Code Inspection

# Interfaces

Please find below the list of Interfaces and the details with respect to every interface used in our code.

**ILoginRepository**

/\*

Component : An interface which declares the methods that need to be defined in 'ILoginRepository.cs'

Author: Abhinav Bhandaram

Use of the component in system design: Good coding practice

Written and revised: 11/14/2016

Reason for component existence: Acts as a contract that specifies the list of all methods that need to be defined in 'ILoginRespository.cs'

Component usage of data structures, algorithms and control(if any):

The component contains the declaration of below methods:

Login(string uName, string uPassword)

Theis method is defined in 'ILoginRespository.cs'

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WhatsUrSay.Models;

namespace WhatsUrSay.Interfaces

{

interface ILoginRepository

{

bool Login(string uName, string uPassword);

}

}

**IUserRepository**

Component : An interface which declares the methods that need to be defined in 'IUserRepository.cs'

Author: Abhinav Bhandaram

Use of the component in system design: Good coding practice

Written and revised: 11/14/2016

Reason for component existence: Acts as a contract that specifies the list of all methods that need to be defined in 'IUserRepository.cs'

Component usage of data structures, algorithms and control(if any):

The component contains the declaration of below methods:

Add(User user), User Find(string uName)

Theis method is defined in 'IUserRepository.cs'

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WhatsUrSay.Models;

namespace WhatsUrSay.Interfaces

{

interface IUserRepository

{

//Declarations of the methods Add and Method Find

User Add(User user);

User Find(string uName);

}

}

**IAnswerRepository**

/\*

Component : An interface which declares the methods that need to be defined in 'AnswerResp.cs'

Author: NikhithaKaza

Use of the component in system design: Good coding practice

Written and revised: 11/14/2016

Reason for component existence: Acts as a contract that specifies the list of all methods that need to be defined in 'AnswerResp.cs'

Component usage of data structures, algorithms and control(if any):

The component contains the declaration of below methods:

GetAll(), Get(int id), Add(Answer Act)

Theis method is defined in 'AnswerResp.cs'

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//It includes all the models from the project WhatsUrSay

using WhatsUrSay.Models;

namespace WhatsUrSay

{

interface IAnswer

{

IEnumerable<Answer> GetAll();

Answer Get(int id);

Answer Add(Answer Act);

}

}

**ISurveyRepository**

/\*

Component : An interface which declares the methods that need to be defined in 'SurveyResp.cs'

Author: NikhithaKaza

Use of the component in system design: Good coding practice

Written and revised: 11/14/2016

Reason for component existence: Acts as a contract that specifies the list of all methods that need to be defined in 'SurveyResp.cs'

Component usage of data structures, algorithms and control(if any):

The component contains the declaration of below methods:

GetAll(), Activity Get(int id), Activity Add(Activity Act);

Theis method is defined in 'SurveyResp.cs'

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//It includes all the models from the WhatsUrSay

using WhatsUrSay.Models;

namespace WhatsUrSay

{

interface ISurvey

{

IEnumerable<Activity> GetAll();

Activity Get(int id);

Activity Add(Activity Act);

}

}

**IPollRepository**

/\*

Component : An interface which declares the methods that need to be defined in 'PollRepository.cs'

Author: Sreedevi Koppula

Use of the component in system design: Good coding practice

Written and revised: 11/14/2016

Reason for component existence: Acts as a contract that specifies the list of all methods that need to be defined in 'PollRepository.cs'

Component usage of data structures, algorithms and control(if any):

The component contains the declaration of below methods:

'GetAll()', 'Get(int id)', 'Add(Activity activity)'

These methods are defined in 'PollRepository.cs'

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using WhatsUrSay.Models;

namespace WhatsUrSay.Interfaces

{

interface IPollRepository

{

IEnumerable<Activity> GetAll();

Activity Get(int id);

Activity Add(Activity activity);

}

}

**IQuestionRepository**

/\*

Component : An interface which declares the methods that need to be defined in 'PollRepository.cs'

Author: Sreedevi Koppula

Use of the component in system design: Good coding practice

Written and revised: 11/14/2016

Reason for component existence: Acts as a contract that specifies the list of all methods that need to be defined in 'PollRepository.cs'

Component usage of data structures, algorithms and control(if any):

The component contains the declaration of below methods:

'GetAll()', 'Get(int id)', 'Add(Activity activity)'

These methods are defined in 'PollRepository.cs'

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using WhatsUrSay.Models;

namespace WhatsUrSay.Interfaces

{

interface IPollRepository

{

IEnumerable<Activity> GetAll();

Activity Get(int id);

Activity Add(Activity activity);

}

}

**IGroupsRepository**

/\* File Name :IGroupsRepository.cs

\* Created By: Raj

\* This file resides in business layer

\* Change History

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* PR Date Author Description

\*\* -- -------- ------- ------------------------------------

\*\* 1 11/15/2016 Raj Created GroupDetails class and incuded properties

\* This interface forces GroupsRepository to implement method specified in this file.

\*

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using WhatsUrSay.Models;

namespace WhatsUrSay.Interfaces

{

public interface IGroupsRepository

{

/// <summary>

/// Method definitions to force GroupRepository to implements these methods

/// </summary>

/// <returns></returns>

IQueryable<Group> GetGroups();

GroupDetails GetGroup(int groupId);

bool UpdatGroup(Group group);

Group CreateGroup(Group group);

Group DeleteGroup(int id);

}

}

# Controllers

In our code we have used 3 types of controllers in our project, they are:

1. MVC controller
2. Web API controllers
3. Angular JS Controllers.

**MVC Controllers**

**Home Controller:** Please find below the details corresponding to the home controller.

/\*

Component : A Web Api controller that invokes the single page application 'View'

Author: Abhinav Bhandaram

Written and revised: 11/5/2016

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WhatsUrSay.Controllers

{

public class HomeController : Controller

{

// GET: Login

public ActionResult Index()

{

return View();

}

}

}

**Web API Controllers**

**User Controller**

/\*

Component : A Web Api controller that invokes the methods defined in 'UserRepository.cs' to serve the client requests in performing 'insert' and 'read' operations on the 'User' table of our project's database (DSE)

