SW Engineering CSC648/848 Summer 2018

# Hat Box Photos

## **Team 5 Amelia Rawlings (**[**Amelia.Rawlings@gmail.com**](mailto:Amelia.Rawlings@gmail.com)**)**

## **Front-End Arnold Jiadong Yu Dalu Li Betty Reaney**

## **Back-End Keawa Rozet U Keong Cheong (Ray) Albert Du**

# Milestone 4

|  |  |  |
| --- | --- | --- |
| Revision Table | | |
| 8/6/2018 | 1.0 | Initial document |
|  |  |  |

# 

**Product Summary**

Welcome, everyone, to the beta launch of HatBoxPhotos. We are delighted to offer you the opportunity to be one of the first to see our brand new free stock photo website. Features to look forward to include:

1. **Fast Search**

Need a high quality image for school, work, or just because? You can quickly find the perfect image for your needs at HatBoxPhotos.

1. **Uploading images**

Share your special moments with other users through our simple image uploading process!

1. **Admin moderation**

HatBoxPhotos enforces a family-friendly image policy. Our admin will either approve or reject uploaded images for live viewing within 24 hours.

Please visit us at <https://hatboxphotos.com/>. We look forward to your feedback.

**Usability Test Plan**

**Test Objective**

This usability test will rate the effectiveness, efficiency, and satisfaction of the search function. The search function is the main function of our application. Therefore, we must ensure that users are able to find specific photos with ease. Things to consider are search results design, search bar and search button placement.

Our effectiveness test will measure task completion and count errors per task. Next, our efficiency test will measure the number of clicks, number of screens, and time to complete the task. Finally, our satisfaction test will measure user satisfaction using a Lickert Scale questionnaire.

**Test Plan**

Intended User: Student needs pictures for Biology class.

Tasks to be accomplished:

1. Search for all pictures in the “Animal” category.
2. Search and download an image of tree from starting point.
3. Search and download an image of lion from starting point.
4. Search and download an image of pizza from starting point.

Starting Point: Home Page (<https://hatboxphotos.com/>)

System Setup 1: (Desktop)

Processor: Intel Core i7-6700k 4.00GHz

Memory: 32.0GB

System Type: Windows 10 64bit

Browser: Firefox Quantum 59.01 (64bit)

System Setup 2: (Laptop)

Processor: 1.8 GHz Intel Core i5

Memory: 4GB

System Type: macOS High Sierra

Browser: Google Chrome Version 67.0.3396.99

**Effectiveness Test:**

Desired time: less than 30 seconds.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test/Use Case** | **% Completed** | **Errors** | **Comments** |
| Search All animal category pictures | 100% | None | Search result showed that the user is seeing all animal pictures. |
| Search/Download a tree picture | 80% | User search city category. | Search result showed that there is 0 result found for tree in city category. |
| Search/Download a lion picture | 90% | User search nature category. | Search result showed that there is 0 result found for lion in nature category. |
| Search/Download a raccoon picture | 100% | None | Search result showed that there is 1 result found for raccoon in all pictures. |

**Efficiency Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test/Use case** | **# of clicks** | **# of screens** | **Time** |
| Search All animal category pictures | 3 | 3 | 15 seconds |
| Search/Download a tree picture | 5 | 5 | 20 seconds |
| Search/Download a lion picture | 4 | 5 | 20 seconds |
| Search/Download a raccoon picture | 5 | 4 | 25 seconds |

**Satisfaction Test**

Fill in the questions below according to your experience on search function.

1. I found the search bar is search to use.

( ) ( X ) ( ) ( ) ( )

Strongly Agree Agree Neutral Disagree Strongly Disagree

2. The search results page was clear.

( ) ( ) ( X ) ( ) ( )

Strongly Agree Agree Neutral Disagree Strongly Disagree

3. I would be able to find any picture with search function.

( ) ( ) ( X ) ( ) ( )

Strongly Agree Agree Neutral Disagree Strongly Disagree

**QA Test Plan**

**Test Objective**

The QA objective is to test search and delete user function. Searching by image category, by image name, as well as delete an existing user shall be performed.

System setup

Processor: Intel Core i7-6700k 4.00GHz

Memory: 32.0GB

System Type: Windows 10 64bit

Browser: Firefox Quantum 59.01 (64bit), Google Chrome 67.0.3396.99 (64bit)

Features to be tested.

Search an image by category.

Search an image by name/description

Delete a user

Test Cases

1. Search an image by category.
2. From the home page search bar, select the ‘Nature’ category and have nothing in the search input. Click on ‘Search’ button. The result shall return 4 thumbnails.
3. Search an image by name/description
4. From the home page search bar, select the ‘All’ category and input “Man” in the search input. Click on ‘Search’ button. The result shall return 4 thumbnails.
5. Delete a user
6. Click on Admin button on Navigation bar. Then click on ‘Manage User’, select ‘[test5@gmail.com](mailto:test5@gmail.com)’ and press delete button. Comfirm by click delete button again on pop-up. The page will refresh and user with email ‘test5@gmail.com’ will not displayed.

