

**SMART SCHOOL**

**Software Design Document**

– HoChiMinh, Jun 2022–

**Table of Contents**

[I. Overview 3](#_gjdgxs)

[1. Code Packages 3](#_30j0zll)

[2. Database Schema 3](#_1fob9te)

[II. Code Designs 4](#_3znysh7)

[1. Login 4](#_2et92p0)

[a. Class Diagram 4](#_tyjcwt)

[b. Class Specifications 4](#_3dy6vkm)

[c. Sequence Diagram(s) 4](#_1t3h5sf)

[d. Database queries 5](#_4d34og8)

[2. Signup](#_2s8eyo1)

3[. S](#_2s8eyo1)earch

4[.](#_2s8eyo1) Update Account

5[.](#_2s8eyo1) Upload Post

6. Approve Post

7. Delete Post

8. List Post 5

[III. Database Tables 5](#_17dp8vu)

[1. tblFeedbackSystem 5](#_3rdcrjn)

[2. tblRole](#_26in1rg)

3. tblUser

4. tblReportType

5. tblReport

6. tblCategory

7. tblPost

8. tblCategoryJob

9. tblJobPost

10. tblApplyJob

11. tbl Question

12. tblAnswer

[5](#_26in1rg)

# I. Overview

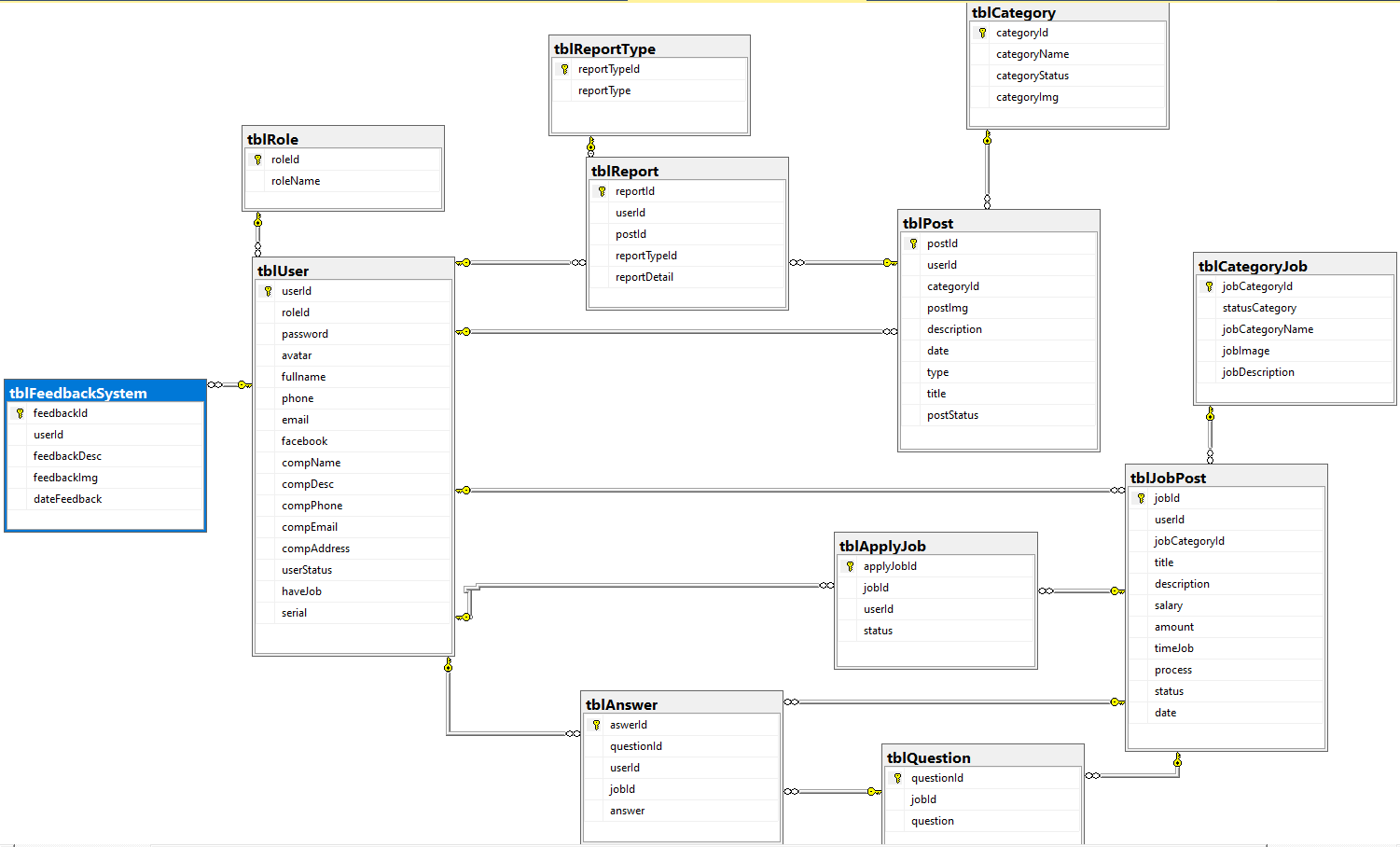
## 1. Code Packages/Namespaces



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

| **No** | **Package** | **Description** |
| --- | --- | --- |
| *01* | *controllers* | *+This package is used to control functions*  *+Class naming convention: capitalize first letters*  *-AddCategoryController*  *-AddJobCategoryController*  *-AdminFreelanceController*  *-AdminLostAndFoundController*  *-ApproveJobController*  *-ApproveJobPostController*  *-CategoryController*  *-ChangePasswordController*  *-DeleteCategoryController*  *-DeleteJobCategoryController*  *-DetailJobController*  *-DisplayAvatarController*  *-DisplayItemController*  *-EditCategoryController*  *-EditJobCategoryController*  *-FeedbackSystemController*  *-FreelancerHomeController*  *-JobCategoryController*  *-ListJobPostController*  *-ListPostController*  *-LoginController*  *-LogoutController*  *-LostAndFoundHomeController*  *-MainController*  *-MyJobPostApproveController*  *-MyJobPostDoneController*  *-MyJobPostProcessController*  *-MyPostController*  *-NoApproveJobController*  *-PostDetailController*  *-ProfileDetailController*  *-SearchJobByCategoryIdController*  *-SearchJobPostController*  *-SearchPostController*  *-SignUpController*  *-UpdateAccountController*  *-UpdateAvatarController*  *-UploadJobPostController*  *-UploadPostController* |
| *02* | *category* | *+Class naming convention: capitalize first letters*  *-CategoryDAO*  *-CategoryDTO* |
| *03* | *feedbackSystem* | *+Class naming convention: capitalize first letters*  *-FeedbackSystemDAO*  *-FeedbackSystemDTO* |
| *04* | *jobCategory* | *+Class naming convention: capitalize first letters*  *-JobCategoryDAO*  *-JobCategoryDTO* |
| *05* | *jobPost* | *+Class naming convention: capitalize first letters*  *-JobPostDAO*  *-JobPostDTO* |
| *06* | *jobPostQuestion* | *+Class naming convention: capitalize first letters*  *-JobPostQuestionDAO*  *-JobPostQuestionDTO* |
| *07* | *post* | *+Class naming convention: capitalize first letters*  *-PostDAO*  *-PostDTO* |
| *08* | *report* | *+Class naming convention: capitalize first letters*  *-ReportDAO*  *-ReportDTO* |
| *09* | *reportType* | *+Class naming convention: capitalize first letters*  *-ReportTypeDTO* |
| *10* | *role* | *+Class naming convention: capitalize first letters*  *-RoleDTO* |
| *11* | *user* | *+Class naming convention: capitalize first letters*  *-UserDAO*  *-UserDTO* |
| *12* | *utills* | *+This package to open the connection port with Database SQL Server*  *+Class naming convention: capitalize first letters*  *-DBUtils*  *-ImageUtils* |

