

CS457A Final Project Report

NPU-ProfessionalNet

04/09/2014

Team #1

Patel Himani (11766)

Shah Dhara (11781)

Unadkat Hima (11113)

Contents

[Final Functionalities 3](#_Toc384774600)

[Final tasks for each team member 4](#_Toc384774601)

[Final design of database 5](#_Toc384774602)

[E-R Diagram 5](#_Toc384774603)

[Table Design 6](#_Toc384774604)

[Normalization 7](#_Toc384774605)

[Specification for table and it’s column 8](#_Toc384774606)

[Specification of view 12](#_Toc384774607)

[Specification of function 12](#_Toc384774608)

[Specification of stored procedure 12](#_Toc384774609)

[Server-side transaction related operations 13](#_Toc384774610)

[Sample execution of query/SPs 13](#_Toc384774611)

[Sample execution of the application 16](#_Toc384774612)

[Final design of application accessing the database 21](#_Toc384774613)

[Final architecture Diagram 22](#_Toc384774614)

[Components Diagram 23](#_Toc384774615)

[Test Plan 24](#_Toc384774616)

[Any potential improvements 25](#_Toc384774617)

[Project postmortem 25](#_Toc384774618)

# Final Functionalities

* Login
* Change Password
* User
  + Sign up user (add user)
  + Update user profile
  + Remove user profile
  + View user profile
* Connection
  + Add connection
  + Remove connection
  + Show connection hierarchy ex. 2nd level, 3rd level
* Interest or Skill
* Add Interest/Skill
* Remove Interest/Skill
* Group
* Join group
* Post and comment on group wall
* Like and unlike group
* Leave group
* Company
* Follow and un-follow company
* Post on company’s wall
* Wall
* Show first level connection’s wall
* Recommendation
* Send recommendation
* Accept recommendation
* Decline recommendation
* Search
  + Own Connection/Colleague/company/group by name
* Suggestion
  + Connection/company/group

**Final Functionalities - Not planned but included**

* + Show Mutual Colleagues
  + Change password
  + Show connection hierarchy ex. 2nd level, 3rd level
  + Leave group
  + Declined Recommendation

**Functionalities – Planned but not Included**

* + Accept & Hide connection, Invitation to join group
  + School wall
  + Comment on company’s wall
  + Endorsement

# Final tasks for each team member

**Patel Himani (11766)**

* Database design
* Application architecture
* User (Add/remove/view/update)
* Search (company/school/connection/group)
* Connection (Add/remove/suggest/view connections with connection level)
* Connection wall
* View Mutual Colleagues
* Unit testing

**Shah Dhara (11781)**

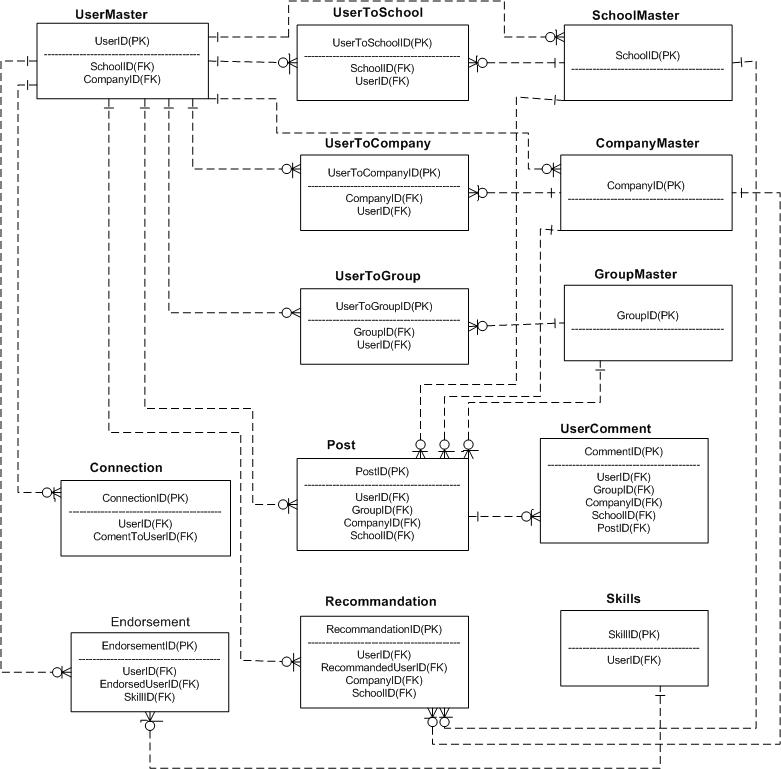
* Database design
* Group (join/leave/view/suggest/like/unlike)
* Group wall (post/comment)
* Company (follow/un-follow/suggest/post)
* Unit testing

**Unadkat Hima (11113)**

* Database design
* Login
* Change Password
* Recommendation (send/view/accept/declined)
* Interest/Skills (Add/Remove/View)
* Unit testing

# Final design of database

### E-R Diagram



### Table Design

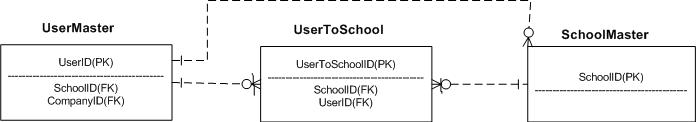
<<Database diagram>>

### Normalization

**Conditions for table to be in 1NF:**

* Table is in 1NF, if Eliminate repeating groups such that all records in all tables can be identified uniquely by a primary key in each table.
* All fields must contain a single value .
* Create a new table to move the repeating groups from the original table.

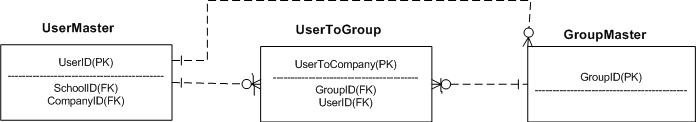
1) To remove redundancy from UserMaster and SchoolMaster, we created a table named “UserToSchool” which is connected to UserMaster and SchoolMaster.In UserToSchool table, we stored the data of users who follow schools. UserId in UserToSchool is Foreign key and refers to UserId of UserMaster. And SchoolID in UserToSchool is Foreign key referring to SchoolID of SchoolMaster.

****

2) To remove redundancy from UserMaster and CompanyMaster, we created a table named “UserToCompany” which is connected to UserMaster and CompanyMaster.In UserToCompany table, we stored the data of users who follow Companies. UserId in UserToCompany is Foreign key and refers to UserId of UserMaster. And CompanyID in UserToCompany is Foreign key referring to CompanyID of CompanyMaster.



3) To remove redundancy from UserMaster and GroupMaster, we created a table named “UserToGroup” which is connected to UserMaster and GroupMaster.In UserToGroup table, we stored the data of users who follow groups or who created group. UserId in UserToGroup is Foreign key and refers to UserId of UserMaster. And GroupID in UserToGroup is Foreign key referring to GroupID of GroupMaster.

****

### Specification for table and it’s column

1. **UserMaster: This table will store information about User detail.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| UserID | uniqueidentifier | PK | Unique User ID.Generated automatically. |
| FName | VARCHAR(25) |  | First name of user |
| LName | VARCHAR(25) |  | Last name of user |
| Email | VARCHAR(35) | Unique | Email address |
| Password | VARCHAR(20) |  | Password |
| Country | VARCHAR(25) |  | Current country |
| ZipCode | INT |  | Zipcode |
| UserType | VARCHAR(20) |  | Employed / Job seeker / Student |
| CompanyID | GUID | FK | User’s company information |
| SchoolID | GUID | FK | User’s School Information |
| CreatedOn | DATETIME |  | To keep track of when the user profile is created. |
| UpdatedOn | DATETIME |  | To keep track of Updation |
| Job Title | Vatchar(25) |  | Title of job. |

1. **Connection: This table will store information about user connected with another user.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| ConnectionID | uniqueidentifier | PK | Unique ConnectionId.Generated automatically. |
| UserID | uniqueidentifier | FK |  |
| ConnectedToUserID | uniqueidentifier |  | Userid of connected userid |
| CreatedOn | DATETIME |  | To show Date and Time |

1. **GroupMaster : This table will store information about different Groups.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| GroupID | uniqueidentifier | PK | Unique GroupId.Generated automatically. |
| GroupName | VARCHAR(25) |  | Name of Group |
| CreatedOn | DATETIME |  | To show Date and Time |

