Task Description:
You are tasked with developing a REST API for a simplified blogging platform using C# and ASP.NET Core. The API should include endpoints for managing users, posts, and comments. Additionally, you are required to establish a many-to-many relationship between posts and comments in the underlying SQL database.

Task Requirements:

1. **User Management API:**
○ Implement APIs for creating, retrieving, updating, and deleting users.
○ Include endpoints for user registration, login, profile updates, and user deletion.

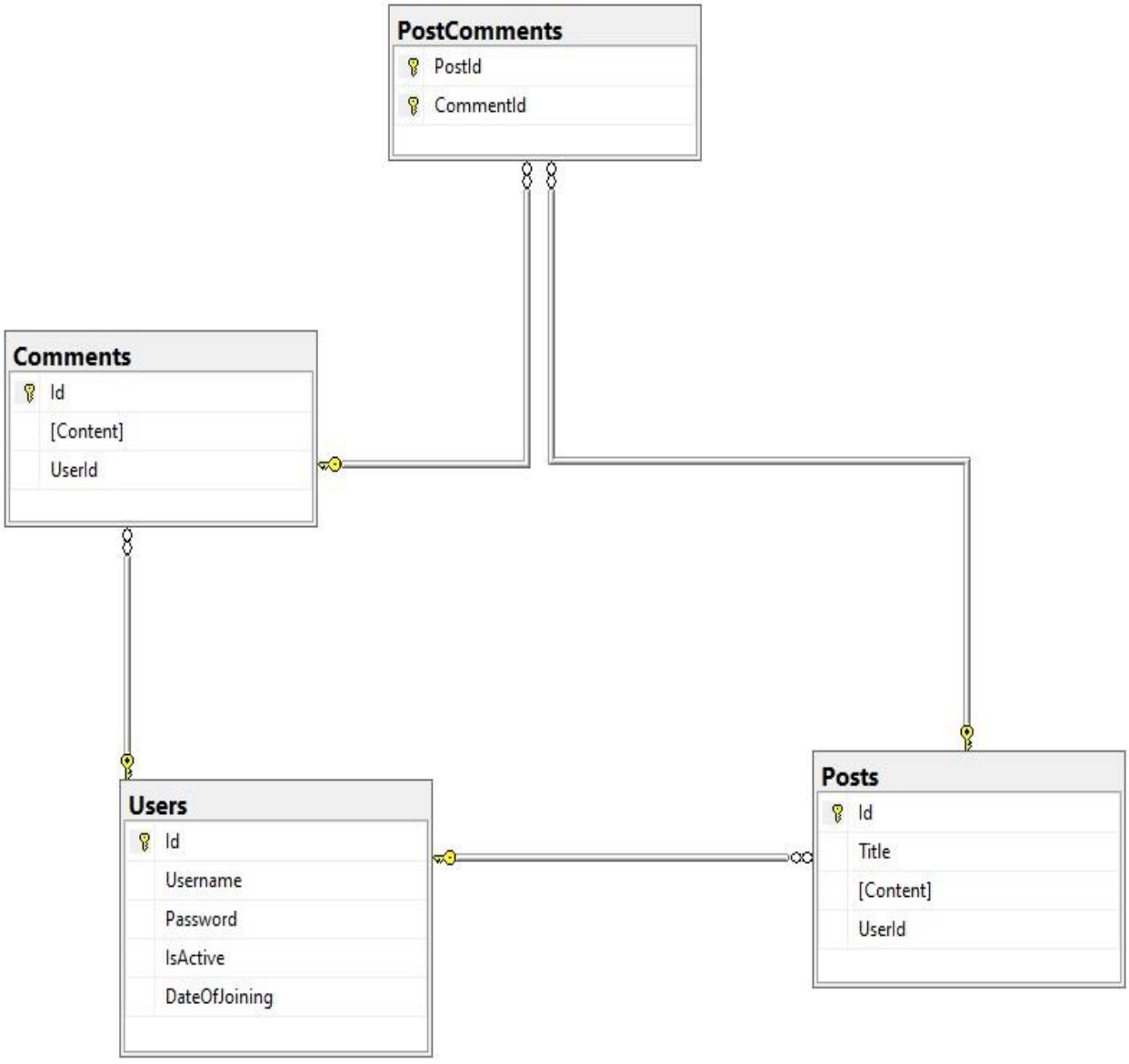
2. **Post Management API:**
○ Develop APIs for managing blog posts, including creating, retrieving, updating, and deleting posts.
○ Implement endpoints for listing all posts, fetching a single post by ID, and updating post content.

3. **Comment Management API:**
○ Create APIs for handling comments on blog posts.
○ Include endpoints for adding comments to posts, listing comments for a specific post, and deleting comments.

4. **Database Relationship:**
○ Establish a many-to-many relationship between posts and comments in the SQL database.
○ Ensure that posts can have multiple comments, and comments can be associated with multiple posts.

To create a simplified blogging platform using C# and ASP.NET Core, we will need to design and implement a REST API with endpoints for managing users, posts, and comments, including the necessary database relationships.

We need these tables : Users, Posts, Comments and PostComments. So this is the Diagram we need:



API Security: JWT

JsonwebToken has been used for securing API.

Controllers

Controllers in this API are divided in 2 controllers.

1. Users controllers
2. Posts controllers

1-Users controllers

Users Controller includes these actions which are all needing authorization except for register and get-token:

1. Register
2. Get-token (log in)
3. Update Users
4. Delete Users
5. Get All Users
6. Get User By ID

Register

Requires to set a username and password. The output will be added to the user table in database.

Get-token (log in)

Requires username and password from user table in database to get JWT for authorization to use other actions.

Update Users

Requires old username and password and new one of each for updating the user and the output will be a new user in database.

Delete Users

Requires UserId for deleting the User which is Authorized and the output will be the deletion of Authorized User in database.

NOTE: it is recommended to build an admin table and set a role for its members to be able to delete users! But in this API it is done as the task has asked.

Get All Users

Requires authorization to retrieve all users and the output will be a list of all users.

Get User By ID

Requires authorization and a userId for retrieving user from user table and the output will be a user from database.

2-Posts controllers

Posts Controllers includes these actions which are all needing authorization:

1. Submit Post
2. Update Post
3. Delete Post
4. Get All Posts
5. Get Post By ID
6. Submit Comment
7. Delete Comment
8. Get All Comments
9. Get Comment By ID

Submit Post

Requires Title and content and the output will be a record in Post table.

Update Post

Requires old Title and content and new one of each for updating post and the output will be a updated record in Post table.

Delete Post

Requires Post Id and the output will be deletion of the record in Post table.

Get All Posts

Requires authorization to retrieve all Posts and the output will be a list of all Posts.

Get Post By ID

Requires authorization and a PostId to retrieve the Post and the output will be retrieved post from post table.

Submit Comment

Requires Content and PostId and the output will be a record in comment table.

Delete Comment

Requires PostId and CommentId and the output will be deletion of the record in Comment table.

Get All Comments

Requires authorization to retrieve all Comments and the output will be a list of all Comments.

Get Comment By ID

Requires authorization and a CommentId to retrieve the Comment and the output will be retrieved Comment from Comment table.

Note:

We could use interfaces and services in a different layer in this API but the best architecture for saving time, energy and load is this.