using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using MongoDB.Driver;

namespace ShareHubServer.Controllers.Api {

[Route("api/SearchGateway")]

public class SearchGatewayController : Controller {

#region search users

internal class UsersResult : ResultBase {

public class QueryReturn {

public string username;

public string[] commonCommunities;

}

public QueryReturn[] queryReturn = null;

}

[HttpPost("Users")]

public ActionResult UsersResponder(string userKey, string query, int skip, int limit) {

if (!Utilities.Sanitized(userKey, query, skip, limit)) {

return StatusCode(405, "one or more arguments were null");

}

return Json(Users(userKey, query, skip, limit));

}

internal UsersResult Users(string userKey, string query, int skip, int limit) {

IMongoCollection<DBEntry.Community> communitiesCollection = Program.Database.GetCollection<DBEntry.Community>("communities");

IMongoCollection<DBEntry.User> usersCollection = Program.Database.GetCollection<DBEntry.User>("users");

List<DBEntry.User> dbUsers = usersCollection.Find(x => x.Cookie == userKey).ToList();

(UserGatewayController.CheckUserResult userResult, DBEntry.User user) = UserGatewayController.CheckUser(userKey);

if (!userResult.success) {

return new UsersResult() {

queryReturn = null,

authorized = userResult.authorized,

message = userResult.message,

success = userResult.success,

type = userResult.type,

};

}

List<DBEntry.Community> communities = communitiesCollection.Find(x => x.Users.Contains(user.Username)).ToList();

List<DBEntry.User> commonUsers = usersCollection.

Find(x => x.Username.Contains(query) && x.Username != user.Username &&

communities.Any(y => y.Users.Contains(x.Username))).Limit(limit + skip).ToList();

List<DBEntry.User> uncommonUsers = usersCollection.

Find(x => x.Username.Contains(query) && x.Username != user.Username &&

commonUsers.Any(y => y.Username == x.Username)).Limit(commonUsers.Count - (limit + skip)).ToList();

List<UsersResult.QueryReturn> queryReturn = new List<UsersResult.QueryReturn>();

commonUsers.ForEach(x => {

List<string> commonCommunities = new List<string>();

communities.ForEach(y => {

if (y.Users.Contains(x.Username)) {

commonCommunities.Add(y.UniqueName);

}

});

queryReturn.Add(new UsersResult.QueryReturn() {

username = x.Username,

commonCommunities = commonCommunities.ToArray()

});

});

uncommonUsers.ForEach(x => {

queryReturn.Add(new UsersResult.QueryReturn() {

username = x.Username,

commonCommunities = null

});

});

return (new UsersResult() {

type = "successful",

authorized = true,

success = true,

message = "all good",

queryReturn = queryReturn.ToArray()

});

}

#endregion

}

}