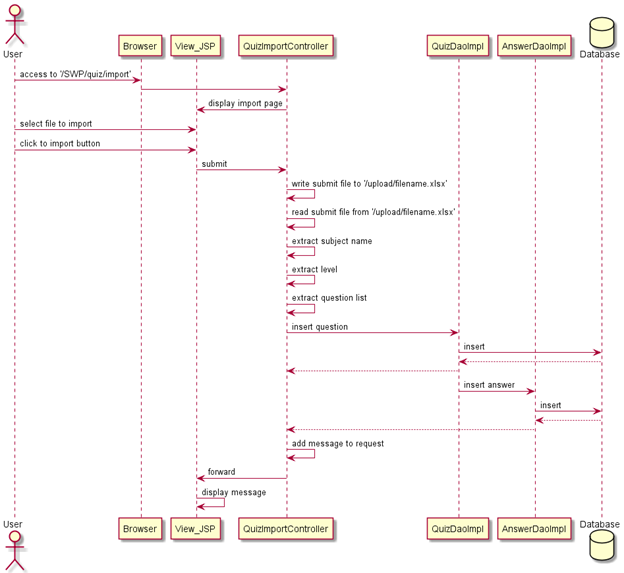
#### QuizDaoImpl Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insert | - Description: insert question to database  - input:  + Question question  - output: N/A  - use: getConnection, PreparedStatement, Connection |

#### AnswerDaoImpl Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertMany | - Description: insert list answer to database  - input:  + List<Answer> answers  - output: N/A  - use: getConnection, PreparedStatement, Connection |

### c. Sequence Diagram(s)



### d. Database queries

# **33.**Quizzes List

### a. Class Diagram

### b. Class Specifications

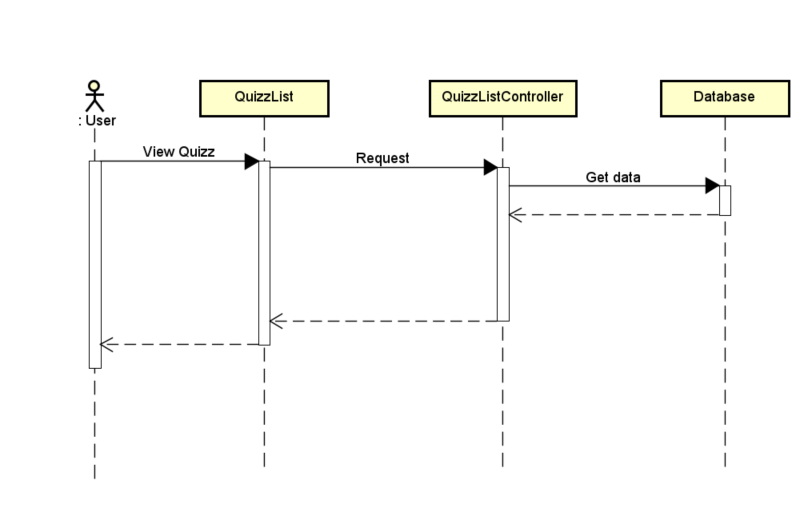
***Class QuizList***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllQuizz | * input: [SWP].[dbo].[Quizz] * output:get all Quizz * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

**Quizlistcontroller**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: all Quizz  - use:Get all Quizz |
| 02 | doPost | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  - use: processRequest |
| 03 | doGet | - input:  + HttpServletRequest request  + HttpServletResponse response  - output: None  use: processRequest |

### c. Sequence Diagram(s)



### d. Database queries

# **34.**Quiz Details

### a. Class Diagram

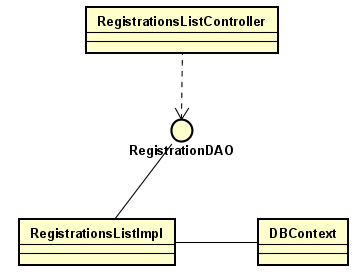
### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

# **35.**Registrations List

### a. Class Diagram

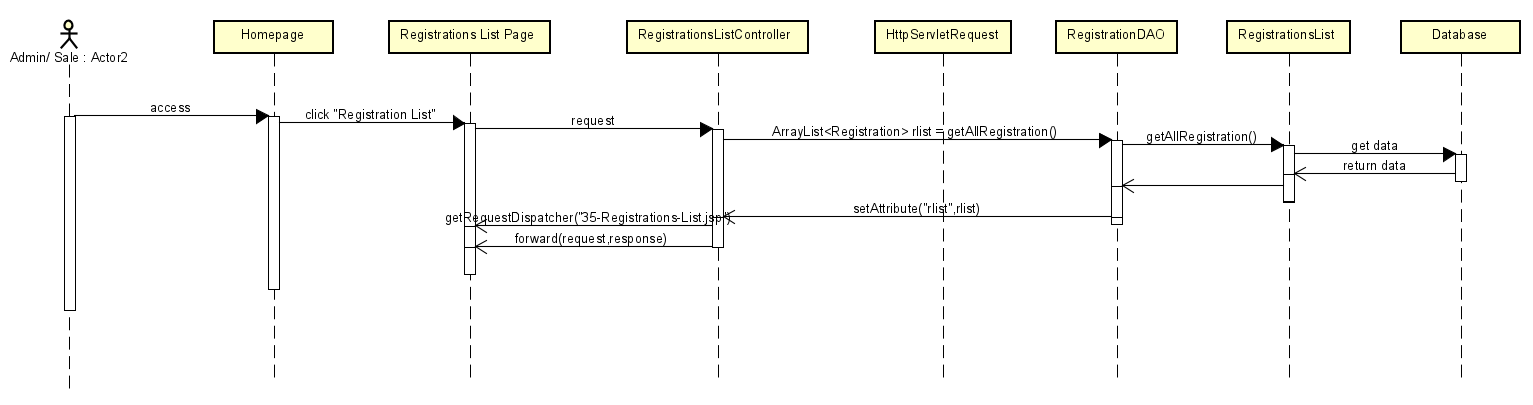


### b. Class Specifications

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | GetAllRegistration | * input: [SWP].[dbo].[Registration] * output: get all Registration of all Users * use: DBContext, getConnection, PrepareStatement |
|  |  |  |

