SW Engineering CSC648/848 Summer 2016

Application: Gatorslist

Milestone 4

August 4th, 2016

Rev 1.0

Team 1

Khalid Alrashed (alrashed@mail.sfsu.edu),
Eric Chen, Robert Chung,
Tai Nguyen, Guoyi Ruan.

Table of Contents

1. Product Summary

Gatorslist is a website where SFSU students can locally sell their items or buy items posted by other students. Our goal is to make exchange of goods between students as simple and convenient as possible, as well as the ease of mind by having the option to pay in advance using our secure payment process.

Gatorslist aims to broaden the options SFSU students have in terms of buying products, as well as selling their unwanted items to their fellow students. The exchange of items between sellers and buyers is all done locally in SFSU campus, giving buyers the chance of inspecting the item before confirming their purchase. The website provides the users with a very simple and intuitive search functionality right on the home page, with the ability to further sort and filter their search to narrow down what their looking for.

Gatorslist provides the users with the following major functions:

- An intuitive search function that is persistent in all of the web pages.
- Search results page generated by the users submitted searches.
- Sorting the search results by price or date, and filtering by price and condition.
- The ability to register as a new user using an SFSU email.
- A user account page where they can see and edit their profile, as well as view their submitted items for sale and editing them.
- Users will be able to sell their items, providing the price, condition, description of the item as well as uploading up to 4 images.
- Each item will have its own item page with details, and the ability contact the seller before buying.
- Administrators will be able to oversee the website's activity, editing or deleting users/items according to the terms and conditions.

Visit our website at: http://sfsuswe.com/~su16g01/gatorslist

2. Usability Test Plan

2.1 Test Objectives

For usability testing, we have decided to test the usability of the purchase function for our website. Since our website is largely used for purchasing items, we think that it's important to make sure that our purchase function is functional. We want to make sure that it's very easy to purchase items.

The users will be told to search for a particular item, view the item, purchase the item and contact the seller. The users shall start their testing from our home page and begin searching for items. Will be asked questions about how easy it was to find the item and purchase it.

2.2 Test Plan

Intended users: SFSU students looking to shop.

Test components

System setup - Computer with chrome, mozilla, or internet explorer.

Starting point - Home page of http://sfsuswe.com/~su16g01/gatorslist

Task 1 - From the home page, search for a book called database management.
Task 2 - From the results, view the item.
Task 3 - After viewing the item, choose to buy the item.
Task 4 - Choose a purchasing method
Task 5 - Place your order and/or contact the seller to meet.

Completion Criteria - User has successfully completed the tasks if he/she is able to successfully purchase an item.

2.3 Questionnaire

Question 1: It was very easy and quick to purchase an item.

Strongly Agree Agree Neutral Disagree Strongly Disagree

Question 2: It was very easy to find the item.

Strongly Agree Agree Neutral Disagree Strongly Disagree

Question 3: It was very easy to view the item.

Strongly Agree Agree Neutral Disagree Strongly Disagree

Question 4: The layout of the website is very simple and clean.

Strongly Agree Agree Neutral Disagree Strongly Disagree

Question 5: If I would like to buy things from SFSU students, I would use this website.

Strongly Agree Agree Neutral Disagree Strongly Disagree

3. QA Test Plan

3.1 Test Objectives

Gatorslist is a PHP based website used to allow individuals to buy and sell personal items on the website. In addition to the PHP framework used, it also incorporates the use of a 'mysql' database. An essential component of the website is the ability for users to purchase items from other users. In this test, the tester will be responsible for testing the product to ensure there is full purchasing functionality. The test team will serve as both pseudo customers and testers in this project.

For this test to be determined as a success, the tester must be able to register an account on the website, search for an item, navigate through to the item's details and finalize the purchase of the item either by contacting the seller or sending their payment in advance. Failures towards the completion of any aforementioned procedures due to software issues will deem this test unsuccessful. For this particular test, the specifications of the item selection and user account are not considered, any item posted for sale on the website and account created by the tester will be valid.

3.2 Hardware and Software Setup

Applicable hardware needed for this test are limited to any desktops or laptops with internet connectivity. Mobile device compatibility will not be considered for this project and thus will not be accounted for in this test. Internet connectivity must be through an ethernet cable; Cat-5/5e/6/6a or via WiFi. In addition, the computer must be connected to both keyboard and mouse peripherals, necessary for performing actions on the website. If using a desktop, the tester must able to have display connectivity to a monitor or any similar form of display hardware. The physical hardware components including but not limited to, the computing processor, motherboard, graphics module, random access memory and hard disk are not important.

The appropriate operating system installed on any desktop or laptop being used for the test must be of either Microsoft's Windows XP,7/8/10, Apple's OSX, or Linux Ubuntu. As long as it is fully compatible with one of the aforementioned operating systems, the computer is compatible with this test. However, any computer with an operating system not listed will produce results not conclusive for the test and are thus disqualified. For software requirements necessary for this test, the tester must have no more than the third oldest software version of Google Chrome, Mozilla Firefox or Microsoft Internet Explorer installed on their respective operating system platform to be used. Other internet browsers are not taken into account for this project and will not be tested for.

