Software Architectures

Assignment 2: MVC using Scala Play

TA: Camilo Velázquez-Rodríguez
camilo.ernesto.velazquez.rodriguez@vub.be
F.10.725

Assignment

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

Deadline: January 7th 2024 by 23:59. The deadline is fixed and will not be extended for any reason. Submissions after the deadline will not be considered for grading and the grade will automatically be **ABS**.

Deliverables A report and source code. The report (in English) explains your solution to the problem and the main components used in your implementation. Please use simple diagrams to illustrate the architecture of your solution.

The report should be handed in as a single PDF file.
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 in a single file. Do not include files or folders resulting from the compilation process of your project, e.g., target/ or *.class. 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 Any suspicion of plagiarism will be reported to the Dean of Faculty without delay. Both user and provider of such code will be reported and will be dealt with according to the plagiarism rules of the examination regulations (cf. OER, Article 118). The dean may decide on (a combination of) the disciplinary sanctions, ranging from 0/20 on the paper of the given programme unit to a prohibition from (re-)enrolling for one or multiple academic years (cf. OER, Article 118$5).

Grading Your solution will be graded and can become the subject of an additional defence upon the lecturer's request.

Problem Description

For this assignment you will implement a social networking website application where users can share their pictures so that other users can see them, like and comment on them. The implementation of the application should follow the Model-View-Controller pattern. Figure 1 depicts the suggested layout for the front page.

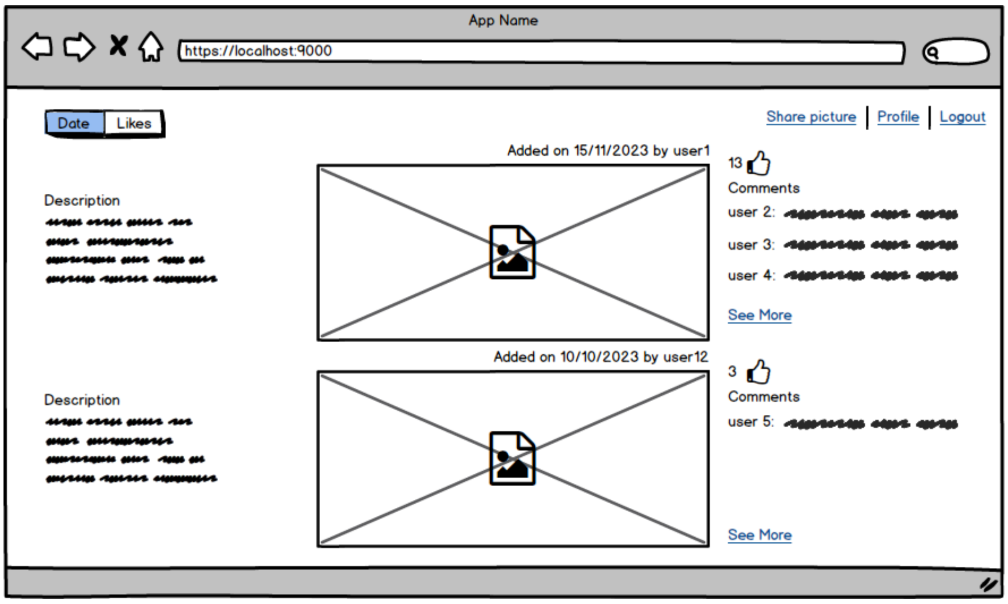

Figure 1: Front page of the website.

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, respectively. The pictures can be sorted in ascending or descending order by their date of upload or their number of likes. 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. Figure 2 depicts the suggested layout for the details page.

Only logged-in users can see shared pictures. Additionally, they can make comments or like the pictures, as well as share new pictures. To share new pictures on the website, you should implement a form that looks like the mockup in Figure 3. This form contains the mandatory fields such as the file upload input and the text area to add a description.

The elements "`Share a picture`" and "`Profile`" should be hidden from the navigation bar at the top when users are logged out. You are also required to implement a page where unregistered users can sign up.

