

**Top TOP Social network**

**Software Design Document**

– DaNang, March 2022 –

**Table of Contents**

[I. Overview 4](#_Toc98834211)

[1. Code Packages/Namespaces 4](#_Toc98834212)

[2. Database Schema 6](#_Toc98834213)

[II. Code Designs 7](#_Toc98834214)

[1. Login 7](#_Toc98834215)

[1.a. Class Diagram 7](#_Toc98834216)

[1.b. Class Specifications 7](#_Toc98834217)

[1.c. Sequence Diagram(s) 8](#_Toc98834218)

[1.d. Database queries 8](#_Toc98834219)

[2.Forget Password 9](#_Toc98834220)

[2.a. Class Diagram 9](#_Toc98834221)

[2.b. Class Specifications 9](#_Toc98834222)

[2.c. Sequence Diagram(s) 11](#_Toc98834223)

[2.d. Database queries 11](#_Toc98834224)

[3.Register 12](#_Toc98834225)

[a. Class Diagram 12](#_Toc98834226)

[b. Class Specifications 13](#_Toc98834227)

[c. Sequence Diagram(s) 15](#_Toc98834228)

[d. Database queries 16](#_Toc98834229)

[4.Dating 16](#_Toc98834230)

[a. Class Diagram 16](#_Toc98834231)

[b. Class Specifications 16](#_Toc98834232)

[c. Sequence Diagram(s) 18](#_Toc98834233)

[d. Database queries 18](#_Toc98834234)

[5. Message 19](#_Toc98834235)

[a. Class Diagram 19](#_Toc98834236)

[b. Class Specifications 19](#_Toc98834237)

[c. Sequence Diagram(s) 21](#_Toc98834238)

[d. Database queries 21](#_Toc98834239)

[6. Add post 22](#_Toc98834240)

[a. Class Diagram 22](#_Toc98834241)

[b. Class Specifications 22](#_Toc98834242)

[c. Sequence Diagram(s) 24](#_Toc98834243)

[d. Database queries 24](#_Toc98834244)

[7. Delete post 25](#_Toc98834245)

[a. Class Diagram 25](#_Toc98834246)

[b. Class Specifications 25](#_Toc98834247)

[c. Sequence Diagram(s) 26](#_Toc98834248)

[d. Database queries 26](#_Toc98834249)

[8. Comment 27](#_Toc98834250)

[a. Class Diagram 27](#_Toc98834251)

[b. Class Specifications 27](#_Toc98834252)

[c. Sequence Diagram(s) 28](#_Toc98834253)

[d. Database queries 29](#_Toc98834254)

[9. edit profile 29](#_Toc98834255)

[a. Class Diagram 29](#_Toc98834256)

[b. Class Specifications 29](#_Toc98834257)

[c. Sequence Diagram(s) 30](#_Toc98834258)

[d. Database queries 30](#_Toc98834259)

[10. like 31](#_Toc98834260)

[a. Class Diagram 31](#_Toc98834261)

[b. Class Specifications 32](#_Toc98834262)

[c. Sequence Diagram(s) 33](#_Toc98834263)

[d. Database queries 33](#_Toc98834264)

[III. Database Tables 34](#_Toc98834265)

[1. <Account> 34](#_Toc98834266)

[2. <UserInformation> 34](#_Toc98834267)

[3. <Participant> 35](#_Toc98834268)

[4. <Conversation> 35](#_Toc98834269)

[5. <Post> 35](#_Toc98834270)

[6. <Like> 35](#_Toc98834271)