### 

### c. Sequence Diagram(s)

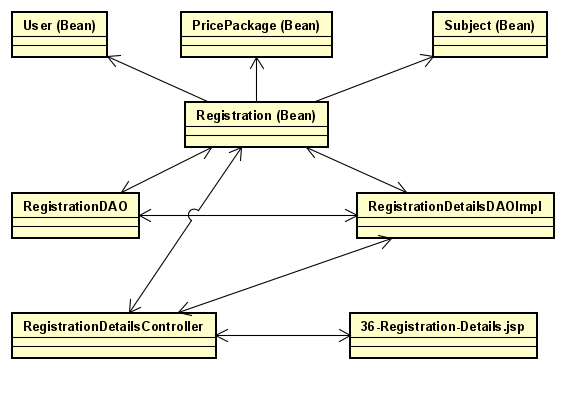


### d. Database queries

* Select \* from Registration

# **36.**Registration Details

### a. Class Diagram



### b. Class Specifications

*Class RegistrationDetailsDAOImpl*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *GetDetailFromRegistration* | * *input: [SWP].[dbo].[Subject] JOIN [PricePackage] JOIN [User] JOIN [Registration]* * *output: get all information by RegistrationID when user buy course* * *use: DBContext, getConnection, PrepareStatement* |
|  |  |  |

### 

### c. Sequence Diagram(s)

### d. Database queries

SELECT Subject.name,\n

+ " PricePackage.listPrice,PricePackage.salePrice,\n"

+ " [User].Fullname,[User].Gender,[User].Email,[User].Phone,\n"

+ " Registration.regisTime,Registration.status,Registration.validFrom,Registration.validto\n"

+ " \n"

+ " FROM Registration\n"

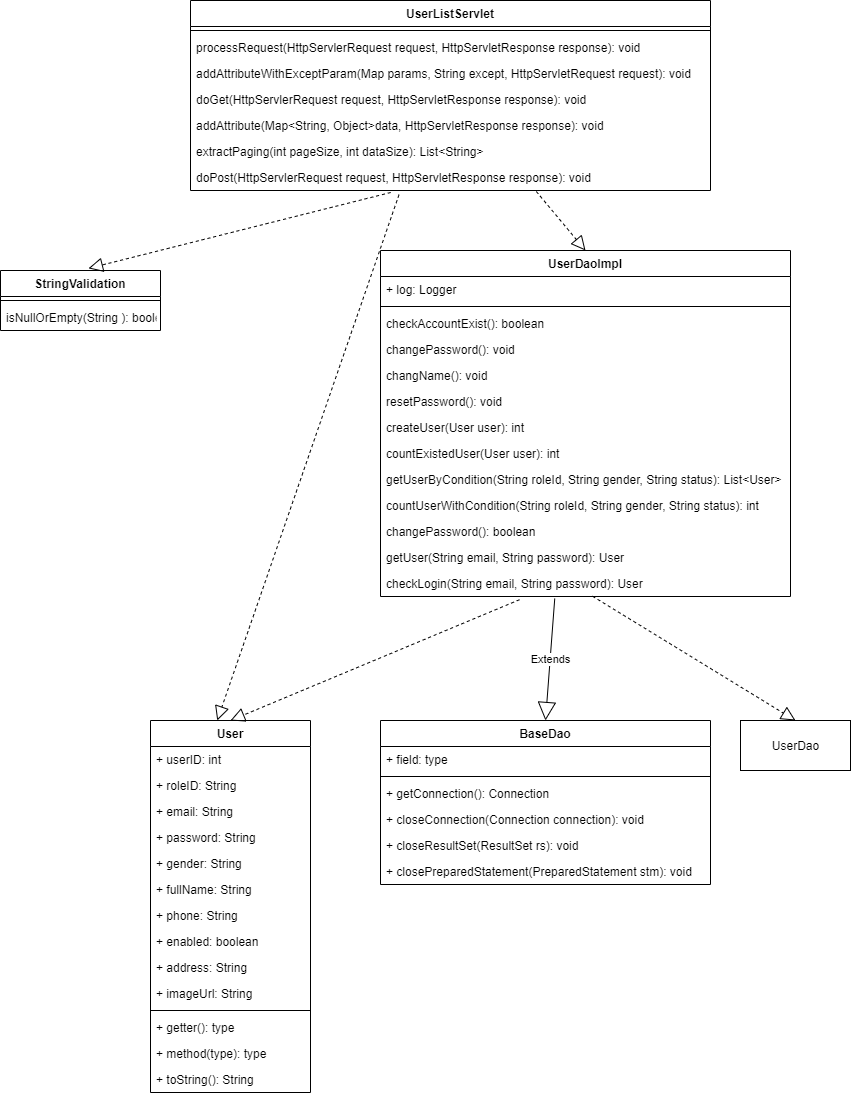
+ " INNER JOIN Subject ON Registration.subjectID = Subject.subjectID\n"

+ " INNER JOIN PricePackage ON Registration.pricepackageID = PricePackage.pricePackageID \n"

+ " INNER JOIN [User] ON Registration.UserID = [User].UserID where registrationID= ?

