Software Architectures

Assignment 2: MVC using Scala Play

Ahmed Zerouali, Camilo Velazquez
Emails: {azeroual, cavelazq}@vub.be
Office: {10F725}

Assignment

The goal of this assignment is to implement a Model-View-Controller (MVC) application using Scala Play.

Deadline: January 10^{th} 2021 by 23:59. The deadline is fixed and will not be extended for any reason.

Deliverables: A report and source code. The report (in English) explains your solution to the problem and the main components used in your implementation. When possible, use simple diagrams to illustrate.

The report should be handed in as a single PDF file of no more than 3 pages (excluding diagrams/screenshots/code). The file should follow the naming schema `FirstName-LastName-SA2.pdf`, for example: `Camilo-Velazquez-SA2.pdf`.

The source code is your implementation of the project zipped or exported from IntelliJ.
Submit the *report* and *source code* as a single zip file on the Software Architectures course page in Canvas, by clicking on *Assignments* > *Assignment 2*.

Plagiarism: Note that copying – whether from previous years, from other students, or from the internet – will not be tolerated, and will be reported to the dean as plagiarism, who will decide appropriate disciplinary action (e.g., expulsion or exclusion from the examination session). If you use any other resources besides those provided in the lectures and in this document, remember to cite them in your report.

Grading: Your solution will be graded and can become subject of an additional defense upon your or the assistants' request.

Problem Description

For this assignment you will implement a social networking website application where users can publicly share their pictures so that other users can see them and like, dislike or comment on them. The implementation of the application should follow the Model-View-Controller pattern. The mockup in Figure 1 could be a motivational example for the front page of your website.

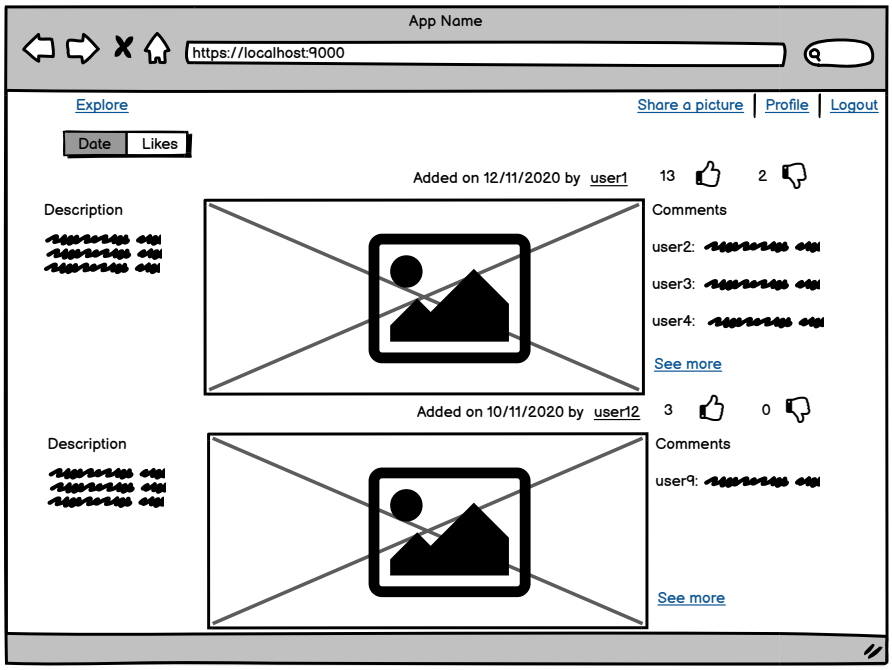

Figure 1: Front page of the website

As shown, you will have to design the layout of the website in a way that pictures are displayed in the middle of the screen, while description, likes and comments are at the left and right sides of the pictures. The pictures can be shown by their date of upload or their number of likes/dislikes. In the comments section, the front page should only show a limited number of comments, to see more comments and see the figure in bigger size, you should create a "`See more`" link that will redirect the users to the page of the corresponding picture. A possible page could look like the mockup in Figure 2.

In the home page in Figure 1 and the picture page in Figure 2, users can see shared pictures regardless of whether they are logged in or not. If they are logged in, they can additionally make comments, like or dislike the pictures, as well as share new pictures.