1. **UserToCompany: This table will store information about User’s Company.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| UserCompanyID | uniqueidentifier | PK | Unique ID.Generated automatically. |
| UserID | uniqueidentifier | FK | UserId, who connected to Company |
| CompanyID | uniqueidentifier | FK | CompanyId,to which user Connected |
| Isfollow | BIT |  | True-if following, False-if not following |
| JoinYear | Int |  | Year when user joined company |
| EndYear | Int |  | Year when user left company. |
| CreatedOn | DATETIME |  | To store date and time. |

1. **UserToSchool: This table will store information about User’s School.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| UserSchoolID | uniqueidentifier | PK | Unique Id.Generated automatically. |
| UserID | uniqueidentifier | FK | UserId of user. |
| SchoolID | uniqueidentifier | FK | SchoolId of school, in which user is connected. |
| Isfollow | BIT |  | True-if following, False-if not following |
| JoinYear | Int |  | Year when user joined School. |
| EndYear | Int |  | Year when user left School. |
| CreatedOn | DATETIME |  | To store date and time. |

1. **UserToGroup: This table will store information about User’s Groups.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| UserGroupID | uniqueidentifier | PK | Unique ID.Generated automatically. |
| UserID | uniqueidentifier | FK | UserId of user. |
| GroupID | uniqueidentifier | FK | GroupId Of group to which user is connected. |
| IsLike | BIT |  | True-if following, False-if not following |
| JoinDate | DATETIME |  | Date when user joined Group. |
| CreatedOn | DATETIME |  | To store date and time. |
| UpdatedOn | DATETIME |  | To keep track of Updation. |

1. **Post: This table will store information about user’s posts in any Group,School or Company.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| PostID | uniqueidentifier | PK | Unique ID.Generated automatically. |
| UserID | uniqueidentifier | FK | UserId of user, who posted on group/school/company. |
| GroupID | uniqueidentifier | FK | GroupID on which User Posted. |
| CompanyID | Uniqueidentifier | FK | CompanyID on which User Posted. |
| SchoolID | Uniqueidentifier | FK | SchoolID on which User Posted. |
| PostedOn | DATETIME |  | To store date and Time. |
| Post | Varchar(Max) |  | Actual post. |

1. **UserComment: This table will store user’s comment on post in any Group, School or Company.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| CommentID | uniqueidentifier | PK | Unique ID. Generated automatically. |
| UserID | uniqueidentifier | FK | User’s ID |
| GroupID | uniqueidentifier | FK | GroupID on which User Posted. |
| CompanyID | uniqueidentifier | FK | CompanyID on which User Posted. |
| SchoolID | uniqueidentifier | FK | SchoolID on which User Posted. |
| PostID | Uniqueidentifier | FK | NOT NULL |
| CommentedOn | DATETIME |  | To store date and time. |
| CommentDesc | Varchar(max) |  | Description of comment. |

1. **CompanyMaster: This table will store all detail about Company.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| CompanyID | uniqueidentifier | PK | Unique ID. Generted automatically. |
| CompanyName | VARCHAR(20) |  | Name of the company |
| IsDelete | BIT |  | True-if deleted  false –if not deleted (By default) |
| CreatedOn | DATETIME |  | To show date and time |

1. **SchoolMaster: This table will store all detail about School.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| SchoolID | uniqueidentifier | PK | Unique Id,Generated automatically |
| SchoolName | VARCHAR(20) | FK | Name of School |
| IsDelete | BIT |  | True-if deleted  false –if not deleted (By default) |
| CreatedOn | DATETIME |  | To show date and time |

1. **Skills: This table will store all information about skills/interests.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| SkillID | uniqueidentifier | PK | Unique ID, generated automatically |
| SkillName | VARCHAR(20) |  | Names of technology of expertise. |
| UserID | uniqueidentifier | FK | User’s ID |
| IsDelete | BIT |  | True-if deleted  false –if not deleted (By default) |
| CreatedOn | DATETIME |  | To show date and time |
| UpdatedOn | DATETIME |  | To show date and time |

1. **Endorsement: This table will store endorsement for given skill by connected User.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| EndorsementID | uniqueidentifier | PK | Unique ID,Generated automatically |
| SkillID | Uniqueidentifier | FK | SkillID,on which user Endorsed other user. |
| SkillName | Varchar(20) |  | Name of Skill of user. |
| UserID | uniqueidentifier | FK | UserID who is endorsing other User for Skill. |
| EndorsedUserID | uniqueidentifier |  | UserId or Endorsed User. |
| CreatedOn | DATETIME |  | To show date and time. |

1. **Recommendation: This table will store recommendation for given experience by connected User.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| RecommendationID | uniqueidentifier | PK | Unique ID,Generated automatically. |
| UserID | uniqueidentifier | FK | UserId who gives recommendation. |
| CompanyID | uniqueidentifier | FK | Name of Company |
| RecommmendedUserID | Uniqueidentifier |  | User to whom Recommandation is given. |
| RecommendationText | VARCHAR(MAX) |  | Text that user writes to give recommendation. |
| Status | Varchar(10) |  | Status of recommendation. |
| CreatedOn | DATETIME |  | To show date and time. |
| SchoolID | Uniqueidentifier | FK | School ID |

1. **CommentLikeDetail: This table is used to store comment like and it will keep track of user’s ID**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| CommentDetailID | uniqueidentifier | PK | Unique ID,Generated automatically. |
| UserID | uniqueidentifier |  | UserId who commented. |
| CommentID | uniqueidentifier |  | Actual comment. |

1. **ErrorDetail : This table gives details of error.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Key** | **Description** |
| ErrorId | int |  | Error ID |
| ErrorNo | Varchar(10) |  | No of Error |
| ErrorSeverity | Varchar(10) |  |  |
| ErrorState | Varchar(10) |  |  |
| ErrorProcedure | Varchar(50) |  |  |
| ErrorLine | Varchar(50) |  |  |
| ErrorMessage | Varchar(50) |  | Actual Error |
| ErrorOn | Datetime |  | Gives date and time |

### Specification of view

|  |  |
| --- | --- |
| **View** | **Purpose** |
| vGetAllRecommendation | Retrieve all recommendations and company, school name for which user recommended and also user name who recommended. |

### Specification of function

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Input Parameter** | **Output** | **Purpose** |
| countPostComment | PostId (Uniqueidentifier) | Count of comments (Int) | Return number of comments on given post |