# **37.**Users List

### a. Class Diagram



### b. Class Specifications

***User List Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest | * input:   + HttpServletRequest request  + HttpServletResponse response   * output: N.A * use: isNullOrEmpty, counExistedUser, createUser, addAttributeWithExceptParam |
| 02 | doPost | * input:   + HttpServletRequest request  + HttpServletResponse response   * output: None * use: processRequest |
| 03 | doGet | * input:   + HttpServletRequest request  + HttpServletResponse response   * output: None   use: processRequest |
| 04 | addAttributeWithExceptParam | * input:   + Map<String, String[]> params  + String except  + HttpServletRequest request   * output: None * use: N.A |
| 05 | addAttribute | * input:   + Map<String, Object> data  + HttpServletRequest request   * output: None * use: N.A |
| 06 | extractPaging | * input:   + int pageSize  + int dataSize   * output: List of page index * use: N.A |

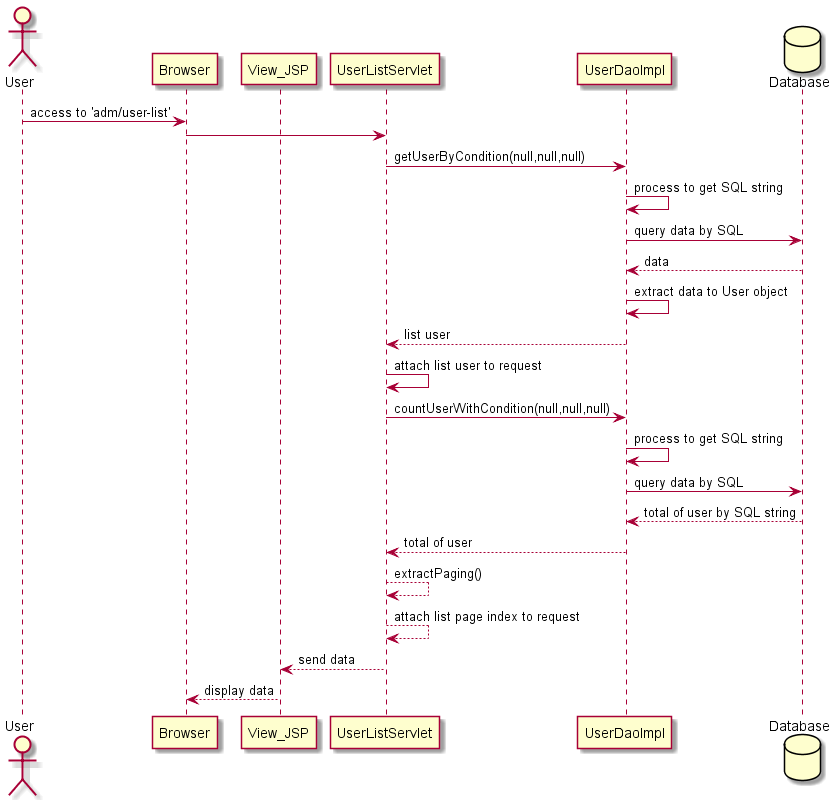
***StringValidation Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isNullOrEmpty | * input:   + String data   * output: true if data inputted is null or empty and false if it’s not * use: N.A |

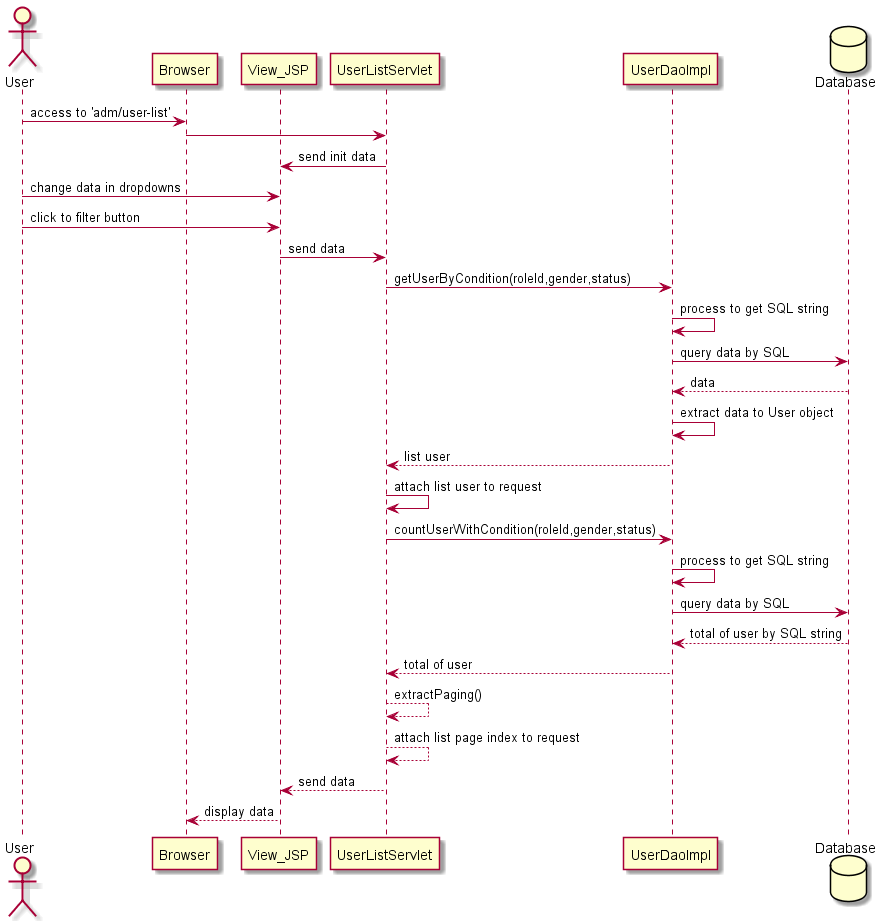
***UserDaoImpl Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createUser | * input:   + User user   * output: number of created user * use: getConnection, PreparedStatement, Connection |
| 02 | countExistedUser | * input:   + User user   * output: number of existed user in database   use: getConnection, PreparedStatement, Connection, ResultSet |