## 2. Database Schema



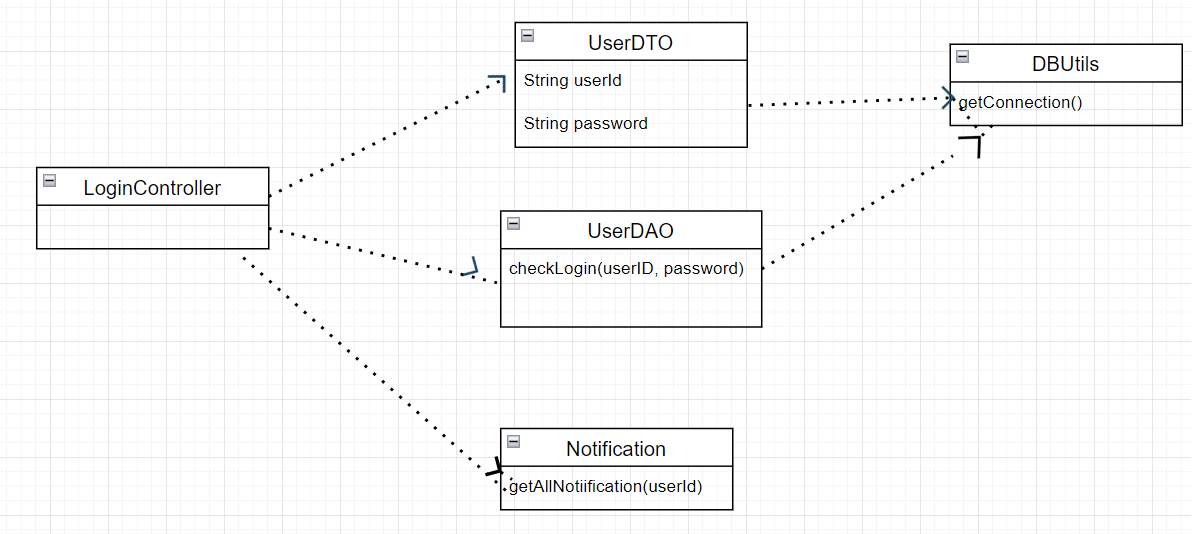
***Table descriptions & package class naming conventions are as below***

| **No** | **Table** | **Description** |
| --- | --- | --- |
| *01* | *tblUser* | *- Primary keys: userId*  *- Foreign keys: roleId* |
| *02* | *tblRole* | *- Primary keys: roleId* |
| *03* | *tblFeedbackSystem* | *- Primary keys: feedbackId*  *- Foreign keys: userId* |
| *04* | *tblReport* | *- Primary keys: reportId*  *- Foreign keys: userId* |
| *05* | *tblReportType* | *- Primary keys: reportTypeId* |
| *06* | *tblCategory* | *- Primary keys: categoryId* |
| *07* | *tblPost* | *- Primary keys: postId*  *- Foreign keys: userId*  *- Foreign keys: categoryId* |
| *08* | *tblCategoryJob* | *- Primary keys: jobCategoryId* |
| *09* | *tblJobPost* | *- Primary keys: jobId*  *- Foreign keys: userId*  *- Foreign keys: jobCategoryId* |
| *10* | *tblApplyJob* | *- Primary keys: applyJobId*  *- Foreign keys: jobId*  *- Foreign keys: userId* |
| *11* | *tblQuestion* | *- Primary keys: questionId*  *- Foreign keys: jobId* |
| *12* | *tblAnswer* | *- Primary keys: answerId*  *- Foreign keys: questionId*  *- Foreign keys: userId*  *- Foreign keys: jobId* |

# II. Code Designs

## 1. Login

### a. Class Diagram



### b. Class Specifications

#### UserDTO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *userId, password* | *input in the corresponding data to compare with the data in the database* |

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *checkLogin(userId, password)* | *Compare the userId and password data entered with the data in the database to see if they match or not* |

#### DBUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *Open the connection port between the project and SQL server* |

#### Notification Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllNotification(userId)* | *Send a notification to the user when the wrong userId or password is entered* |

#### LoginController Servlet

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* |  | *Control Login function* |

