SW Engineering CSC 648/868

Section 01 Spring 2017
Group 3

Gator Swap

Milestone 4

Mark Reynolds, Ahmed Salem, Rushabh Vora, Skylar Miles, Shayn Hoy, Lowell Castillo

shmarkus@mail.sfsu.edu

Revision	Date
1.0	5/4/17

1) **Product Summary**

Gator Swap is website for students to buy and sell what they no longer need. Gator Swap will allow anyone to search for listings either by category or by a search bar. Anyone will be able to view the details of each listings, such as the asking price, description, and the pick-up location, if the seller provides one. After registering, users can post their own listings for sale as well as contact other sellers to purchase an item. What really makes Gator Swap unique is the ability for sellers to specify where they want buyers to pick up the item being sold allowing buyers to know before hand if they will be able to pick up an item.

Functional Specifications at Launch:

- All Users can...
  - View listings by category.
  - Search for listings by keywords.
  - View listing details.
  - Register.
- Registered Users can…
  - Post new listings for sale.
  - Upload images for their listings.
  - Set a pickup location for their listings.
  - Message other users about their listings.
- Administrators can…
  - View all users.
  - View all listings.
  - Remove illegal listings.
  - Ban registered users users.

Site URL: www.sfsuse.com/~sp17g03/

2) <u>Usability Test Plan</u>

"Search" Function

- **Test Objectives:**
  To test the function of Search on the web application Gator Swarp
- **Test Plan:**
  - System setup:
    Our group is using Laptops running on Mac OS and Windows. We plan on having users be able to use laptops, desktops, as well as mobile devices.

  - Starting Point:
    Users will start on the homepage where they will be greeted with a welcome message and categories to search through. At the top of the page the user will see a nav bar consisting of the website logo, search bar, and links to different parts of the website.

  - Task to be accomplished:
    The Search function is to help direct users of our web applications to specific listings or listings similar to their requests. The search function shall also utilize categories to filter through the various listings posted on the site.

  - Who is the intended user:
    The intended users for our web application are San Francisco State University Students and Faculty wishing to buy or sell their stuff. The website itself will not require a user to login if they only wish to browse listings. Any other
    features i.e. (messaging, buying, and selling) will require a valid San Francisco State University Email to login.

  - Completion Criteria:
    Upon completing testing of the Search function a user should be able to find items using categories such as electronics or furniture. After searching electronics listings such as iPhone, MacBook Pro, and Microsoft Surface appeared. After searching furniture listings such as sofa bed and bunk bed appeared. After searching all categories all items were shown.

  - URL of the System to be tested:

http://www.sfsuse.com/~sp17g03/

3) QA Test Plan:

Test Objectives:

Ensure the usability and functionality of searching products with and without a filter.

HW and SW Setup:

Website supports web browsers (FireFox, Chrome, etc). Internet connection is required in order to interact with the website.

Feature to be Tested:

Searching for products with and without a filter.

Test Plan:

Test Number	Description	Test Input	Expected Output	PASS /FAIL
1	-Open the webpage: http://www.sfsuse.com/~sp17g03/ -Change the category from " All categories " to " Books" and search it with an empty search bar.	**Category**: "Books" **Search**: " " (Empty)	-one listing –item titled "old book"	
2	-Open the webpage: http://www.sfsuse.com/~sp17g03/ -leaving the category as " all categories", type the input into the search bar and search it.	**Category**: "All Categories" **Search**: "bed"	- Two listings -item 1: "Sofabed" -item 2: "Bunk Beds"	

3	-Open the webpage: http://www.sfsuse.com/~sp17g03/ -Change the category from “ All categories “ to “ electronics”, type the input into the search bar and search it.	**Category**: “Electronics” **Search**: “iPhone“	-one listing - item titled “iPhone 6”	
4	-Open the webpage: http://www.sfsuse.com/~sp17g03/ -Change the category from “ All categories “ to “ electronics”, type the input into the search bar and search it.	**Category**: “Electronics” **Search**: “fridge“	- no listings - a message should read” sorry there are no results for this search”	

- **Questionnaire Form**:
    1. The Search Bar was easy to find.

    [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Agree

    Comments:

    2. The Search Bar used categories to help find listings.

    [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Agree

    Comments:

    3. The Search Bar provided results of listings that the user requested or similar to what the user requested.

    [ ] Strongly Agree [ ] Agree [ ] Neutral [ ] Disagree [ ] Strongly Agree

    Comments:

4) <u>Code Review</u>

a) Coding Style:

- Variable and function names are in camelcase
- Classes and namespaces are in title case
- Four spaces indentation
- Code documentation done with Doxygen
- Database tables named with lowercase and underscore
- Table column names done in all uppercase

b) Peer Reviewed Code:

Shayn:

Thanks Rush for looking it over. I'll implement your suggestions right away

On Thu, May 4, 2017 at 10:03 PM, Rushabh Rajivratna Vora <rvora@mail.sfsu.edu> wrote:

Well done on getting search to work as required!

