**Documentation of Social Network**

**Purpose of the application:**

- What is the general purpose of the application?

The application is a social network app. Its purpose is very similar to Instagram, where you can share pictures, as well as leave comments and like them. There is also a chat feature, where you can send direct messages to other users. In the messages screen, there is also a call feature that was never implemented.

**Back-end:**

Chat Route:

**Endpoint: /api/chat/getChatRooms/**

* Method: POST
* Middleware: checkAuth
* Controller: chatController.getChatRooms
* Purpose: Retrieve chat rooms for the authenticated user.

**Endpoint: /api/chat/getMessagesForRoom**

* Method: POST
* Middleware: checkAuth, chatValidator.getMessagesForRoom
* Controller: chatController.getMessagesForRoom
* Purpose: Retrieve messages for a specific chat room for the authenticated user.

**Endpoint: /api/chat/sendImage**

* Method: POST
* Middleware: checkAuth, chatController.upload, chatValidator.sendImage, chechRoom
* Controller: chatController.createImageMessage
* Purpose: Upload and send an image as a message in a chat room for the authenticated user.

**Endpoint: /api/chat/call**

* Method: POST
* Middleware: checkAuth, chatValidator.handleCall, chechRoom
* Controller: chatController.handleCall
* Purpose: Initiate a call in a chat room for the authenticated user.

**Endpoint: /api/chat/answer**

* Method: POST
* Middleware: checkAuth, chatValidator.answer, chechRoom
* Controller: chatController.answer
* Purpose: Answer a call in a chat room for the authenticated user.

**Endpoint: /api/chat/sendMessage**

* Method: POST
* Middleware: checkAuth, chatValidator.sendMessage, chechRoom
* Controller: chatController.sendMessage
* Purpose: Send a text message in a chat room for the authenticated user.

**Endpoint: /api/chat/readMessages**

* Method: POST
* Middleware: checkAuth, chatValidator.readMessages, chechRoom
* Controller: chatController.readMessages
* Purpose: Mark messages as read in a chat room for the authenticated user.

Comment Route:

**Endpoint: /api/comment/getComments/**

* Method: POST
* Middleware: checkAuth, commentValidator.getComments
* Controller: commentController.getCommentsForPost
* Purpose: Retrieve comments for a specific post for the authenticated user.

**Endpoint: /api/comment/getCommentReplies/**

* Method: POST
* Middleware: checkAuth, commentValidator.getCommentReplies
* Controller: commentController.getRepliesForComment
* Purpose: Retrieve replies for a specific comment for the authenticated user.

**Endpoint: /api/comment/addComment/**

* Method: POST
* Middleware: checkAuth, commentValidator.addComment
* Controller: commentController.addComment
* Purpose: Add a new comment to a post for the authenticated user.

**Endpoint: /api/comment/addCommentReply/**

* Method: POST
* Middleware: checkAuth, commentValidator.addCommentReply
* Controller: commentController.addCommentReply
* Purpose: Add a reply to a comment for the authenticated user.

**Endpoint: /api/comment/getCommentLikes**

* Method: POST
* Middleware: checkAuth, commentValidator.getCommentLikes
* Controller: commentController.getCommentLikes
* Purpose: Retrieve the likes for a specific comment.

**Endpoint: /api/comment/getCommentReplyLikes**

* Method: POST
* Middleware: checkAuth, commentValidator.getCommentReplyLikes
* Controller: commentController.getCommentReplyLikes
* Purpose: Retrieve the likes for a specific comment reply.

**Endpoint: /api/comment/likeComment/**

* Method: POST
* Middleware: checkAuth, commentValidator.likeComment
* Controller: commentController.likeComment
* Purpose: Like a specific comment for the authenticated user.

**Endpoint: /api/comment/likeCommentReply/**

* Method: POST
* Middleware: checkAuth, commentValidator.likeCommentReply
* Controller: commentController.likeCommentReply
* Purpose: Like a specific comment reply for the authenticated user.

