

# SW Engineering CSC 648/848 Fall 2018

## Section 01

**GatorTrader**



### **Team # 02 (Local)**

Gayatri Pise, **Team Lead** | [gpise@mail.sfsu.edu](mailto:gpise@mail.sfsu.edu)

Peter Malolepszy

Divam Shah

Dylan Wright

Moses Martinez

Charul Ramtekkar

Regine Manuel

### **“Milestone 4”**

Date: 12/2/2018

**History Table**

Version	Date	Owner(s)	Description
1.0	12/02/18	Gayatri	Submitted for Review

## 1) PRODUCT SUMMARY

Name of our product - **GatorTrader**

Below is the list of ALL major committed P1 functions for GatorTrader.

- Home Page
- Search (with search text validation)
- Search results
- Item details
- Login and Registration
- Item posting
- Message from buyer to seller
- Seller dashboard (for seller items and messages)
- Admin dashboard (back up is to use DB native admin tool but must show ability to approve posted images)

GatorTrader is a buying and selling website specially designed for San Francisco State University students. Gator Trader will be marketed as a website created by SFSU students for SFSU students. This website would serve as a platform for students to sell their unused items, old class materials, etc. as well as buy essential items at a cheaper rate compared to the market price. It will provide an affordable and safe marketplace for students to exchange their belongings.

Moreover, this website builds direct communication between buyer and seller. This unique approach will eliminate long wait times and no shipping costs would be involved. The basic attraction of our website is its simplicity and user-friendliness. Also, the good design of GatorTrader can be eventually used as a template for other universities looking to design their own buy and sell website.

URL to our website: <https://se-csc648-fa18-section1-team02.herokuapp.com>

## 2) USABILITY TEST PLAN

### Test Objective:

The goal of usability testing is to identify any usability problems, collect quantitative data on participants' performance as well as determine user satisfaction for the website. If a website is difficult to navigate or doesn't clearly articulate a purpose, users will leave it. Ideally, usability testing is done before a site launches, and then many times after. Currently, our website is in its development and testing phase. Hence, usability testing will help us to find out the flaws or weakness in the usability flow at an early stage. User comments or reviews will help us to judge and improve our website in terms of usability.

We will be conducting a usability test for posting an item on our website. This will allow the user to navigate through a series of usage flows touching on the main components of the website. It will focus on logging in with user details on GatorTrader, posting an item for sale and then reviewing the posted item in seller dashboard. Also, test users will be asked to fill a survey consisting of a Likert scale. Success factors would then be determined based on feedback from the test users.

### Test plan:

#### Purpose:

The purpose of this test is to identify any usability problems within our website.

#### System Setup:

A laptop with an internet connection is required and access to any web browsers such as Google Chrome, Mozilla Firefox, Safari, etc. We will be testing our website on two latest versions of all major browsers in order to check the consistency while using our website on different browsers.

#### Starting point:

The starting point will be the homepage of our website located on below URL.

URL: <https://se-csc648-fa18-section1-team02.herokuapp.com/>

#### Task Description:

There are different tasks that the user needs to go through in order to successfully post an item and then view the posted item. This consists of opening the website, logging in with credentials, posting an item for sale and then checking the posted item in their respective seller dashboard and view item details.

Task	Machine State	Successful Completion criteria	Benchmark
Open GatorTrader website on browser	URL just loaded	Homepage of the website displayed	Completed in 30 sec
Login on Gator Trader with credentials	Login Page loaded	Login successful and search page displayed consisting item list	Completed in 30 sec

Post an item for sale	Empty post form loaded	Filled out all the required fields and posted the item successfully.	Completed in 30 sec
Checking the posted item in My Dashboard and View Details of the item	My Dashboard page loaded	Located the posted item in My Dashboard and viewed item details successfully.	Completed in 30 sec

### Completion Criteria:

User has successfully completed the following tasks: -

1. Launched the website URL on browser.
2. Found the Login button and Logged in successfully.
3. Found the Post button and posted an item successfully.
4. Found My Dashboard button and could see the item posted in My Dashboard.
5. Found the View Details button for the posted item and could see the details of the item successfully.

### User Satisfaction Questionnaire

Website user survey. Select one option for below questions: -

Task	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The GUI is simple and user friendly.					
It was easy to post an item for sale on Gator Trader.					
It was easy to locate the posted item in My Dashboard after posting.					
The item details were displayed correctly after clicking on View Details in My Dashboard.					

