Image Based Social Media Application

#### Design Document By: Athena, Edvin, Noris, Phillip, Yasir

# General Description of the application

One of the most popular aspects of the internet is social media. Imagine being able to message, to share your life, within seconds of posting something within an app for the whole world to see. Due to social media becoming so popular and having such a strong influence on people's daily lives, we decided to create our own interpretation of Instagram.

This app will be image-based: think Instagram, a pioneer of social media and the most popular social media app in the world. This app will have the limited abilities of Instagram. Planned functionalities for the application are listed below:

* Register / Login
* Follow / Unfollow other users
* Create / Delete posts
* Like / Unlike posts
* Comment / Uncomment posts

This is what we plan to do on the **second phase** of this project if we have enough time:

* Set user accounts private or public
* Explore page
* Notifications
* Message/Chat
* Stories
* Search bar

Technology Overview

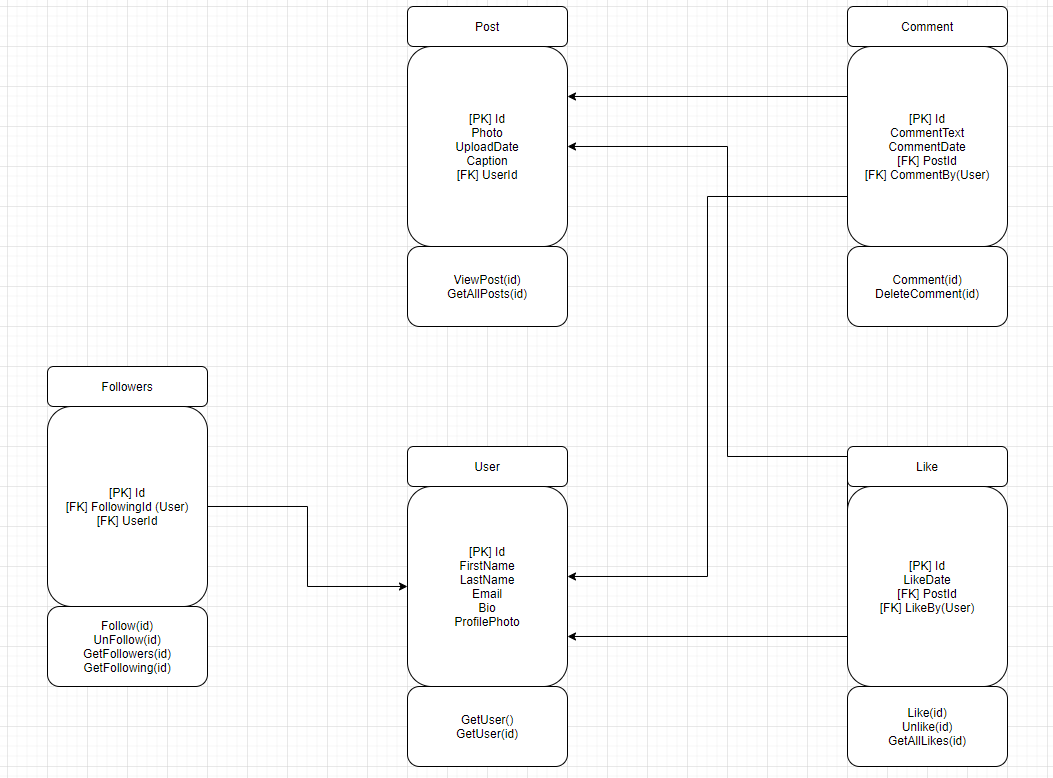
* C#.
* SQL.
* React.
* JWT (JSON Web Tokens).
* HTML5.
* .NET Core Web API.

# Models

* Users
* Posts
* Comments
* Likes
* Followers
* Register/Login (for authentication only)
* Response (for json response)

# Database Schema

\*Subject to change



# Unit Testing

Unit testing will be done in a BDD (Behavorial Driven Development) fashion.

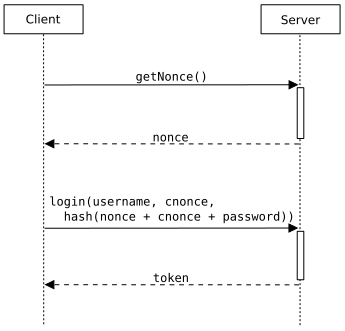
Front testing using either Jest or Mocha/Chai.