Author: Abhinav Bhandaram

Use of the component in system design: Serves the requests of the clients

Written and revised: 11/5/2016

Reason for component existence: To serve the requests of the clients in performing 'insert' and 'read' operations on the 'User' table

Component usage of data structures, algorithms and control(if any):

The component contains the below methods:

Get(), Get(string uName), Post([FromBody]User user)

These methods are invoked by the clients upon their request for a particular service on 'User' table

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using WhatsUrSay.Repositories;

using WhatsUrSay.Interfaces;

using WhatsUrSay.Models;

namespace WhatsUrSay.Controllers

{

public class UserController : ApiController

{

static readonly IUserRepository objRepository = new UserRepository();

// GET api/<controller>/5

//Purpose: Invokes 'Get(string uNmae)' method

//Input: None

//Output: A list of answer records from the 'User' table

public User Get(string uName)

{

return objRepository.Find(uName);

}

// POST api/<controller>

//Purpose: Invokes 'Post([FromBody]User user)' method

//Input: None

//Output: A list of answer records from the 'User' table

public bool Post([FromBody]User user)

{

User objUser = objRepository.Add(user);

if (User != null)

{

return true;

}

else

{

return false;

}

}

}

}

**Poll Controller**

/\*

Component : A Web Api controller that invokes the methods defined in 'PollRepository.cs' to serve the client requests in performing 'insert' and 'read' operations on the 'Activity' table of our project's database (DSE)

Author: Sreedevi Koppula

Use of the component in system design: Serves the requests of the clients

Written and revised: 11/5/2016

Reason for component existence: To serve the requests of the clients in performing 'insert' and 'read' operations on the 'Activity' table

Component usage of data structures, algorithms and control(if any):

The component contains the below methods:

'GetAllPolls()', 'GetPoll(int id)', 'PostPoll(Activity activity)'

These methods are invoked by the clients upon their request for a particular service on 'Activity' table

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using WhatsUrSay.Models;

using WhatsUrSay.Repositories;

namespace WhatsUrSay.Controllers

{

public class PollController : ApiController

{

PollRepository repo = new PollRepository();

//Purpose: Invokes 'GetAll()' method of PollRepository.cs that returns all the records of type 'poll' from the 'Activity' table

//Input: None

//Output: A list of poll records from the 'Activity' table

public IEnumerable<Activity> GetAllPolls()

{

return repo.GetAll();

}

//Purpose: Invokes 'Get(int id)' method of PollRepository.cs that returns a record from the 'Activity' table whose key is 'id'

//Input: 'id' of the required record

//Output: a record from 'Activity' table whose key is 'id'

public Activity GetPoll(int id)

{

return repo.Get(id);

}

//Purpose: Invokes 'Add(Activity activity) method of 'PollRepository.cs' that adds an object 'activity' in the 'Activity' table

//Input: 'activity' object of type 'Activity.cs'

//Output: Returns the object 'activity' upon its successful addition in the table

public Activity PostPoll(Activity activity)

{

return repo.Add(activity);

}

}

}

**Survey Controller**

Component : A Web Api controller that invokes the methods defined in 'SurveyRepo.cs' to serve the client requests in performing 'insert' and 'read' operations on the 'Activity' table of our project's database (DSE)

Author: Nikhitha Kaza

Use of the component in system design: Serves the requests of the clients

Written and revised: 11/5/2016

Reason for component existence: To serve the requests of the clients in performing 'insert' and 'read' operations on the 'Activity' table

Component usage of data structures, algorithms and control(if any):

The component contains the below methods:

GetAllSurveys(), PostSurvey(Activity Act)

These methods are invoked by the clients upon their request for a particular service on 'Activity' table

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using WhatsUrSay.Models;

namespace WhatsUrSay.Controllers

{

public class SurveyController : ApiController

{

//It reads the Isuvery interface

static readonly ISurvey Survey1 = new SurveyResp();

//GetAllSurveys api/<controller>

//Purpose: Invokes 'GetAllSurveys()' method of SurveyRepo.cs that returns all the records of type 'survey' from the 'Survey' table

//Input: None

//Output: A list of answer records from the 'Survey' table

public IEnumerable<Activity> GetAllSurveys()

{

return Survey1.GetAll();

}

//PostSurvey api/<controller>

//Purpose: Invokes 'PostSurvey(Activity Act) method of 'ActivityRepo.cs' that adds an object 'activity' in the 'Activity' table

//Input: 'activity' object of type 'Activity.cs'

//Output: Returns the object 'activity' upon its successful addition in the table

public Activity PostSurvey(Activity Act)

{

return Survey1.Add(Act);

}

}

}

**Answer Controller**

/\*

Component : A Web Api controller that invokes the methods defined in 'AnswerRepo.cs' to serve the client requests in performing 'insert' and 'read' operations on the 'Answer' table of our project's database (DSE)

Author: Nikhitha Kaza

Use of the component in system design: Serves the requests of the clients

Written and revised: 11/5/2016

Reason for component existence: To serve the requests of the clients in performing 'insert' and 'read' operations on the 'Answer' table

Component usage of data structures, algorithms and control(if any):

The component contains the below methods:

GetAnswers(), PostSurvey(Answer Act)

These methods are invoked by the clients upon their request for a particular service on 'Answer' table

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

//It includes all the models from the WhatsUrSay

using WhatsUrSay.Models;

using WhatsUrSay.Interfaces;

namespace WhatsUrSay.Controllers

{

public class AnswerController : ApiController

{

// It reads the IAnswer interface.

static readonly IAnswer Answer1 = new AnswerResp();

//GETAnswers api/<controller>

//Purpose: Invokes 'GetAnswers()' method of AnswerRepo.cs that returns all the records of type 'answer' from the 'Answer' table

//Input: None

//Output: A list of answer records from the 'Answer' table

public IEnumerable<Answer> GetAnswers()

{

return Answer1.GetAll();

}

//PostSurvey api/<controller>

//Purpose: Invokes 'PostSurvey(Answer Act) method of 'AnswerRepo.cs' that adds an object 'answer' in the 'Answer' table