### General Comments/Feedback: -

---



---



---

### 3) QA TEST PLAN

#### Test Objective:

Quality assurance (QA) is a systematic process of verifying whether a software meets required specifications and customer expectations. The primary goal of QA is to find defects and resolve defects prior to product release. A quality assurance system increases customer confidence and the company's credibility, by enabling a company to better compete with others.

#### HW and SW setup:

- A laptop with an internet connection is required and access to any web browsers such as Google Chrome, Mozilla Firefox, Safari, etc.
- Browse to the website using the following URL. This will direct to the homepage of our website.
- URL: <https://se-csc648-fa18-section1-team02.herokuapp.com/>
- Unregistered users need to sign up using the Sign-Up clickable text located at the top right corner of the homepage.
- After successful registration, User should login by clicking on the Login text located at the top right corner of the home page.
- Enter Email and Password and Click Login Button.
- User should be logged in successfully and the search page should be displayed.
- Once logged in, test the following cases for posting an item.

**Feature to be tested:** The post an item feature for registered and logged in users.

#### Tasks:

Test ID	User Type	Test Description	Test Input	Expected Correct Output	Test Results on Chrome	Test Results on Firefox
1.	Registered and Logged In User	Open the following URL in browser.  <a href="https://se-csc648-fa18-section1-team02.herokuapp.com/post">https://se-csc648-fa18-section1-team02.herokuapp.com/post</a>  Users is logged in and shall post an item for sale.	Click on Sell An Item text located at the top right of the page.	If user is logged in, Post Item page should be loaded successfully consisting of form fields as below. Title, Category, Price Description, Image upload button.	Pass	Pass

				Post and Cancel button.		
2.		<p>Open the following URL in browser.</p> <p><a href="https://se-csc648-fa18-section1-team02.herokuapp.com/post">https://se-csc648-fa18-section1-team02.herokuapp.com/post</a></p> <p>Users is logged in and shall post an item for sale.</p>	<p>Click on Sell An Item text located at the top right of the page.</p> <p>Enter details as below: -</p> <p>Title - Cap Category - Clothes Price - \$5 Description - Unused cap for sale at cheaper price Image - Upload any cap image. Click - Post</p>	<p>Item should be posted successfully displaying item details such as Item Image, Item ID, Item Name, Price, Category Description and Contact Seller Button. The Item details displayed should match the entered item details.</p>	Pass	Pass
3.		<p>Open the following URL in browser.</p> <p><a href="https://se-csc648-fa18-section1-team02.herokuapp.com/post">https://se-csc648-fa18-section1-team02.herokuapp.com/post</a></p> <p>Users is logged in and shall post an item for sale.</p>	<p>Click on Sell An Item text located at the top right of the page.</p> <p>Enter details as below: -</p> <p>Title - Category - Books Price - \$100 Description - Cracking the coding Interview book Image - Upload the book image. Click - Post</p>	<p>Item should not be posted and a message 'Please fill out this field should be displayed for Title Field.</p>	Pass	Pass

4.		<p>Open the following URL in browser.</p> <p><a href="https://se-csc648-fa18-section1-team02.herokuapp.com/dashboard">https://se-csc648-fa18-section1-team02.herokuapp.com/dashboard</a></p> <p>Users is logged in and posted an item successfully. User shall check the posted item in My Dashboard.</p>	<p>Click on My Dashboard located at the top right of the page</p>	<p>My dashboard page should display the recently posted item in Item List. My Dashboard page should contain following fields for the posted item:-</p> <p>Thumbnail Title Price Date Description Status Messages</p>	Pass	Pass
----	--	---	---	--	------	------



## 4) CODE REVIEW

DS

Divam Hiren Shah  
Fri 11/30/2018 9:22 PM  
To: @ Dylan Travis Wright; @ Gayatri Pise

👤

🔄

Reply all

▼

Hello Dylan,

Hope things are going smooth at your end, as discussed I have pasted the following code of posting an item for sale, for reviewing. Kindly please let me know, if there are any changes required.

```
/**
 * This script is responsible for allowing the user to post images for sell an item
 * functionality. The user can upload a single file from there system. The accepted
 * file formats are jpeg or png.
 *
 * Authors: Dylan Wright, Divam Shah
 */
var express = require("express");
var router = express.Router();
var db = require('./db');
var hash = require('object-hash');

//formidable is used to read/parse incoming form from the web page.
var formidable = require('formidable');
//This npm helps to resize, crop and render image in thumbnails.
var im = require('node-imagemagic');
//Helps to move the file from one path to another.
var fs_extra = require('fs-extra');
//Hosting service, to host images of item.
var imgur = require('imgur');
//Used to render a page with associated handlebars file.
var render = require('../models/loginCheck').renderUserAndCategory;

var filename,thumbnail;
var fields,id;
var response, request;
let resizeWidth = 200;

/**
 * This function is responsible for cropping the image.
 * The image gets cropped with minimum dimension, as its width and height
 * @param err This throws an error if the page fails to render.
 * @param features Detects the features of image. Eg: Width and Height
 * @return calls a cropImageCallback function.
 */
function cropImageCallback({
```