3.3 Features to be tested

Components within the purchase feature will include the confirmation, checkout and post checkout features. The tester will be able to operate within the website to reach these three checkpoints in this order. Once they have completed through the post checkout feature and it is successful, the test has come to a conclusion.

3.4 Test Cases

Test Number	Test Process	Test Input	Expected Output	Pass/Fail
1	Navigate to the website www.su16g01.com and search "macbook"; select the first item and select buy on the details page.	User : Non-Registered Search Criteria : Macbook	The webpage redirects the user to register for an account before they can make a purchase.	PASS
2	Navigate to the website www.su16g01.com, log into a registered account and search "macbook"; select the first item and select buy on the details page. Select the option to contact the seller instead of paying in advance.	User : Registered User Search Criteria : Macbook	The web page redirects to the user to confirm their purchase and afterwards presents a choice of either contacting the seller directly or paying in advance; if the buyer chooses to contact the seller and not yet pay for the item, the buyer is notified that their message is sent and is redirected to home page within 5 seconds.	PASS
3	Navigate to the website www.su16g01.com, log into a registered account and search "macbook"; select the first item and select buy on the details page. Select the option to pay in advance and fill out all of the forms regarding billing information, credit card information before submitting.	User : Registered User Search Criteria : Macbook	The web page redirects to the user to confirm their purchase and afterwards presents a choice of either contacting the seller directly or paying in advance; if the buyer chooses to pay for the item, the buyer is notified that their message is sent to the seller and that their order has been placed. They are redirected to the home page within 5 seconds.	PASS

4. Code Review

Codes were originally developed by using bootstrap. This simplified the process of designing the skeleton of all sites. The further formats and styles were written by following mini framework. The advantages of using mini framework offer the convenient way to add class objects, and id objects in the uniform way to be managed and edited. Sometimes, to specialize some characteristics of the specific div, additional styles are added up directly into tag to attract the concern for other developers.

Upload Feature:

- **View**

```
<div class="container" style="">

  <!-- register form -->
  <div class="box">
    <div class="col-lg-6 col-lg-offset-3" style="margin-top:1%" >
      <!-- Panel container -->
      <div class="panel panel-default ">
        <!-- panel head-->
        <div class="panel-heading" style="text-align:center; font-weight:800; font-size:20px">Sell your item</div>
      <!-- panel body -->
        <form class="panel-body col-lg-offset-0" style="font-size: 16px;background-color: #87CEFA"  role="form"
action="<?php echo URL; ?>sell/createitem" method="POST" enctype="multipart/form-data">

            <div class="form-group form-inline">
              <div class="form-group" >
                <label class="" for="item title" >Item title:</label>
                <div class="">
                  <input type="text" class="form-control input-md" name='Title' id="item_title" style="font-size:14px"
placeholder="Enter item name">
                </div>
              </div>

              <div class="form-group" style="margin-left:20%">
                <label class="" for="price" >Price: </label>
                <div class="">
                  <input type="number" class="form-control input-md " name='Price' id="item_price"
style="font-size:14px" placeholder="">
                </div>
              </div>
            </div>

              <div class="form-group">
                <label class="" for="categories" >Choose Categories:</label>
```

```
                <div class=""  id="categories">
                  <select name="Category_Id" class="form-control  ">
                    <option value="">All Categories</option>
                    <option value="1">Books</option>
                    <option value="2">Furniture</option>
                    <option value="3">Electronics</option>
                    <option value="4">Office Supplies</option>
                    <option value="5">Clothing</option>
                    <option value="6">Other</option>
                  </select>
                </div>
              </div>

              <div class="form-group">
                <label class="" for="condition" >Choose Condition:</label>
                <div class="" id="item_condition" style="margin-left:0%">
                  <select name="Condition" class="form-control ">
                    <option value="">All Kinds</option>
                    <option value="new">New</option>
                    <option value="used">Used</option>
                  </select>
                </div>
              </div>
            <div class="form-group">
              <label class="" for="condition">Description: </label>
              <textarea class="form-control" rows="5"  name='Description' id="description" placeholder="Describe
your item here..."></textarea>
            </div>

          <div class="form-group">
            <label for='image' >Select image to upload:</label>
            <input type="file" name="fileToUpload" id="fileToUpload" />
          </div>

          <div class="form-group">
            <label for='image' >Select image to upload:</label>
            <input type="file" name="fileToUpload2" id="fileToUpload2" />
          </div>

          <div class="form-group">
            <label for='image' >Select image to upload:</label>
            <input type="file" name="fileToUpload3" id="fileToUpload3" />
          </div>

          <div class="form-group">
            <label for='image' >Select image to upload:</label>
            <input type="file" name="fileToUpload4" id="fileToUpload4" />
          </div>

          <div class="form-group" >
```

```
                <div class="" >
                  <button type="submit" class="btn btn-default col-lg-offset-4" name="submit" value="submit"
id="submit">Submit</button>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div>
```