//Input: 'activity' object of type 'Answer.cs'

//Output: Returns the object 'answer' upon its successful addition in the table

public Answer PostSurvey(Answer Act)

{

return Answer1.Add(Act);

}

}

}

**QuestionController**

/\*

Component : A Web Api controller that invokes the methods defined in 'QuestionRepository.cs' to serve the client requests in performing 'insert' operation on the 'Question' table of our project's database (DSE)

Author: Sreedevi Koppula

Use of the component in system design: Serves the requests of the clients

Written and revised: 11/5/2016

Reason for component existence: To serve requests of the clients in performing 'insert' operation on the 'Question' table

Component usage of data structures, algorithms and control(if any):

The component contains the below methods:

'PostQuestion(Question question)'

This method is invoked by the clients upon their request for service to add a question in the 'Question' table

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using WhatsUrSay.Repositories;

using WhatsUrSay.Models;

namespace WhatsUrSay.Controllers

{

public class QuestionController : ApiController

{

QuestionRepository repo = new QuestionRepository();

//Purpose: Invokes 'Add(Question question)' method of 'QuestionRepository.cs' that adds an object 'question' in the 'Question' table

//Input: 'question' object of type 'Question.cs'

//Output: Returns the object 'question' upon its successful addition in the table

public Question PostQuestion(Question question)

{

return repo.Add(question);

}

}

}

**GroupsController**

/\* File Name :GroupsController.cs

\* Created By: Raj

\* This File resides in Business Layer

\*

\* Change History

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* PR Date Author Description

\*\* -- -------- ------- ------------------------------------

\*\* 1 11/7/2016 Raj Created GroupsController class and incuded methods to create ,delete,edit group details

\* This file hold business logic related to group related activities and provides services to UI layer up on request.

\*

\*/

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using System.Web.Http.Description;

using WhatsUrSay.Models;

using WhatsUrSay.Repositories;

namespace WhatsUrSay.Controllers

{

public class GroupsController : ApiController

{

GroupsRepository groupRepository = new GroupsRepository();

// GET: api/Groups

/// <summary>

/// This method returns all groups listed in database

/// </summary>

/// <returns></returns>

public IQueryable<Group> GetGroups()

{

return groupRepository.GetGroups();

}

// GET: api/Groups/5

/// <summary>

/// This method get specific group details

/// </summary>

/// <param name="id"></param>

/// <returns>group details</returns>

[ResponseType(typeof(GroupDetails))]

public IHttpActionResult GetGroup([FromUri]int id)

{

GroupDetails groupDetails = groupRepository.GetGroup(id);

return Ok(groupDetails);

}

// PUT: api/Groups/5/

/// <summary>

/// This method updates specific group details

/// </summary>

/// <param name="group"></param>

/// <returns> status code</returns>

[ResponseType(typeof(void))]

public IHttpActionResult PutGroup(Group group)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

try

{

groupRepository.UpdatGroup(group);

}

catch (DbUpdateConcurrencyException)

{

throw;

}

return StatusCode(HttpStatusCode.NoContent);

}

// POST: api/Groups

/// <summary>

/// This method creates new group

/// </summary>

/// <param name="group"></param>

/// <returns>Route of created group</returns>

[ResponseType(typeof(Group))]

public IHttpActionResult PostGroup(Group group)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

try

{

groupRepository.CreateGroup(group);

}

catch (DbUpdateException)

{

throw;

}

return CreatedAtRoute("DefaultApi", new { id = group.id }, group);

}

// DELETE: api/Groups/5

/// <summary>

/// This method deletes specific group

/// </summary>

/// <param name="id"></param>

/// <returns> deleted group details</returns>

[ResponseType(typeof(Group))]

public IHttpActionResult DeleteGroup([FromUri]int id)

{

Group group;

try

{

group = groupRepository.DeleteGroup(id);

}

catch (Exception)

{

throw;

}

return Ok(group);

}

}

}

**Login Controller**

/\*

Component : A Web Api controller that invokes the methods defined in 'LoginRepository.cs' to serve the client requests in performing 'Login' operation on our

Author: Abhinav Bhandaram

Use of the component in system design: Serves the Login requests of the clients

Written and revised: 11/5/2016

Reason for component existence: To serve Login requests of the clients

Component usage of data structures, algorithms and control(if any):

The component contains the below methods:

'Login(UserInput userInfo)'

This method is invoked by the clients upon their request for login into the application.

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using WhatsUrSay.Repositories;

using WhatsUrSay.Models;

using WhatsUrSay.Interfaces;

namespace WhatsUrSay.Controllers

{

public class LoginController : ApiController

{

//Purpose: To proces the login requests of clients into our application.

//Input: 'userInfo' object of type 'UserInput.cs'

//Output: a boolean variable representing the status of the login request.

public bool Login(UserInput userInfo)

{

ILoginRepository objLoginRepository = new LoginRepository();

return objLoginRepository.Login(userInfo.uName, userInfo.uPassword);

}

}

}

**Angular JS Controllers**

**Home Controller**

/\*

Component : An AngularJS controller that invokes the methods defined in 'HomeController.cs' to serve the client requests and the Home Page.

Author: Abhinav Bhandaram

Use of the component in system design: Serves the Login requests of the clients

Written and revised: 11/5/2016

Reason for component existence: To serve Home page and client side functions related to Home.html view.

\*/

(function () {

'use strict';

//Reigstering the Controller with the Angular JS module defined for our application.

angular

.module('app')

.controller('HomeController', HomeController);

//Injecting the Angular JS Scope to be used for model binding between the view and the controller.

HomeController.$inject = ['$scope'];

function HomeController($scope) {

$scope.title = 'Home';

activate();

function activate() {

}

}

})();

**Login Controller**

/\*

Component : An AngularJS controller that invokes the methods defined in 'LoginController.cs' to serve the client login requests and the Login Page.

Author: Abhinav Bhandaram

Use of the component in system design: Serves the Login requests of the clients

Written and revised: 11/5/2016

Reason for component existence: To serve Login page and client side functions related to Login.html view.