### Specification of stored procedure

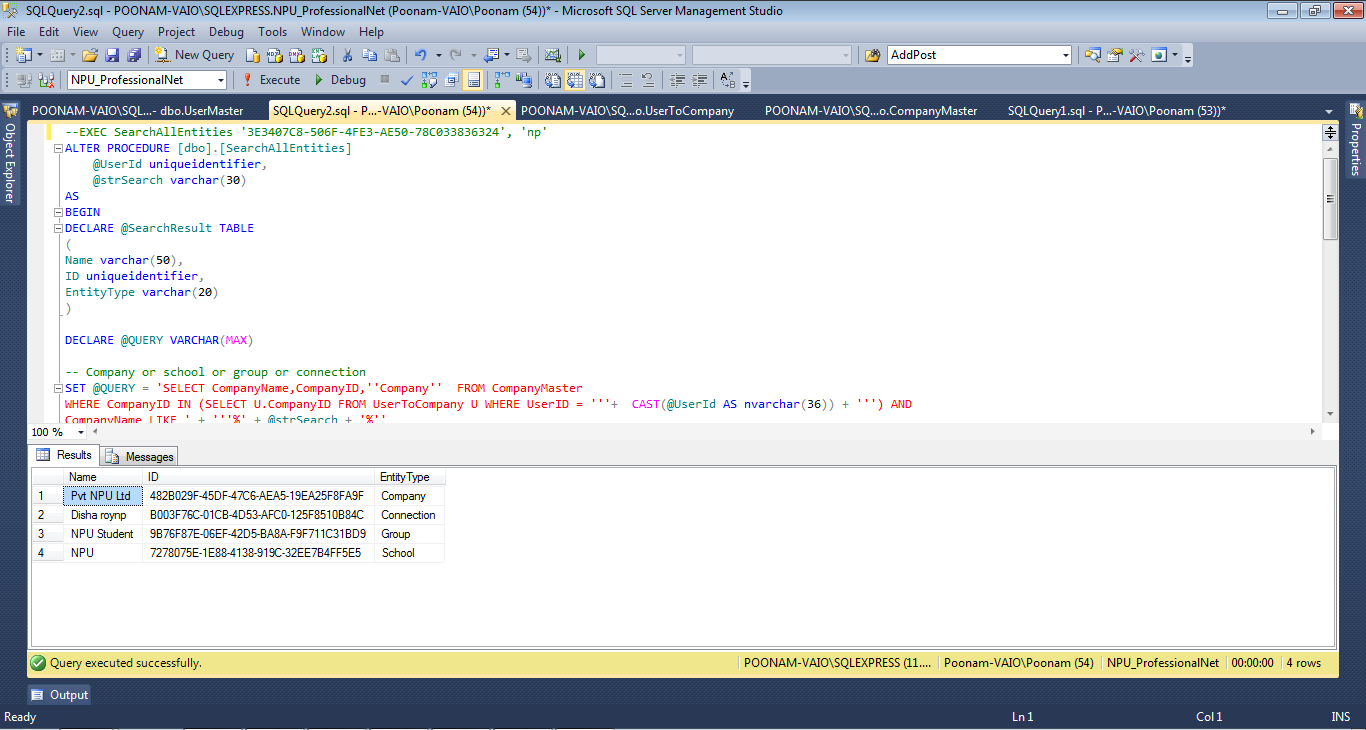
|  |  |  |
| --- | --- | --- |
| **No** | **Stored Procedure** | **Purpose** |
| 1 | AddConnection | When a user wants to connect to other user. |
| 2 | AddPost | When user posts on company/group |
| 3 | AddSkill | When user wants to Add any skill to his/her profile. |
| 4 | AddComment | To add comment on group/company/school |
| 5 | AddRecommandation | To give recommendation to users. |
| 6 | AddUserToCompany | To add any company as a past or present experience to profile. |
| 7 | AddUserToSchool | To add Past/Present School to Profile. |
| 8 | ChangePassword | When User wants to change password,this procedure will ask for the current password,if it matches in database,then it will allow user to change password. |
| 9 | DeleteConnection | To delete existing connections. |
| 10 | DeleteSkill | To delete existing skill in profile. |
| 11 | DeleteUser | To delete user account |
| 12 | GetAllCompanyList | To display all the companies. |
| 13 | GetAllGroups | To display all the available groups. |
| 14 | GetAllSchoolList | To display all the schools. |
| 15 | GetAvailableConnection | To list connections which are not yet connected to logged in user. |
| 16 | GetCommentByPostID | To list comment based on post ID. |
| 17 | GetConnectionByUserID | To get List of connections by User ID |
| 18 | GetCountforConnectionbyUserID | To display No of Connections of user. |
| 19 | GetCountForGroupByUserId | To display No of groups by user ID. |
| 20 | GetGroupByUserId | Retrieve joined groups by userid |
| 21 | GetGroups | To get list of groups, which are not joined by user. |
| 22 | GetMutualColleague | Retrieve connection those are in the same company |
| 23 | GetPostByGroupID | To display all the Posts of Group by GroupID |
| 24 | GetRecommendationByUserId | Retrieve recommendation by user id |
| 25 | GetSkillByUserID | To list the Skill of User by userID |
| 26 | GetUserByEmailAndPassword | To retrieve user based on email and password (log in process) |
| 27 | GetUserToGroupByGroupID | To list groups created by User ,by groupId |
| 28 | GetUserByUserID | To retrieve all the information of User |
| 29 | GetFollowingCompanyList | To retrieve list of Company which is followed by logged-in user |
| 30 | GetUnFollowingCompanyList | To retrieve list of Company which is not followed by logged-in user. It will suggest user to follow that compnay |
| 31 | GetPostByCompanyID | To retrieve created posts by userid |
| 32 | FollowCompany | To list followed company by user |
| 33 | UnfollowCompany | To list Un-Followed company by user |
| 34 | IsEmailExists | To check weather email address is exists in database or not. |
| 35 | JoinGroup | To join the group by user |
| 36 | LeaveGroup | To leave the group. |
| 37 | SearchAllEntities | To search by connection/company/group/school/etc. |
| 38 | SignUpUser | To store user information, who is creating account for the first time. If the email address given by the user is matches with the emails stored in database, this will give error to the user and asks to choose other email address. |
| 39 | UpdateGroupLike | To update user like/unlike for any group |
| 40 | UpdateRecommendationStatus | Update recommendation status approve, decline |
| 41 | UpdateUser | To edit user’s Profile |
| 42 | Usp\_getErrorInfo | To get error information like Error message, Error Line,Error Number,Error State,etc |

### Server-side transaction related operations

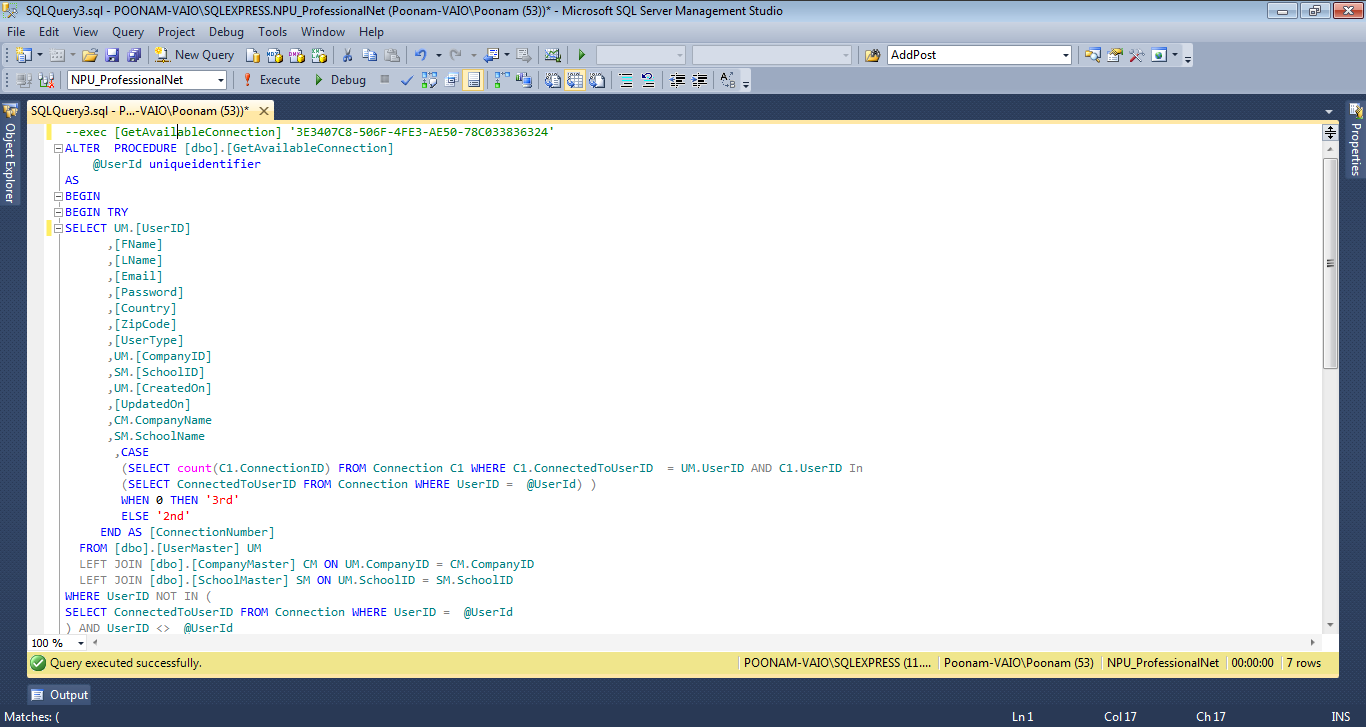
1. Login to the system
2. Signup for new user
3. Change Password
4. Verify that email id exist or not
5. Update/View/Remove Profile
6. Add/View/Delete Connection
7. Search own company, school, connection, group
8. Suggest group, company, connection
9. Join/Leave/Like/Unlike Group
10. Post in group/company
11. Comment on group post
12. Add/Delete Skills
13. Send/Accept/Declined Recommendation
14. View Mutual Colleagues