DW

Dylan Travis Wright  
Today, 8:38 AM  
Divam Hiren Shah; Gayatri Pise

🔄

Reply all

▼

Divam,

Fantastic! The code you've sent looks great! I've reviewed it with great ease due to your generous use of comments throughout.

I've noticed a few points in the code to consider changing:

router.post:

- In the callback function, consider removing the 'next' parameter. It's unused, and JavaScript doesn't explicitly check that all parameters passed to an anonymous function are declared.
- The 'return' line in the header should specify that the function returns void.
- Consider commenting the purpose of 'response = res', 'request = req', and 'fields = field' (e.g. That these variables are needed for the above callback functions).
- Image Magic supports a lot more formats than just jpeg and png. Maybe remove the conditional checking since we are now hosting images to imgur, and naming conventions are automatic. And using the built in error checking of the Image Magic functions might lead to more predictable outcomes.
- If the file type conditional statement is kept, within the else statement, render a more user-friendly error message, instead of the res.write(). Also, add a 'return;' so the rest of the function won't be called after the error.
- Since we moved hosting to imgur, we don't need to move the images to a new folder within our app. This also means the hashed Date() file name is no longer necessary. However, the thumbnail still needs a unique name, as many users may be processing thumbnails at the same time. Consider removing this thumbnail file when finished to save space, and so our file system doesn't become overloaded with user thumbnails that are hosted elsewhere.

identifySizeCallback:

- The '@param err' should say that it is the error returned if identifying the image fails.
- The '@return' should specify that the function returns void

cropImageCallback:

- The '@param err' should say that it is the error returned if cropping the image fails.
- The '@return' should specify that the function returns void
- 'stdout' and 'stderr' can be safely removed, as they are not used in the final callback function.

resizeImageCallback:

- The '@param err' should say that it is the error returned if cropping the image fails.
- The '@return' should specify that the function returns void
- 'stdout' and 'stderr' can be safely removed, as they aren't used.
- Instead of removing all the apostrophes from the title and description (since they cause errors inserting into the database), replacing with '&apos;' would be safe, and still preserve the original intent of the user. Note that the handlebars file would need to be updated to use three brackets instead of two (e.g. {{{ }} instead of {{ }}) when displaying the data.
- The final catch should render an error page for the user.

If you have any questions about anything please feel free to reach out to me, and I'll get back to you as soon as I can. And as always, I am open to performing any other code reviews you need in the future!

Hope the rest of your day is fantastic!

Respectfully,  
Dylan Wright

# CSC648- SOFTWARE ENGINEERING - CODE REVIEW

## - POST AN ITEM

Dylan Travis Wright

Today, 8:38 AM

Divam,

Fantastic! The code you've sent looks great! I've reviewed it with great ease due to your generous use of comments throughout.

I've noticed a few points in the code to consider changing:

### **router.post:**

- In the callback function, consider removing the 'next' parameter. It's unused, and JavaScript doesn't explicitly check that all parameters passed to an anonymous function are declared.
- The @return line in the header should specify that the function returns void.
- Consider commenting the purpose of 'response = res', 'request = req', and 'fields = field' (e.g. That these variables are needed for the above callback functions).
- Image Magick supports a lot more formats than just jpeg and png. Maybe remove the conditional checking since we are now hosting images to imgur, and naming conventions are automatic. And using the built-in error checking of the Image Magick functions might lead to more predictable outcomes.
- If the file type conditional statement is kept, within the else statement, render a more user-friendly error message, instead of the res.write(). Also, add a 'return;' so the rest of the function won't be called after the error.
- Since we moved hosting to imgur, we don't need to move the images to a new folder within our app. This also means the hashed Date () file name is no longer necessary. However, the thumbnail still needs a unique name, as many users may be processing thumbnails at the same time. Consider removing this thumbnail file when finished to save space, and so our file system doesn't become overloaded with user thumbnails that are hosted elsewhere.