\*/

(function () {

'use strict';

//Reigstering the Controller with the Angular JS module defined for our application.

angular

.module('app')

.controller('LoginController', LoginController);

//Injecting the Angular JS Scope and required modules to be used for model binding between the view and the controller, making http requests etc.

LoginController.$inject = ['$scope', '$http','$location','$mdDialog'];

function LoginController($scope, $http,$location,$mdDialog) {

$scope.title = 'Login';

$scope.userRoles = [{ id: 0, name: "Basic User" }, { id: 1, name: "Group Leader" }, { id: 3, name: "Admin" }];

$scope.userName = "";

$scope.password = "";

$scope.LoginStatus = false;

activate();

function activate() {

}

//Purpose: To proces the login requests of clients into our application.

//Input: $event to prevent the default JS behaviour of the link button.

//Output: An alert showing the login status of the user.

$scope.Login = function ($event) {

$event.preventDefault();

var userInfo = { uName: $scope.userName, uPassword: $scope.password }

$scope.LoginStatus = $http.post('api/Login/Login', userInfo).then(function success(response) {

var alert\_text = "";

if (response.data) {

alert\_text = "Login Successfull.";

} else {

alert\_text = "Please enter valid username and password.";

}

$mdDialog.show(

$mdDialog.alert()

.parent(angular.element(document.querySelector('#popupContainer')))

.clickOutsideToClose(true)

.title('Login')

.textContent(alert\_text)

.ariaLabel('alert')

.ok('Ok')

);

}, function error(response) {

$location.path('/Error');

});

}

}

})();

# Repositories

Please find below the list of Repositories and the details with respect to every repository used in our code.

**AnswerRepository**

/\*

Component : A class that does CRUD operations on the 'Answer' table of our project's database (DSE)

Author: Nikhitha Kaza

Use of the component in system design: Used for performing CRUD operations on the 'Answer' table

Written and revised: 11/5/2016

Reason for component existence: To perform CRUD operations on the 'Answer' table

Component usage of data structures, algorithms and control(if any):

The component contains the below methods:

GetAll(), Get(int id), Add(Answer Ans), Update(Answer Ans), Delete(int id)

These methods are invoked by 'AnswerController', a Web Api controller that serves the clients' requests

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

//includes all the models from the WhatsUrSay

using WhatsUrSay.Models;

namespace WhatsUrSay

{

public class AnswerResp: IAnswer

{

DSEEntities Db2 = new DSEEntities();

//Purpose: Gets all the records of type 'answer' from the 'Answer' table

//Input: None

//Output: A list of answer records from the 'Answer' table

public IEnumerable<Answer> GetAll()

{

// TO DO : Code to get the list of all the records in database

return Db2.Answers;

}

//Purpose: Gets a record from the 'Answer' table whose row id is 'id'

//Input: 'id' of the required record

//Output: a record from 'Answer' table whose key is 'id'

public Answer Get(int id)

{

// TO DO : Code to find a record in database

return Db2.Answers.Find(id);

}

//Purpose: Adds an object 'answer' in the 'Answer' table

//Input: 'answer' object of type 'Answer.cs'

//Output: Returns the object 'answer' upon its successful addition in the table

public Answer Add(Answer Ans)

{

if (Ans == null)

{

throw new ArgumentNullException("Answer");

}

// TO DO : Code to save record into database

Db2.Answers.Add(Ans);

Db2.SaveChanges();

return Ans;

}

}

}

**GroupsRepository**

/\* File Name :GroupsRepository.cs

\* Created By: Raj

\* This File resides in Business Layer

\*

\* Change History

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* PR Date Author Description

\*\* -- -------- ------- ------------------------------------

\*\* 1 11/6/2016 Raj Created GroupsRepository class and incuded methods to modify group data

\* 2 11/15/2016 Raj implemeted IGroupsRepository methods

\* This file performs CRUD operations on Group Table

\*

\*/

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Linq;

using System.Web;

using WhatsUrSay.Interfaces;

using WhatsUrSay.Models;

namespace WhatsUrSay.Repositories

{

public class GroupsRepository:IGroupsRepository

{

private DSEEntities db = new DSEEntities();

/// <summary>

/// This method returns all groups

/// </summary>

/// <returns></returns>

public IQueryable<Group> GetGroups()

{

return db.Groups;

}

/// <summary>

/// This method returns specific group details

/// </summary>

/// <param name="groupId"></param>

/// <returns></returns>

public GroupDetails GetGroup(int groupId)

{

Group group = db.Groups.Find(groupId);

if (group == null)

{

throw new ArgumentNullException("Group not found");

}

GroupDetails groupDetails;

try

{

groupDetails = new GroupDetails();

groupDetails.group = group;

groupDetails.UserList = new List<UserDetails>();

string[] users = group.user\_ids.Split(',');

foreach (var userId in users)

{

User user = db.Users.Find(Int32.Parse(userId));

groupDetails.UserList.Add(new UserDetails() { Id = user.id, Name = user.name });

}

}

catch (Exception)

{

throw;

}

return groupDetails;

}

/// <summary>

/// This method updates group details

/// </summary>

/// <param name="group"></param>

/// <returns>true or false </returns>

public bool UpdatGroup(Group group)

{

db.Entry(group).State = EntityState.Modified;

try

{

db.SaveChanges();

}

catch (DbUpdateConcurrencyException)

{

return false;

}

return true;

}

/// <summary>

/// This method returns created group details

/// </summary>

/// <param name="group"></param>

/// <returns> created group object</returns>

public Group CreateGroup(Group group)

{

db.Groups.Add(group);

try

{

db.SaveChanges();

}

catch (DbUpdateException)

{

if (GroupExists(group.id))

{

throw new Exception("Conflit");

}

else

{

throw;

}

}

return group;

}

/// <summary>

/// Deletes group

/// </summary>

/// <param name="id"></param>

/// <returns> deteleted group object</returns>

public Group DeleteGroup(int id)

{

Group group = db.Groups.Find(id);

if (group == null)

{

throw new ArgumentNullException("Group not found to delete");