### c. Sequence Diagram(s)



### d. Database queries

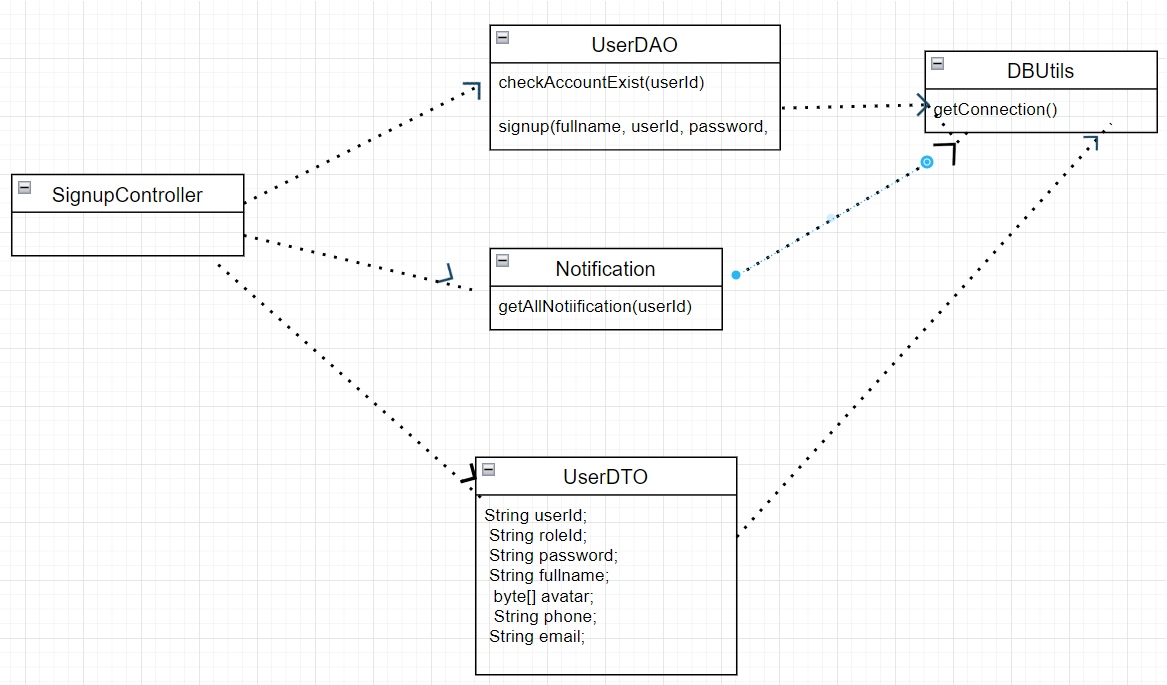
*SELECT u.userId, u.roleId, u.password, u.fullname, u.avatar, u.phone, u.email, u.facebook, u.userStatus, u.haveJob, r.roleName*

*FROM tblUser as u, tblRole as r*

*WHERE u.roleId = r.roleId AND userId = ? AND password = "*

## 2. Signup

### a. Class Diagram



### b. Class Specifications

#### UserDTO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *userId, password, password, fullName, avatar, phone, email* | *input in the corresponding data to compare with the data in the database* |

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *checkAccountExist(userId)* | *Compare the entered userId data with the data in the database to see if they exist* |
| *02* | *signup(fullName,userId, password* | *Transfer data fullName, userId, password to SQL server database to create an account* |

#### DBUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *Open the connection port between the project and SQL server* |

#### Notification Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllNotification(userId)* | *Send a notification to the user when the wrong user or password is entered, and notify when the username has been duplicated with another account* |

#### SignupController Servlet

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* |  | *Control Signup function* |

### c. Sequence Diagram(s)

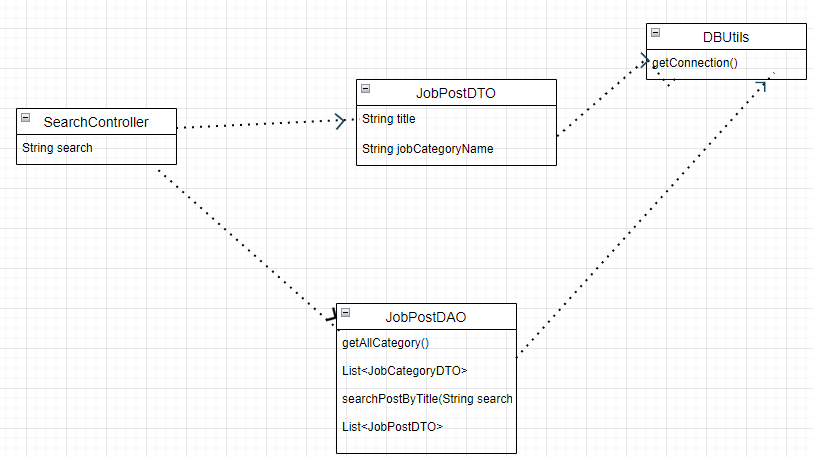


### d. Database queries

*INSERT INTO tblUser(fullname, userId, password, email, phone, roleId) VALUES (?, ?, ?, ?, ?, 'US')*