Users that are not logged in should not be able to perform the actions described above. The elements "`Share a picture`" and "`Profile`" should be hidden from the navigation bar at the top when users are logged out. To add new pictures on the website, you should implement a form that looks like the mockup in Figure 3. This form contains

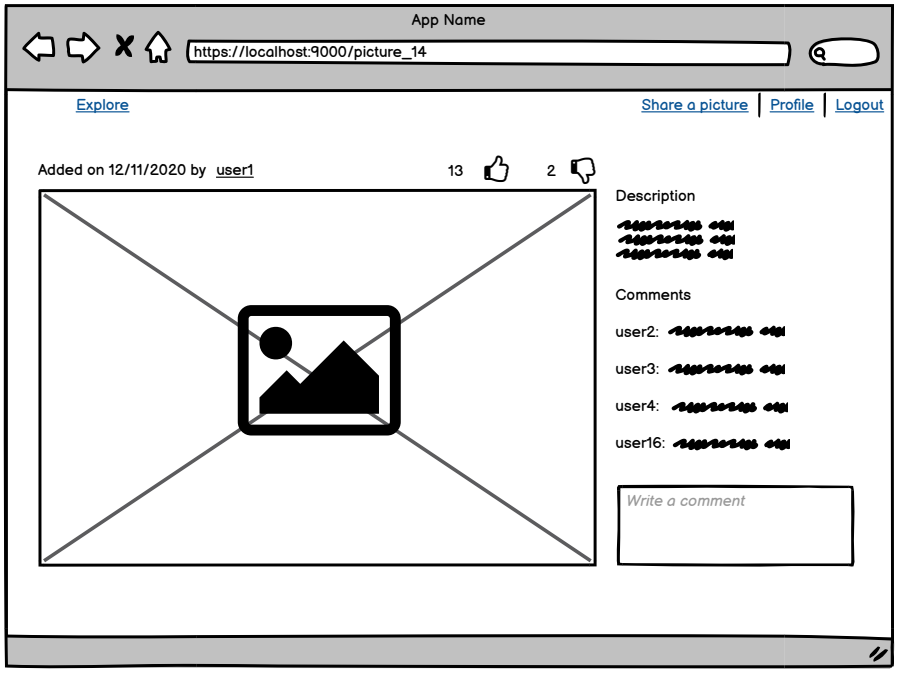

Figure 2: Picture in full size with all comments.

the mandatory fields such as the file upload input and text area, etc. Once the shared picture is selected and submitted, any logged-in user should be able to see, comment or like/dislike it.

All users should be able to visit each other's profile. Moreover, logged-in users should be able to remove their own shared pictures when they access their profiles. You are also required to implement a page where unregistered users can sign up.

As a final recommendation, you can use any public template for your web page design (e.g. Twitter's Bootstrap). Do not use a database for your project, just keep the data in memory or store them in a json file. Pictures should be stored in a public folder within the project.

The above functionalities should be implemented using Model-View-Controller (MVC) pattern with Scala Play. In addition to the general approach, you will be evaluated according to how you implemented the different functionalities, views and controllers. You should take into account issues such as validations, error handling and any other abstractions that you deem fit for the scenario. Motivate your design decisions in the report as per the problem description and illustrate your approach using concepts in Scala Play.

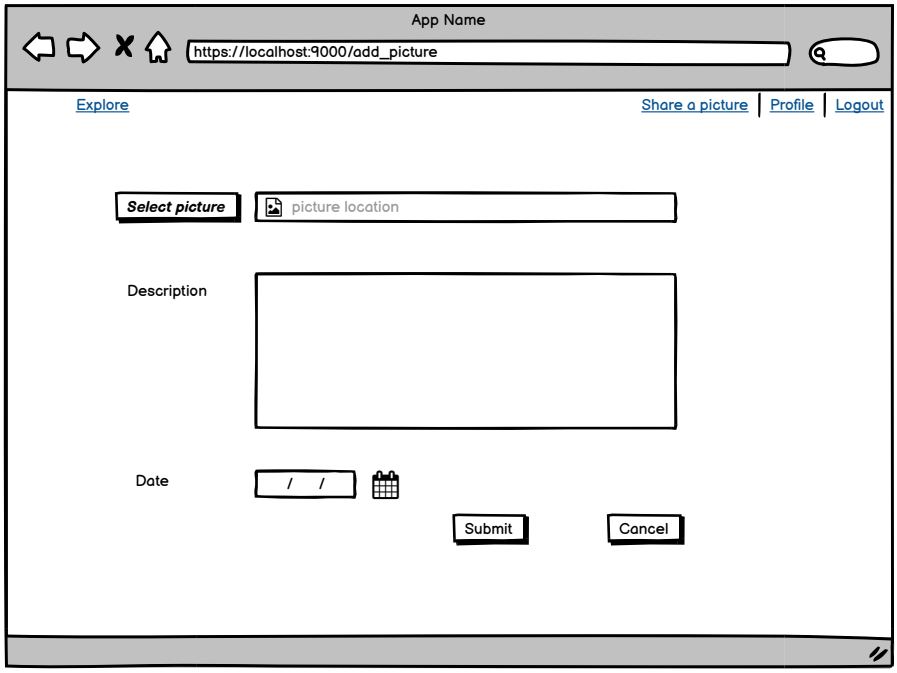

Figure 3: Share a new picture.

More information on Scala Play

- `https://www.playframework.com/`