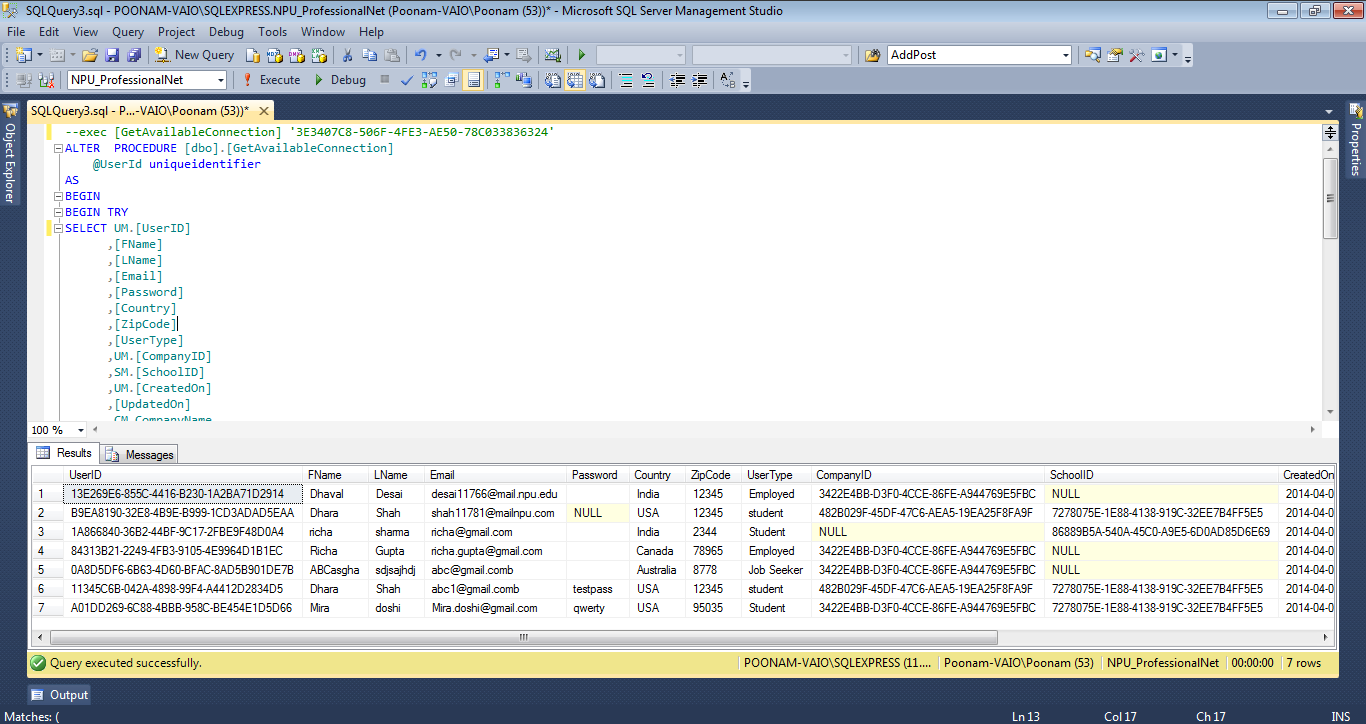
### Sample execution of query/SPs

* Search own company, school, connection, group by name

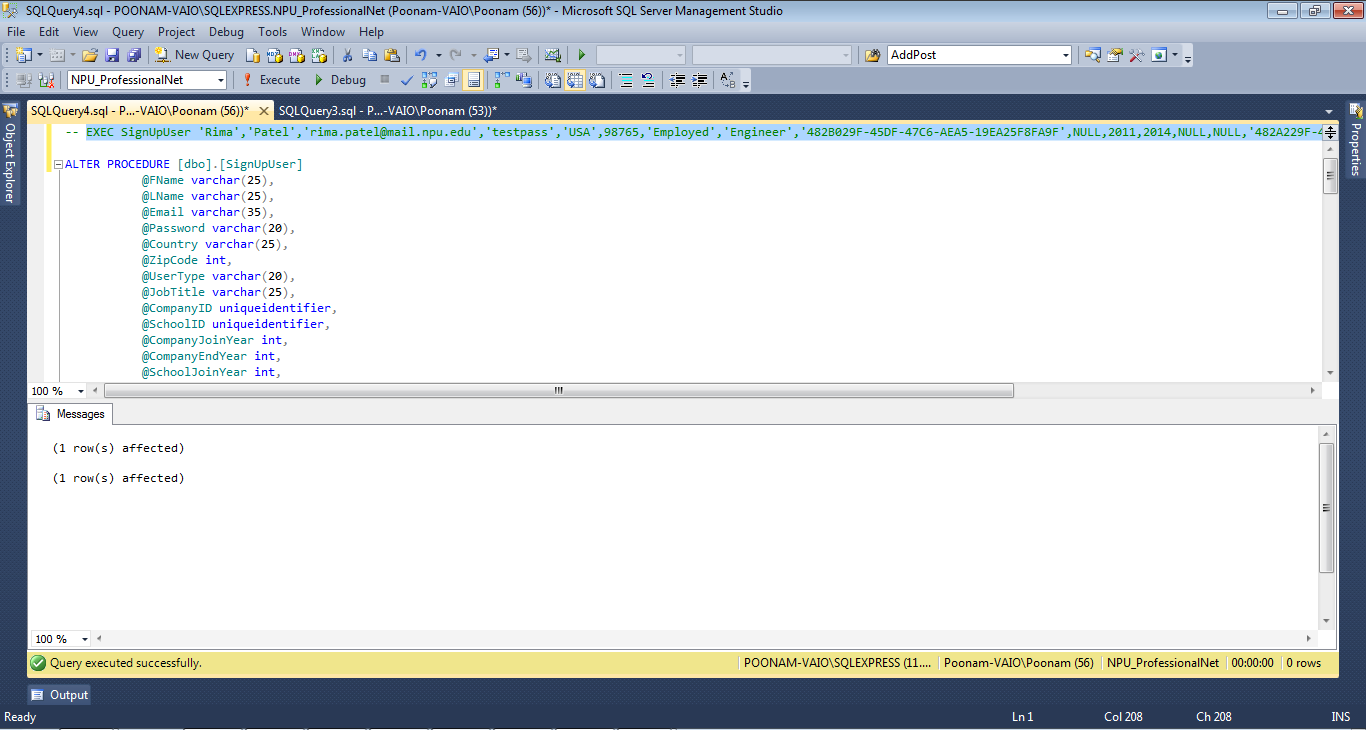


* Retrieve all connection (3rd and 2nd) level connections with whom logged in user not connected yet