## 3. Search

### a. Class Diagram



### b. Class Specifications

#### JobPostDTO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *String title* | *Get title of post data from SQL server to web page* |
| *02* | *String jobCategoryName* | *Get Category Name of post data from SQL server to web page* |

#### JobPostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCategory()* | *Get all category names from provided web upload database* |
| *02* | *searchPostByTitle(String search)* | *Get all post titles from the provided web upload database* |
| *03* | *List<JobCategoryDTO>* | *List of all categories* |
| *04* | *List<JobPostDTO>* | *List of all job post* |

#### DBUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *Open the connection port between the project and SQL server* |

#### SearchController Servlet

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *String search* | *Get search data was enter to return to return value match* |

### c. Sequence Diagram(s)

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



### d. Database queries

*SELECT j.jobId, j.userId, j.jobCategoryId, j.title, j.description, j.salary, j.amount, j.timeJob, j.process, j.date, j.status, c.jobCategoryName, u.fullname, u.compName*

*FROM tblJobPost as j, tblCategoryJob as c, tblUser as u*

*WHERE j.jobCategoryId=c.jobCategoryId AND j.userId=u.userId AND status=1 AND (dbo.removeMark(title) LIKE ? OR title LIKE ?) AND (process = 'new' OR process='process')*

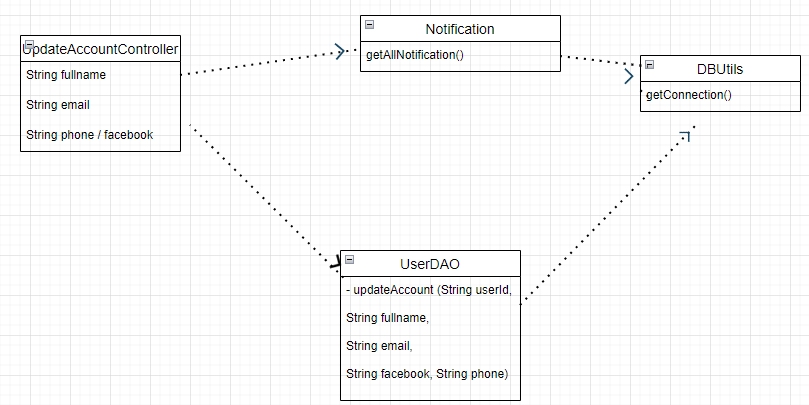
*ORDER BY jobId DESC*

* *get all category*

*SELECT\* FROM tblCategoryJob WHERE statusCategory=1*

## 4. Update Account

### a. Class Diagram



### b. Class Specifications

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateAccount()* | *Enter new user information data to replace previously entered data* |

#### DBUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *Open the connection port between the project and SQL server* |

#### Notification Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllNotification(userId)* | *Send notifications to users when updating new information fails* |

#### UpdateAccountController Servlet

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *String fullname* | *Enter new data to replace previously saved data* |
| *02* | *String email* | *Enter new data to replace previously saved data* |
| *03* | *String phone* | *Enter new data to replace previously saved data* |
| *04* | *String facebook* | *Enter new data to replace previously saved data* |

