SW Engineering CSC 648/848 Section 02 Spring 2017

Professor D. Petkovic and Anthony Souza, San Francisco State University

GatorSell

Milestone 4

May 8, 2017

Team 10 (Local)

Amanda Robinson anikkole@mail.sfsu.edu
Ronald Rieger
Priya Krishnakumar
Jason Bockover
Rainier Hui
Tony Filippo

History of Revisions		
Version 1.0	5/8/2017	Initial Milestone 4

1. Product Summary

On May 27, 2017, Team Ten is releasing GatorSell: the best website for SFSU students to buy and sell items. Our website provides guest and registered users the option to browse through our catalogue of items or search items by their category. Guest users can then choose to create a registered user account and once their account is activated, they can list items for sale, buy items on the website and contact buyers and sellers through our messaging system. To become a registered user, guest users must fill out the registration page and include their name, SFSU email address, phone number, password, and agree to the terms of service. Guest users are not initially registered upon completion of the registration page and must wait for their account to be activated. If a registered user decides to sell an item on GatorSell, they must include the item's name, the price they wish to charge for that item, the category that best fits the item they are selling, and a Safe Meeting location.

The Safe Meeting location is the heart of our website and what makes our website stand out from our competition. Safe Meeting locations are on-campus facilities at SFSU (Thornton, Cesar Chavez, etc.) that ensure the safety of the buyers and sellers as they complete their transactions. Sellers will be able to see the Safe Meeting location they have chosen via a map that illustrates the location they have chosen. The seller has the option to include more details about their item by providing a picture or a description. Once an item is bought, the seller can simply remove the post from the website.

For more information, please visit http://www.gatorsell.com

2. Usability Test Plan

Test Objectives

In this usability plan, the purpose is to determine whether GatorSell's add new item feature is efficient and easy for the targeted demographic (San Francisco State University students) to use. The registered user will be already logged in to the website and will complete the task of posting a new item to GatorSell with a corresponding picture.

Test Plan

- **System setup:** The two most recent versions of Mozilla, Safari, or Chrome
- **Starting point:** User will begin logged into GatorSell on the homepage
- **Task to be accomplished:** The user must post an item up for sale with an correlated picture
- **Intended user:** Registered SFSU student
- **Completion criteria:** User will have completed the task when the submitted page directs the user to the confirmation page and can be seen on list of posted items (browse page).
- **Url of the system to be tested:** http://www.gatorsell.com/items/new_item

Questionnaire
Check one circle that applies to your experience

1. I found GUI self-explanatory

   ◯ Strongly disagree ◯ Disagree ◯ Neutral ◯ Agree ◯ Strongly agree

2. It was easy to find the sell/add new post feature

   ◯ Strongly disagree ◯ Disagree ◯ Neutral ◯ Agree ◯ Strongly agree

3. I found the process to post an item to be efficient

   ◯ Strongly disagree ◯ Disagree ◯ Neutral ◯ Agree ◯ Strongly agree

   Comments:

3. QA Test Plan

Formal QA Test Plan

Test Objectives

In this QA plan, the goal is to discover how well each form of hardware and software is able to perform each test case when posting a new item.

HW and SW Setup

Browser #1 - FireFox 53.0.2 (32-bit) - i5 - 2.4 GHz CPU, 8 GB RAM
Browser #2 - Chrome 58.0 (64-bit) - i7 - 2.3 GHz CPU, 6 GB RAM

Feature to be Tested

The add new item feature will be tested by checking the validity of each input field and when the item is added to the database.

Test Cases

Our first test case is to see if the page will tell the user that the name is invalid when he or she tries to post an item without anything in the Name text field.

Our second test case is to see if the page will tell the user that the price is invalid when he or she tries to post an item with a number of 0 or less or an non-number.

Our final test case is to see if the page will tell the user that the image size is too big when he or she tries to post an item with a image file of 2mb or greater.

Results of Testing

Test #	Test case	Test Description	Test Input	Expected Correct Output	Test Results (PASS/FAIL)
1	Valid Name	User leaves the name input box empty when adding a new item	Name = “”	Should prompt user to fill out the Name Field	✔
2	Valid Price	User puts up a new item to cost $0	0	Should prompt user to have the Price field to be greater than $0.00	✔
3	Valid Photo	User uploads a photo of 2mb or more for an item to be added	big.jpg	Should prompt the user to upload a picture smaller than 2mb	✔

Browser #1 Test - Firefox

- Setup of HW and SW
  - i5 2.4 GHz CPU, 8 GB RAM
  - Browser: FireFox 53.0.2 (32-bit)
  - URL: http://www.gatorsell.com/items/new_item

Test #	Test case	Test Description	Test Input	Expected Correct Output	Test Results (PASS/FAIL)
1	Valid Name	Using Space as name	" "	The Name field is required.	✔
2	Valid Price	Input a Fake price	asdsdf	The Price field must contain a number greater than 0.	✔
3	Valid Photo	Added 4mb Photo	msg-1-fc-40.jpg	Invalid Photo (Currently uploads as no image)	X

Browser #2 Test - Chrome

- Setup of HW and SW
  - i7 - 2.3 GHz CPU, 6 GB of RAM
  - Browser: Chrome 58.0 (64-bit)
  - URL: http://www.gatorsell.com/items/new_item

Test #	Test case	Test Description	Test Input	Expected Correct Output	Test Results (PASS/FAIL)
1	Valid Name	Using Space as name	" "	The Name field is required.	✔
2	Valid Price	Using 0 as price	0	The Price field must contain a number greater than 0.	✔
3	Valid Photo	Added 4.3mb photo	moon.jpg	Invalid Photo (Currently uploads as no image)	X

4. Code Review

Coding Style: MVC

Subject: Re: Code Review
From: Jason Bockover <jbockove@gmail.com>
To: "zebraone100@gmail.com" <zebraone100@gmail.com>

Sure thing Ron.