- **Controller**

```
class Sell extends Controller
{
  /**
   * PAGE: index
   * This method handles what happens when you move to http://../sell/index
   */
  public function index()
  {
     // load views.Specify different conditions for login user or guest.
     $categories = $this->model->getProductCategory();

     if (isset($_SESSION['loggedInUser_id'])) {

         require APP . 'view/_templates/header.php';
         require APP . 'view/sell/index.php';
         require APP . 'view/_templates/footer.php';

     }else {

         require APP . 'view/_templates/header.php';
         require APP . 'view/users/index.php';
         require APP . 'view/_templates/footer.php';
     }
  }
  /**
   * ACTION: createItem -- add product with image
   */
  public function createItem()
  {
     $categories = $this->model->getProductCategory();

     $image1 = file_get_contents($_FILES["fileToUpload"]["tmp_name"]);

     if(($_FILES['fileToUpload2']['tmp_name']) != ""){
        $image2 = file_get_contents($_FILES['fileToUpload2']['tmp_name']);
     }else{
```

```
        $image2 = NULL;
    }

    if(($_FILES['fileToUpload3']['tmp_name']) != ""){
        $image3 = file_get_contents($_FILES['fileToUpload3']['tmp_name']);
    }else{
        $image3 = NULL;
    }

    if(($_FILES['fileToUpload4']['tmp_name']) != ""){
        $image4 = file_get_contents($_FILES['fileToUpload4']['tmp_name']);
    }else{
        $image4 = NULL;
    }

    $date = date("Y-m-d");
    $seller_id =  $_SESSION['loggedInUser_id'];
    $this->model->createItem($seller_id,$_POST["Title"], $_POST["Description"], $_POST["Price"],
$_POST["Condition"],$date, $_POST["Category_Id"],$image1,$image2,$image3,$image4);

    header('location: ' . URL . 'sell/index');
  }
 }
```

- **Model**

```
// add product with images
public function createItem($seller_id, $title, $description, $price, $condition, $date, $category_Id,$image1,$image2,
$image3,$image4)
{
  $parameters = [
    ":seller_id" => $seller_id,
    ":title" => $title,
    ":description" => $description,
    ":price" => $price,
    ":condition" => $condition,
    ":date" =>$date,
    ":category_Id" =>$category_Id,
    ":image1" => $image1,
    ":image2" => $image2,
    ":image3" => $image3,
    ":image4" => $image4,
  ];
  $this->dao->create($parameters, "item");
}
```

- **DAO -Data Analysis Object**

```
// add product with images
public function create($parameters, $target) {
  if($target == "item"){
     $seller_id = $parameters[":seller_id"];
     $title = $parameters[":title"];
     $description = $parameters[":description"];
     $price = $parameters[":price"];
     $condition = $parameters[":condition"];
     $postdate = $parameters[":date"];
     $category_Id =  $parameters[":category_Id"];
     $image1 =  $parameters[":image1"];
     $image2 =  $parameters[":image2"];
     $image3 =  $parameters[":image3"];
     $image4 =  $parameters[":image4"];

     $sql1 = "INSERT INTO image (Image_blob1,Image_blob2,Image_blob3,Image_blob4) VALUES (:image1, :image2,
:image3, :image4)";
     $query1 = $this->db->prepare($sql1);
     $parameters1 = array(':image1' => $image1, ':image2' => $image2, ':image3' => $image3, ':image4' => $image4 );
     try {
        if ($query1->execute($parameters1)) {

           $sql = "INSERT INTO product (Seller_id, Title, Description, Price, ItemCondition, Postdate,
Category_Id,Image_id)
           VALUES ('".$seller_id."','".$title."' , '".$description."', '".$price."', '".$condition."','".$postdate."' ,
'".$category_Id."','".$this->db->lastInsertId()."')";
           $query = $this->db->prepare($sql);
           try {
              if ($query->execute()) {
                 return true;
              } else {
                 return false;
              }
           } catch(PDOException $e) {
              echo $e->getMessage();
           }
        } else {
           return false;
        }
     } catch(PDOException $e) {
        echo $e->getMessage();
     }
  }
}
```

Comments: Good formatting, no missing or mismatched end tags;
Code is readable and easy to understand;
No redundant or duplicated code;
Good to use try- catch statements.

5. Adherence to Original 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 have to be explicitly approved by Marc Sosnick 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 shall render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome and IE. It shall degrade nicely for different sized windows using class approved programming technology and frameworks. **DONE**
5. Data shall be stored in the database on the class server in the team's account. **DONE**
6. Application shall be served from the team's account. **DONE**
7. No more than 50 concurrent users shall be accessing the application at any time.
8. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **DONE**
9. The language used shall be English. **DONE**
10. Application shall be very easy to use and intuitive. No prior training shall be required to use the website. **DONE**
11. Google analytics shall be added for major site functions.
12. Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services. **DONE**
13. Site security: basic best practices to be applied (as covered in the class). **DONE**
14. Modern SE processes and practices must be used as specified in the class, including collaborative and continuous SW development, using the tools approved by instructors. **DONE**
15. The website shall prominently display the following text on all pages "SFSU/FAU/Fulda Software Engineering Project, Summer 2016. For Demonstration Only". (Important so as to not confuse this with a real application). **DONE**