Notification Route:

**Endpoint: /api/notification/readNotifications/**

* Method: POST
* Middleware: checkAuth, notificationValidator.readNotifications
* Controller: notificationController.readNotifications
* Purpose: Mark notifications as read for the authenticated user.

**Endpoint: /api/notification/getNotifications/**

* Method: POST
* Middleware: checkAuth, notificationValidator.getNotifications
* Controller: notificationController.getNotifications
* Purpose: Retrieve notifications for the authenticated user.

Post Route:

**Endpoint: /api/post/getPostsByHashtag**

* Method: POST
* Middleware: checkAuth, postValidator.getPosts, userController.getFollowings
* Controller: postController.getPosts
* Purpose: Retrieve posts for the authenticated user based on their followings.

**Endpoint: /api/post/getPostsByHashtag**

* Method: POST
* Middleware: checkAuth, postValidator.getPostsByHashtag
* Controller: postController.getPostsByHashtag
* Purpose: Retrieve posts containing a specific hashtag for the authenticated user.

**Endpoint: /api/post/getPostLikes**

* Method: POST
* Middleware: checkAuth, postValidator.getPostLikes
* Controller: postController.getPostLikes
* Purpose: Retrieve the likes for a specific post.

**Endpoint: /api/post/getPostsByLocation**

* Method: POST
* Middleware: checkAuth, postValidator.getPostsByLocation
* Controller: postController.getPostsByLocation
* Purpose: Retrieve posts based on location for the authenticated user.

**Endpoint: /api/post/addPost**

* Method: POST
* Middleware: checkAuth, postController.upload, postValidator.createPost
* Controller: postController.createPost
* Purpose: Upload and create a new post for the authenticated user.

**Endpoint: /api/post/getPost**

* Method: POST
* Middleware: checkAuth, postValidator.getPost
* Controller: postController.getPost
* Purpose: Retrieve a specific post for the authenticated user.

**Endpoint: /api/post/likePost/**

* Method: POST
* Middleware: checkAuth, postValidator.likePost
* Controller: postController.likePost
* Purpose: Like a specific post for the authenticated user.

**Endpoint: /api/post/delete/**

* Method: POST
* Middleware: checkAuth, postValidator.deletePost
* Controller: postController.deletePost
* Purpose: Delete a specific post for the authenticated user.

User Route:

**Endpoint: /api/user/signup**

* Method: POST
* Middleware: userValidator.addUser
* Controller: userController.addUser
* Purpose: Sign up a new user.

**Endpoint: /api/user/login**

* Method: POST
* Middleware: userValidator.loginUser, verificationCheck.verificationCheck
* Controller: userController.loginUser, userController.sendUserData
* Purpose: Log in an existing user and send user data.

**Endpoint: /api/user/getNewUsers**

* Method: POST
* Middleware: userValidator.getNewUsers
* Controller: userController.getNewUsers
* Purpose: Retrieve a list of new users.

**Endpoint: /api/user/passwordreset**

* Method: POST
* Middleware: checkAuth, userValidator.resetPassword
* Controller: userController.resetPassword
* Purpose: Reset the password for the authenticated user.

**Endpoint: /api/user/email/activate/:token**

* Method: GET
* Controller: userController.activate
* Purpose: Activate a user account using the activation token.

**Endpoint: /api/user/sendVerificationEmail**

* Method: POST
* Middleware: checkEmailEnv, userValidator.sendVerificationEmail
* Controller: userController.sendVerificationEmail
* Purpose: Send a verification email to the authenticated user.

**Endpoint: /api/user/sendforgotPasswordEmail**

* Method: POST
* Middleware: checkEmailEnv, userValidator.sendVerificationEmail
* Controller: userController.sendforgotPasswordEmail
* Purpose: Send a password reset email to the authenticated user.