### c. Sequence Diagram(s)



### d. Database queries

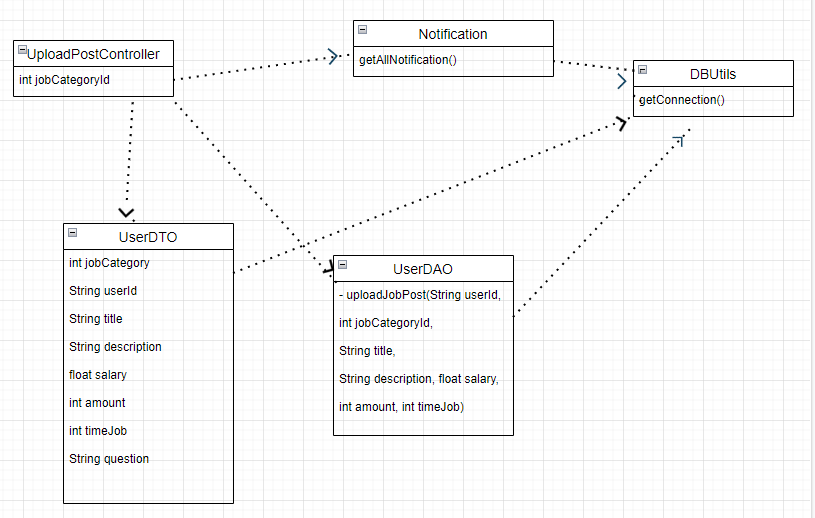
*UPDATE tblUser*

*SET fullname = ?, email = ?, facebook = ?, phone = ?*

*WHERE userId= ?";*

## 5. Upload Post

### a. Class Diagram



### b. Class Specifications

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *uploadJobPost()* | *Provide informative data for the post* |

#### UserDTO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *int jobCategory* | *provide data for posts to SQL server* |
| *02* | *String userId* | *provide data for posts to SQL server* |
| *03* | *String title* | *provide data for posts to SQL server* |
| *04* | *String description* | *provide data for posts to SQL server* |
| *05* | *float salary* | *provide data for posts to SQL server* |
| *06* | *int amount* | *provide data for posts to SQL server* |
| *07* | *int timeJob* | *provide data for posts to SQL server* |
| *08* | *String question* | *provide data for posts to SQL server* |

#### DBUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *Open the connection port between the project and SQL server* |

#### Notification Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllNotification(userId)* | *Send a notification to the user when the post is successful* |

#### UploadPostController Servlet

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *int jobId* | *Get the information of the post from SQL server to put on the web* |

### c. Sequence Diagram(s)



### d. Database queries

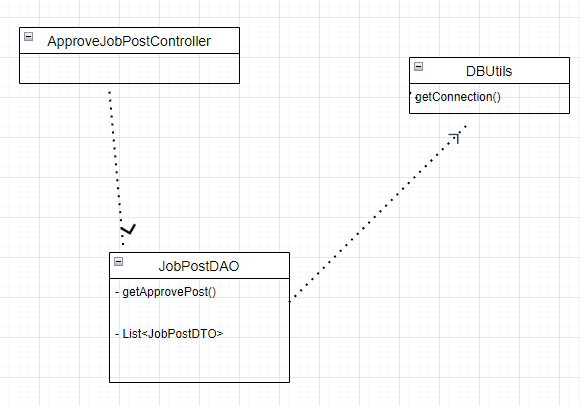
*UPDATE tblUser*

*SET fullname = ?, email = ?, facebook = ?, phone = ?*

*WHERE userId= ?";*

## 6. Approve Post

### a. Class Diagram



### b. Class Specifications

#### JobPostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getApprovePost()* | *Get posts pending approval* |
| *02* | *List<JobPostDTO>* |  |

#### DBUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *Open the connection port between the project and SQL server* |

#### ApproveJobPostController Servlet

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *int jobId* | *Get the information of the post from SQL server to put on the web* |

### c. Sequence Diagram(s)



### d. Database queries

*SELECT j.jobId,j.userId, j.jobCategoryId, j.title, j.description, j.salary, j.amount, j.timeJob, j.process, j.date, j.status, c.jobCategoryName, u.avatar, u.fullname, u.compName*

