

Contour

Bartosz Kaczmarek 327244

October 2024

1 Introduction

Contour is a web application, inspired by Artstation, which allows users to share their artworks with everyone. The "everyone" may be understood figuratively, but I also take pride in its more literal meaning - in the spirit of Progressive Enhancement, the basic functionality (that being viewing the pictures' gallery) uses no JavaScript¹, to ensure it does not stop anyone from accessing the website.

2 Functionality

1. Anonymous users can:
 - (a) View gallery of pictures
 - (b) Filter pictures by author and/or combinations of tags
2. Registered users can also:
3. Save their favourite pictures to the Pictmark tab so they can find them easily later.
4. Post their own pictures (and delete them if they ever need to!).²
5. Express themselves using changeable avatars!
6. If they ever forget their password, a handy recovery functionality is also added.
7. Managers and Admins:

¹Original idea of the application was to implement all functionality without any front-end code execution, while allowing it for users who enable JavaScript, but it was changed to adhere to the requirements of the project.

²The editing functionality was considered, but the potential abuse could be tragic. Consider scenario, where a user posts a perfectly normal photo, people save it to Pictmark, and then a malicious agent replaces it with some graphic content. Therefore, the preferred way to fix mistakes is to delete picture and post it anew, if necessary.

- (a) Managers have the power and responsibility to delete images that violate the rules of the service, but, so as to not make a mistake, first they can restrict the visibility of a picture until the decision is final.
- (b) Admins additionally may ban repeat offenders, to ensure the community's well-being.

3 Technical Information

The application uses the following technologies:

3.1 Data

Data is stored in **SQLite3** database. This might seem inappropriate, due to the nature of SQLite3 existing as singular file. And, although creators of SQLite denounce this claim: Appropriate Uses For SQLite, it could be problematic if significantly many users were to start using it at once. However, SQLite3 was chosen to allow the easiest possible deployment, which seems like a good choice for a project made for educational purposes.

3.2 Backend

The programming language used for serving the web application is **Racket**. As it is not as well known as other options, I will provide a short introduction in case it is necessary: Racket is a general purpose programming language. As a dialect of Lisp, it leans towards the functional paradigm, with heavy emphasis on metaprogramming. I chose it for this project because of its straightforward functionality - the web framework it provides doesn't offer "automagical" solutions, allowing me to work "close to the metal" with other technologies and standards like HTTP.

3.3 Rendering Pages

The layout is handled using normal **HTML**, with **Scribble** (which is part of Racket) used for templating. For the styling, I used **Less**, to allow easier colour-scheme management than in pure CSS. To render icons, I used **Google's material icons**.

3.4 Interactivity

The web application uses exactly four **JavaScript** scripts:

1. **Umbrella JS** - A lightweight, modern jQuery substitute, which is used as a dependency in the other 3 scripts.
2. `authors.js` - A script that showcases dynamic client-side rendering of list of authors.

3. buttons.js and forms.js - A generic scripts, inspired by HTMX, I have written to create data-driven, enhanced functionality for HTML forms and buttons. As long as the components have the right data-* attributes and structure, those scripts allow them to nicely configure their behaviour without need of writing specific scripts for each use case.

3.5 Automatic Tests

The project has two kinds of automatic tests, which run in a separate database from the one used in main code.

1. Database Tests - First, it is necessary to check that the database functionality is working correctly. We comprehensively check that database is left in consistent state after applying some of the operations described in the Functionality section to the model.
2. API Tests - Afterwards, knowing that we can trust the database, we ensure that the API endpoints receive and respond with data in proper form, by simulating requests and checking that the server succeeds or fails accordingly to the data sent.

3.6 Details

The following information pertains to small fragments of the codebase:

3.6.1 Sending Emails

For sending emails, I use **Google Gmail's SMTP server** functionality.

3.6.2 Security

The user's passwords are encrypted using **OpenSSL's** libcrypto library, and uses the password-based key-derivation function called **scrypt**. For user authentication and authorization we use cookie-based session tokens.