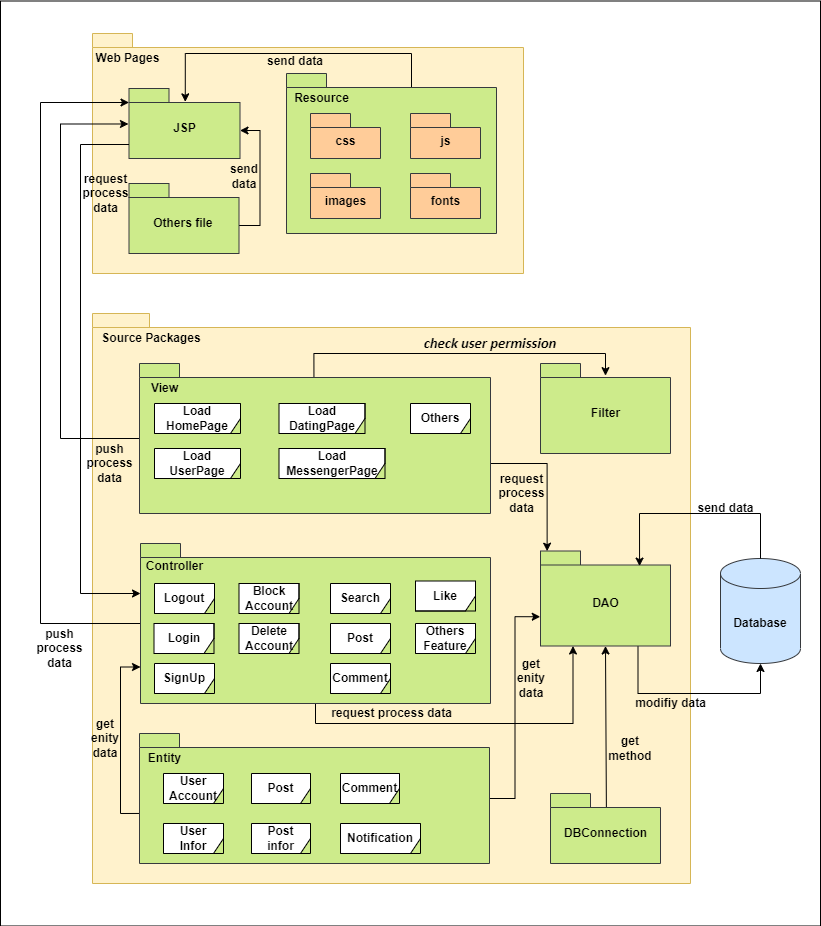
[7. <Comment> 35](#_Toc98834272)

[8. <Dating> 35](#_Toc98834273)

[9. <DatingProfile> 36](#_Toc98834274)