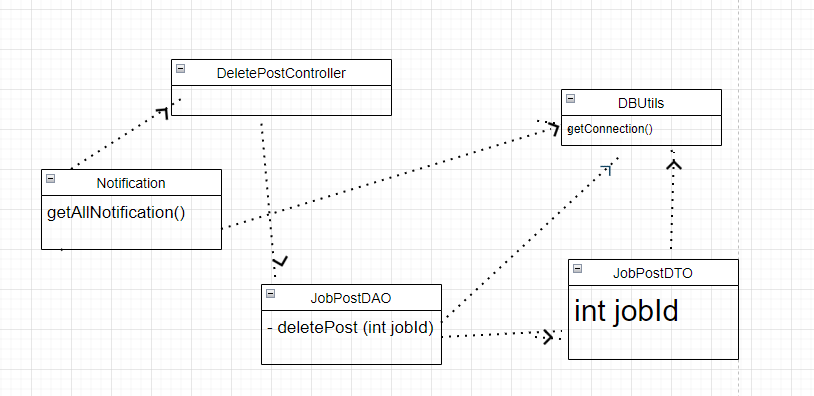
*FROM tblJobPost as j, tblCategoryJob as c, tblUser as u*

*WHERE j.jobCategoryId=c.jobCategoryId AND j.userId=u.userId AND status=1 AND process='approving'*

*ORDER BY jobId DESC*

## 7. Delete Post

### a. Class Diagram



### b. Class Specifications

#### JobPostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *deletePost(int jobId)* | *Get post by jobId to do delete* |

#### JobPostDTO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *int jobId* | *Get post by jobId to do delete* |

#### DBUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *Open the connection port between the project and SQL server* |

#### DeletePostController Servlet

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *int jobId* | *Get the information of the post from SQL server to put on the web* |

#### Notification Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllNotification(userId)* | *Send notifications to users when a successful post is deleted* |

### c. Sequence Diagram(s)

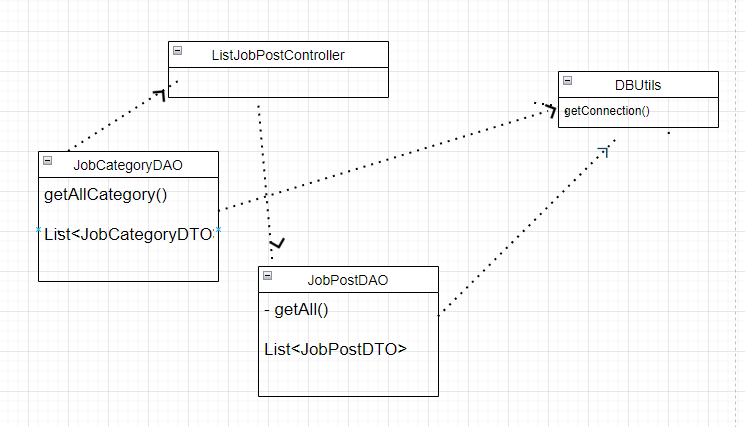


### d. Database queries

*UPDATE tblJobPost SET status=0 WHERE jobId=?*

## 8. List Post

### a. Class Diagram



### b. Class Specifications

#### JobPostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAll()* | *Get all post from databse* |
| *02* | *List<JobPostDTO>* |  |

#### JobCategoryDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCategory* | *Get all Category Name from database* |
| *02* | *List<JobCategoryDTO>>* |  |

#### DBUtils Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *Open the connection port between the project and SQL server* |

#### ListJobPostController Servlet

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* |  | *Show all posts* |

### c. Sequence Diagram(s)



### d. Database queries

*SELECT j.jobId, j.userId, j.jobCategoryId, j.title, j.description, j.salary, j.amount, j.timeJob, j.process, j.date, j.status, c.jobCategoryName, u.fullname, u.compName, u.avatar*

*FROM tblJobPost as j, tblCategoryJob as c, tblUser as u*

*WHERE j.jobCategoryId=c.jobCategoryId AND j.userId=u.userId AND status=1 AND (process='new' OR process='process')*

*ORDER BY jobId DESC*

# III. Database Tables

## 1. tblFeedbackSystem

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | feedbackId | int |  |  | x | PK |  |
| 2 | userId | varchar | 30 |  | x | FK |  |
| 3 | feedbackDesc | nvarchar | max |  |  |  |  |
| 4 | feedbackImg | image |  |  |  |  |  |
| 5 | dateFeedback | datetime |  |  |  |  |  |

## 2. tblRole

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | roleId | varchar | 10 |  | x | PK | US, AD,EM |
| 2 | roleName | nvarchar | 50 |  |  | FK | User, Admin,Employeer |