}

db.Groups.Remove(group);

db.SaveChanges();

return group;

}

/// <summary>

/// Finds if group exists

/// </summary>

/// <param name="id"></param>

/// <returns> true or false</returns>

private bool GroupExists(int id)

{

return db.Groups.Count(e => e.id == id) > 0;

}

}

}

**LoginRepository**

/\*

Component : A class that does CRUD operations on the 'User' table of our project's database (DSE)

Author: Abhinav Bhandaram

Use of the component in system design: Used for performing CRUD operations on the 'User' table

Written and revised: 11/5/2016

Reason for component existence: To perform CRUD operations on the 'User' table

Component usage of data structures, algorithms and control(if any):

Uses Entity framework class 'User.cs' to do CRUD operations on 'User' table

The component contains the below methods:

'Login(uName, uPassword)'

These methods are invoked by 'UserController', a Web Api controller that serves the clients' requests

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

//Includes all the Models from the WhatsUrSay

using WhatsUrSay.Models;

using WhatsUrSay.Interfaces;

namespace WhatsUrSay.Repositories

{

public class LoginRepository:ILoginRepository

{

//Purpose: Finds the row whose key is uName, uPassword

//Input: Strings uName, uPassword for the Login method

//Output: Returns 'true' upon successfully finding of record in the table

// In case of 'null' input, the method throws 'ArgumentNullException'

public bool Login(string uName, string uPassword)

{

UserRepository objUserRepository = new UserRepository();

User user = objUserRepository.Find(uName);

if (user == null)

{

return false;

}

else

{

if(String.Equals(user.pwd, uPassword))

{

return true;

}

else

{

return false;

}

}

}

}

}

**PollRepository**

PollRepository.cs:

/\*

Component : A class that does 'insert' and 'read' operations on the 'Activity' table of our project's database (DSE)

Author: Sreedevi Koppula

Use of the component in system design: Used for performing 'insert' and 'read' operations on the 'Activity' table

Written and revised: 11/5/2016

Reason for component existence: To perform 'insert' and 'read' operations on the 'Activity' table

Component usage of data structures, algorithms and control(if any):

Uses Entity framework class 'Activity.cs' to do 'insert' and 'read' operations on 'Activity' table

The component contains the below methods:

'GetAll()', 'Get(int id)', 'Add(Activity activity)'

These methods are invoked by 'PollController', a Web Api controller that serves the clients' requests

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using WhatsUrSay.Models;

namespace WhatsUrSay.Repositories

{

public class PollRepository

{

private DSEEntities db = new DSEEntities();

//Purpose: Gets all the records of type 'poll' from the 'Activity' table

//Input: None

//Output: A list of poll records from the 'Activity' table

// If any exception occurs, the exception message is printed and the exception is thrown

public IEnumerable<Activity> GetAll()

{

try

{

return db.Activities.ToList();

}

catch (Exception e)

{

Console.WriteLine("Error occured: " + e.Message);

throw e;

}

}

//Purpose: Gets a record from the 'Activity' table whose row id is 'id'

//Input: 'id' of the required record

//Output: a record from 'Activity' table whose key is 'id'

// If any exception occurs, the exception message is printed and the exception is thrown

public Activity Get(int id)

{

try

{

return db.Activities.Find(id);

}

catch (Exception e)

{

Console.WriteLine("Error occured: " + e.Message);

throw e;

}

}

//Purpose: Adds an object 'activity' in the 'Activity' table

//Input: 'activity' object of type 'Activity.cs'

//Output: Returns the object 'activity' upon its successful addition in the table

// If any exception occurs, the exception message is printed and the exception is thrown

public Activity Add(Activity activity)

{

if (activity == null)

throw new ArgumentNullException("activity");

try

{

db.Activities.Add(activity);

db.SaveChanges();

return activity;

}

catch (Exception e)

{

Console.WriteLine("Error occured: " + e.Message);

throw e;

}

}

}

}

**QuestionRepository**

/\*

Component : A class that does 'insert' operation on the 'Question' table of our project's database (DSE)

Author: Sreedevi Koppula

Use of the component in system design: Used for performing 'insert' operation on the 'Question' table

Written and revised: 11/5/2016

Reason for component existence: To perform 'insert' operation on the 'Question' table

Component usage of data structures, algorithms and control(if any):

Uses Entity framework class 'Question.cs' to do 'insert' operation on 'Question' table

The component contains the below method:

'Add(Question question)'

These methods are invoked by 'QuestionController', a Web Api controller that serve the clients' requests

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using WhatsUrSay.Models;

namespace WhatsUrSay.Repositories

{

public class QuestionRepository

{

private DSEEntities db = new DSEEntities();

//Purpose: Adds an object 'question' in the 'Question' table

//Input: 'question' object of type 'Question.cs'

//Output: Returns the object 'question' upon its successful addition in the table

// If any exception occurs, the exception message is printed and the exception is thrown

public Question Add(Question question)

{

if (question == null)

throw new ArgumentNullException("question");

try

{

db.Questions.Add(question);

db.SaveChanges();

return question;

}

catch (Exception e)

{

Console.WriteLine("Error occured: " + e.Message);

throw e;

}

}

}

}

**SurveyRepository**

/\*

Component : A class that does CRUD operations on the 'Activity' table of our project's database (DSE)

Author: Nikhitha Kaza

Use of the component in system design: Used for performing CRUD operations on the 'Activity' table

Written and revised: 11/5/2016

Reason for component existence: To perform CRUD operations on the 'Activity' table

Component usage of data structures, algorithms and control(if any):

Uses Entity framework class 'Activity.cs' to do CRUD operations on 'Activity' table

The component contains the below methods:

'GetAll()', 'Get(int id)', 'Add(Activity activity)', 'Update(Activity activity)', 'Delete(int id)'

These methods are invoked by 'SurveyController', a Web Api controller that serves the clients' requests

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using WhatsUrSay.Models;

namespace WhatsUrSay

{

public class SurveyResp : ISurvey

{

DSEEntities Db1 = new DSEEntities();

