Software Engineering

CSC648/848 Summer 2018

# Infinite Image

**Team 6:**
Zac Henney
zhenney@mail.sfsu.edu
Paul Ancajima
Andre Leslie
Teodora Caneva
Abdullah Amir
Joe Phabmixay
Rohit Nair

Milestone 4

August 4th, 2018

First Submission

## 1) Product Summary

**COME VISIT ONE OF THE BEST STOCK PHOTO SITES ON THE WEB! JOIN FREE TODAY!**

**INFINITE IMAGE**

**http://18.191.194.127**

Ever visit a website and feel overwhelmed with an excessive amount of content? We at Infinite Images designed a simple yet sleek site that is free from all that clutter. The goal is to make our website the easiest to use stock photo site out there. Which is why we only value essential functions, such as:

- **Search**: Our search allows narrowing images within the category.
- **Thumbnails**: For quick browsing, we display thumbnail to make images load smoother.
- **Sign Up**: Registering is **FREE** for everyone!
- **Login**: User login is quick and easy will not take up any essential viewing time.
- **Upload**: Our upload feature supports uploading more than one image at a time.
- **Download**: Downloading images is **FREE** for people who are registered.
- **Admin:** In order to filter images for their content and quality.

In addition to keeping things simple, we pride ourselves with offering a free platform for high-quality images that are filtered by our admins in-order to ensure both quality and content. We understand that when browsing for images, especially quality images, it is very difficult to tell whether or not the image is under copyright. Well here at Infinite Images our database hosts a wide variety of royalty free images that are free and reusable. And for all the artists and creative minds out there that would love to share their passion, inspire others by joining today and contributing your work!

## 2) Usability Test Plan

**Test Objective:** Within this test, the intended user will be introduced to our website and product - InfiniteImage. The primary objective of this test is to indirectly expose usability flaws. User will be given a simple task, that would test our search functionality. User should be able to search and find an image of an animal, and view/expand it. We want to measure effectiveness and efficiency.

**Test Plan:**

| **TASK / CRITERIA** | **DESCRIPTION** |
|---|---|
| Task to be completed | Search and find an image of an animal (wild or domestic) |
| System Setup | Website is loaded on a computer with an internet connection. |
| URL of System | **http://18.191.194.127** |
| Starting Point | User starts on the homepage of InfiniteImage - search is visible. |
| Completion Criteria | User was able to use search, find an animal image, and expand it. |
| Benchmark | Task was completed in under 40 seconds. |