# I. Overview

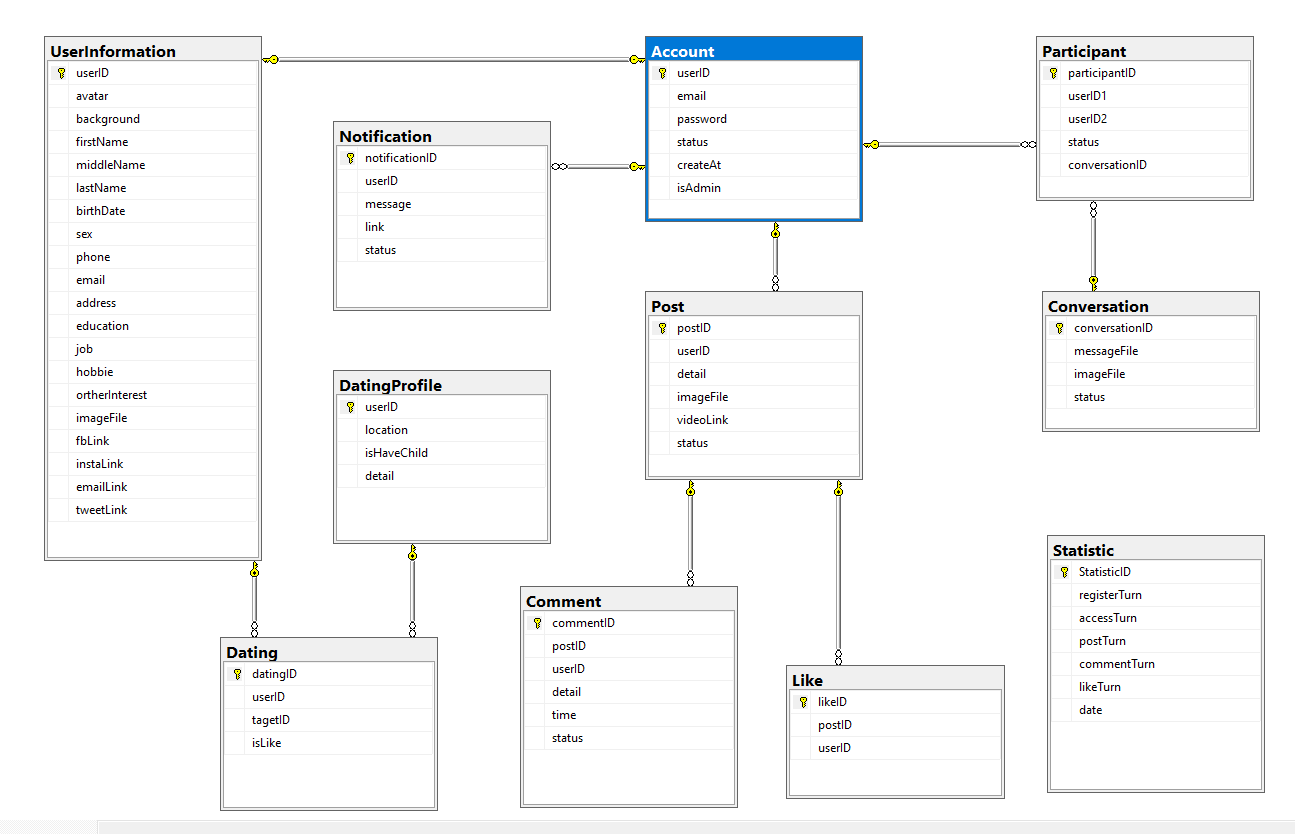
## 1. Code Packages/Namespaces



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

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| *01* | *Web Pages* | *Contains folders and files which create the interface of the system to interact directly with the user. Mainly about front-end. Contain folders: JSP, Resource, Others files.*  *Class naming convention: Camel Cases.* |
| *02* | *JSP* | *Contains all jsp, html files.*  *Class naming convention: Camel Case.* |
| *03* | *Resource* | *Contains folders css, js, image, fonts.*  *Class naming convention: Camel Case.* |
| *04* | *Others file* | *Generated files related to the project.*  *Class naming convention: Camel Case.* |
| *05* | *Source Package* | *Contains folders that are primarily responsible for modeling objects and folders that handle user requests through the front-end. Mainly about Back-end.*  *Class naming convention: Camel Case.* |
| *06* | *DAO* | *These folders do not contain entities, but contain classes that have methods that work with those entities from the database.*  *Class naming convention: Camel Case .* |
| *07* | *Controller* | *Contains folders that contain servlet files.*  *Class naming convention: Camel Case.* |
| *08* | *View* | *Contains folders that contain servlet files*  *Class naming convention: Camel Case* |
| *09* | *Filter* | *Folder contains classes that have methods used to check the user permission for the purpose of preventing the user from accessing unauthorized sites.*  *Class naming convention: Camel Case.* |
| *10* | *DBConnection* | *Folder contains classes that have methods used to connect to the database.*  *Class naming convention: Camel Case.* |
| *11* | *Entity* | *Contains the class that generates the entities used in the project.*  *Class naming convention: Camel Case.* |

## 2. Database Schema



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

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| *01* | *Account* | *Account’s information.*  *- Primary keys: userID* |
| *02* | *UserInformation* | *User’s information.*  *- Primary keys: userID* |
| *03* | *Participant* | *Friendship with others.*  *- Primary keys: participantID*  *- Foreign keys: conversationID* |
| *04* | *Conversation* | *Conversation between two people.*  *- Primary keys: conversationID* |
| *05* | *Post* | *Post’s information.*  *- Primary keys: postID*  *- Foreign keys: userID, commentID, likeID* |
| *06* | *Comment* | *Comment’s information.*  *- Primary keys: commentID* |
| *07* | *Like* | *A person's like activities.*  *- Primary keys: likeID* |
| *08* | *DatingProfile* | *A person's dating profile.*  *- Primary keys: userID* |
| *09* | *Dating* | *Dating relationship between two people.*  *- Primary keys: targetID* |

# II. Code Designs

## 1. Login

This feature is used to categorize users like viewers, people who have accounts but have not confirmed gmail, people who have verified accounts gmail, admin. These people have different permissions and different levels of access, the login function will help control that.

### 1.a. Class Diagram

A screenshot of a computer

Description automatically generated with low confidence

### 1.b. Class Specifications

#### loadLoginPage

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: load login.jsp page  Input: HttpServletRequest request, HttpServletResponse response  Output: void |

#### 

#### Login\_controller

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: get email and password, use dao class to check valid email and password in method checkLogin(email,password)  If account is valid forward to loadHomePage serverlet  Else forward to loadLoginPage serverlet. |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | checkLogin(email,password) | Description: create a connection by package DBContext with method getConnection() to connect to SQL server. Create a query for data from SQL Server:  Query: Select \* from Account where email = email and password = password  Return new Account if available. |

#### 

#### DBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Description: connect to the system's database  Input:  Output: Connection |

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

Calendar

Description automatically generated

### 1.d. Database queries

#### 

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | checkLogin(email,password) | Description: select account if available  Query: select \* from Account where email = email, password = password |

## 2.Forget Password

This feature is used to classify users like viewers, people who have accounts but they forget their password.