4 Preview

Lastly, here are some screenshots of what users can expect of the Contour website:

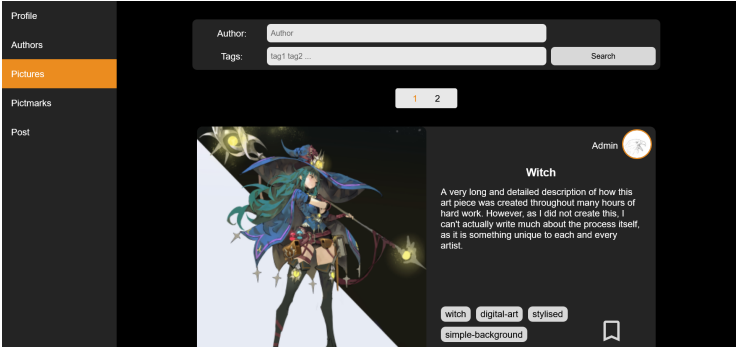


Figure 1: Picture Gallery As Seen On Desktop

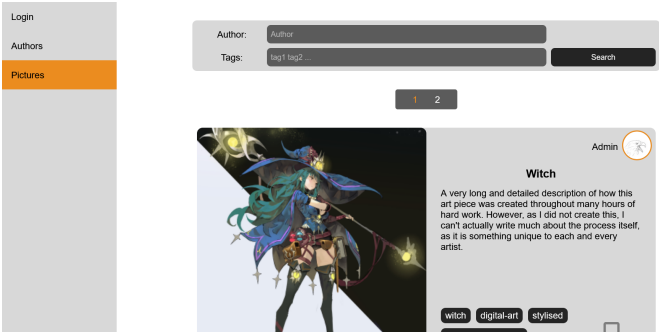


Figure 2: Picture Gallery As Seen On Desktop - Light Mode

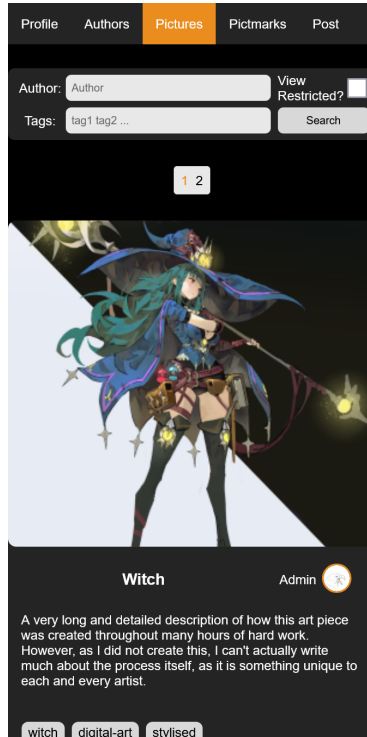


Figure 3: Picture Gallery As Seen On Mobile Phone (With some additional UI for Managers and Admins)