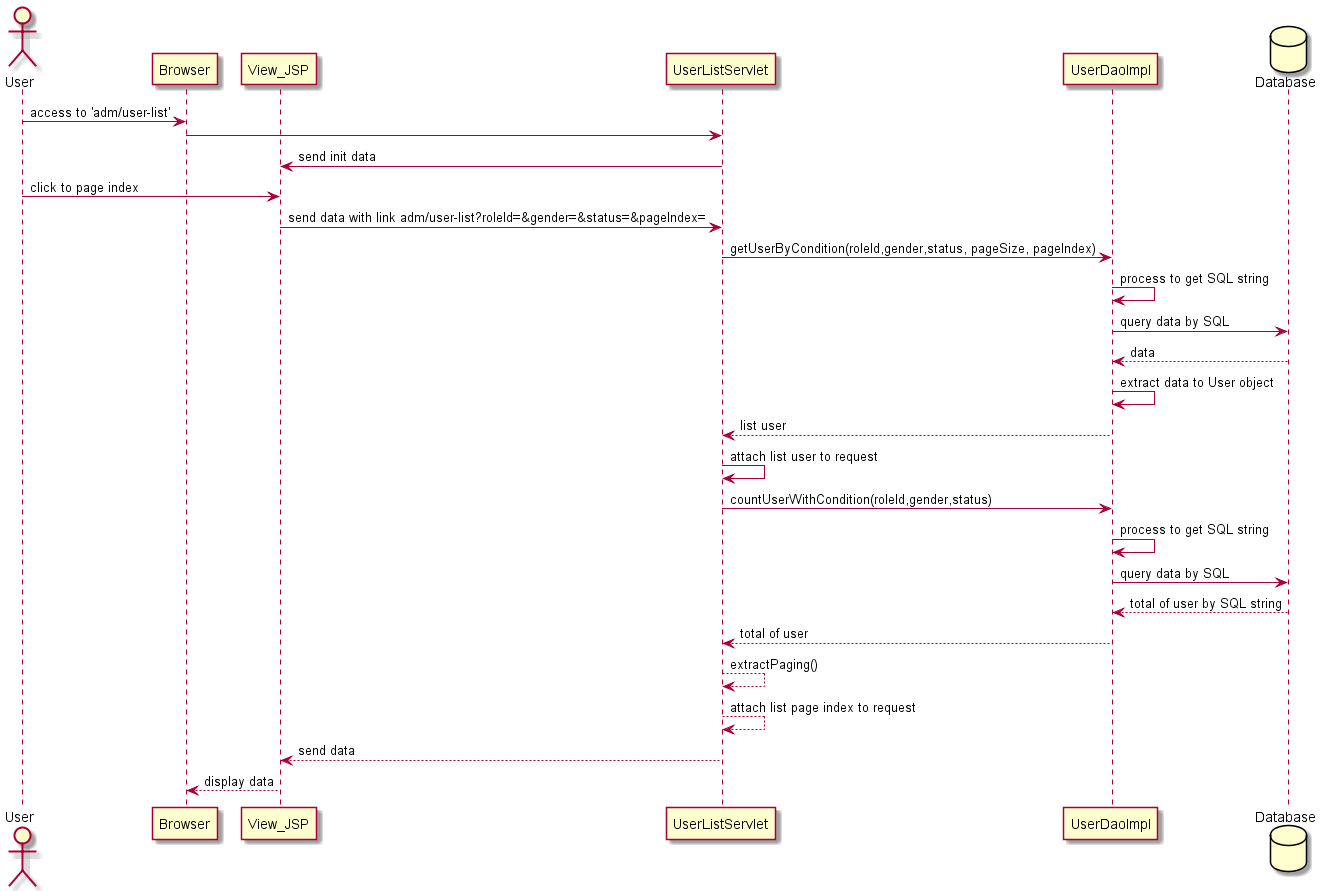
### c. Sequence Diagram(s)



- Filter



- Paging



d. Database queries

"SELECT \* FROM dbo.[User]"

+ "WHERE email = ? AND password = ? AND [enabled] = 'True' ";