**Endpoint: /api/user/getUserData**

* Method: POST
* Middleware: checkAuth, userValidator.getUserData
* Controller: userController.getUserData, userController.getUserPosts, userController.sendUserData
* Purpose: Retrieve and send user data for the authenticated user.

**Endpoint: /api/user/getPosts**

* Method: POST
* Middleware: checkAuth, userValidator.getPosts
* Controller: userController.getPosts
* Purpose: Retrieve posts for the authenticated user.

**Endpoint: /api/user/getProfilePageData**

* Method: POST
* Middleware: checkAuth, userValidator.getUserProfileData
* Controller: userController.getUserProfileData, userController.getUserPosts, userController.sendUserData
* Purpose: Retrieve and send user profile data for the authenticated user.

**Endpoint: /api/user/getUserProfileFollowers**

* Method: POST
* Middleware: checkAuth, userValidator.getUserProfileFollowers
* Controller: userController.getUserProfileFollowers
* Purpose: Retrieve followers for a user's profile.

**Endpoint: /api/user/getUserProfileFollowings**

* Method: POST
* Middleware: checkAuth, userValidator.getUserProfileFollowings
* Controller: userController.getUserProfileFollowings
* Purpose: Retrieve followings for a user's profile.

**Endpoint: /api/user/addProfiePicture**

* Method: POST
* Middleware: checkAuth, userController.upload
* Controller: userController.changeProfilePicture
* Purpose: Upload and change the profile picture for the authenticated user.

**Endpoint: /api/user/updateUser**

* Method: POST
* Middleware: checkAuth, userValidator.updateUser
* Controller: userController.updateUser
* Purpose: Update user information for the authenticated user.

**Endpoint: /api/user/searchByUsername**

* Method: POST
* Middleware: checkAuth, userValidator.searchByUsername
* Controller: userController.searchUsersByUsername
* Purpose: Search for users by username for the authenticated user.

**Endpoint: /api/user/followUser**

* Method: POST
* Middleware: checkAuth, userValidator.followUser, checkUser
* Controller: userController.followUser
* Purpose: Follow a user for the authenticated user.

**Endpoint: /api/user/delete/**

* Method: POST
* Middleware: checkAuth
* Controller: userController.deleteUser
* Purpose: Delete the authenticated user's account.

**Front-end:**

There is a front-end in this application and it consists of several pages. Here are the screenshots of every page:

* The different views (take screenshots to document them)

Register page:  
A screenshot of a computer

Description automatically generated

Login page:

A screenshot of a computer

Description automatically generated

Home page:  
A screenshot of a dog

Description automatically generated

Location page:  
A screenshot of a map with a dog

Description automatically generated

Messenger page:  
A screenshot of a chat

Description automatically generated

Not found page:  
A screenshot of a computer

Description automatically generated

Password Reset page:  
A screenshot of a computer

Description automatically generated  
  
Post page:  
A squirrel eating food on a table

Description automatically generated

Post Upload page:

A screenshot of a computer

Description automatically generated

Profile page:

A screenshot of a social media account

Description automatically generated

* What are the actions possible?

In this app, a user can create an account and log in to it, to upload images, to see friends' photos (if his/her friends created an account in this app) in their profile, to chat with friends and to indicate his/her location in the app.

* Is there authentication required?

Yes, the user is authenticated by the jwt token, when the user logs in the app, the user gets a jwt token in the local storage of the app and that token is used to keep the user interacting with other pages in the app. If the user is not logged in then the user can’t access the authenticated routes.

**Database:**

This app uses MongoDB as a database. The name of the database is “finalProject” and it has several collections.

* The different collections/table names

There are ten collections inside the “finalProject” database, which are: chatrooms, followers, followings, messages, notes, notifications, postlikes, posts, sessions, and users.

* What are the models used in the database?

There are twelve models used in this app. These models are ChatRoom.js, Comment.js, CommentLike.js, CommentReply.js, CommentReplyLike.js, Followers.js, Following.js, Message.js, Notification.js, Post.js, PostLike.js, and User.js.