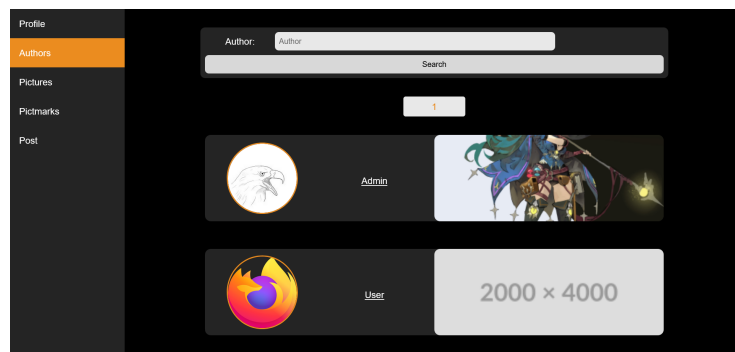


Figure 4: Authors List

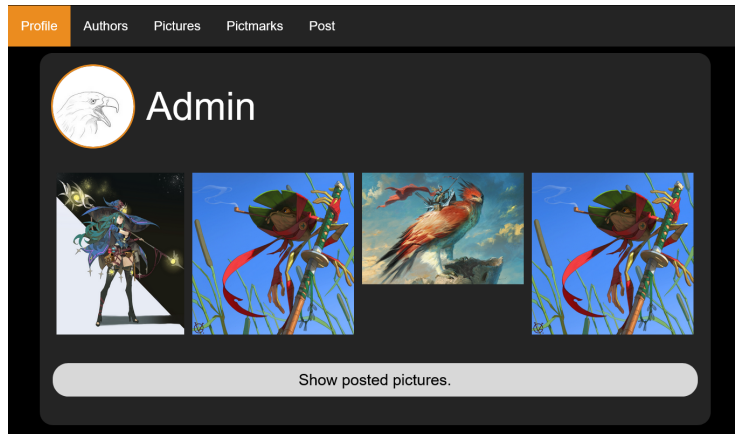


Figure 5: Profile View

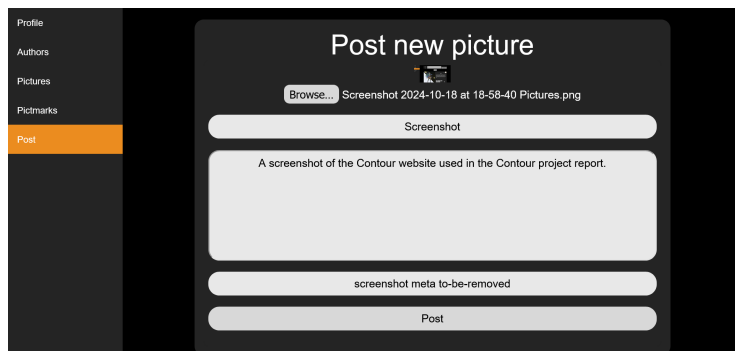


Figure 6: Posting new picture

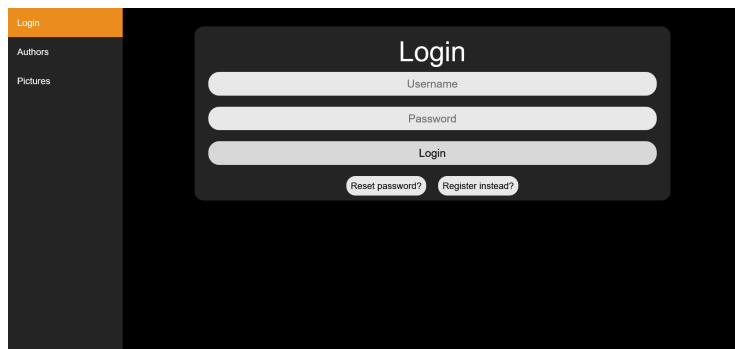
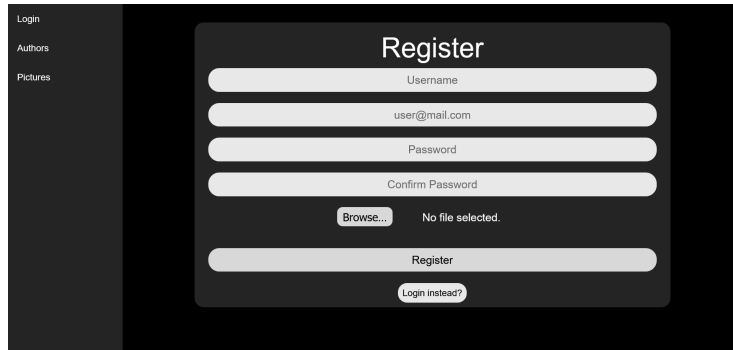


Figure 7: Login page



The screenshot shows a web application with a dark theme. On the left is a sidebar with links: 'Login', 'Authors', and 'Pictures'. The main content area is titled 'Register' and contains a form with the following fields: 'Username' (with the placeholder 'user'), 'Email' (with the placeholder 'user@mail.com'), 'Password', and 'Confirm Password'. Below these is a file upload section with a 'Browse...' button and the text 'No file selected.'. At the bottom of the form are two buttons: 'Register' and 'Login instead?'.

Figure 8: Registration page



The screenshot shows the same web application interface as Figure 8. The main content area is titled 'Password recovery' and contains a form with two fields: 'Email' (with the placeholder 'example@mail.com') and a 'Send recovery email.' button.

Figure 9: Password Recovery

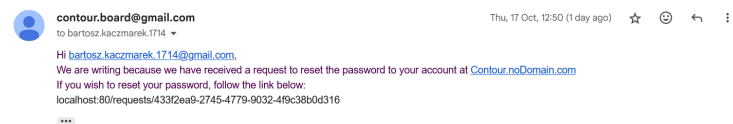


Figure 10: And we received a nice email regarding our request!