For starters, The documentation is very detailed and that is a great thing for future maintenance. The code is properly formatted using the default netbeans format tool so it is easy to read. The function names imply what they do and the variable names mostly imply what they mean.

One item need to be addressed: private function upload_photo($id){

This is a minor thing, but should still be brought up. $id should be $item_id since all of our data structures have an integer id. From the context, it is not exactly clear what it stands for. Is it for item or for a photo id?

Other than that, it looks awesome.

Have a great day,

Jason

On Mon, May 1, 2017 at 1:53 PM, Ron Rieger <zebraone100@gmail.com> wrote:

> Jason, can you review my code please?
>
> Thanks,
>
> Ron
>
>
> public function new_item() {
> must_be_logged_in();
>
> // set up the rules to validate the form data
> $this->form_validation->set_rules('name', 'Name', 'required');

```
>         $this->form_validation->set_rules('price', 'Price',
> 'required|greater_than[0]');
>         $this->form_validation->set_rules('category_id', 'Category',
> 'required|greater_than[0]');
>         $this->form_validation->set_rules('location_id', 'Location',
> 'required|greater_than[0]');
>
>         // execute the form validation
>         if ($this->form_validation->run() == FALSE) {
>             // form is incorrect or hasn't been run yet so show the add
> item page
>             gator_view('Add New Item', 'pages/Add_New_Post');
>         } else {
>             // data is good.  add new item to the database
>             $this->load->model('items_model');
>             $item = $_POST;
>             $item['photo_id'] = 0;
>             $item['seller_id'] = $_SESSION['registered_user']['id'];
>             $item_id = $this->items_model->add_item($item);
>
>             //upload photo if there is one
>             $photo_id = $this->upload_photo($item_id);
>             $this->items_model->update_photo_id($item_id,$photo_id);
>
>             redirect("items/post_confirm/$item_id");
>         }
>     }
>
>     private function upload_photo($id) {
>         $this->load->library('upload');
>
>         // is there a file uploaded?
>         if (!$this->upload->do_upload('photo')) {
>             return 0;
>         } else {
>             $img = $this->upload->data();
>
>             // set up the image manipulation api
>             $config['image_library'] = 'gd2';
>             $config['source_image'] = $img['full_path'];
>             $config['create_thumb'] = false;
>             $config['maintain_ratio'] = TRUE;
>             $config['new_image'] = $img['full_path'] . ".jpg";
>             $config['quality'] = 100;
>             $config['width'] = 100;
```

```
>             $config['height'] = 100;
>
>             // create the thumbnail
>             $this->load->library('image_lib', $config);
>             $this->image_lib->resize();
>
>             // upload the blob to the database
>             $this->load->model('photos_model');
>             $photo_id = $this->photos_model->upload_photo($$id,
> $img['full_path'], $img['full_path'] . ".jpg", $img['image_type']);
>
>             // delete the temporary files
>             unlink($img['full_path']);
>             unlink($img['full_path'] . ".jpg");
>
>             return $photo_id;
>         }
>     }
```

5. Self-Check on Best Practices for Security

Protected Major Assets

- User's password
- User's phone number
- User's received and sent messages

Encrypted Password in Database

- Status: DONE

Input Data Validation

- Search bar input
    - Status: DONE
    - Test inputs such as 'chair' gives accurate results

- Email registration with SFSU-only email
    - Status: DONE
    - Test inputs such as 'bob@sgsu.edu' is not allowed, only SFSU email addresses are allowed

6. Adherence to Non-Functional Specs

---

1. Application shall be developed using class provided LAMP stack. **DONE**
2. Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks must be be explicitly approved by Anthony Souza on a case by case basis. **DONE**
3. Application shall be hosted and deployed on Amazon Web Services as specified in the class. **DONE**
4. Application shall be optimized for standard desktop/laptop browsers, and must render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. **DONE**
5. Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed. **DONE**
6. Data shall be stored in the MySQL database on the class server in the team's account. **DONE**
7. Application shall be served from the team's account. **DONE**
8. No more than 50 concurrent users shall be accessing the application at any time. **DONE**
9. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **DONE**
10. The language used shall be English. **DONE**
11. Application shall be very easy to use and intuitive. No prior training shall be required to use the website. **DONE**
12. Google analytics shall be added **ON TRACK**
13. Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services. **DONE**
14. Pay functionality (how to pay for goods and services) shall not be implemented. **DONE**
15. Site security: basic best practices shall be applied (as covered in the class) **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 text on all pages *"SFSU Software Engineering Project, Spring 2017. For Demonstration Only"*. (Important so as to not confuse this with a real application). **DONE**