CS194/W README - ScrImage

Matthew Pick, Tope Olagbemi, Michelle Pan

**\*\* scrimage.herokuapp.com \*\***

**Codebase**

The app is built using Ruby on Rails and Postgres. Here is a general overview of the important folders/files in our codebase followed by a detailed file-by-file breakdown:

/app/assets/

Holds CSS files for the entire app and Javascript files for modals, carousels, hover icons, etc.

/app/controllers/

Home of all controllers: favorite, home, photos, scrimages, searches, sessions (handles login/logout), users

/app/helpers/

Helper methods that are used in controllers

/app/models/

Home of all models: comments, favorites, notifications, photos, scrimages, users, and scrimages/user (makes sure users only vote once per scrimage)

/app/views/

The views for each user-facing page in our app, organized by folders corresponding to each model

/db/migrate/

All of the database migrations that are run when rake db:migrate is called, we usually run rake db:migrate:reset and rake db:seed in succession to properly clear and load the database

/db/seeds.rb

Loads the initial seed data into our database when rake db:seed is called

/public/images/

Where all of our edited images and user profile pictures are stored

/public/site\_images/

Where all of the main icons and graphics needed for the site are stored

# File Overview - Controllers

Favorite Photos Controller

create

Creates a new favorite object with current user and photo

destroy

Deletes the favorite object from the database with given user and photo IDs

set\_project

Finds the photo object based on the given photo ID

Home Controller

index

Sets variables for the four most recent photos as well as a random scrImage open for voting

Photos Controller

show

Displays an individual photo’s page including comments and the edit history tree

vote

Checks to see if user has already voted in photo’s scrImage, if not, increments the vote count for the current photo

create\_comment

Called via Ajax on an individual photo’s page to add a new comment and display the new comment on the page via javascript

save\_edited\_photo

Handles the process of locally storing any photos edited by the Aviary API. Called via Ajax every time a user saves an API-edited image.

draw\_tree

Draws the edit history tree on an individual photo’s page. Is called when the page loads

getPhotoForTree

Called via Ajax when the user clicks the button to expand the history tree

Scrimages Controller

show

Displays an individual scrimage

create

Creates a new srimage and original photo

uploadEditedImage

Called via Ajax on the scrimage show page to create a new photo when the user uploads an edit

render\_children

Displays all of the edits within a scrimage on the scrimage page

add\_winner\_badges

Determines which photos in a given scrimage have the most votes, ensuring that said photos have the winner badge displayed next to them. Also creates notifications for users whose photos were winning photos. Called via Ajax after the voting period of a timed scrimage ends.

show\_winning\_images

Stores the winning photos for a particular scrimage in an array, passing this array over to a partial (winningImages.html.erb) that renders any winning images for a scrimage at the top of the scrimage page. Called via Ajax after the winner badges for winning images have been drawn.

add\_share

Toggles the appearance of Facebook share links for the images on a scrimage page. Called via Ajax when 1) the voting period for a scrimage first opens and 2) when the voting period for a scrimage closes. When a scrimage’s voting period first opens, a Facebook share link will appear for each image, allowing users to ask for votes from their friends on Facebook. When a scrimage’s voting period closes, the Facebook share link will disappear, signifying that a user can no longer solicit for votes.

Searches Controller

show

Draws the initial page when the user gets to search page

searchData

Is called by Ajax when a user types a search query. Shows all scrimages if no search parameters are given

Sessions Controller

create

Creates a new session on user login, redirects to user’s profile

destroy

Deletes current session on logout

Users Controller

show

Shows the homepage for an individual scrimage including checking for a user’s notifications and setting the winner of scrimages that have closed

new

Creates a new user and redirects to their profile page

edit

Finds the given user based on id for editing

update

After edit form submitted - updates user’s attributes in database

set\_scrimage\_winners

Finds all timed scrimages that have closed but have not had their winner set and sets the winners and sends notifications to the winner(s)

getNotifications

Retrieves all the notifications for a given user

File Overview - Helpers

Scrimages Helper

remaining\_time(scrimage)

Returns the time remaining in the given scrimage

voting\_time(scrimage)

Returns the amount of time left in the given scrimage’s voting period

Sessions Helper

log\_in(user)

Saves the user’s ID in the session hash

current\_user

Returns the current user object, or nil if none

current\_user?(user)

Checks if the given user is the current user

get\_current\_user\_id

Returns the ID of the current user

logged\_in?

Checks to see if user is currently logged in

log\_out

Deletes the logged in user’s ID from the session hash and sets current user to nil

# File Overview - Javascripts

In addition to the javascript located within the view files, there are other javascripts within app/assets/javascripts

hoverIcons.js

This js file contains methods for the hover buttons that are used to submit photo edits

initHoverIcons(): initializes the hover icons on the scrimage page

uploadEditedImage(imageID): when a user uploads an image they edited offline

modal.js

This file contains methods that handle the showing and hiding of modals as well as some of the text they display

showModal(modalID, imageID)

closeModal(modalID)

showPostSaveModal()

closePostSaveModal()

showSaveSuccessText()

showSaveErrorText()

carousel.js

Controls/displays the Bootstrap carousel

transition.js

This file is used for the javascript carousel sliding transitions

spin.min.js

This file is used for the spinner that appears while saving an image edited with the Aviary API

feather.js

The javascript needed to initialize the Aviary API

# Additional Comments

The only element of our website that we did not write ourselves is the online photo editor. The editor, an API made by the company Aviary, was something that we chose to integrate because of its extensive capabilities as a photo-editing platform. Apart from allowing users to change various parts of an image -- the image’s color, text, orientation, etc. -- the API provides developers with an easy way to keep track of any photos that people edit. Thus, we were able to use the API to not only save newly-edited photos to the database but also update already-saved images. We also had the option of selecting which of the API’s editing features we wanted enabled, which allowed us to select only the features that we felt users would want/need, removing features that were confusing or redundant.

We did our own custom CSS styling rather than using a pre-styled layout such as Bootstrap and wrote our own Javascript tools such as the hover buttons to edit and upload images.

Changes from User Testing

We did user testing during week 9 and made some tweaks to the app based on the responses. Here are the things we changed:

* Users wanted a way to share their photos with friends, so we added Facebook sharing
* There was no time zone set on the scrImage start/end dates, which was confusing to people, so we changed everything to be in PST
* Users wanted the clocks to be displayed in a 12-hour format rather than military
* Our forms did not handle error checking, which was a big issue we ran into during user testing; we later went back and made sure our forms handled these errors properly
* Users wanted to know if/when they won a scrImage, so we implemented a notification system to ping them if they win
* It was unclear to the users what the trophy in the corner of some images meant (winning image), so we changed the image to a more intuitive winner’s badge