//Purpose: Gets all the records of type 'poll' from the 'Activity' table

//Input: None

//Output: A list of poll records from the 'Activity' table

public IEnumerable<Activity> GetAll()

{

// TO DO : Code to get the list of all the records in database

return Db1.Activities;

}

//Purpose: Gets a record from the 'Activity' table whose row id is 'id'

//Input: 'id' of the required record

//Output: a record from 'Activity' table whose key is 'id'

public Activity Get(int id)

{

// TO DO : Code to find a record in database

return Db1.Activities.Find(id);

}

//Purpose: Adds an object 'activity' in the 'Activity' table

//Input: 'activity' object of type 'Activity.cs'

//Output: Returns the object 'activity' upon its successful addition in the table

public Activity Add(Activity Act)

{

if (Act == null)

{

throw new ArgumentNullException("Activity");

}

// TO DO : Code to save record into database

Db1.Activities.Add(Act);

Db1.SaveChanges();

return Act;

}

//Purpose: Updates the row whose key is updated.id with the details of object 'updated' in the 'Activity' table

//Input: 'updated' object of type 'Activity.cs'

//Output: Returns 'true' upon the successful updation of record in the table

// In case of 'null' input, the method throws 'ArgumentNullException'

public bool Update(Activity Act)

{

if (Act == null)

{

throw new ArgumentNullException("Activity");

}

// TO DO : Code to update record into database

var Survey1 = Db1.Activities.Single(a => a.id == Act.id);

Survey1.heading = Act.heading;

Survey1.description= Act.description;

Survey1.type= Act.type;

Survey1.category = Act.category;

Survey1.group\_ids = Act.group\_ids;

Db1.SaveChanges();

return true;

}

//Purpose: Deletes the row whose key 'id' in the 'Activity' table

//Input: 'id' of the activity that needs to be deleted

//Output: Returns 'true' upon the successful deletion of record from the table

public bool Delete(int id)

{

// TO DO : Code to remove the records from database

Activity products = Db1.Activities.Find(id);

Db1.Activities.Remove(products);

Db1.SaveChanges();

return true;

}

}

}

**UserRepository**

/\*

Component : A class that does CRUD operations on the 'User' table of our project's database (DSE)

Author: Abhinav Bhandaram

Use of the component in system design: Used for performing CRUD operations on the 'User' table

Written and revised: 11/5/2016

Reason for component existence: To perform CRUD operations on the 'User' table

Component usage of data structures, algorithms and control(if any):

Uses Entity framework class 'User.cs' to do CRUD operations on 'User' table

The component contains the below methods:

Add(User)

These methods are invoked by 'UserController', a Web Api controller that serves the clients' requests

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using WhatsUrSay.Interfaces;

using WhatsUrSay.Models;

namespace WhatsUrSay.Repositories

{

public class UserRepository:IUserRepository

{

DSEEntities objEntities = new DSEEntities();

//Purpose: Adds the user

//Input: 'user' for the Add method

//Output:

// In case of 'null' input, the method throws 'ArgumentNullException'

// ELSE adds user and saves the changes and returns the output as the user

public User Add(User user)

{

if (user == null)

{

throw new ArgumentNullException("user");

}

objEntities.Users.Add(user);

objEntities.SaveChanges();

return user;

}

//Purpose: Finds the row whose key is uName

//Input: Strings uName for the Find method

//Output: Returns objEntities.Users upon finding of record in the table

// In case of 'null' input, the method throws 'ArgumentNullException'

public User Find(string uName)

{

if (String.IsNullOrEmpty(uName))

{

throw new ArgumentNullException("user name");

}

return objEntities.Users.Find(uName);

}

}

}

# Views

There are 2 types of views that we have used in our project, they are:

1. CSHTML Views
2. HTML Views

**CSHTML View**

**Index**

<!--

The file is named: Error.

The file is of .cshtml extention.

The file is created by Abhinav.

This is basically .cshtml file named Index which provides the basic layout and acts as master page for the application.-->

<!--Inlcides all the required scripts and css files for the application.-->

<!DOCTYPE html>

@{

ViewBag.Title = "Home";

Layout = null;

}

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

<link href="~/Content/css/foundation.css" rel="stylesheet" />

<link href="~/Content/Site.css" rel="stylesheet" />

<link rel="stylesheet" href="http://ajax.googleapis.com/ajax/libs/angular\_material/1.1.0/angular-material.min.css">

<link href="~/Content/css/foundation-icons/foundation-icons.css" rel="stylesheet" />

</head>

<body ng-app="app" ng-controller="HomeController">

<div class="top-bar">

<div class="row">

<div class="top-bar-left">

<ul class="menu">

<li><a href="#">Create <i class="fi-plus"></i></a></li>

<li><a href="#">Participate</a></li>

</ul>

</div>

<div class="top-bar-title" style="margin-left:20%">

<ul class="menu">

<li class="menu-text">

Whats Ur Say !!

</li>

</ul>

</div>

<div class="top-bar-right">

<ul class="menu">

<li><a href="#/join">Join</a></li>

<li><a href="#/login">Log In</a></li>

</ul>

</div>

</div>

</div>

<div id="popupContainer"></div>

<div class="row body-content" ng-view></div>

<footer class="secondary callout">

<div class="row">

<div class="small-3 large-6 columns">&copy;DSE's</div>

<div class="small-3 large-6 columns">

<ul class="menu">

<li>

<a href="#">About Us</a>

</li>

<li>

<a href="#">Team</a>

</li>

</ul>

</div>

</div>

</footer>

<script src="~/scripts/jquery-1.10.2.min.js"></script>

<script src="~/scripts/js/foundation.min.js"></script>

<script src="~/scripts/js/what-input.js"></script>

<script src="~/scripts/angular.min.js"></script>

<script src="~/scripts/angular-route.min.js"></script>

<script src="~/scripts/angular-animate.min.js"></script>

<script src="~/scripts/angular-aria.min.js"></script>

<script src="~/scripts/angular-messages.min.js"></script>

<script src="http://ajax.googleapis.com/ajax/libs/angular\_material/1.1.0/angular-material.min.js"></script>

<script src="~/scripts/app/app.js"></script>

<script src="~/scripts/app/HomeController.js"></script>

<script src="~/scripts/app/LoginController.js"></script>

<script>$(document).ready(function () { $(document).foundation(); });</script>

</body>

</html>

**HTML View**

**Error**

<!--

The file is named: Error.