### 2.a. Class Diagram

A picture containing text, wall

Description automatically generated

### 2.b. Class Specifications

#### loadLoginPage

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: open login.jsp  Input: HttpServletRequest request, HttpServletResponse response  Output: void |

#### forgetPassword.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Click button | Description: click button in login.jsp to forward to forgetPassword.jsp |

#### forgetPasswordController

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: get email from request and check email exist or not.  If exist send new code to confirm and forward to confirmForgetPass.jsp  Else forward to forgetPassword.jsp  Input: email  Output: code for confirm. |

#### ConfirmForgetPass.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Forward from forgetPasswordController | Description: open ConfirmForgetPass.jsp |

#### getNewPasswordController

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: get code and email from request and input code.  Check lifeTime of code:  if invalid forward to register.jsp  else : check code and input code  if valid forward to setNewPass.jsp  else forward to confirmForgetPass.jsp |

#### setNewPass.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Forward from getNewPasswordController | Description: open setNewPass.jsp |

#### setNewPass

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: get email and password and confirm password from request.  Check password and confirm password:  if invalid forward to loadHomePage  else forward to setNewPass.jsp |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | checkEmail(email) | Description: check email exist or not  Input: email  Output: userID |
| 02 | ramdomCofirmCode() | Description: create a new code for confirm  Input:  Output: confirm code |
| 03 | sendCofrimCode(email, code) | Description: send confirm code to email user |
| 04 | updatePass(email,newPass) | Description: update new password with email in Database |

#### DBContex

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Connect to Sql Server |

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

A screenshot of a computer

Description automatically generated with low confidence

### 2.d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | checkEmail(email) | Description: check email exist or not  Query: select \* from Account where email = email |
| 02 | dao | updatePass(email,newPass) | Description: update new password for uesr  Query: update Account set password = newPass where email = email |

## 3.Register

Users use this attribute to register to use the site.

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

#### login.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: click to forward to registser.jsp. |

#### Register.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: user input their information and send request to registerController |

#### registerController

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: use dao() to check user’s information if valid send code to user’s email and forward to confirmRegister.jsp else forwar to register.jsp and print error  Input: user’s information |

#### confirmRegister.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: user input confrirm code and send request to confirmAccount |

#### confirmAccount

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: check confirm code, if valid user dao() to creat Account in database, if create successed use logincontroller servlet to login system else forward to register.jsp and print error  If confirm code invalid check if time out forward to register.jsp. If wrong code forward to confirmRegister.jsp and print error  Input: user’s information and confirm code |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | checkEmailExit(email) | Description: check email exists or not  Input: email  Output: boolean |
| 02 | ramdomCofirmCode() | Description: create new code to confirm. |
| 03 | sendCofrimCode(email, code) | Description: send confirm code to user’s email.  Input: email, code  Output: void |
| 4 | creatIDUser() | Description: create id of user |
| 5 | addNewAccount() | Description: add account to database  Input: user infor account  Output: boolean |
| 6 | addUserInformation(u) | Description: add user information to database  Input: user information  Output: |
| 7 | createDatingProfile(p) | Description: create dating profile in database  Input: user dating information |

#### DBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Description: connect database |

### c. Sequence Diagram(s)

Calendar

Description automatically generated

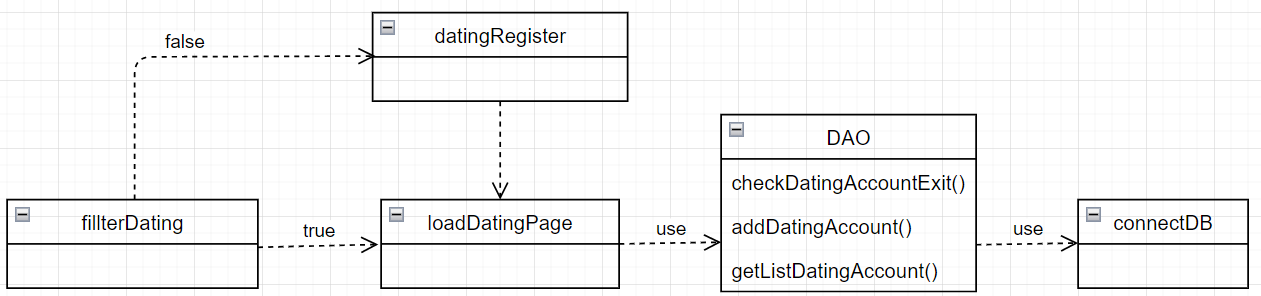
### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | checkEmailExit(email) | Description: check email is exist or not. Query: select \* from account where email = email |
| 02 | dao | addNewAccout(account) | Description: insert new account to Account Query: Insert into Account values(userID,email,password,status,'False' ) |
| 03 | dao | addUserInformation(u) | Description: insert new user information to userInformation  Query: insert into UserInformation values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);  ? = user Information |
| 4 | dao | createDatingProfile(p) | Description: insert new datingprofile to datingProfile  Query: insert into DatingProfile values(?,?,?,?,?,?,?,?,?,?,?,?)  ? = user dating profile |

## 4.Dating

This feature allows users who already have a dating account to view and search for the match people to connect.