## 3. tblUser

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | userId | varchar | 30 |  | x | PK |  |
| 2 | roleId | varchar | 10 |  | x | FK | AD, US,EM |
| 3 | password | varchar | 30 |  | x |  |  |
| 4 | avatar | varbinary | max |  | x |  |  |
| 5 | fullname | nvarchar | 50 |  | x |  |  |
| 6 | phone | varchar | 15 |  |  |  |  |
| 7 | email | varchar | 30 |  | x |  |  |
| 8 | facebook | varchar | 100 |  |  |  |  |
| 9 | compName | nvarchar | 50 |  |  |  |  |
| 10 | compDesc | nvarchar | max |  |  |  |  |
| 11 | compPhone | varchar | 15 |  |  |  |  |
| 12 | compEmail | varchar | 30 |  |  |  |  |
| 13 | compAddress | nvarchar | max |  |  |  |  |
| 14 | userStatus | bit |  |  |  |  |  |
| 15 | haveJob | bit |  |  |  |  |  |
| 16 | serial | int |  |  | x |  |  |

## 4. tblReportType

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | reportTypeId | int |  |  | x | PK |  |
| 2 | reportType | nvarchar | max |  | x |  |  |

## 5. tblReport

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | reportId | int |  |  | x | PK |  |
| 2 | userId | varchar | 30 |  | x | FK |  |
| 3 | postId | int |  |  | x | FK |  |
| 4 | reportTypeId | int |  |  | x |  |  |
| 5 | reportDetail | nvarchar | max |  |  |  |  |

## 6. tblCategory

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | categoryId | int |  |  | x | PK |  |
| 2 | categoryName | nvarchar | 30 |  | x |  |  |
| 3 | categoryStatus | bit |  |  |  |  | 0: false ; 1:true |
| 4 | categoryImg | varchar | max |  |  |  |  |

## 7. tblPost

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | postId | int |  |  | x | PK |  |
| 2 | userId | varchar | 30 |  | x | FK |  |
| 3 | categoryId | int |  |  | x | FK |  |
| 4 | postImg | varbinary | max |  |  |  |  |
| 5 | description | nvarchar | max |  |  |  |  |
| 6 | date | datetime |  |  | x |  |  |
| 7 | type | varchar | 10 |  | x |  |  |
| 8 | title | nvarchar | 50 |  | x |  |  |
| 9 | postStatus | varchar | 10 |  |  |  |  |

## 8. tblCategoryJob

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | jobCategoryId | int |  |  | x | PK |  |
| 2 | statusCategory | bit |  |  | x |  | 0:false , 1:true |
| 3 | jobCategoryName | nvarchar | max |  |  |  |  |
| 4 | jobImage | varchar | max |  |  |  |  |
| 5 | jobDescription | nvarchar | max |  |  |  |  |

## 9. tblJobPost

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | jobId | int |  |  | x | PK |  |
| 2 | userId | varchar | 30 |  | x | FK |  |
| 3 | jobcategoryId | int |  |  |  | FK |  |
| 4 | title | nvarchar | max |  |  |  |  |
| 5 | description | nvarchar | max |  |  |  |  |
| 6 | salary | float |  |  |  |  |  |
| 7 | amount | int |  |  |  |  |  |
| 8 | timeJob | int |  |  |  |  |  |
| 9 | proccess | varchar | 10 |  |  |  | approving, process, new |
| 10 | status | bit |  |  |  |  | 0:false ; 1:true |
| 11 | date | int |  |  |  |  |  |

## 10. tblApplyJob

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | applyJobId | int |  |  | x | PK |  |
| 2 | userId | varchar | 30 |  | x | FK |  |
| 3 | jobId | int |  |  | x | FK |  |
| 4 | status | bit |  |  |  |  | waiting, done, deny, duing |

## 11. tblQuestion

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | questionId | int |  |  | x | PK |  |
| 2 | jobId | int |  |  | x | FK |  |
| 3 | question | nvarchar | max |  |  |  |  |

## 12. tblAnswer

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | aswerId | int |  |  | x | PK |  |
| 2 | userId | varchar | 30 |  | x | FK |  |
| 3 | questionId | int |  |  | x | FK |  |
| 4 | jobId | int |  |  | x | FK |  |
| 5 | answer | nvarchar | 1 |  | x |  |  |