Test Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Test Title | Test Description | Test Input | Expected Output | Firefox | Chrome |
| 1 | Search an image by category. | Test category search functionality | “Nature” without input on search | 4 thumbnails | Pass | Pass |
| 2 | Search an image by name/description | Test input search functionality | Input “Man” | 4 thumbnails | Pass | Pass |
| 3 | Delete a user | Test delete user functionality | None | Delete user successfully | Pass | Pass |

**Code Review**

**Coding Style**

* JSDoc documentation is used for class headers and function headers
* Inline comments are used to document non-headers

**Emails:**

**Chosen Code for Peer Review Email:**

Albert Du <adu1@mail.sfsu.edu>

To:Keawa.Rozet@yahoo.com

Hello Keawa,

I finished writing code for the search route in UploadedImageRouter.js. Could you check it out?

router.get("/search-thumbnails", function(req, res) {

   var noMatch = false;

   // Handles if the user enters something into the search bar

   if(req.body.search) {

       const regex = new RegExp(escapeRegex(req.body.search), 'gi');

       // Get all images and image thumbnails from image db

       // based on search and category

       Images.find({'image\_category': req.body.image\_category, tags: regex}, function(err, allImages) {

          thumbnails = allImages.map(function (allImages) {return allImages.image\_thumbnail\_path;});

          if(err) {

            console.log(err);

          } else {

          // if no images were found, return all images in selected category

          if (allImages.length < 1) {

              noMatch = true;

              // Get all images and thumbnails from db in only the selected category

              Images.find({'image\_category': req.body.image\_category}, function(err, restOfImages){

                thumbnails = restOfImages.map(function (restOfImages)        {return restOfImages.image\_thumbnail\_path;});

                 if(err){

                   console.log(err);

                 } else {

                   res.json({images: restOfImages, thumbnails: thumbnails, noMatch: noMatch});

                 }

            });

Thanks

Albert Du

**Peer Feedback Email:**

Keawa Rozet <keawa.rozet@yahoo.com>

To:Albert Du

Hey Albert!

The code looks really solid and is almost complete. I was testing it out using Postman and I was able to figure out some bugs that were preventing it from connecting.

I commented in the areas with "//**FIX ME** - " where you need to take a look.

router.get("/search-thumbnails", function(req, res) {

    var noMatch = false;

// Handles if the user enters something into the search bar

//**FIX ME** – don’t use body because that is for x-www-form-urlencoded

//instead use req.query since we are using a Form

    if(req.body.search) {

        //**FIX ME** – use req.query

        const regex = new RegExp(escapeRegex(req.body.search), 'gi');

        // Get all images and image thumbnails from image db

        // based on search and category

        //**FIX ME** – include finding by status: 1, that way we return only accepted images

        // use req.query

        Images.find({'image\_category': req.body.image\_category, tags: regex}, function(err, allImages) {

          thumbnails = allImages.map(function (allImages) {return allImages.image\_thumbnail\_path;});

          if(err) {

            console.log(err);

          } else {

            // if no images were found, return all images in selected category

            if (allImages.length < 1) {

              noMatch = true;

              // Get all images and thumbnails from db in only the selected category

              //**FIX ME** – req.query, status: 1

              Images.find({'image\_category': req.body.image\_category}, function(err, restOfImages){

                thumbnails = restOfImages.map(function (restOfImages) {return restOfImages.image\_thumbnail\_path;});

                  if(err){

                    console.log(err);

                  } else {

                    res.json({images: restOfImages, thumbnails: thumbnails, noMatch: noMatch});

                  }

             });

Send me what you have when you fix these areas, thanks Albert!

Best,

Keawa Rozet

**Self-check On Best Practices for Security**

* Major assets being protected include email addresses, usernames, and passwords
* Passwords in the database have been encrypted using the bcrypt hashing function
* Input validation is present in login, registration and the search bar
  + The Mongoose-unique-validator plugin prevents duplicate registration
  + Mongoose-unique-validator combined with the uniqueCaseInsensitive Mongoose schema validator perform case-sensitive validation for emails and usernames during login & registration
  + Username and password length is validated during registration by the Mongoose schema validators minlength and maxlength
  + Search bar only allows characters and spaces

**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). (**DONE**)
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**)

1. Data shall be stored in the team’s chosen database technology on the team’s deployment server. (**DONE**)
2. No more than 50 concurrent users shall be accessing the application at any time. (**DONE**)
3. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. (**DONE**)
4. The language used shall be English. (**DONE**)
5. Application shall be very easy to use and intuitive. (**ISSUES**)

* We are struggling to connect front-end and back-end. We cannot test fully on localhost because of "Access-Control-Allow-Origin" problem. Have tried many ways to solve it but failed.
* We’re having trouble with sign in and sign up pages.

1. Application shall render well on mobile devices (UI shall be responsive). (**DONE**)
2. Google analytics shall be added. (**ON TRACK**)
3. No e-mail clients shall be allowed. (**DONE**)
4. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. (**DONE**)
5. Site security: basic best practices shall be applied (as covered in the class) (**DONE**)
6. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development (**DONE**)
7. The website shall prominently display the following exact 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**)