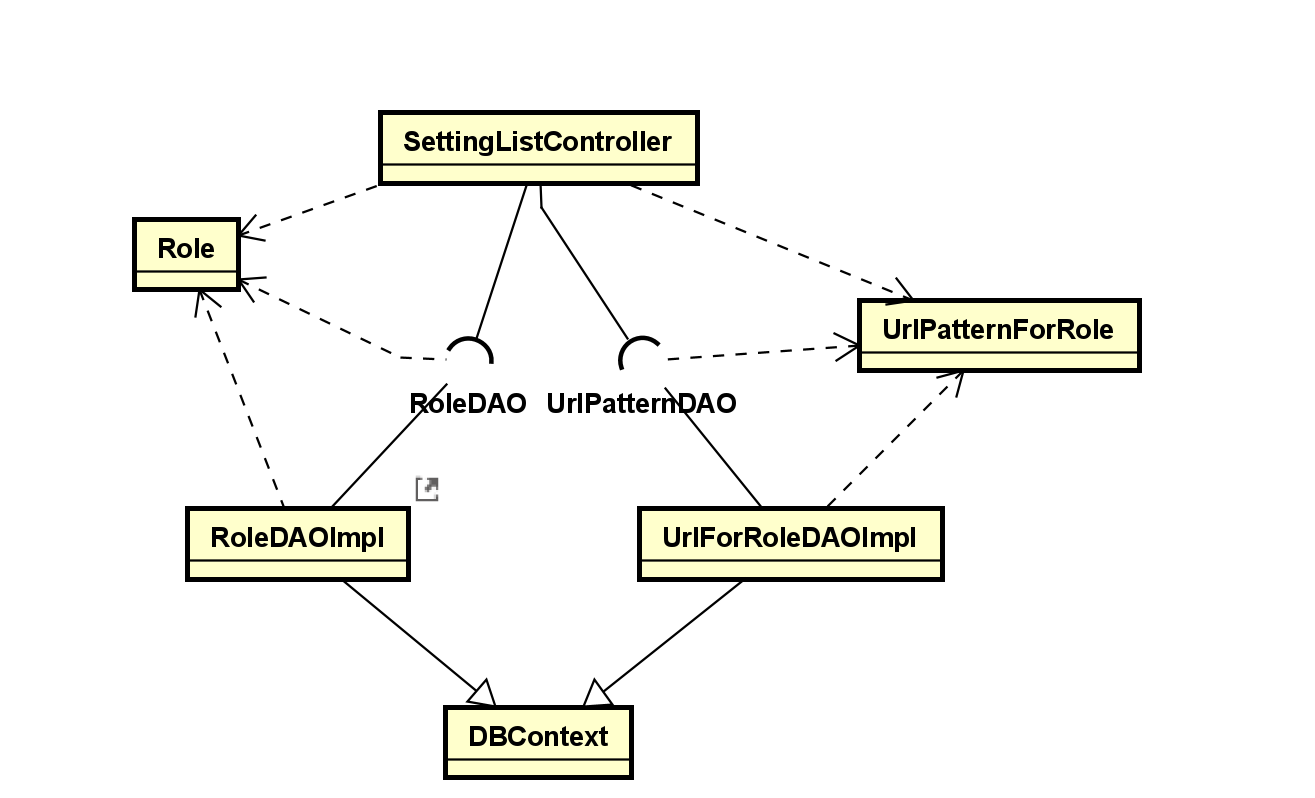
### d. Database queries

"SELECT \* FROM dbo.[User]"

+ "WHERE email = ? AND password = ? AND [enabled] = 'True' ";

# **39.**Settings List

### a. Class Diagram



### b. Class Specifications

*Class RoleDAOImpl*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRoleDesList() | - input: none  - output: arraylist get all information of all Role  - use: DBContext, getConnection, PrepareStatement, ResultSet |

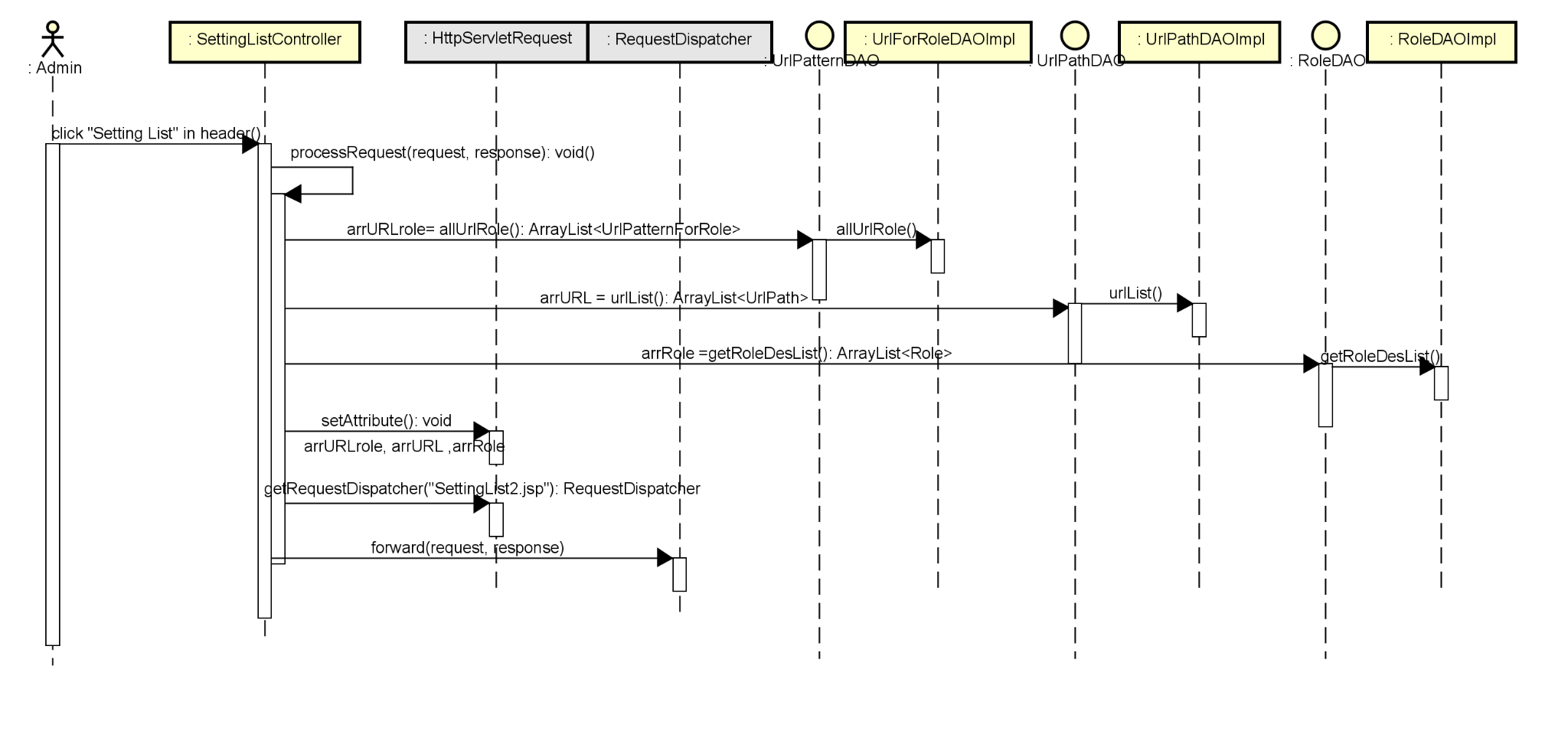
*Class UrlFilterDAOImpl*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | ArrayList<UrlPatternForRole> allUrlRole() | - input: none  - output:: arraylist get all information of all URL pattern  - use: DBContext, getConnection, PrepareStatement, URLPatternDAO |