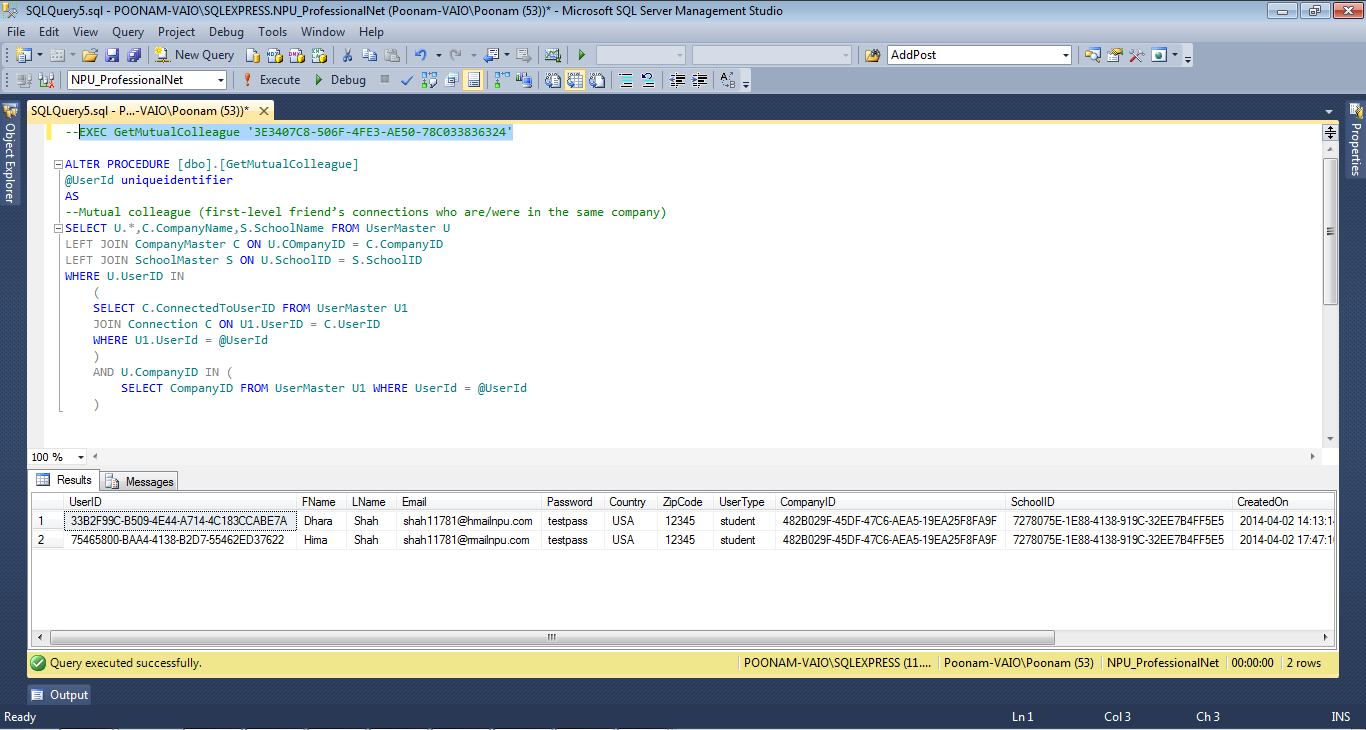
****

****

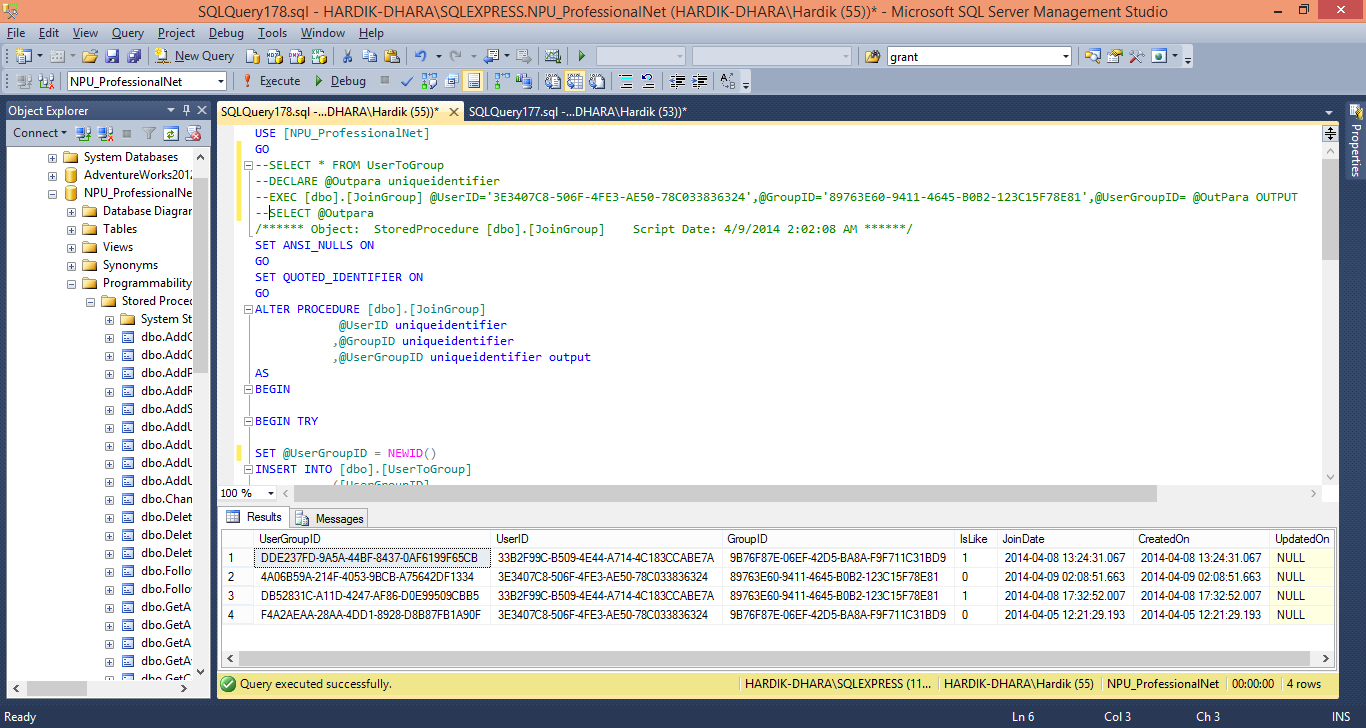
* Sign up user functionality (This SP will add user details to User Master table and insert entry to UserToCompany or UserToSchool table based on company and school selection)



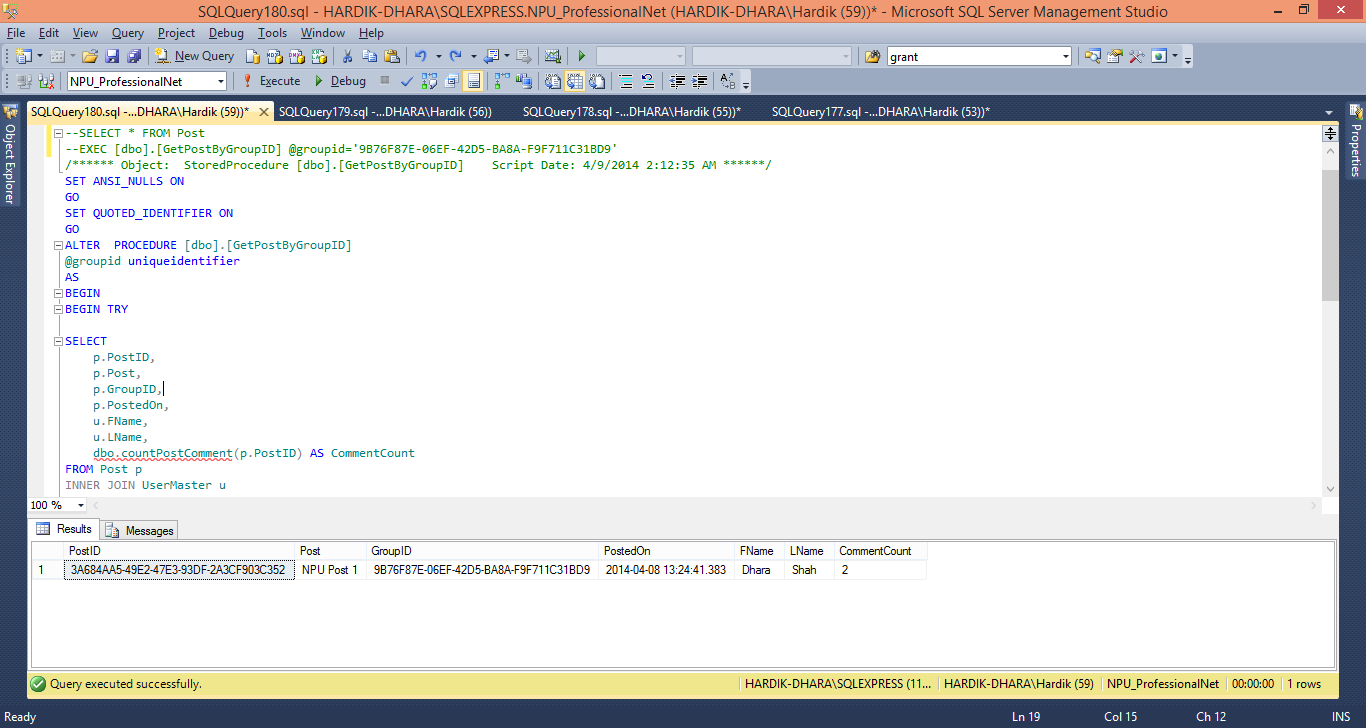
* Retrieve Mutual Colleagues(Mutual colleague (first-level friend’s connections who are/were in the same company)