**Questionnaire:**

1. Searching images was responsive and user-friendly

☐ Strongly Agree ☐ Agree ☐ Neither ☐ Disagree ☐ Strongly Disagree

2. Image results loaded quickly

☐ Strongly Agree ☐ Agree ☐ Neither ☐ Disagree ☐ Strongly Disagree

3. I understood the process of searching for and downloading images

☐ Strongly Agree ☐ Agree ☐ Neither ☐ Disagree ☐ Strongly Disagree

## 3) QA test plan

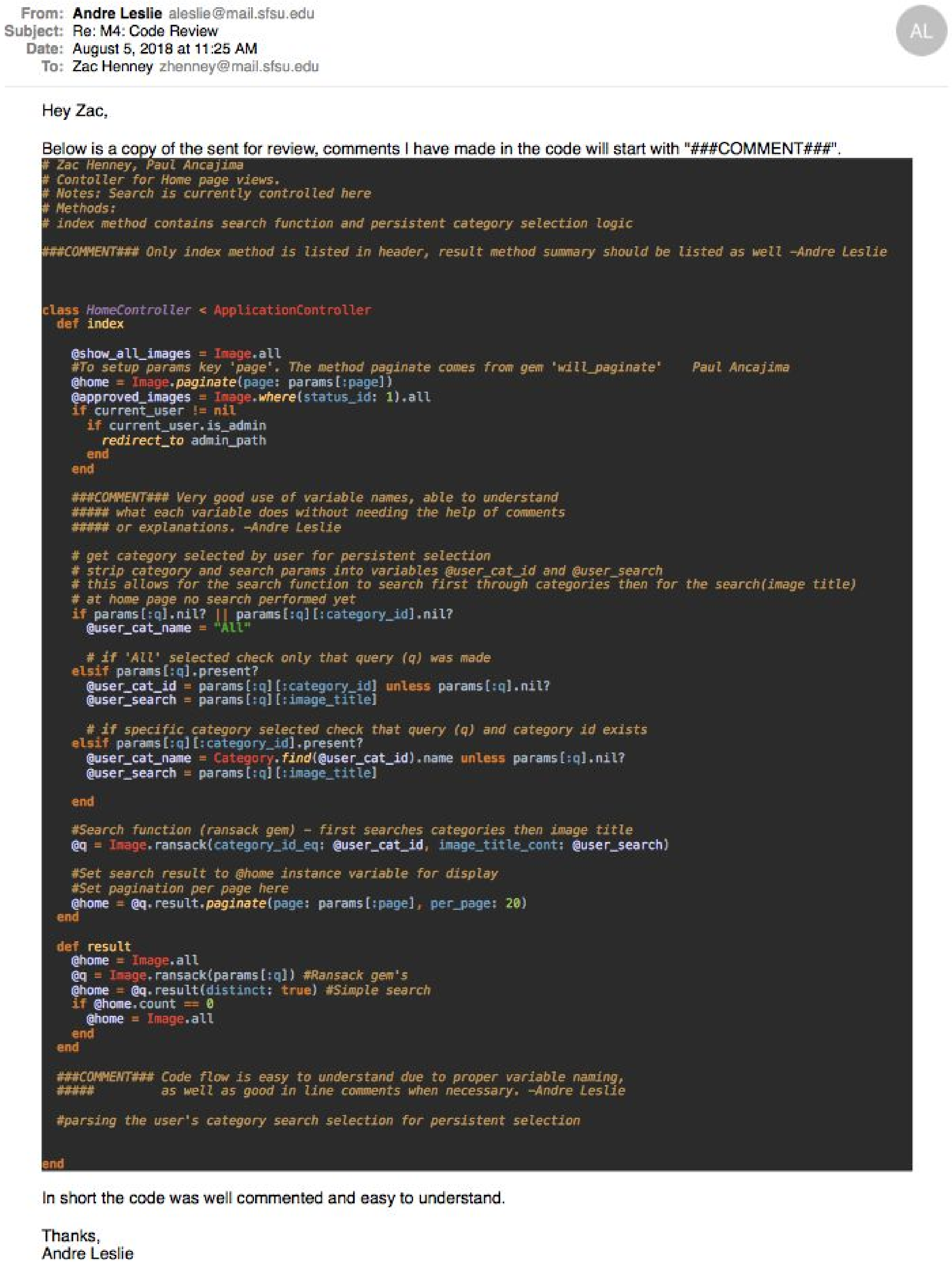
**Test Objectives:**
The following tests are conducted to ensure complete functionality of our website's search and display functions for stored images. Proper error messages should be displayed if users provide illegal inputs.

**Hardware and Software Setup:**
*Hardware*: A computer or laptop with MacOS or Windows and internet connection
*Deployment*: Amazon Web Services
*Web Browsers:* Chrome (v66.0.3359 & v67.0.3396) and Firefox (v60.0.1 & v60.0.2).
*Open-Source:* ImageMagick and Elasticsearch installed (if running locally)

**Features to be Tested:**
- Image searching through database using categories (optional text field)
- Rerouting to suggested searches on non existing queries.
- Error handling with illegal inputs

| Test # | Title | Description | Input | Expected Output | Pass/Fail |
|---|---|---|---|---|---|
| 1 | Image search with no additional text | Select an image category and submit the search | **Category**: 'Cars'' **Text Field:** *NULL* | A populated result page of images with all images of category: Cars | PASS |
| 2 | Image search with no existing results in database | Narrow search query using optional text field. | **Category:** 'Animals' **Text Field:** 'Purple dyed Alaskan Bull Worm' | A result page with no retrieved images. Display error message and suggest sample searches. | PASS |
| 3 | Image search with improper text | A search query with an illegal | **Text Field:** '-1' | Error dialogue on search bar, asking for a valid input | PASS |

## 4) Code Review



**From:** **Andre Leslie** aleslie@mail.sfsu.edu
**Subject:** Re: M4: Code Review
**Date:** August 5, 2018 at 11:25 AM
**To:** Zac Henney zhenney@mail.sfsu.edu

