SW Engineering CSC648/848 Spring 2020
SFSU Access
Team 02

Team lead: Yanrui Xu

Team lead email: yxu13@mail.sfsu.edu

Back end lead & Github Master: Kevin Luong

Front end lead: JunMinLi

Team members: Cody Xu, Aitor Elvira, David Lin

Milestone 4

05/18/2020

History

Date Submitted	Date Revised
05/18/2020	

1. Product summary

The name of our project is SFSU Access.

The major committed functions are:

Home page:

1. Search text box shall only allow up to 40 alphanumeric characters

2. Search and search results shall be responsive

Non-registered users:

1. All listings provided by registered users shall be made viewable by non-registered users.

2. Users shall be able to search and browse all items

3. Users shall be able to view item details after clicking an item

4. Users shall register before being able to purchase or download items.

5. Users shall be able to create an account to engage in activity on the service. Registration shall require an SFSU email address.

Registered users:

1. Users shall be able to do everything that Non-registered users can.

2. Posting items to the store shall be subject to Admin approval.

3. Users shall be able to log in on the website.

4. Users shall be able to send and receive messages.

5. Users shall be able to post items for sale or download.

6. Users shall be able to remove their posted items.

7. Users shall be able to view all posted items and messages

8. Users shall be able to message seller using in-site message prefilled with product info

Administrator:

1. The admin shall be able to approve or deny pending listings.

The unique point of our project is it is specially designed for SFSU students and faculties. By targeting a specific user demographic allows us to address the specific needs of the users better, and be more efficient in marketing. Users need to provide a valid SFSU ID number to register the website, and contents could be organized according to SFSU course numbers for students to look up easily.

Website URL: http://54.177.237.30/

2. Usability test plan

The major function being tested is upload/posting.

i. Test objectives:

We want to test the process and correct functionality of file uploading and posting. As SFSU Access is an e-commerce website, it is important for our users to be able to upload their files so that we can process their product and list it for other users to see. Should users encounter issues with our file uploading and posting process, they will not be able to create their listing and will be unable to continue with merchandising. Having our valued users turn away from our platform due to usability issues that we could resolve is completely unacceptable.

ii. Test background and setup:

- System setup
    - Any modern browser that can run Javascript, such as Google Chrome. In order to reduce potential interference by browser extensions or add-ons, we recommend launching the browser in Guest mode.
- Starting point
    - To start head over to http://54.177.237.30/. This is our homepage. We recommend signing in first. If you do not have an account, just click Sign Up on the top right. Next, click Post Item next to the search bar.
- Intended users
    - We intend for our users to generally be college students, with some faculty in the mix. Our users won't need to be technologically adept at understanding how our platform works, because we've streamlined our posting process.
- URL of system & measurements
    - We are testing our functionality at http://54.177.237.30/Postitem. We are measuring the ease of use and user satisfaction with using the function. A small likert scale questionnaire, posted below, will be used to gauge and record those measurements.

iii. Usability task description:

TASK	**DESCRIPTION**
Task	Post a new item
Machine state	Post item page is not filled

Successful completion criteria	Item is posted
Benchmark	Completed in 1 minute

Task: Please post the given example.jpg image to SFSU Access. You will start on our homepage at http://54.177.237.30/ while already logged in to a user account. Any extraneous details during the posting process is up to you. You have one minute to post the image.

We would measure effectiveness by determining how many users are able to complete the task on time and record any errors thrown from the form. We expect high effectiveness as our form has minimal restrictions for inputting the required data.

We would measure efficiency by determining how quickly the user completes the posting form. We expect the average time to post an item to not exceed 1 minute, though this may vary due to differing description word counts and time to search and upload files.

Likert scale:

- I found the form easy to navigate (check one)

__ Strongly agree __ Agree __ Neither agree nor disagree __ Disagree __ Strongly disagree

- I found browsing for my file to be ___ (check one)

__ Very easy __ Easy __ Neither easy nor hard __ Hard __ Very hard

- I had no issues with filling in the form (check one)

__ Strongly agree __ Agree __ Neither agree nor disagree __ Disagree __ Strongly disagree

3. QA test plan

Test objectives: Perform QA testing on the file upload/post process to ensure there are no issues. Posting should take no longer than one minute on average. Confirm that all character count limits are reasonable, and that the form is secure against abuse.

Setup: (Optional) Register or log in first. Go to http://54.177.237.30/Postitem to get started.

Features to be tested: Test the process of uploading/posting files. Test the limits of the upload form.

QA Test Plan Table:

Test #	Test Title	Description	Inputs	Expected correct output	Test Results
1	Uploading unsupported file type: .exe	To test the file type limit of our system, we will upload an .exe file.	Title: Image Magick Category: Other Price: 0 License: Copyrighted Description: “The Image Magick exe installs Image Magic 7.0.10-10-Q16-x64 dll.” File: ImageMagick-7.0.10-10-Q16-x64-dll.exe	The file should not be posted and the file upload field will specify it is an invalid file type.	Google Chrome Version 81.0.4044.138 - PASS Microsoft Edge 44.18362.449.0 - PASS
2	Upload a .jpg file with max length character limits	To test the form input character limit, we will upload a .jpg file with the title and description filling out the limit. The limit is currently 30	Title: Example of Sbox and encryption Category: Notes Price: 0 License: Free use & modification Description: “Many different	The file should be posted with a successful message. Upon reviewing the item in the dashboard or item detail page, all text entered	Google Chrome Version 81.0.4044.138 - PASS Microsoft Edge 44.18362.449