****

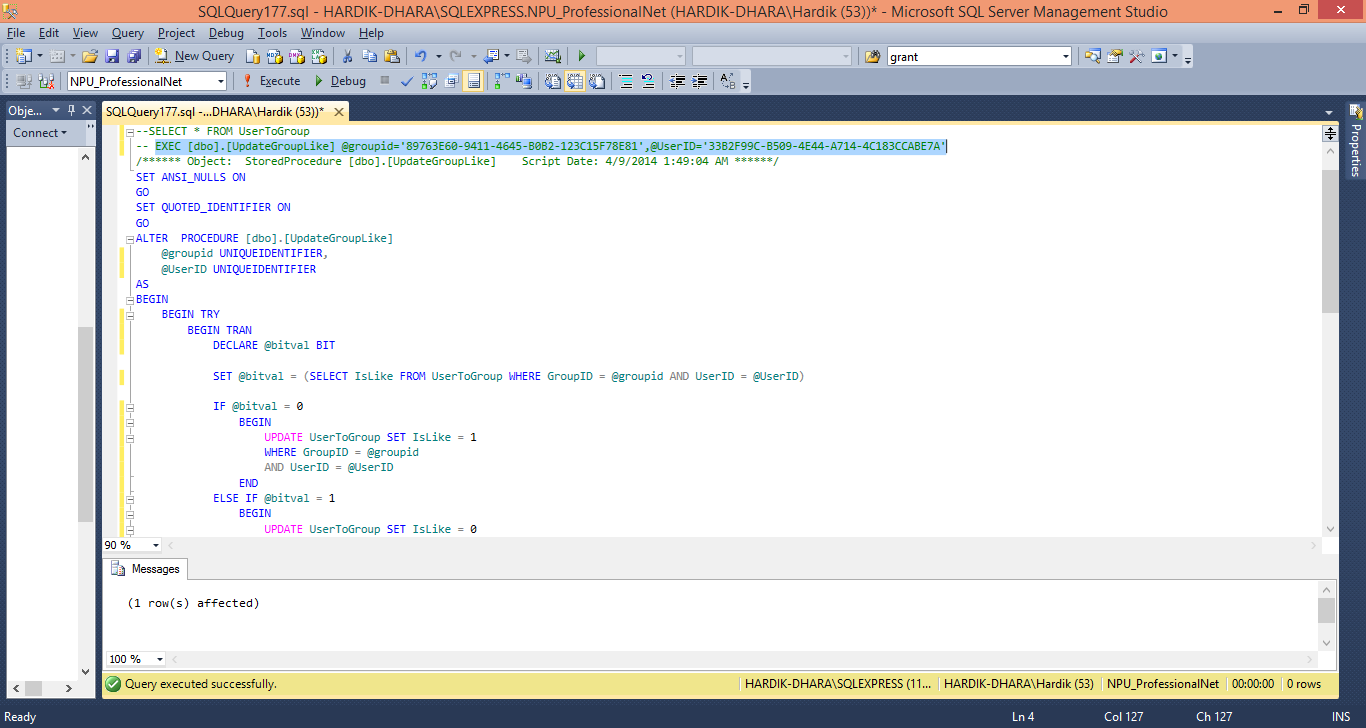
* Join Group: It will insert data in to table when user wants to join any group



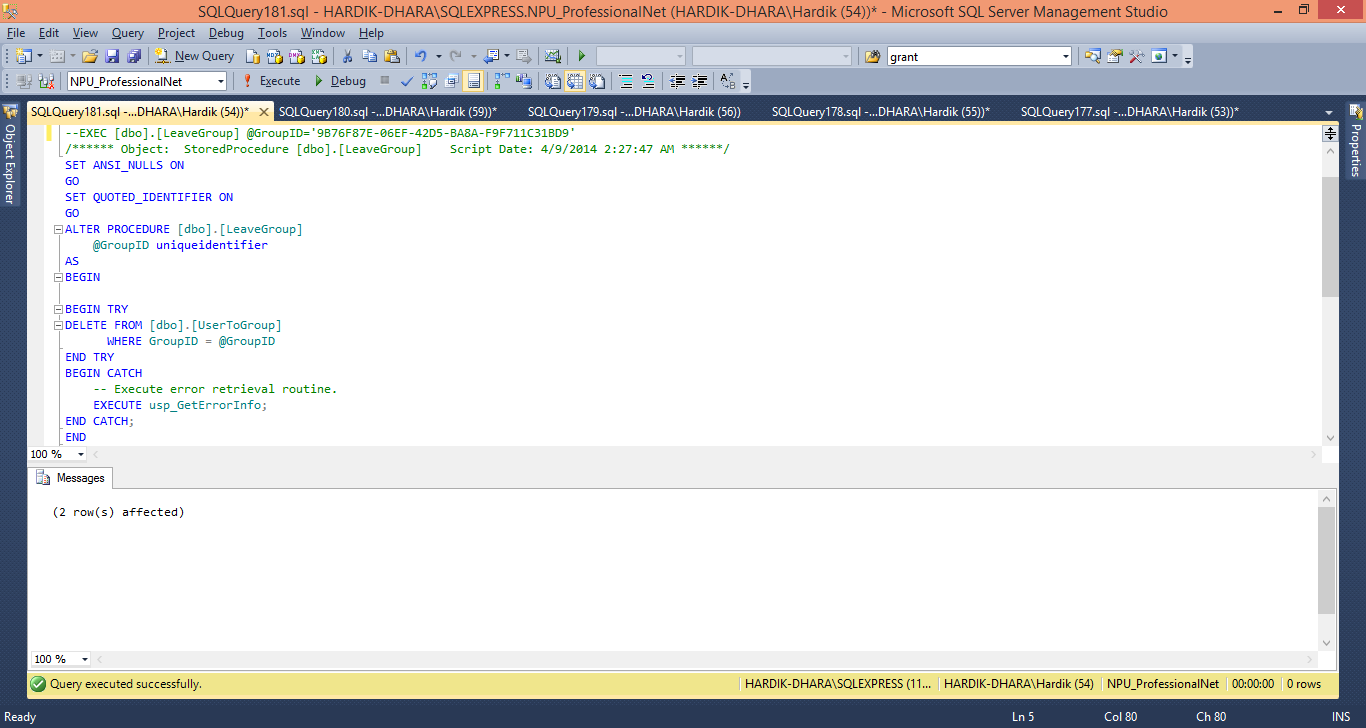
* This will return total posts created by user for particular group(It will also get count of comments if any on that post)



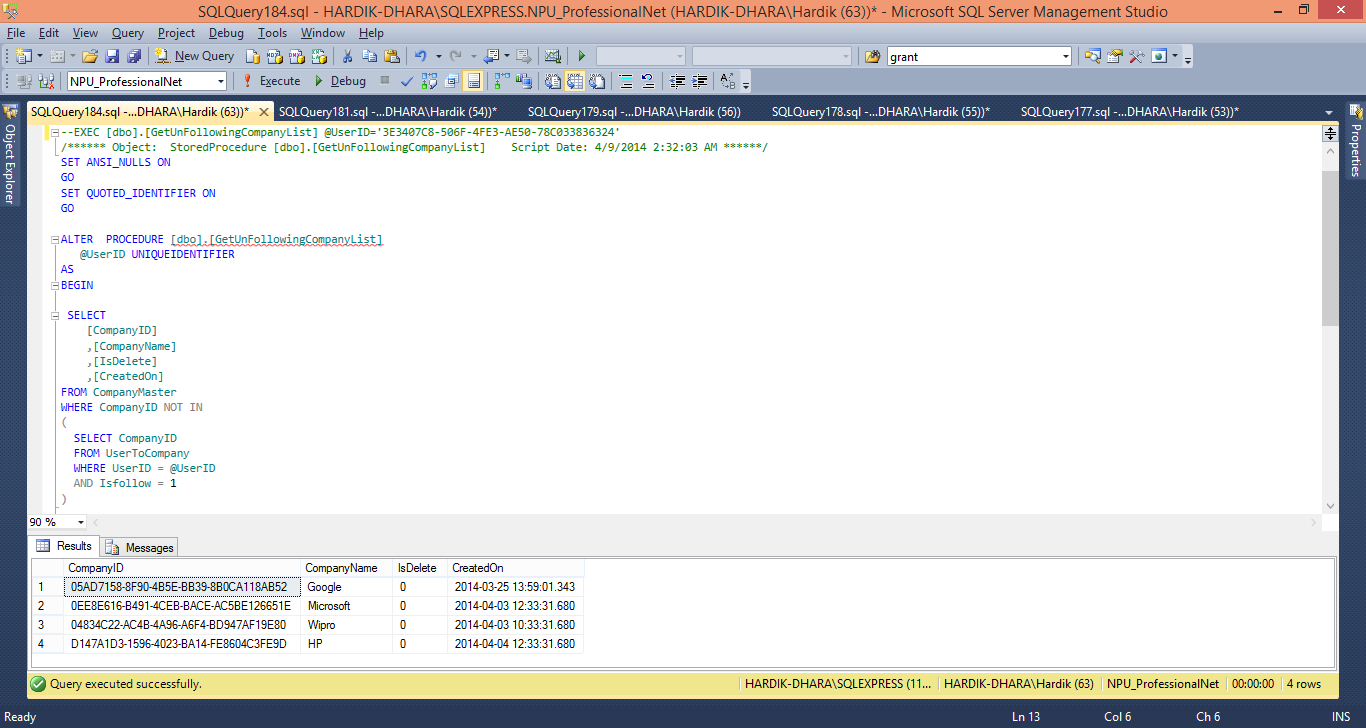
* This will update Like/Unlike for Group