### a. Class Diagram



### b. Class Specifications

#### fillterDating

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: the person accessing the dating function has an account  then forward to loadDdatingPage. Else, forward to datingRegister. |

#### datingRegister

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost | Description: get the values from form. Use the method “checkDatingAccountExit” in DAO to get all the accounts’s name in the system, then compare if there are any accounts with the same user id. If no then add new Account to database by call method “addDatingAccount” in DAO and forward to loadDatingPage, otherwise forward to loadHomePage. |

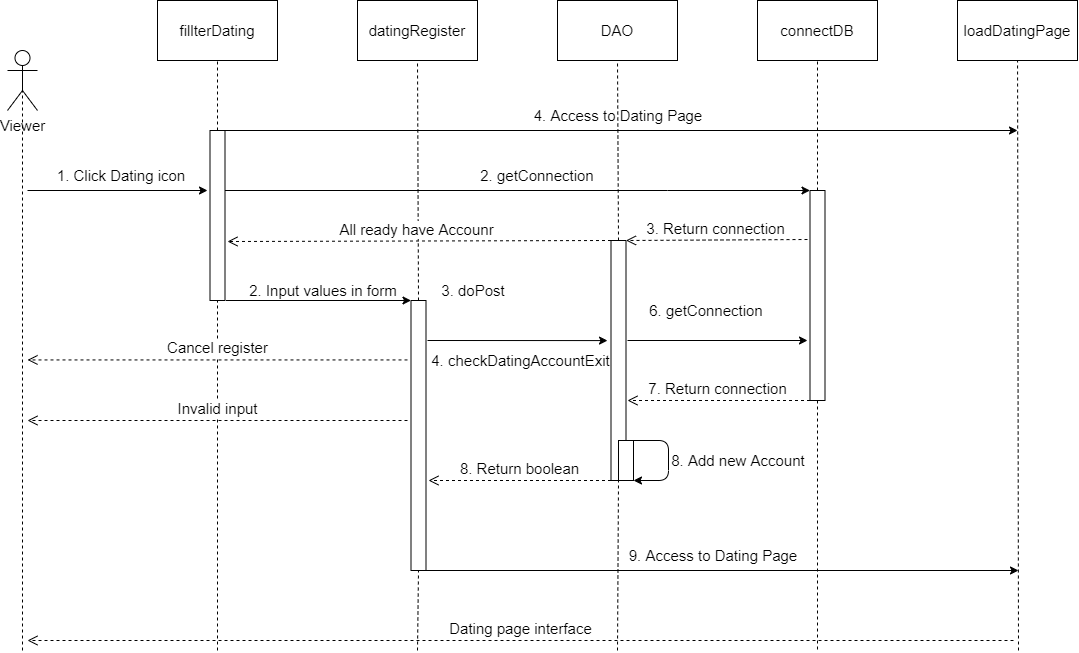
#### DAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | checkDatingAccountExist | Description: Call the method "getonnection" to connect to the database, then search “userID” is entered. If return result, return true, else return false.  Input: userID  Output: boolean |
| 02 | addDatingAccount | Description: Connect to the database using the "getConnection" method. Then insert into the database the information about a new Account, if the insert is successful, return true, if it fails, return false  Input: String birthDate, String sex, String location, String education, String job, String hobbie, String isHavChild, String infoDetail, String image.  Output: boolean |
| 03 | getListDatingAccount | Description: Connect to the database using the "getConnection" method. Then select the information about "DatingProfile" table from the database.  Input: null  Output: ArrayList<String> |

#### DBConnection

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection | Description: connect to the system's database  Input:  Output: Connection |

### c. Sequence Diagram(s)



### d. Database queries

#### 

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | DAO | checkDatingAccountExist | Description: check Dating account is exist or not. Query: select \* from DatingProfile  where userID = ?  1: String uID |
| 02 | DAO | addDatingAccount | Description: insert new account into DatingProfile Query: insert into DatingProfile(username,userPassword,userEmail,userFullname,isAdmin,createDate,facebookURL,userImage,)"  + " values(?,?,?,?,0,?,'','')  1: String username 2: String password 3: String email 4: String fullname 5: LocalDateTime.now().toString() |
| 03 | DAO | getListDatingAccount | Description: check Dating account is exist or not. Query: select \* from DatingProfile D, UserInformation U where D.userID = U.userID |

## 5. Message

Users use this attribute to message with user’friend.