The file is of .html extention.

The file is created by Abhinav.

This is basically .html file named Error which provides the basic Error page when there is an error that occurrs in the application".

-->

<!--For Registration the fiels provided are email id, password, Confirm Password, Name, Role, Reason -->

<div>

<span style="color:red">An Error Occurred while procesing the request. Please Try Again Later or Contact your System Administrator</span>

</div>

**Home**

<!--

The file is named: Home.

The file is of .html extention.

The file is created by Abhinav.

This is basically .html file named Home which shows the message on the home page.

-->

<div>

<!-- This is the Div which deals with the image "Business-poll-survey2-2401" on the home page -->

<div class="callout" style="padding:0px">

<img src="../../Content/img/business-poll-survey2-2401.jpg" style="width:100%;max-height:300px;" />

</div>

<!--This is the Div dealing with the meaasge on the home page-->

<div class="callout">

<h5>For All your Polling And Survey Needs.</h5>

<p>

WhatsUrSay!!, As the name suggests is an application were you can express your views through the means of Polls and Surveys.

</p>

<p>

Have Something to Ask, Create a Poll or a Survey.

</p>

</div>

<div>

<!-- This is the paragraph tag which provides the Register and Login links -->

<p>

New to WhatsUrSay!! Register today <a href="#/join">Register</a>,<br /> Already a Member?? <a href="#/login">Sign In.</a>

</p>

</div>

</div>

**Login**

<!--

The file is named: Login.

The file is of .html extention.

The file is created by Abhinav.

This is basically .html file named Login which provides the basic details needed for the user to "login".

-->

<!--The login has the fields email id, password, Login and Cancel buttons-->

<form>

<div class="primary callout">

<div class="row">

<div class="large-12 columns">

<label>

Email Id

<input type="text" placeholder="Email Id" />

</label>

</div>

</div>

<div class="row">

<div class="large-12 columns">

<label>

Password

<input type="password" placeholder="Password" />

</label>

</div>

</div>

<div class="row">

<div class="large-3 columns">

<a href="#" class="button round">Login</a>

<a href="#" class="button radius round">Cancel</a>

</div>

</div>

</div>

</form>

**Register**

<!--

The file is named: Register.

The file is of .html extention.

The file is created by Abhinav.

This is basically .html file named Register which provides the basic details needed for the new user to "Register".

-->

<!--For Registration the fiels provided are email id, password, Confirm Password, Name, Role, Reason -->

<form data-abide novalidate>

<div class="primary callout">

<div class="row">

<div class="large-12 columns">

<label for="emailId" class="error" role="alert">

Email Id <small>required</small>

<input type="text" placeholder="Email Id" name="emailId" id="emailId">

</label>

</div>

</div>

<div class="row">

<div class="large-12 columns">

<label for="password">

Password <small>required</small>

<input type="password" id="password" placeholder="Password" name="password" />

</label>

</div>

</div>

<div class="row">

<div class="large-12 columns">

<label for="confirmPassword" class="">

Confirm Password <small>required</small>

<input type="password" id="confirmPassword" placeholder="Confirm Password" name="confirmPassword">

</label>

</div>

</div>

<div class="row">

<div class="large-12 columns">

<label>

Name <small>required</small>

<input type="text" id="name" placeholder="Name" name="name">

</label>

</div>

</div>

<div class="row">

<div class="large-12 columns">

<label>

Role <small>required</small>

<select id="role" name="role">

<option ng-repeat="userRole **in** userRoles" value="userRole.id">**{{**userRole.name**}}**</option>

</select>

</label>

</div>

</div>

<div class="row">

<div class="large-12 columns">

<label>

Reason: <small>required</small>

<textarea></textarea>

</label>

</div>

</div>

<div class="row">

<div class="large-3 columns">

<button type="button" class="button round">Register</button>

<a href="#/home" class="button round">Cancel</a>

</div>

</div>

</div>

</form>

<script>$(document).foundation();</script>

**CreatePoll**

<!--

Component : UI for creating a poll by the authorized users

Author: Sreedevi Koppula

Use of the component in system design: Provides UI that is accessed by the users to create polls

Written and revised: 11/11/2016

Reason for component existence: Provides an interface for the user to create a poll in the system

Component usage of data structures, algorithms and control (if any):

The component displays the appropriate fields to the user to create a poll. The fields include:

Poll Heading, Poll Question, Poll Type (Public/private), Group Names (if poll type is private), Buttons 'Create Poll' and 'Cancel'

The authorized user needs to fill all the fields and click on 'Create Poll' button to create the poll in the system

The user can click on 'Cancel' button to cancel the poll creation

-->

<form>

<div class="primary callout">

<div class="row">

<div class="large-12 columns">

<label>

Title <small>required</small>

<input type="text" placeholder="Title" />

</label>

</div>

</div>

<div class="row">

<div class="large-12 columns">

<label>

Description <small>required</small>

<input type="text" placeholder="Enter the poll description" />

</label>

</div>

</div>

<div ng-app="myApp">

<div class="row">

<div class="large-12 columns">

<label>

Poll Type <small>required</small>

<select id="pollType" name="pollType" ng-model="pollType">

<option value="">--Select Poll Type--</option>

<option value="public">Public</option>

<option value="private">Private</option>

</select>

</label>

</div>

</div>

<div class="row" ng-show="(pollType=='private')">

<div class="large-12 columns">

<label>

Groups' Names <small>required</small>

<input type="text" placeholder="Enter the names of groups" />

</label>

</div>

</div>

</div>

<div class="row">

<div class="large-3 columns">

<a href="#" class="button round">Create Poll</a>

<a href="#" class="button radius round">Cancel</a>

</div>

</div>

</div>

</form>

# Scripts

Please find below the script files and details corresponding to the script files below:

**App.JS**

/\*

Component : An AngularJS module definition that is used to define the angular js application used in the application and the different routes

identified in the application.