* This procedure will delete rows when user wants to leave that particular group



* This will give suggestion to user for the company if he/she would like to follow

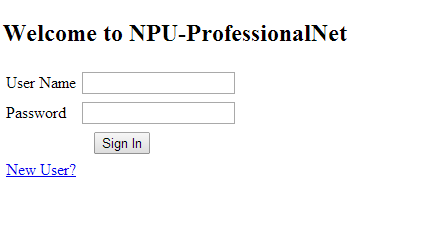


**Final design of accessing the database**

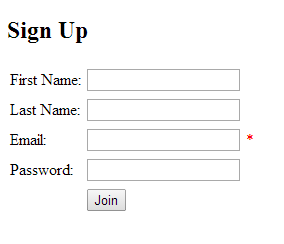
|  |  |
| --- | --- |
| Technology, framework, language, etc | ASP.NET framework 4.5 ,C# |
| DB access technology | SQL SERVER 2012 |
| client-side transaction | NO |

### Sample execution of the application

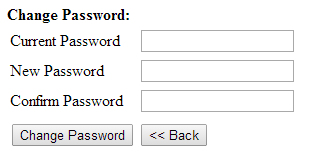
1. **Login page**



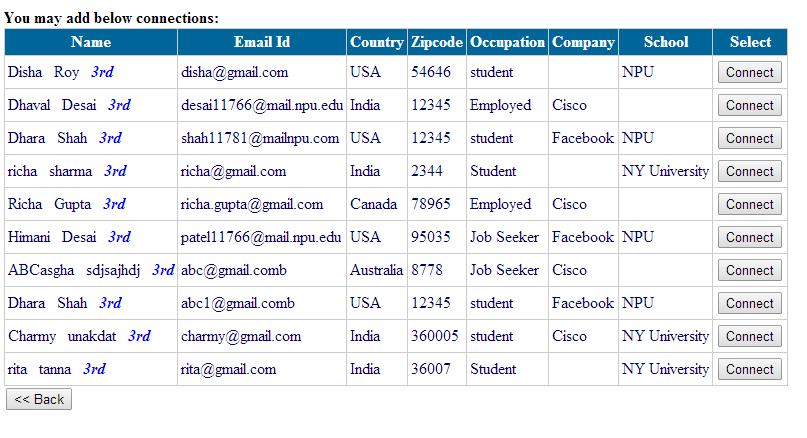
1. **Sign up page**



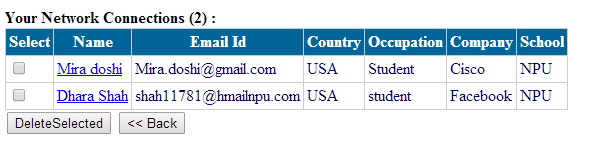
1. **User home page**
2. **Change Password**