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

#### newsfeed.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: send request to loadChatPage. |

#### loadChatPage

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: get userID, load data message and forward to chat.jsp |

#### Chat.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: view message, history message and send message |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getUserInfo(ID) | Description: get information of user |
| 2 | getListFriendContacYet(id) | Description: get list messaged |
| 3 | getConverByUser(user, i) | Description: get content in box chat of user i |
| 4 | getListFriendNotContacYet(id) | Description: get list friend not message |
| 5 | getChatBox(u1,u2) | Description: check messaged or not  If not yet send new mess  Else send mess |
| 6 | updateParticipant() | Description: create new box chat |
| 7 | sendMessage(mess) | Description: send message to user (add to database) |

#### DBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Description: connect database |

### c. Sequence Diagram(s)

A screenshot of a computer

Description automatically generated with low confidence

### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | getUserInfo(id) | Description: get userInfor Query: select \* from userInformation where userID = id |
| 02 | dao | getListFriendContacYet(id) | Description: get list messaged Query: select \* from UserInformation u, (select userID2 from Participant where userID1 = id and conversationID is not null where u.userID = p.userID2" |
| 03 | dao | getConverByUser(user, i) | Description: get content box chat  Query: select top 1 c.conversationID,u.firstName + ' ' + u.middleName + ' ' + u.lastName [name],m.content,m.[time],m.[type],m.[status1],m.senderID,a.[status] [on\_off], m.sttMessage from [Conversation] c, (select conversationID from Participant where userID1 = id1 and userID2 = id2 and conversationID is not null) p,  MessageContent m,Account a,UserInformation u  where c.conversationID = p.conversationID and m.ConversationID = p.conversationID and a.userID = m.senderID and u.userID = m.senderID  order by m.sttMessage DESC |
| 4 | dao | getListFriendNotContacYet(id) | Description: get list friend not message  Query: select p.userID2,u.avatar, u.firstName +' '+u.middleName + ' ' + u.lastName [fullname] , a.[status] [active] ,p.[status] from UserInformation u ,(select userID2,[status] from Participant where userID1 = id and conversationID is null and [status] = 1) p,Account a where u.userID = p.userID2 and a.userID = p.userID2 |
| 5 | dao | getChatBox(senderID, recieveID) | Description: get chatbox Query: select conversationID from Participant where userID1 = senderID and userID2 = recieved |
| 6 | dao | creatBoxChat(id) | Description: create chatBox  Query: Insert into Conversation values (id,'','1','1') |
| 7 | dao | updateParticipant(senderID,reciveID, id) | Description: create chatBox  Query: Update Participant set conversationID = id where userID1 = sender and userID2 = recieved |
| 8 | dao | sendMessage(messgae) | Description: send message  Query: Insert into MessageContent values(?,?,?,?,?,?,?,?)  ? = message information |

## 6. Add post

Users use this attribute to message with user’friend.

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

#### newsfeed.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: input content post and send request. |

#### addPost

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost | Description: get content of post from newsfeed.jsp, check type post if type is image use dao to save image to folder and add to post table in database else if type is text add to post table in database |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | addTextPostByUser(post,hostID) | Description: add text post. |
| 2 | addImagePostByUser(contentPost, user.getUserId(),"image"); | Description: add imgae post |
| 3 | addImageAttachment(newPostID, nameFile); | Description: add url image of post |

#### DBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Description: connect database |

### c. Sequence Diagram(s)

Diagram

Description automatically generated

### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | addTextPostByUser(contentPost, user.getUserId()) | Description: get userInfor Query: INSERT INTO Post (postID, userID, detail, posted\_time,imageFile) VALUES(?,?,?,?,?) |
| 02 | dao | addImagePostByUser(contentPost, user.getUserId(),"image") | Description: get list messaged Query: INSERT INTO Post (postID, userID, detail, imageFile, posted\_time) VALUES(?,?,?,?,?) |
| 03 | dao | addImageAttachment(newPostID, nameFile) | Description: get content box chat  Query: INSERT INTO Attachments (postID, attachmentURL) VALUES(?,?) |

## 7. Delete post

Users use this attribute to message with user’friend.

