VRIJE
UNIVERSITEIT
BRUSSEL

ASSIGNMENT 2: MVC

Software Architectures

Anisha Sachdeva

August 21, 2022

Prof. Coen De Roover

1 Introduction

We design a social networking app named as "Say it on Social" using ScalaPlay framework. The implementation follows the Model-View-Controller pattern. The basic working of the application includes the following functionalities,

1. A user can register on the web application
2. A user can login using the correct username and password on the web application
3. As soon as a user logins the application, he/she can see the explore page with posts posted by the user profiles he/she is following along with the hashtags (topics), he/ she is subscribed to.
4. A user can follow another user profile or subscribe to a topic
5. A user can share a new post. The post can be an image or text, or combination of both. The only necessary thing is to have a hashtag assigned to a post.
6. A user can see, like and comment on posts.
7. A user can delete their own post by visiting their profile.
8. A user can see the explore page in two available sorting options, i.e. Like and Date.

Above mentioned are the functionalities that a user can perform. More details are followed in the next section.

2 Implementation

To make our web application using MVC patter in Scala Play Framework, we start by making a login and registration function. For this we define the following:

1. LoginController - LoginController contains all the methods required for login and registration functionality. It contains a loginForm which basically gets the information like username and password from the user and checks if the user exists with the same correct credentials. If so, it allows the user to see the Explore page and creates a session for the existing user.

   If a user does not have credentials and wants to sign up to use the application, the user can navigate to the Registration page. The user fills a form on registration page with the username and desired password. Then the system verifies, if the user does not exist already and fulfills all the checks for a password (like password should contain atleast one upper case, etc) then the user is registered and is show the blank Explore page as the user is not yet following any user or subscribed to a topic.
2. UserDAO (Model) - UserDAO contains the necessary information regarding a user like username, password, list of following users and a list of subscribed topics. For ease of demo for the application, some user data is prefilled.
3. Login and Registration views - As soon as a user opens the application, he/she is navigated to a login page view to login. However, if the user is not yet registered then he/she can simply click on the Registration link given on the top right of the login view. The user is then navigated to the Registration view wherein they can register.

Once our login and registration of new users is functional, we move forward to displaying posts, following other users, subscribing to new topics, sharing a new post etc.

1. PostController: PostController contains all the logic behind displaying posts and how to behave while certain action, for example, to like a post or to delete a post is taken.

   PostController starts with the method defined to display posts on the Explore page. Here we define that the explore page will contain the post, which are either posted by the user themselves or the posts of the other users they are following or the posts of the topic they are subscribed to. A decision that we took here was, in the project description it was mentioned to have only the posts of the following users on the Explore page, however in second session there was an extension to subscribe to topic. It was mentioned that when a user is subcribed to a topic and a new post on that topic is made available, the user should also get that in their feed. With both the description statements, we came to the conclusion that a post with a subscribed topic should be made available on the Explore page even if that post is posted by a user who the current user is not following. On the basis of subscribed topic, the user will get the post on explore page.

   Further on ever page, the posts can be sorted on the basis of date or number of likes.

   Next functionality included is to enable a user to add a comment to a post. For this we used a commentForm and basis on the input from the user, we display the comment. As part of a comment the user can also upload a picture from their computer to the web application. We can dynamically add and are not dependent on the pictures saved in the public folder of our play application.

   The user can also like any post. If the user would have already liked the post, the counter will not increment but give the user a nice message, that they have already liked the post.

   Another functionality is to add a new post. On clicking the "Share a thought" navigation link on top right, user will be shown a form where they can add a post in the form of an image or text or combination of both. However, they need to add a hashtag(topic) to their post, else the post wont be uploaded. Again the user can upload a picture from their desktop dynamically and it will get stored in the public folder of play. The new post will then be available to all the users following the user who made the post and ofcourse on user's own profile.