Author: Abhinav Bhandaram

Use of the component in system design: Angular JS application definition and routes defined in the application.

Written and revised: 11/5/2016

Reason for component existence: Angular JS application definition..

\*/

(function () {

'use strict';

//Defining the agularjs application 'app' and the required dependencies for the application.

var app = angular.module('app', ['ngRoute', 'ngMaterial']);

//Defining the routes identified in the application.

app.config(['$routeProvider', function ($routeProvider) {

$routeProvider.when('/login', {

templateUrl: 'features/Login/Login.html',

controller:'LoginController'

}).when('/join', {

templateUrl:'features/Login/Register.html'

}).when('/error', {

templateUrl:'features/Error/Error.html',

})

.otherwise({

templateUrl: 'features/Home/Home.html',

controller:'HomeController'

})

}]);

})();

# DataBase Scripts

Please find below the database script file created for our project and the details corresponding to the database schema.

**WhatsUrSayTables.sql**

/\*

Component : Database script that is used for creating our project's database and necessary tables

Author: Sreedevi Koppula

Use of the component in system design: Specifies the structure of the database that stores our project's data

Written and revised: 11/9/2016

Reason for component existence: To create project's database and necessary tables

Component usage of data structures, algorithms and control(if any):

aContains the SQL queries for performing the below actions:

Create Database - 'DSE'

Create tables - User, Group, User\_Group, User\_Request, Activity, Question, Answer

\*/

/\* Creating our project's database. Name of the database - DSE \*/

CREATE DATABASE [DSE]

GO

/\*Uses the database 'DSE' to perform the below SQL operations\*/

USE [DSE]

GO

/\* Object: Table [dbo].[User] \*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[User](

[id] [int] NOT NULL,

[name] [ntext] NOT NULL,

[role] [ntext] NULL,

[emailId] [ntext] NOT NULL,

[pwd] [ntext] NOT NULL,

[status] [ntext] NOT NULL,

CONSTRAINT [PK\_User] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\* Object: Table [dbo].[Group] \*/

CREATE TABLE [dbo].[Group](

[id] [int] NOT NULL,

[name] [ntext] NOT NULL,

[user\_ids] [ntext] NULL,

[createdby] [int] NOT NULL,

CONSTRAINT [PK\_Group] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Group] WITH CHECK ADD CONSTRAINT [FK\_Group\_User] FOREIGN KEY([createdby])

REFERENCES [dbo].[User] ([id])

GO

ALTER TABLE [dbo].[Group] CHECK CONSTRAINT [FK\_Group\_User]

GO

/\* Object: Table [dbo].[User\_Group] \*/

CREATE TABLE [dbo].[User\_Group](

[id] [int] NOT NULL,

[user\_id] [int] NOT NULL,

[group\_ids] [ntext] NULL,

CONSTRAINT [PK\_User\_Group] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[User\_Group] WITH CHECK ADD CONSTRAINT [FK\_User\_Group\_User] FOREIGN KEY([user\_id])

REFERENCES [dbo].[User] ([id])

GO

ALTER TABLE [dbo].[User\_Group] CHECK CONSTRAINT [FK\_User\_Group\_User]

GO

/\* Object: Table [dbo].[User\_Request] \*/

CREATE TABLE [dbo].[User\_Request](

[id] [int] NOT NULL,

[user\_id] [int] NOT NULL,

[description] [ntext] NULL,

[status] [int] NOT NULL,

CONSTRAINT [PK\_User\_Request] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[User\_Request] WITH CHECK ADD CONSTRAINT [FK\_User\_Request\_User] FOREIGN KEY([user\_id])

REFERENCES [dbo].[User] ([id])

GO

ALTER TABLE [dbo].[User\_Request] CHECK CONSTRAINT [FK\_User\_Request\_User]

GO

/\* Object: Table [dbo].[Activity] \*/

CREATE TABLE [dbo].[Activity](

[id] [int] NOT NULL,

[heading] [ntext] NOT NULL,

[description] [ntext] NULL,

[type] [ntext] NOT NULL,

[category] [ntext] NOT NULL,

[group\_ids] [ntext] NULL,

[createdby] [int] NOT NULL,

CONSTRAINT [PK\_Activity] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Activity] WITH CHECK ADD CONSTRAINT [FK\_Activity\_User] FOREIGN KEY([createdby])

REFERENCES [dbo].[User] ([id])

GO

ALTER TABLE [dbo].[Activity] CHECK CONSTRAINT [FK\_Activity\_User]

GO

/\* Object: Table [dbo].[Question] \*/

CREATE TABLE [dbo].[Question](

[id] [int] NOT NULL,

[description] [ntext] NOT NULL,

[activity\_id] [int] NOT NULL,

CONSTRAINT [PK\_Question] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Question] WITH CHECK ADD CONSTRAINT [FK\_Question\_Activity] FOREIGN KEY([activity\_id])

REFERENCES [dbo].[Activity] ([id])

GO

ALTER TABLE [dbo].[Question] CHECK CONSTRAINT [FK\_Question\_Activity]

GO

/\* Object: Table [dbo].[Answer] \*/

CREATE TABLE [dbo].[Answer](

[id] [int] NOT NULL,

[description] [ntext] NOT NULL,

[question\_id] [int] NOT NULL,

[activity\_id] [int] NOT NULL,

[count] [int] NULL,

CONSTRAINT [PK\_Answer] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Answer] WITH CHECK ADD CONSTRAINT [FK\_Answer\_Activity] FOREIGN KEY([activity\_id])

REFERENCES [dbo].[Activity] ([id])

GO

ALTER TABLE [dbo].[Answer] CHECK CONSTRAINT [FK\_Answer\_Activity]

GO

ALTER TABLE [dbo].[Answer] WITH CHECK ADD CONSTRAINT [FK\_Answer\_Question] FOREIGN KEY([question\_id])

REFERENCES [dbo].[Question] ([id])

GO

ALTER TABLE [dbo].[Answer] CHECK CONSTRAINT [FK\_Answer\_Question]

GO