AL

Hey Zac,

Below is a copy of the sent for review, comments I have made in the code will start with "###COMMENT###".

```ruby
# Zac Henney, Paul Ancajima
# Contoller for Home page views.
# Notes: Search is currently controlled here
# Methods:
# index method contains search function and persistent category selection logic

###COMMENT### Only index method is listed in header, result method summary should be listed as well -Andre Leslie



class HomeController < ApplicationController
  def index

    @show_all_images = Image.all
    #To setup params key 'page'. The method paginate comes from gem 'will_paginate'    Paul Ancajima
    @home = Image.paginate(page: params[:page])
    @approved_images = Image.where(status_id: 1).all
    if current_user != nil
      if current_user.is_admin
        redirect_to admin_path
      end
    end

    ###COMMENT### Very good use of variable names, able to understand
    ##### what each variable does without needing the help of comments
    ##### or explanations. -Andre Leslie

    # get category selected by user for persistent selection
    # strip category and search params into variables @user_cat_id and @user_search
    # this allows for the search function to search first through categories then for the search(image title)
    # at home page no search performed yet
    if params[:q].nil? || params[:q][:category_id].nil?
      @user_cat_name = "All"

      # if 'All' selected check only that query (q) was made
    elsif params[:q].present?
      @user_cat_id = params[:q][:category_id] unless params[:q].nil?
      @user_search = params[:q][:image_title]

      # if specific category selected check that query (q) and category id exists
    elsif params[:q][:category_id].present?
      @user_cat_name = Category.find(@user_cat_id).name unless params[:q].nil?
      @user_search = params[:q][:image_title]

    end

    #Search function (ransack gem) - first searches categories then image title
    @q = Image.ransack(category_id_eq: @user_cat_id, image_title_cont: @user_search)

    #Set search result to @home instance variable for display
    #Set pagination per page here
    @home = @q.result.paginate(page: params[:page], per_page: 20)
  end

  def result
    @home = Image.all
    @q = Image.ransack(params[:q]) #Ransack gem's
    @home = @q.result(distinct: true) #Simple search
    if @home.count == 0
      @home = Image.all
    end
  end

  ###COMMENT### Code flow is easy to understand due to proper variable naming,
  #####         as well as good in line comments when necessary. -Andre Leslie

  #parsing the user's category search selection for persistent selection


end
```

In short the code was well commented and easy to understand.

Thanks,
Andre Leslie

**5) Self-check on best practices for security**

**Assets Protected:**
- Passwords
- Name
- Emails
- Images

**Password Encryption:**
- Passwords are encrypted by the clearance gem using BCrypt algorithm, which creates a hash that is stored in the database.

**Validations:**
1. Email cannot be left blank, must be unique, and must be in format (example@domain.com). This is enforced by clearance gem.

2. Password cannot be left blank and must be over 7 characters. This is also enforced by clearance gem.

3. Password must match password confirmation. Added :password_confirmation to user_params in user_controller.rb. Added following lines to models/user.rb
attr_accessor :password_confirmation
validates_confirmation_of :password, :message => "Password does not match."

4. Following validations are pertaining to image files:
    a. Validates that upload types are correct is done by following code:

```
    validate :correct_uploads_type

  def correct_uploads_type
    if uploads.attached? == false
    errors.add(:uploads, 'Must have image files')
     end
    uploads.each do |u|
    if !u.content_type.in?(%('image/jpeg image/png image/gif'))
                errors.add(:uploads, 'Must be jpeg or png file')
    end
     end
  End
```

b. The following code validates that the length of the description is between 5 and 40

```
 validates_length_of :description, :minimum => 5, :maximum => 40,
:allow_blank => true
```

5. Each image is associated with an user, the following code validates that:
validates_associated :images

6. The search bar validates that input values are either letters or numbers. The code is:

```
<%= f.search_field :image_title, :value => @user_search,
placeholder: 'Search for an image. (i.e. Animals, Cars, Food,
Nature, People, Travel)',
class: 'form-control col-8', pattern: '^[a-zA-Z0-9 ]+',
oninvalid: 'setCustomValidity("Please enter valid search")',
oninput: "setCustomValidity('')" %>
```

**6) Self-check: Adherence to original Non-functional specs**

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO). - ON TRACK
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. - ON TRACK
3. Data shall be stored in the team's chosen database technology on the team's deployment server. ON TRACK
4. No more than 50 concurrent users shall be accessing the application at any time.
5. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. ON TRACK
6. The language used shall be English. DONE
7. Application shall be very easy to use and intuitive. DONE
8. Application shall render well on mobile devices (UI shall be responsive). DONE
9. Google analytics shall be added. DONE
10. No e-mail clients shall be allowed. DONE
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. DONE
12. Site security: basic best practices shall be applied (as covered in the class). DONE
13. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. ON TRACK
14. The website shall <u>prominently</u> display the following <u>exact</u> text on all pages *"SFSU Software Engineering Project CSC 648-848, Summer 2018. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application). - DONE