   Next, on clicking to 'Profile' link on top right of the Explore page, the user can see their own profile wherein they get the option to delete a post, if they want. A user can delete a post on by visiting their own profile. Again the posts can be sorted on number of likes or date.

   Now, what if a user is not following someone they want to follow? Simple, they can search for the user in the search bar given on the right of the screen. To search for a user profile, they need to type '@' followed by the name of user. They can simply go the user profile, see their post and if want to follow them (and of course are not following yet) can follow. The user's post whom they just followed will become available on the Explore page. Similarly a user can also search for a topic to subscribe. To search for a topic, they need to type '' followed by the topic. They can visit the topic and subscribe to it. Users can subscribe to n number of topics.

2. PostDAO(Model): PostDAO basically contains the information related to a post like postId, description, date, post uploaded by, image filename, list of users who liked the post, list of comments on post and a mandatory topic.

3. For posts views, we create several views. A view is created depicting how a post should look like. Another 'Explore' view is created to define the view of an Explore page. Then the 'userProfile' view depicts how a user profile should look like. But interestingly, the only when a user is on their own profile, they get the option to delete a post else on other users profile they of course dont get the option. Another view is created to show posts of a same topic altogether. Last and final view to 'Share a Thought' is created which depicts a form that user fill in turn to post something new.

In the routes file, we write the appropriate routes.

3 Results

Results in form of images are displayed below.