*Class SettingListController*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | processRequest | - input: request, response  - output: arraylist all role, arraylist of all url pattern  - use: requestDispatcher |

c. Sequence Diagram(s)

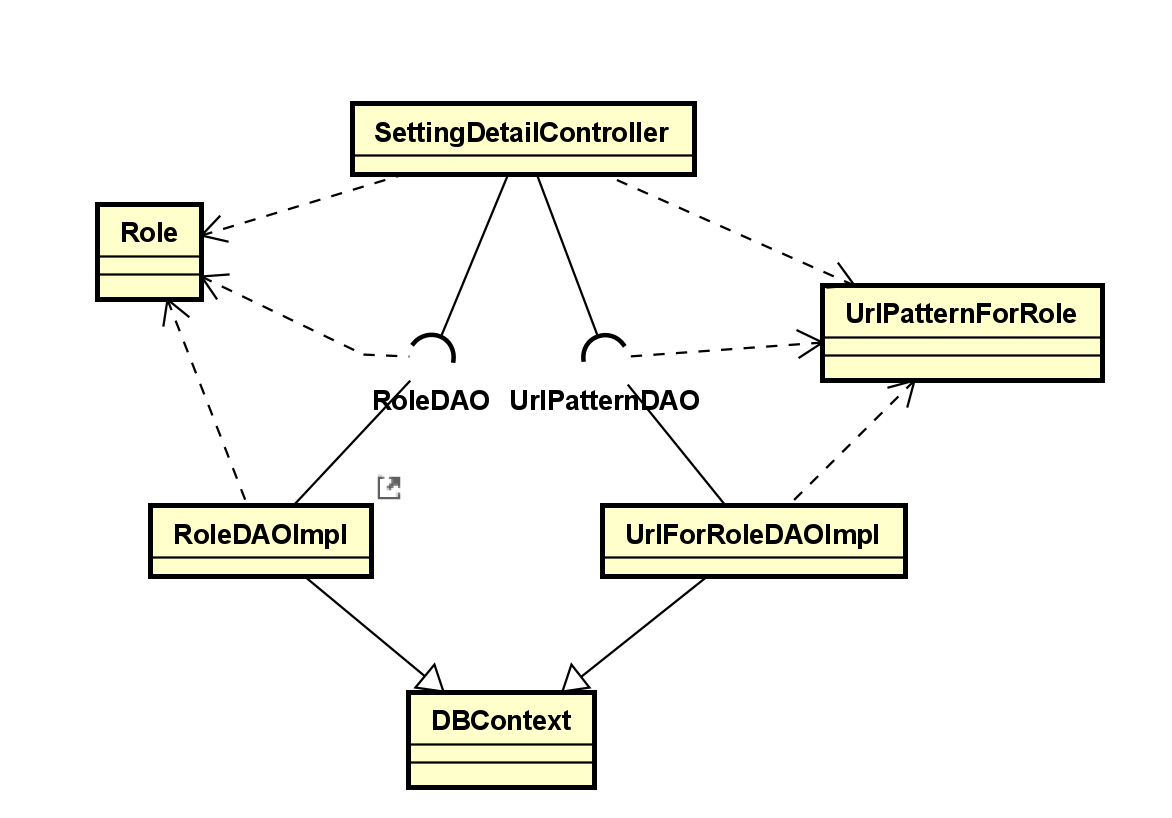


### d. Database queries

* SELECT \* FROM dbo.[Role]
* SELECT \* from UrlPatternOfRole

# **40.**Setting Details

### a. Class Diagram



### b. Class Specifications

*Class RoleDAOImpl*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRoleDesList() | - input: none  - output: arraylist get all information of all Role  - use: DBContext, getConnection, PrepareStatement, ResultSet |