### **identifySizeCallback:**

- The '@param err' should say that it is the error returned if identifying the image fails.
- The '@return' should specify that the function returns void

### **cropImageCallback:**

- The '@param err' should say that it is the error returned if cropping the image fails.
- The '@return' should specify that the function returns void
- 'stdout' and 'stderr' can be safely removed, as they are not used in the final callback function.

### **resizeImageCallback:**

- The '@param err' should say that it is the error returned if cropping the image fails.
- The '@return' should specify that the function returns void
- 'stdout' and 'stderr' can be safely removed, as they aren't used.

- Instead of removing all the apostrophes from the title and description (since they cause errors inserting into the database), replacing with ' &apos;' would be safe, and still preserve the original intent of the user. Note that the handlebars file would need to be updated to use three brackets instead of two (e.g. {{{ }}} instead of {{ }} ) when displaying the data.
- The final catch should render an error page for the user.

If you have any questions about anything please feel free to reach out to me, and I'll get back to you as soon as I can. And as always, I am open to performing any other code reviews you need in the future!

Hope the rest of your day is fantastic!

Respectfully,

Dylan Wright

**Divam Hiren Shah**

**Reply all**

**Fri 11/30, 9:22 PM**

**Dylan Travis Wright;**

**Gayatri Pise**

Hello Dylan,

Hope things are going smooth at your end, as discussed, I have pasted the following code of posting an item for sale, for reviewing. Kindly please let me know, if there are any changes required.

/\*\*

**\* This script is responsible for allowing the user to post images for sell an item**

**\* functionality. The user can upload a single file from there system. The accepted**

**\* file formats are jpeg or png.**

**\***

**\* Authors: Dylan Wright, Divam Shah**

**\*/**

var express = require("express");

var router = express.Router();

var db = require('../db');

var hash = require('object-hash');

```

//formidable is used to read/parse incoming form from the web page.
var formidable = require('formidable');

//This npm helps to resize, crop and render image in thumbnails.
var im = require('node-imagemagick');

//Helps to move the file from one path to another.
var fs_extra = require('fs-extra');

//Hosting service, to host images of item.
var imgur = require('imgur');

//Used to render a page with associated handlebars file.
var render = require('../models/loginCheck').renderUserAndCategory;

var filename,thumbnail;
var fields,id;
var response, request;
let resizeWidth = 200;

/**
 * This function is responsible for cropping the image.
 * The image gets cropped with minimum dimension, as its width and height
 * @param err This throws an error if the page fails to render.
 * @param features Detects the features of image. Eg: Width and Height
 * @return calls a cropImageCallback function.
 */
var identifySizeCallback = function (err, features) {
  if (err) throw err;

  // Get height and width
  let width = features.width;
  let height = features.height;

  // Get smallest dimension
  let smallestDimension = Math.min(height, width);

  // Crop width from center

```

```

console.log("Cropping...")

im.crop({
  srcPath: './public/images/user_images/' + filename,
  dstPath: './public/images/user_images/' + thumbname,
  width: smallestDimension,
  height: smallestDimension,
  quality: 1,
  gravity: "Center"
}, cropImageCallback);
}

/**
 * This function is responsible for resizing image after getting cropped.
 * @param err throws error if page fails to render.
 * @param stdout Standard output
 * @param stderr Standard error
 * @return calls resizeImageCallback function.
 */

var cropImageCallback = function(err, stdout, stderr){
  if (err) throw err;
  console.log('Cropped');
  console.log('Resizing');
  // Resize image
  im.resize({
    srcPath: './public/images/user_images/' + thumbname,
    dstPath: './public/images/user_images/' + thumbname,
    width: resizeWidth
  }, resizeImageCallback(err, stdout, stderr));
}

/**

```

- \* This function is responsible for uploading image on imgur hosting service.
- \* It uploads the original file as well as thumbnail version of it.
- \* It also populates the image database through pg-promise.
- \*
- \* @param err throws error if page fails to render.
- \* @param stdout Standard output
- \* @param stderr Standard error
- \* @return calls renderUserAndCategory through render with corresponding post handlebars
- \* and css.
- \*/

```

var resizeImageCallback = function(err, stdout, stderr){
  if (err) throw err;
  console.log('Resized');
  //Uploading of original file
  imgur.uploadFile('./public/images/user_images/' + filename)
    //returns a json file.
    .then( (filejson) => {
      //Uploading of thumbnail version of file.
      imgur.uploadFile('./public/images/user_images/' + thumbname)
        //returns a json file of thumbnail version.
        .then( thumbjson => {
          console.log(filejson.data.link);
          console.log(thumbjson.data.link);
          console.log(fields.category)
          //Removes all the '/' from incoming form.
          let title = fields.title.replace(/'/g, "");
          let description = fields.description.replace(/'/g, "");
          // Insert new item into database
          db.any(
            `INSERT INTO item(
              item_title,