Some observations I made:

1) It’s a good idea to keep the comments updated as you add functionality. As searching within description was added later, the function description comment should be updated to reflect the same.

2) Coding style states that variable names are in camel case, so a comment should be added explaining why “$_GET” is not camel cased.

3) I’ve added some comments below in the code as well.

From: Shayn Hoy <swauvenx@gmail.com>
Date: Thursday, May 4, 2017 at 21:45
To: Rushabh Rajivratna Vora <rvora@mail.sfsu.edu>
Subject: Code Review

```
<?php
namespace App\Controller;

use App\Controller\AppController;
use Cake\ORM\TableRegistry;

/**
 * Controller for listing.
 */
```

```
class ListingsController extends AppController
{
      /**
      * Action for viewing a listing.
      *
      * @param id The id of the listing to view.
      *
      * @post Set a string for listingName, listingDescription,
listingImage, and listingPrice.
      */
      public function view($id)
      {
      $query = $this->Listings->find();
      $query->where(['ListingsId' => $id]);

      foreach ($query as $listing){
           $this->set('listingName', $listing->TITLE);
           $this->set('listingPrice', $listing->PRICE);
           $this->set('listingDescription',$listing->DESCRIPTION);
           $this->set('listingImage',$listing->PICTURE);
           break;
      }
      $this->render();
      }

      /**
      * Action for searching for a listing.
      *
      * @param query The text to search for. Searches both the title
and the description to find matching results
      * @param category The category to search in. (Not yet
implemented.)
      *
      * @post Set an array for results. Each result should be an array
containing strings for listingName, listingShortDescription,
listingImage, listingPrice, and listingID.
      */
      public function search()
      {
      if (array_key_exists('query',$_GET))
      {
           $query = htmlspecialchars(stripslashes($_GET['query']));
      }
      else  // This allows a category to be searched without a specific
search term. (i.e. lists everything in the category.)
      {
            $query = '';
      }
      if (array_key_exists('category', $_GET))
      {
            $category =
htmlspecialchars(stripslashes($_GET['category']));
      }
      else
      {
            $category = 'All Categories';
      }

      if ($category == 'All Categories')
      {
            $queryResults = $this->Listings->find()->where(['Title LIKE'
=> "%$query%"])->orWhere(['Description LIKE' => "%$query%"]);  //
Query methods can also be chained! Added search with description *Peer
Review – the meaning of this is not clear, maybe give an example in comments?*
      }
      else
      {
            $queryResults = $this->Listings->find()->where(['Title LIKE'
=> "%$query%", 'Category' => $category])->orWhere(['Description LIKE'
=> "%$query%", 'Category' => $category]); //Search either title or
description
      }

      $results = array();
      foreach ($queryResults as $result){
            $results[] = ['listingName' => $result->TITLE,
'listingShortDescription' => $result->SHORTDESCRIPTION, 'listingImage'
=> $result->THUMBNAILS, 'listingPrice' => $result->PRICE, 'listingID'
=> $result->LISTINGSID];
      }
      $this->set('results',$results);
      $this->render();
      }*Peer Review – Apart from documenting the function, I think it's a good
idea to document the logical loops*
```

5) Self-Check on Best Practices for Security

Major Assets Being Protected:

- User Passwords
- User Email Addresses

Passwords are hashed before being stored in database.

Data Validation:

- - Front End
    - Limit search to 30 characters.
    - Limit username and password to 30 characters.
    - Limit email to 60 characters.
    - Limit search to alphanumeric characters, spaces, and periods.
    - Limit username to alphanumeric characters and periods.
- Back End
    - Search queries are stripped of slashes.
    - Search category is checked against list of known categories, and ignored if not a pre-existing category.
    - At registration, email is checked to end with "@mail.sfsu.edu".

Backend search validation as well as additional backend registration validation will be added by release.

6) Adherence to original Non-Functional specs

Non-Functional Specs	Status
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	On Task
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.	On Task
6. Data shall be stored in the MySQL database on the class server in the team's account.	On Task
7. Application shall be served from the team's account.	On Task
8. No more than 50 concurrent users shall be accessing the application at any time.	On Task
9. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.	On Task
10. The language used shall be English.	On Task
11. Application shall be very easy to use and intuitive. No prior training shall be required to use the website.	On Task
12. Google analytics shall be added.	On Task
13. Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services.	On Task

14. Pay functionality (how to pay for goods and services) shall not be implemented.	On Task
15.Site security: basic best practices shall be applied (as covered in the class).	On Task
16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.	On Task
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