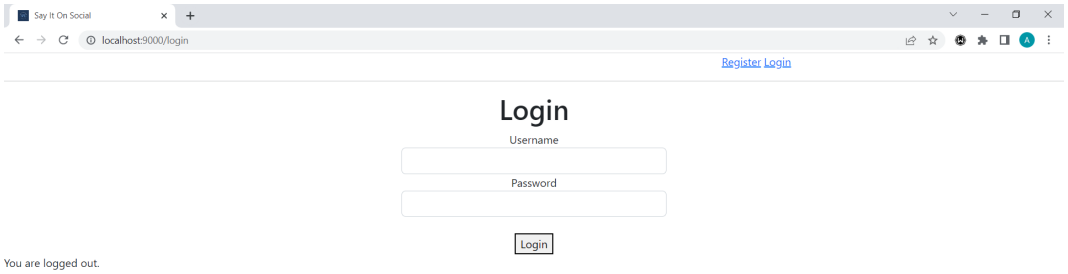

Figure 1: Login Page View

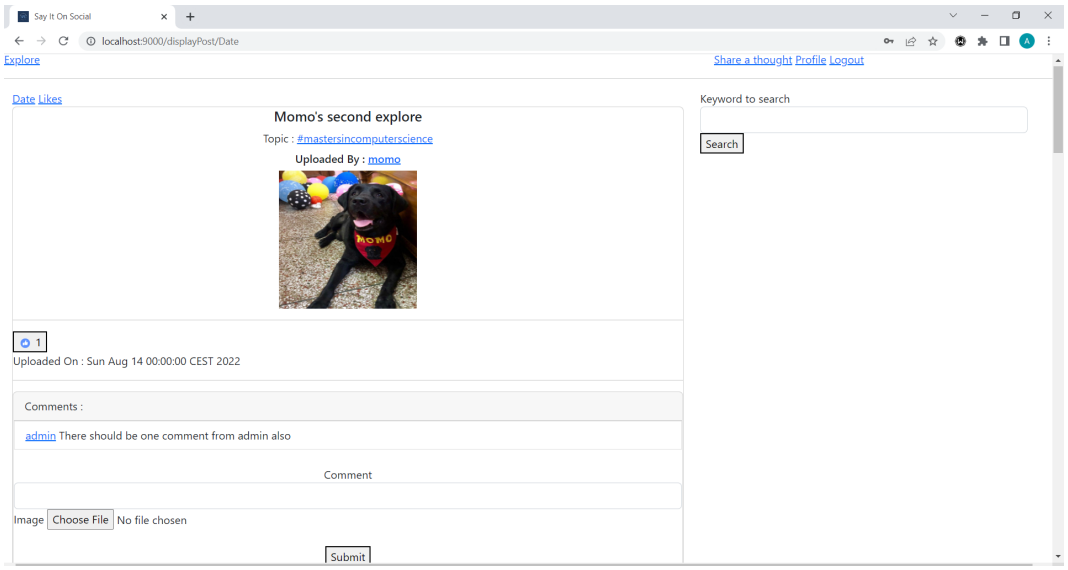

Figure 2: Explore Page View

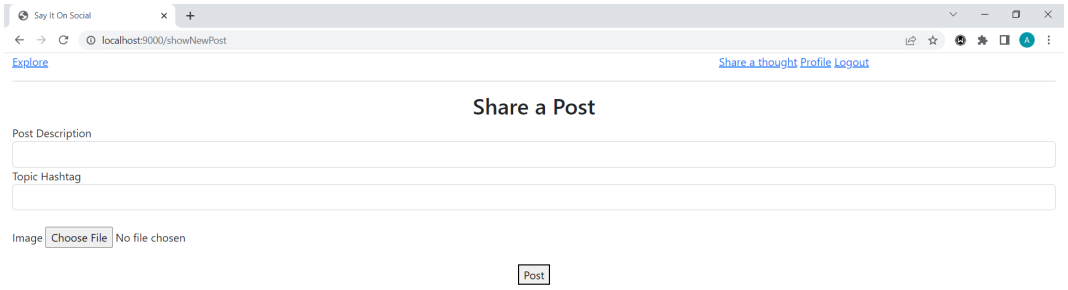

Figure 3: A view to add a new Post i.e. Share a thought

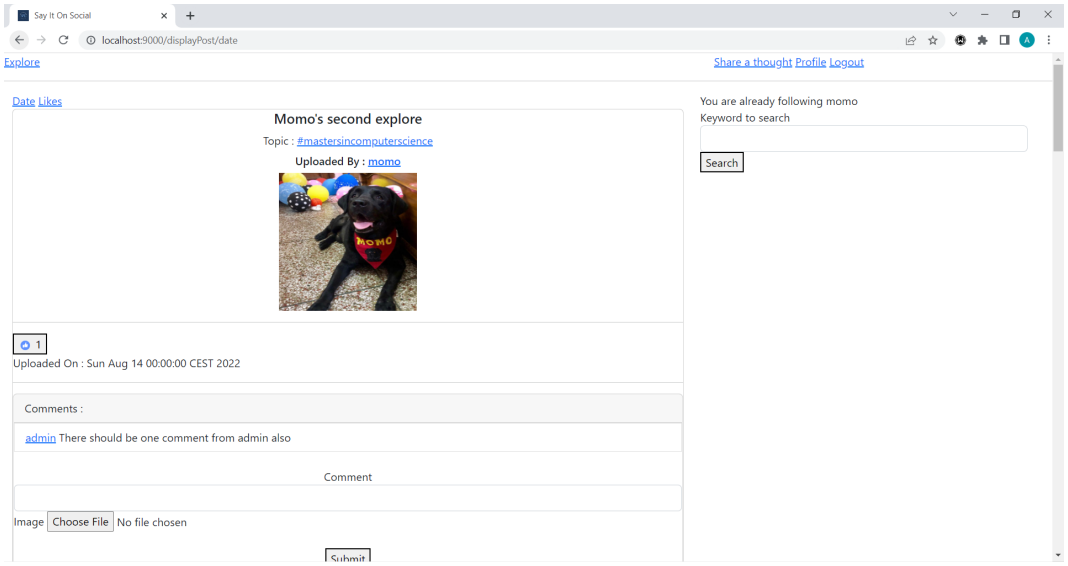

Figure 4: System is robust to show errors. In the image above it can be seen that is a user is already following other user, they cannot follow them again.