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

#### newsfeed.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: choose post which you want to delete and send request. |

#### deletePost

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: delete post and all related stuff |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | deleteLikeOfPost(postID) | Description: delete all like of post. |
| 2 | deleteCmtOfPost(postID); | Description: delete all comment of post. |
| 3 | deleteRelaFile(postID); | Description: delete all image of post (if any) |
| 4 | deletePost(postID); | Description: delete post |

#### DBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Description: connect database |

### c. Sequence Diagram(s)

Diagram

Description automatically generated

### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | deleteLikeOfPost(postID) | Description: delete Like Query: delete from [like] where postID = postID |
| 02 | dao | deleteCmtOfPost(postID) | Description: delete comment Query: delete from [comment] where postID = postID |
| 03 | dao | deleteRelaFile(postID) | Description: delete image  Query: delete from [Attachments] where postID = postID |
| 04 | dao | deletePost(postID) | Description: delete image  Query: delete from [Post] where postID = postID |

## 8. Comment

Users use this attribute to message with user’friend.

### a. Class Diagram

Diagram

Description automatically generated

### b. Class Specifications

#### newsfeed.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: choose post which you want to comment, input comment and send request. |

#### addComment

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: send commnt to post |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | addCommentToPost(postID, userID, detail) | Description: add comment to post |
| 2 | createNotificationID(); | Description: create notifications for post owner. |
| 3 | getHostByPostID(postID) | Description: get post owner |
| 4 | getListCarePost(postID, host); | Description: get list user commented on post |
| 5 | createNotification(noti); | Description: send notifications to list user |

#### DBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Description: connect database |

### c. Sequence Diagram(s)

Diagram

Description automatically generated

### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | addCommentToPost(postID, userID, detail) | Description: delete Like Query: INSERT INTO Comment (commentID, postID, userID, detail, time) VALUES(?,?,?,?,?) |
| 02 | dao | checkNotificationID(id) | Description: delete comment Query: select \* from Notification where notificationID = ? |
| 03 | dao | getHostByPostID(postID) | Description: delete image  Query: select userID from Post where postID = ? |
| 04 | dao | createNotification(noti) | Description: delete image  Query: insert into Notification values(?,?,?,?,?,?,?,?) |

## 9. Edit profile

Users use this attribute to message with user’friend.

### a. Class Diagram

A screenshot of a computer

Description automatically generated with medium confidence

### b. Class Specifications

#### managerInforUser.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: input infor and send request |

#### acceptChangeInfor

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: check password, if valid show button change |

#### changeInfor

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: get infor from managerInforUser.jsp and update owner infor |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | maHoaMatKhau(pass) | Description: User Password Encryption |
| 2 | updateInfor(new\_u) | Description: update information. |

#### DBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Description: connect database |

### c. Sequence Diagram(s)

A picture containing text, indoor, map

Description automatically generated

### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | maHoaMatKhau(pass) | Query: SELECT CONVERT(VARCHAR(32), HashBytes('MD5', '" + mk + "'), 2) as md5 |
| 02 | dao | updateInfor(new\_u) | Description: delete comment Query: update UserInformation set "  + "avatar = ?,"  + "background = ?,"  + "firstName = ?,"  + "middleName = ?,"  + "lastName = ?,"  + "birthDate = ?,"  + "sex = ?,"  + "phone = ?,"  + "email = ?,"  + "address = ?,"  + "education = ?,"  + "job = ?,"  + "hobbie = ?,"  + "ortherInterest = ?,"  + "imageFile = ?,"  + "fbLink = ?,"  + "instaLink = ?,"  + "emailLink = ?,"  + "tweetLink = ?,"  + "createAt = ?"  + " where userID = ? |

## 10. Like

Users use this attribute to message with user’friend.

### a. Class Diagram

A screenshot of a computer

Description automatically generated with low confidence

### b. Class Specifications

#### newsfeed.jsp

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | no | Description: request like post |

#### likePost

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest | Description: like post |

#### dao

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getPostByID(postID) | Description: get post |
| 2 | getIsLikePost(String uID, String postID) | Description: check like or not |
| 3 | unLikePost(postID, a.getUserId()) | Description: if liked then unlike. |
| 4 | likePost(postID, a.getUserId()) | Description: like post |
| 5 | createNotification(d.createNotificationID(), p.getUserID(), a.getUserId(), "Đã thích bài viết của bạn!", "loadPostDetailPage?postID=" + postID, "fa fas-heart"); | Description: create notification |

#### DBContext

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getConnection() | Description: connect database |