Backend will be using Moq, XUnit and InMemoryDatabase.

# Authentication & Authorization

Access to the functionality of the app will be behind authentication. Users will require an account to be able to view and post images.

The login screen will initiate a login authentication token (such as nonce but instead we will use JWT), JWTs are good for information exchange, these JWTs securely transmit information between client and server because these tokens can be signed which means you know the sender is real and not a fraud. This will prevent any outside scripts from having any external influence on the UX (effectively foiling any attempt at CSRF).



### Controllers

AuthenticationController

* RegisterUsers
  + For users who want to create a new account.
* SignInUsers
  + Gives the ability for people to login to their account from anywhere.

UsersController

* GetUserDetails(id) Profile? GET
  + This method will get details for a specific user to display on their profile
* EditUserDetails(id) PUT or PATCH (bio and username)
  + This method will ensure that the people who take 15 hours to decide on a bio will be able to edit their bio to their hearts content.
* DeleteUser (from user settings)
  + This method will delete the user from the database, in case a user wishes to scrub their account.

FollowsController

* AddNewFollow(~~userid~~, targetid) POST
  + When a user follows another user, this method facilitates the logic and communication between them.
* RemoveFollow(~~userid~~, targetid)
  + When a user unfollows another user, this method will help facilitate that logic.
* ~~GetFollowedUsers(userid)~~ GetFollowers(userid)
  + This method is to be able to see which user is following which user from an individual user's perspective. (Users who are following you)
* GetFollowing(userid)
  + Get users who you are following

PostsController

* AddPost(~~userid~~) POST (current user posts for him/herself)
  + When you want to post your selfie, this is the method to allow it.
* DeletePost(postid) DELETE
  + When you realize you look stupid, you can delete your post.
* EditPost(postid) PUT or PATCH (caption only)
  + Sometimes you want to edit your post, or make yourself look a little better. This method gives you that ability.
* GetPostsByUser(userid)
  + This method is so you can be able to make fun of how other people look in their posts. Or just admire their food/nature shots.
* GetAllPostsfromFollows(userid)
  + Whenever you feel like life is empty, you just login into the app and you see this screen, the screen of where the users that you follow, and you get to see what they’re posting.

CommentsController

* GetPostComments(postid)
  + Retrieves comments posted on a post by a GET request.
* AddComment(postid)
  + Adds a comment on a post.
* DeleteComment(commentid)
  + Deletes a comment.

LikesController

* GetLikes(postid)
  + Retrieves likes on a post.
* AddLike(postid, ~~userid~~)
  + When you like someone's post.
* RemoveLike (postid,~~userid~~)
  + When you realize that you dislike the post, you can show your dislike by removing your like.

# Logging and General Exception Handling

Use:

* Make sure not to give away too much information when sending error info to user, could be a security risk.
* Default error page, Database errors.
* Text file to log user log in and user API calls.
* Text file to log API Errors and exceptions.

# Configurable Parameters

* Connection string.
* Web API URL and Routing
* "JWT": {

"ValidAudience": "<http://localhost:44345>",

"ValidIssuer": "<http://localhost:44391>",

"Secret": "ByYM000OLlMQG6VVVp1OH7Xzyr7gHuw1qvUC5dcGt3SNM"

}

# Routes

https://localhost:5001/register

https://localhost:5001/login

https://localhost:5001/edituserprofilephoto

https://localhost:5001/edituser

https://localhost:5001/getcurrentuserdetails

https://localhost:5001/getuserdetails/{username}

https://localhost:5001/deleteuser

https://localhost:5001/addpost

https://localhost:5001/deletepost/{postId}

https://localhost:5001/editpost/{postId}

https://localhost:5001/getcurrentuserposts

https://localhost:5001/getuserposts/{username}

https://localhost:5001/addcomment/{postId}

https://localhost:5001/deletecomment/{postId}

https://localhost:5001/getPostComments/{postId}

https://localhost:5001/addNewFollow/{username}

https://localhost:5001/removeFollow/{username}

https://localhost:5001/getFollowing/{username}

https://localhost:5001/getFollowers/{username}

https://localhost:5001/addLike/{postId}

https://localhost:5001/removeLike/{postId}

https://localhost:5001/getLikes/{postId}