Do not spend too much time on the aesthetics of the website. A bare-bones layout without CSS suffices. 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

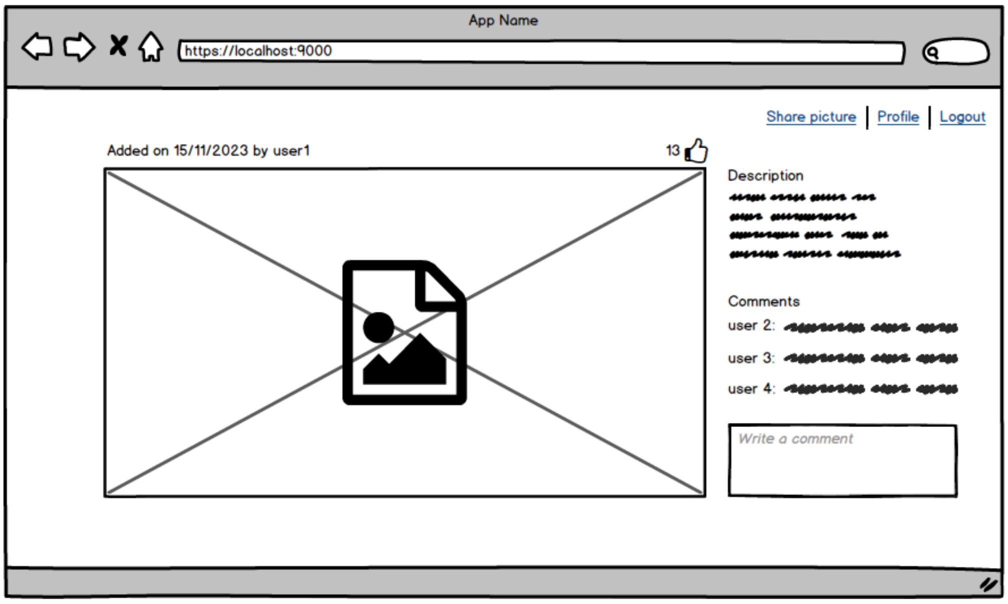

Figure 2: Details page with all comments.

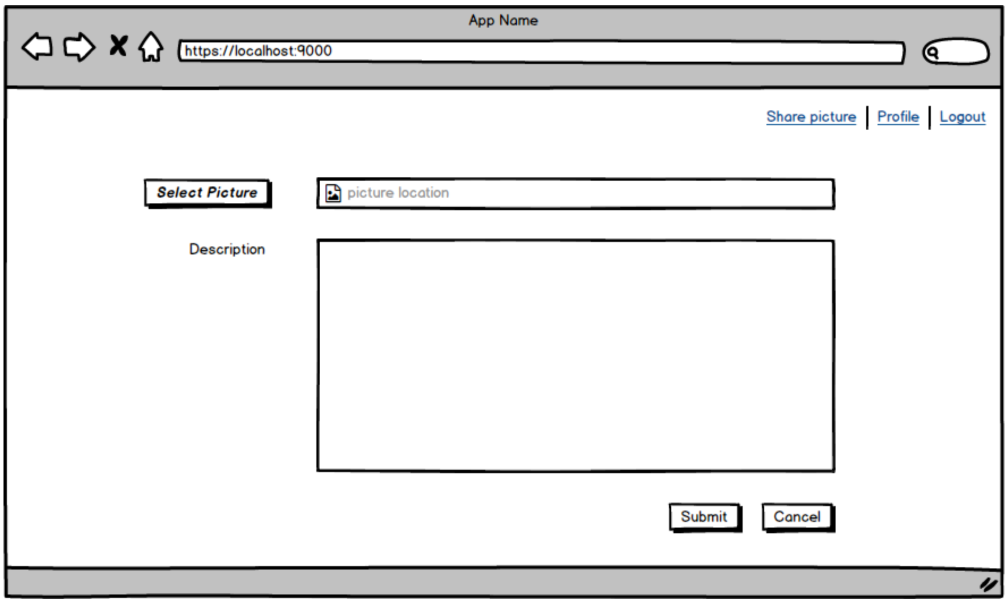

Figure 3: Submit a new picture.

within the project.

The above functionalities should be implemented using Model-View-Controller (MVC) pattern with Scala Play and Scala 3. 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.

More information on Scala Play

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