		characters for the title, and 500 characters for the description.	block ciphers use a special substitution called an ``S-box''. The AES also has these S-boxes, which it terms the ``SubBytes Transformation". S-boxes provide an invertible (reversible) transformation of segments of plaintext during encryption, with the reverse during decryption. With the AES it is a single simple function applied over and over again to each byte during stages of the encryption, returning a byte. For more info, visit: http://www.cs.utsa.edu/~wagner/laws/SBoxes.html" File: Untitled2.png	should be present.	.0 - PASS
3	Upload a .jpg file after resetting the form once	To test the form's reset integrity, we will upload a .jpg image and fill in all information as needed, then reset the entire form. Fill in the	Title: Image QA Category: Other Price: 0 License: Copyrighted Description: "For testing" File: wallpaper.jpg	The file should be posted with a successful message. All information should stay the same.	Google Chrome Version 81.0.4044.138 - PASS Microsoft Edge 44.18362.449.0 - PASS

		form again with different information and .jpg file and submit.	Title: Image QA2 Category: Video Price: 1 License: Free for Project Description: “For testing again” File: 1500x500.jpg		

4. Code Review

We are using proper variables and function names to help identify what they are for and what they do. We use comments for the beginning of each function, and in-line comments when a certain line of code may not be immediately clear of why it is there. We are using proper tabbing and line returns when necessary and to ensure lines of code do not extend too far beyond the screen (such as on GitHub).

```
useEffect (() => {
    axios.get('/api/search').then(response =>{setList(response.data)}).catch(error=>console.log(error));
    console.log("isLoggedin? "+ user_isloggedin); // we don't need this in the final release

    if(typeof cookies.post_item !== 'undefined'){
      setName(cookies.post_item.product_name);
      setCategory(cookies.post_item.product_category);
      setPrice(cookies.post_item.product_price);
      setLicense(cookies.post_item.product_license);
      setDescription(cookies.post_item.product_description);
    }
  },[]);

//Reset user input on post item form.
const resetForm = ()=>{
  document.getElementById("itemForm").reset();
  setFileName("Upload File here...");
}

//this function is used to post cookies item
const postItem =()=>{
```

```
if(document.getElementById("file").files.length !== 0){
    var form_data = new FormData();
    for ( var key in cookies.post_item ) { // we should try to normalize how we space apart parentheses
        if(key !== "file") // we should normalize using curly brackets here, which is used above
            form_data.append(key, cookies.post_item[key]);
    }
    form_data.append("user_id", user_id);
    form_data.append("product_author", cookies.first_name);
    form_data.append("file", product_file);

    setPosting(true);

    axios.post('/api/product',form_data) // state route information and what it returns?
        .then((response) =>{
            console.log("Item has been posted successfully."); // remove for final
            setSuccessully(true); // misspelled here, need to adjust
            setFileName('Upload File here...');
            removeCookies('post_item');
            ReactGA.event({
             category: 'PostItem',
             action: 'Item Posted',
             transport: 'beacon'
            });
        })
        .catch((error) => console.log(error))
    setShow(false); // what does this do?
```

5. Self-check on best practices for security

Major assets we are protecting:

User information and media accessibility.

Major threats for each asset above:

Threats for user information includes snooping for user information within information supplied to the client by the server. Threats of unauthorized media accessibility includes directly downloading uploaded assets through means that bypass regular e-commerce processes.

How we are protecting each asset:

Privacy of users - The application minimizes the usage and accessibility of user information. Only necessary information is sent to the client to allow client-side processing, such as their names. We never send sensitive information, such as emails and passwords, which are only processed server-side. All account passwords are encrypted and stored in the database.

Media accessibility - Small resolution thumbnails of uploaded files are used when displaying details of the file, such as on the search results and in the item's detail page. The full resolution or original file is presented for download if the item is listed as free, else the media can only be accessed by contacting the seller

Input data validation list:

Input data is being validated by Formik and Yup custom validation.

Home page:

- Search bar input for up to 40 alphanumeric characters.

Sign in / Sign up:

- All fields are mandatory.
- First name and last name must be 15 characters or less
- Email must be a SFSU email which shall either be "@mail.sfsu.edu" or "@sfsu.edu".
- PW must be more than 8 characters.

- PW and confirmPW fields must be matched.

Post item:

- All fields are mandatory.
- Product name must be 30 characters or less.
- Price must be a real number.
- File must be picked
- File type must be supported (mp3, mp4, pdf, jpg, png)
- Product description must be 500 characters or less.

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

List of non-functional requirements

1. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers. DONE

2. Selected application functions must render well on mobile devices. ON TRACK

3. Data shall be stored in the team's chosen database technology on the team's deployment Server. DONE

4. Full resolution free media shall be downloadable directly, and full resolution media for selling shall be obtained after contacting the seller/owner. DONE

5. No more than 50 concurrent users shall be accessing the application at any time. DONE

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 (no localization needed) DONE

8. Application shall be very easy to use and intuitive. DONE

9. Google analytics shall be used DONE

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 in UI. DONE

12. Site security: basic best practices shall be applied (as covered in the class) for main data items. DONE

13. Media formats shall be standard as used in the market today. DONE

14. Media material shall be either free or for sale, as determined by media owner. DONE

15. Each media material shall have its license info as one of the following: a) free use and modification; b) free but only allowed for SFSU related projects; c) for sale. DONE

16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. DONE

17. The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2020. For Demonstration Only" at the top of the WWW page. (Important so as to not confuse this with a real application). DONE