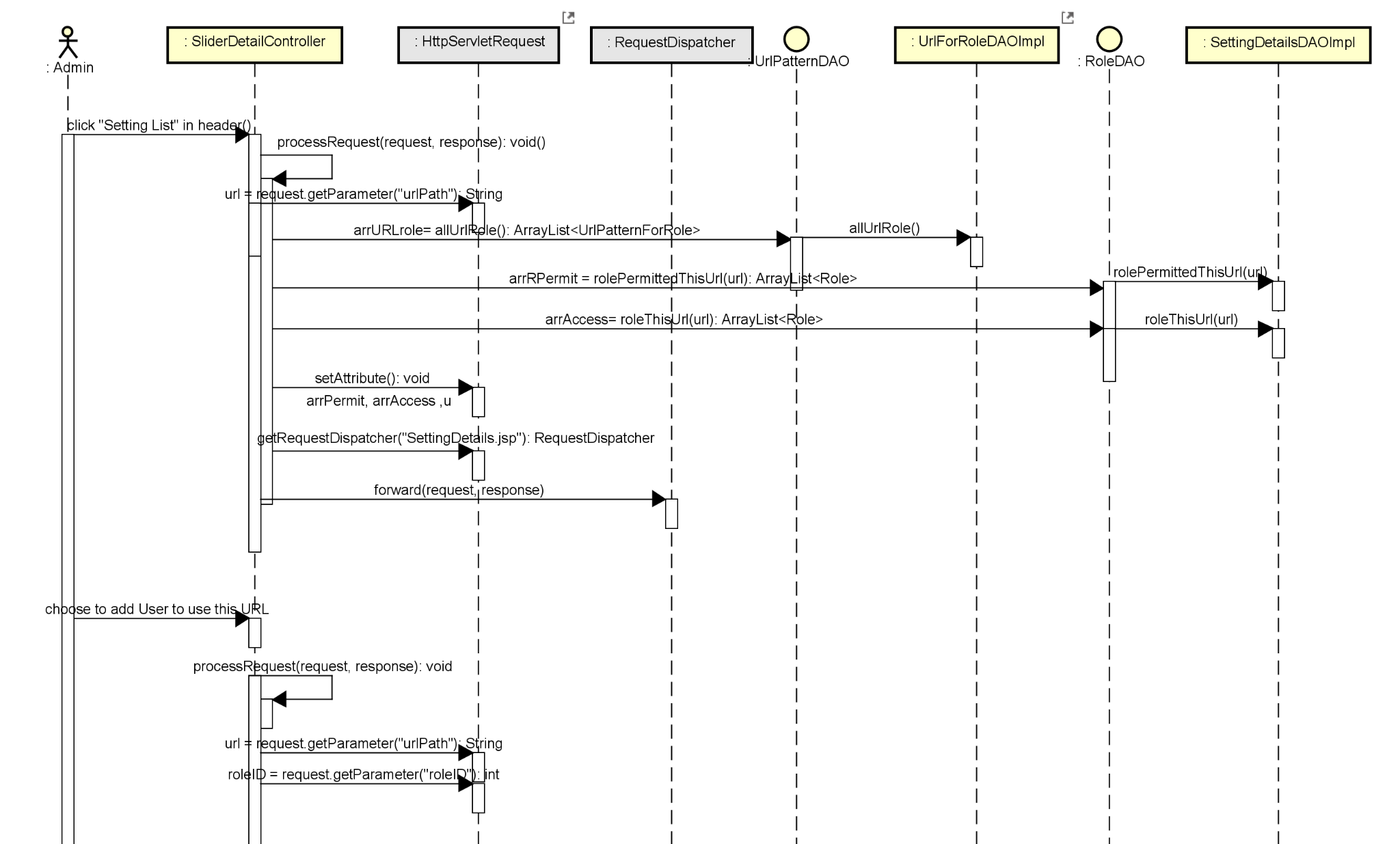
*Class UrlFilterDAOImpl*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | ArrayList<UrlPatternForRole> urlRoleUrl | - input: String urlPattern  - output:: arraylist get all information of all URL pattern  - use: DBContext, getConnection, PrepareStatement, URLPatternDAO |

*Class SettingListController*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | processRequest | - input: request, response  - output: arraylist all role, arraylist of all url pattern  - use: requestDispatcher |

### c. Sequence Diagram(s)



### d. Database queries

* SELECT \* from UrlPatternOfRole where urlPattern = ?
* SELECT \* FROM dbo.[Role]
* SELECT \* from UrlPatternOfRole

# III. Database Tables

## 1. <Question>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | questionID | int |  | ✔ |  | ✔ | primarykey |
| 2 | quizID | int |  | ✔ |  | ✔ | foreign key (quizID) references Quiz(quizID) |
| 3 | content | nvarchar | 50 |  | ✔ |  |  |
| 4 | "level" | nvarchar | 50 |  | ✔ |  |  |
| 5 | lessoniD | int |  | ✔ | ✔ | ✔ | foreign key (lessoniD) references Lesson(lessoniD) |
| 6 | subjectID | int |  | ✔ | ✔ | ✔ | foreign key (subjectID) references "Subject"(subjectID) |
| 7 | dimensionID | int |  | ✔ | ✔ | ✔ | foreign key (dimensionID) references SubjectDimension(dimensionID) |

## 2. <Category>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | categoryID | int |  | ✔ |  | ✔ | primarykey |
| 2 | categoryName | nvarchar | 50 |  | ✔ |  |  |
| 3 | type | nvarchar | 36 |  | ✔ |  |  |

## 3. <Answer>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | answerID | int |  | ✔ |  | ✔ | primarykey |
| 2 | questionID | int |  | ✔ |  | ✔ | Foreign key (questionID) references Question(questionID) |
| 3 | result | nvarchar | 50 |  | ✔ |  |  |
| 4 | content | nvarchar | 500 |  | ✔ |  |  |

## 4. <Blog>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | blogID | int |  | ✔ |  | ✔ | primary key |
| 2 | userID | int |  | ✔ |  | ✔ | Foreign key (userID) references "User"(userID) |
| 3 | fullname | nvarchar | 50 |  | ✔ |  |  |
| 4 | categoryID | int |  | ✔ | ✔ | ✔ | Foreign key (categoryID) references Category(categoryID) |
| 5 | thumbnail | nvarchar | 400 |  |  |  |  |
| 6 | title | nvarchar | 300 |  | ✔ |  |  |
| 7 | updatedDate | nvarchar | 50 |  | ✔ |  |  |
| 8 | blogContent | nvarchar | 3000 |  | ✔ |  |  |
| 9 | flag | bit |  |  |  |  |  |
| 10 | status | bit |  |  |  |  |  |

