**HopeLine.API.Hubs.ICall.cs**

namespace HopeLine.API.Hubs

{

public interface ICall

{

Task CallDisconnected(string roomId);

Task ConnectCall(string roomId);

Task RequestToVideoCall(string roomId);

}

}

**HopeLine.API.Hubs.IChat.cs**

namespace HopeLine.API.Hubs.v2

{

public interface IChat

{

Task AddMentor(string mentorId);

Task RequestToTalk(string userId);

Task RemoveUser(string userId, string roomId, bool isUser);

Task LoadMessage(string room);

Task SendMessage(string user, string message, string room);

Task AddTopics(string roomId, List<int> ids);

}

}

**HopeLine.Service.Interfaces**

namespace HopeLine.Service.Interfaces {

/// <summary>

///

/// </summary>

public interface ICommonResource {

IEnumerable<CommunityModel> GetCommunities ();

IEnumerable<ResourceModel> GetResources ();

IEnumerable<LanguageModel> GetLanguages ();

IEnumerable<TopicModel> GetTopics ();

bool AddResources (ResourceModel resource);

bool EditResource (ResourceModel resource);

bool AddCommunity (CommunityModel resource);

bool EditCommunity (CommunityModel resource);

bool AddTopics (TopicModel topic);

bool AddLanguage (LanguageModel language);

bool RemoveResource (int id);

bool RemoveCommunity (int id);

void SaveTopic ();

void SaveResource ();

void SaveCommunity ();

void SaveLanguage ();

Task SaveTopicAsync ();

Task SaveResourceAsync ();

Task SaveCommunityAsync ();

Task SaveLanguageAsync ();

}

}

**HopeLine.Service.Interfaces**

namespace HopeLine.Service.Interfaces {

/// <summary>

///

/// </summary>

public interface ICommunication {

//For Admin Only

IEnumerable<ConversationModel> GetConversations ();

//For Mentor Only

IEnumerable<ConversationModel> GetConversationsByMentorId (string mentorId);

//For Registered User Only

IEnumerable<ConversationModel> GetConversationsByUserId (string userId);

//For all kind of users

ConversationModel GetConversationById (int id);

ConversationModel GetConversationByPIN (string pin);

bool EditConversation (ConversationModel conversation);

string GenerateConnectionId ();

bool AddConversation (ConversationModel conversation);

}

}

**HopeLine.Service.Interfaces.IMessage**

namespace HopeLine.Service.Interfaces

{

public interface IMessage

{

void NewMessage(MessageModel model);

Task NewMentorAvailable(string mentorId, string connectionId);

Task RemoveMentor(string connectionId);

Task DeleteAllMessages(string roomId);

IEnumerable<MessageModel> GetAllMessages(string roomId);

IEnumerable<OnlineMentorModel> ListAvailableMentor();

Task SetMentorOnCall(string mentorId, string connectionId);

string GetRoomForUser(string userId, bool isGuest);

Task AndUsersToRoom(string mentorId, string guestId, string roomId);

}

}

**HopeLine.DataAccess.Interfaces.IRepository**

namespace HopeLine.DataAccess.Interfaces

{

//TODO : add implementation

public interface IRepository<T>

{

IEnumerable<T> GetAll(string include = null);

void Insert(T obj);

void Update(T obj);

T Get(object id);

void Delete(T obj);

void Remove(object id);

void Save();

Task SaveAsync();

}

}

**HopeLine.DataAccess.Interfaces.IServices**

interface IServices

{

//void SomeFunc();

}

**HopeLine.Security.Interfaces**

namespace HopeLine.Security.Interfaces

{

/// <summary>

///

/// </summary>

public interface ITokenService

{

object GenerateToken(string username, HopeLineUser user);

ClaimsPrincipal GetClaimsPrincipalFromExpiredToken(string expiredToken);

Task<object> SignInUser(string username, string password, bool isguest);

Task<object> RegisterUser(RegisterModel model);

}

}

**HopeLine.Service.Interfaces.IUserService**

namespace HopeLine.Service.Interfaces

{

public interface IUserService

{

IEnumerable<UserModel> GetAllUsers();

IEnumerable<UserModel> GetAllUsersByAccountType(string userType);

//Can be refactored

#region Users and Mentors

IEnumerable<ActivityModel> GetUserActivities(string userId);

IEnumerable<ActivityModel> GetMentorActivities(string mentorId);

IEnumerable<ConversationModel> GetUserConversations(string username);

IEnumerable<ConversationModel> GetMentorConversations(string mentorId);

IEnumerable<ScheduleModel> GetMentorSchedules(string mentorId);

IEnumerable<SpecializationModel> GetMentorSpecializations(string mentorId);

#endregion

bool UpdateUserProfile(UserModel model);

IEnumerable<string> ListMentorIdByAvailability(bool available);

}

}