```

```

        item_description,
        item_price,
        item_status,
        user_id,
        category_id,
        item_image,
        item_availability,
        item_image_thumbnail
    )
VALUES
(
    " + title + `,
    " + description + `,
    " + fields.price + `,
    'Pending',
    ` + id + `,
    ` + fields.category + `,
    " + filejson.data.link + `,
    true,
    " + thumbjson.data.link + `
)
RETURNING item_id`
)

// Query returns with data called 'myData'
.then(function(myData) {
    response.redirect('/item/' + myData[0].item_id);
})

//If there is an error, it is caught here.
.catch(function(error) {
    render(request, response, 'post', 'POST PAGE', 'post');
});
})

```

```

    })
    .catch(function (err) {
        console.error(err.message);
    });
}

/**
 * This is the main function route, which parses the incoming form and then performs
 * the above mentioned functions and callbacks. It also moves the image file from original
 * file directory to a new directory.
 * @param req it is a request to the http
 * @param res it is a response to the http
 * @returns newpath of the file
 */
router.post('/', function(req, res, next) {
    id = req.session.user_id;
    response = res;
    request = req;
    // Get incoming form
    var form = new formidable.IncomingForm();
    // Parse incoming form
    form.parse(req, function (err, field, files) {
        fields = field

        // Hash the time now for uncollidable filenames
        var name = hash.shal(Date.now());

        // If the file is a jpeg...
        if (files.image_file.type === 'image/jpeg') {
            // Append the correct suffix
            filename = name + '.jpg';
            thumbname = name + '_thumb.jpg';
        }
        // If the file is a png...

```



```
else if (files.image_file.type === 'image/png') {  
    // Append the correct suffix  
    filename = name + '.png';  
    thumbname = name + '_thumb.png';  
}  
else {  
    res.write("Not an image file");  
    res.end();  
}  
console.log(filename);  
// Get temp path of uploaded image  
var oldpath = files.image_file.path;  
// Specify new path for uploaded image  
var newpath = './public/images/user_images/' + filename;  
// Move and rename image file to public/images/  
fs_extra.move(oldpath, newpath, function (err) {  
    if (err) throw err;  
    // Read height and width  
    im.identify(newpath, identifySizeCallback);  
});  
});  
});  
module.exports = router;
```

Regards,

Divam Shah

Back End Team

## **5) SELF-CHECK ON BEST PRACTICES FOR SECURITY**

### **Major assets we are protecting:**

1. User Information - User Information is securely stored in DB and is visible only to the authorized administrator of the website.
2. Password Protection - Passwords are encrypted.
3. Image Uploads - Rename the file on upload to ensure the correct file extension, or to change the file permissions.
4. Watching out for SQL Injection - Search bar input validation of less than 40 alphanumeric characters has been taken care of.
5. Append a user agreement in the terms and conditions while signing up for our website - This will help to assure that all image uploads belong to user and any copyright issues will be the user responsibility.
6. The website shall prominently display the following text on all pages “SFSU-Fulda Software Engineering Project CSC 648-848, Fall 2018. For Demonstration only” in order to avoid conflicts with other buy and sell websites.

### **Confirm of the password in the DB**

Yes, the password is encrypted.

### **Confirm Input data validation**

- Search bar input validation - less than 40 characters.
- Search bar input validation - no special characters (alphanumeric only).
- Search bar input validation - empty search shows all the item list.
- Search bar output validation - provided valid search, provides some result.

## 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) - **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 - **DONE**
3. Selected application functions must render well on mobile devices - **ON TRACK**
4. Data shall be stored in the team's chosen database technology on the team's deployment server - **DONE**
5. No more than 50 concurrent users shall be accessing the application at any time - **ON TRACK**
6. Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users - **DONE**
7. The language used shall be English - **DONE**
8. Application shall be very easy to use and intuitive - **ON TRACK**
9. Google analytics shall be added - **ON TRACK**
10. No email 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 - **DONE**
14. The website shall prominently display the following exact text on all pages **"SFSU-Fulda Software Engineering Project CSC 648-848, Fall 2018. For Demonstration Only"** at the top of the WWW page. (Important so as to not confuse this with a real application). - **DONE**