## 5. <[Subject]>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | subjectID | int |  | ✔ |  | ✔ | primary key |
| 2 | categoryID | int |  | ✔ |  | ✔ | Foreign key (categoryID) references Category(categoryID) |
| 3 | thumbnail | nvarchar | 100 |  | ✔ |  |  |
| 4 | name | nvarchar | 200 |  | ✔ |  |  |
| 5 | "Description" | nvarchar | 600 |  | ✔ |  |  |
| 6 | Status | nvarchar | 20 |  |  |  |  |
| 7 | Featured | nvarchar | 20 |  |  |  |  |

## 6. <Lesson>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | lessonID | int |  | ✔ |  | ✔ | primarykey |
| 2 | lessonType | nvarchar | 20 | ✔ |  | ✔ |  |
| 3 | lessonName | nvarchar | 100 |  | ✔ |  |  |
| 4 | status | nvarchar | 50 |  | ✔ |  |  |
| 5 | youtubeLink | nvarchar | 1000 |  | ✔ |  |  |
| 6 | content | nvarchar | 4000 |  | ✔ |  |  |
| 7 | subjectID | nvarcha | 20 | ✔ |  |  | Foreign key (subjectID) references [Subject](subjectID) |

## 7. <PricePackage>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | pricePackageID | int |  | ✔ | ✔ | ✔ | primarykey |
| 2 | name | nvarchar | 50 |  | ✔ |  |  |
| 3 | accessDuration | int |  |  |  |  |  |
| 4 | "status" | nvarchar | 20 |  | ✔ |  |  |
| 5 | listPrice | float, | 50 |  | ✔ |  |  |
| 6 | salePrice | float, | 50 |  | ✔ |  |  |
| 7 | "description" | nvarchar | 50 |  | ✔ |  |  |
| 8 | subjectID | nvarchar | 20 | ✔ |  | ✔ | Foreign key (subjectID) references "Subject"(subjectID) |

## 8. <Quiz>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | quizID | int |  | ✔ | ✔ | ✔ | primarykey |
| 2 | lessonID | int |  | ✔ | ✔ | ✔ | Foreign key (lessonID) references Lesson(lessonID) |
| 3 | quizName | nvarchar | 50 |  | ✔ |  |  |
| 4 | [level] | nvarchar | 50 |  | ✔ |  |  |
| 5 | numOfQuestion | int |  |  |  |  |  |
| 6 | duration | int |  |  |  |  |  |
| 7 | passRate | float |  |  |  |  |  |
| 8 | quizType | nvarchar | 50 |  | ✔ |  |  |
| 9 | lessonID | int |  | ✔ | ✔ | ✔ | Foreign key (lessonID) references Lesson(lessonID)) |

## 9. <Registration>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | registrationID | int |  | ✔ |  | ✔ | primarykey |
| 2 | regisTime | date |  |  |  |  |  |
| 3 | "status" | nvarchar |  |  |  |  |  |
| 4 | validFrom | nvarchar | 20 |  | ✔ |  |  |
| 5 | validto | nvarchar | 20 |  | ✔ |  |  |
| 6 | totalPrice | float |  |  | ✔ |  |  |
| 7 | UserID | int |  | ✔ |  | ✔ | foreign key (UserID) references "User"(UserID) |
| 8 | UserEmail | nvarchar | 50 |  | ✔ |  |  |
| 9 | SubjectID | nvarchar | 20 |  |  | ✔ | foreign key (subjectID) references [Subject](subjectID) |

## 10. <Role>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | roleID | nvarchar | 8 | ✔ |  | ✔ | primarykey |
| 2 | roleName | nvarchar | 50 |  | ✔ |  |  |
| 3 | roleDescription | nvarchar | 50 |  | ✔ |  |  |

## 11. <Score>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | scoreID |  |  |  |  | ✔ | primarykey |
| 2 | quizID | int |  | ✔ |  | ✔ | Foreign key (quizID) references Quiz(quizID) |
| 3 | score | int |  |  |  |  |  |
| 4 | userID | int |  |  | ✔ |  | foreign key (UserID) references "User"(UserID) |

## 12. <Slider>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | sliderID | int |  | ✔ |  | ✔ | primarykey |
| 2 | title | nvarchar | 50 |  | ✔ |  |  |
| 3 | backlink | nvarchar | 50 |  |  |  |  |
| 4 | "image" | nvarchar | 300 |  |  |  |  |

## 13. < "User" >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | UserID | int |  | ✔ |  | ✔ | primarykey |
| 2 | RoleID | nvarchar | 8 | ✔ |  | ✔ | foreign key (RoleID) references "Role"(RoleID) |
| 3 | Email | nvarchar | 50 |  | ✔ |  |  |
| 4 | "Password" | nvarchar | 20 |  |  |  |  |
| 5 | Gender | nvarchar | 20 |  | ✔ |  |  |
| 6 | Fullname | nvarchar | 50 |  | ✔ |  |  |
| 7 | Phone | nvarchar | 50 |  | ✔ |  |  |
| 8 | status | nvarchar | 10 |  |  |  |  |
| 9 | image | nvarchar | 100 |  |  |  |  |

## 14. < Setting >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | SettingID | int |  | ✔ |  | ✔ | primarykey |
| 2 | [order] | int |  | ✔ |  | ✔ | foreign key (regisID) references Registration(regisID) |
| 3 | "type" | nvarchar | 20 |  | ✔ |  |  |
| 4 | value | float |  |  | ✔ |  |  |
| 5 | "status" | nvarchar | 20 |  | ✔ |  |  |
| 6 | "Description" | nvarchar | 50 |  |  |  |  |