1. **Add Connection**



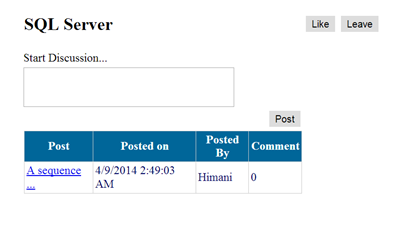
1. **Manage Connection**



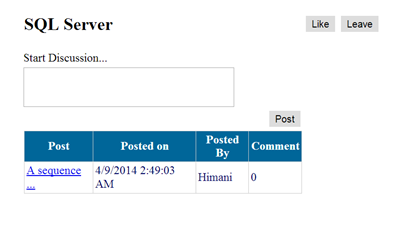
1. **Connection Wall**
2. **Group**

****

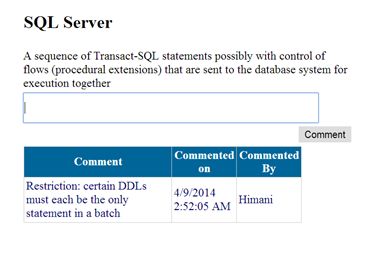
1. **Group Wall**

****

1. **Post on Group wall**

****

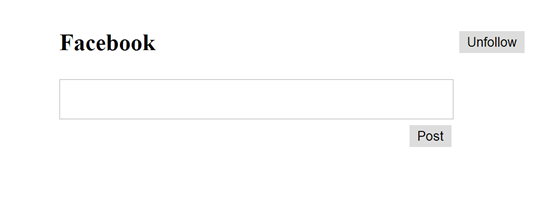
1. **Post detail**

****

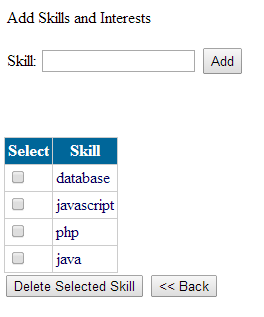
1. **Company**

****

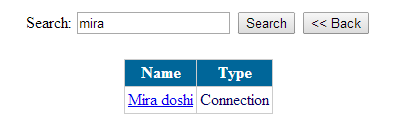
1. **Post on Followed Company**

****

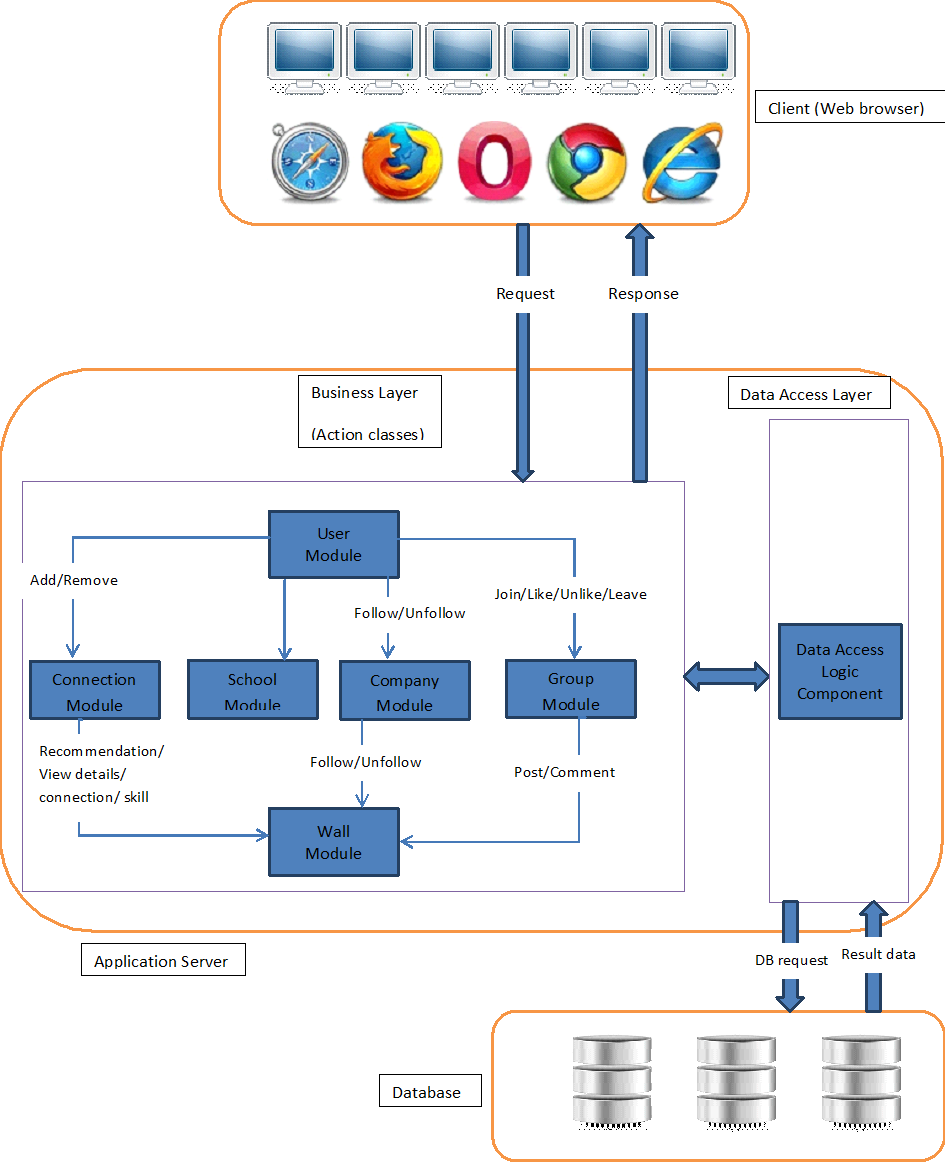
1. **Skills**



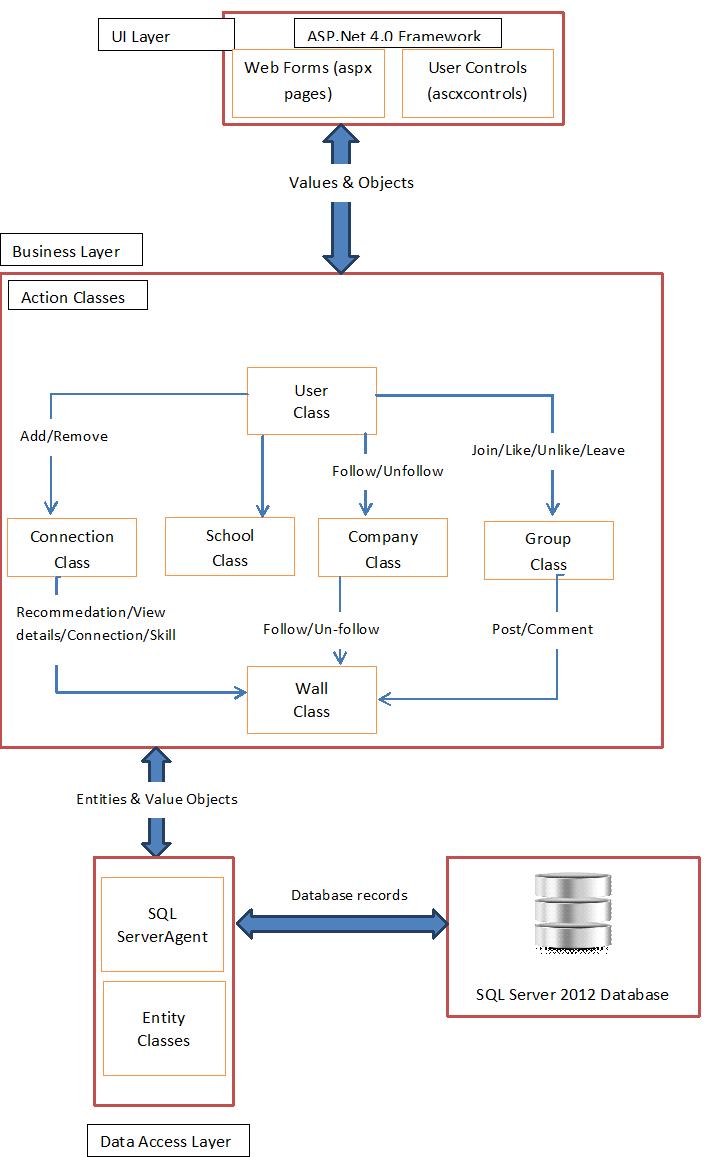
1. **Recommendation**
2. **Mutual Colleagues**
3. **Search**



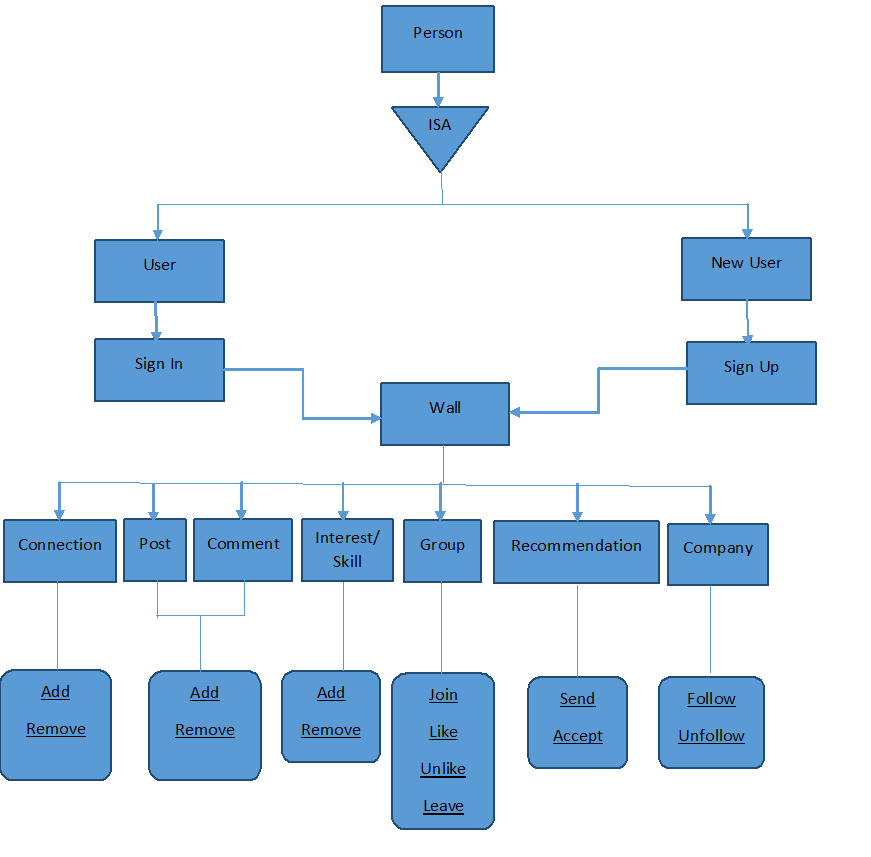
# Final design of application accessing the database

****

# Final architecture Diagram

****

# Components Diagram



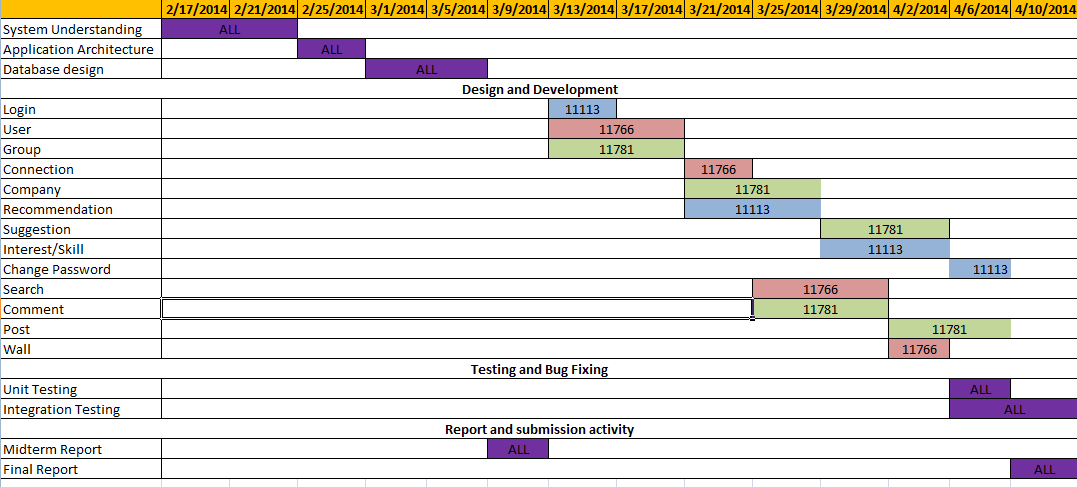
**Major design decisions**

1. We added UserToSchool, UserToCompany and userToGroup3 extra tables to normalize the data we stored in tables. If we don’t include these tables, the data was stored more than one time in different table. In that tables, we added column, by which we can easily retrieve.

**Reason for Design Decision**

1. Normalization is necessary for removing redundancy in tables. So instead of storing same value multiple times in different tables, we use normalization.

**Final major milestones and ToDo by each members**



# Test Plan

* Prepare test data
* Unit Testing
  + High level end to end testing of each module
  + Validate data entry
* Integration testing
  + Test all functionalities with test data
  + Check application execution flow

# Any potential improvements

# Project postmortem