### c. Sequence Diagram(s)

Calendar

Description automatically generated

### d. Database queries

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class** | **Method** | **Query** |
| 01 | dao | getPostByID(postID) | Query: SELECT CONVERT(VARCHAR(32), HashBytes('MD5', '" + mk + "'), 2) as md5 |
| 02 | dao | getIsLikePost(String uID, String postID) | Description: delete comment Query: select \* from [Like] where postID=? and userID = ? |
| 3 | dao | unLikePost(postID, a.getUserId()) | Query: delete from [Like] where postID = ? and userID = ? |
| 4 | dao | likePost(postID, a.getUserId()) | Query: Insert into [Like] values (?,?) |
| 5 | dao | createNotification(d.createNotificationID(), p.getUserID(), a.getUserId(), "Đã thích bài viết của bạn!", "loadPostDetailPage?postID=" + postID, "fa fas-heart"); | Query: insert into Notification values(?,?,?,?,?,?,?,?) |

# III. Database Tables

## 1. <Account>

*[Give some lines about the table here>>*

*[Table fields, in the form of table format as below]*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | userID | nvarchar | 10 | x | x | PK |  |
| 2 | userName | nvarchar | 50 | x | x |  |  |
| 3 | passWord | nvarchar | 50 |  | x |  |  |
| 4 | email | nvarchar | 50 | x | x |  |  |
| 5 | avata | nvarchar | MAX |  |  |  |  |
| 6 | credit | nvarchar | MAX |  |  |  |  |
| 7 | status | nvarchar | 30 |  |  |  |  |
| 8 | createdAt | Date |  |  |  |  |  |
| 9 | isAdmin | bit |  |  |  |  |  |

## 2. <UserInformation>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
|  | userID | nvarchar | 10 | x | x | PK |  |
|  | firstName | nvarchar | 50 |  | x |  |  |
|  | middleName | nvarchar | 50 |  |  |  |  |
|  | lastName | nvarchar | MAX |  | x |  |  |
|  | birthDate | nvarchar | MAX |  | x |  |  |
|  | phone | nvarchar | 10 | x |  |  |  |
|  | email | nvarchar | 50 | x | x |  |  |
|  | address | nvarchar | MAX |  |  |  |  |
|  | education | nvarchar | MAX |  |  |  |  |
|  | job | nvarchar | MAX |  |  |  |  |
|  | hobbie | nvarchar | MAX |  |  |  |  |
|  | otherIntersest | nvarchar | MAX |  |  |  |  |
|  | imageFile | nvarchar | MAX |  |  |  |  |
|  | fbLink | nvarchar | MAX |  |  |  |  |
|  | instaLink | nvarchar | MAX |  |  |  |  |
|  | emailLink | nvarchar | MAX |  |  |  |  |
|  | tweetLink | nvarchar | MAX |  |  |  |  |

## 3. <Participant>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
|  | participantID | nvarchar | 10 | x | x | PK |  |
|  | userID | nvarchar | 10 |  | x | FK |  |
|  | isFriend | bit |  |  |  |  |  |
|  | conversationID | nvarchar | 10 |  | x | FK |  |

## 4. <Conversation>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
|  | conversationID | nvarchar | 10 | x | x | PK |  |
|  | messageFile | nvarchar | MAX |  |  |  |  |
|  | imageFile | nvarchar | MAX |  |  |  |  |
|  | status | nvarchar | 20 |  |  |  |  |

## 5. <Post>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
|  | postID | nvarchar | 10 | x | x | PK |  |
|  | userID | nvarchar | MAX |  | x | FK |  |
|  | detail | nvarchar | MAX |  | x |  |  |
|  | imageFile | nvarchar | MAX |  |  |  |  |
|  | videoLink | nvarchar | MAX |  |  |  |  |
|  | status | nvarchar | 20 |  |  |  |  |

## 6. <Like>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
|  | likeID | nvarchar | 10 | x | x | PK |  |
|  | postID | nvarchar | 10 |  | x | FK |  |
|  | userID | nvarchar | 10 |  | x | FK |  |
|  | status | nvarchar | 20 |  |  |  |  |

## 7. <Comment>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
|  | commentID | nvarchar | 10 | x | x | PK |  |
|  | postID | nvarchar | 10 |  | x | FK |  |
|  | userID | nvarchar | 10 |  | x | FK |  |
|  | detail | nvarchar | MAX |  | x |  |  |
| 4 | time | Date |  |  | x |  |  |
|  | status | nvarchar | 20 |  |  |  |  |

## 8. <Dating>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
|  | targetID | nvarchar | 10 | x | x | PK |  |
|  | userID | nvarchar | 10 |  | x | FK |  |
|  | isLike | bit |  |  |  |  |  |

## 9. <DatingProfile>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
|  | userID | nvarchar | 10 | x | x | PK |  |
|  | location | nvarchar | 10 |  | x |  |  |
|  | isHaveChild | 50 |  |  | x |  |  |
|  | detail | nvarchar | MAX |  | x |  |  |