SW Engineering CSC648/848 Section 01 Spring 2017
Team 05

Gator 2 Gator

Milestone 4

May 4, 2017

Andrew Lesondak
Tim Bauer
David Rodriguez
James Lee
Mayara Dusheyko
Chengjiu Hong
Jeffrey Hu

History

Initial Draft	v1.0
Update to Usability Test Plan	v1.1
Updates to Test Plan	v1.2
Update to QA Test Form	v1.3

1. Product Summary

Gator 2 Gator is our web application is for SFSU students to buy and/or sell their possessions in a safe, exclusive community.

As a user of the site you will be able to the do the following:
1. Browse the site and view the posted items for sale
2. Create an account if you have a valid SFSU email address to register with
3. Login to a previously created account if you have already registered
4. Send messages to communicate with the sellers
5. Post an item to sell
6. Edit your posts
7. Delete your posts

What makes Gator to Gator unique is its exclusive membership for SFSU students. This means every buyer and seller will be a verified SFSU student, granting the additional ease of mind when going through the financial transaction and physical meet up.

Also, due to the campus being a common meeting area, it serves as a very convenient way to exchange items in both a familiar and public location. Products being sold will also be relevant to students needs as many share the same or similar courses.

Site URL: http://sfsuse.com/~sp17g05

2. Usability Test Plan - Post an Item

The purpose of this usability test plan is to provide us with insight into our user population and the actions and events that occur while they use our application in its current state. It will allow us to predict their performance and allow us to make necessary changes for the future.

Test Metrics

1. Completion Rate

The percentage of test users who successfully finish the task without experiencing critical errors is the Completion Rate. We define a critical error as being an error which results in an incorrect or unsatisfactory outcome. Our goal is to have a completion rate of 100% for the usability task.

2. Error Free Rate

We define the Error Free Rate as the percentage of test users who finish the task without experiencing any critical or Non-Critical Errors. An error that does not affect the final output of the task, but would result in a less efficient experience is a Non-Critical Error. Our goal is to have an Error Free Rate of 80% for the usability task.

3. Time on Task

The time to complete the task is defined as the Time on Task. Test users are expected to complete the task within a specific time limit. Test users will be measured from the time they begin the scenario to the time of completion.

4. Subjective Measures

At the end of the test, users will rate their satisfaction with the task experience. Subjective opinions about the task in general, time to perform, features, and functionality will be surveyed.

Test Plan

1. Intended User

The intended user has the background and abilities which are representative of the average SFSU student. Effective results are obtained only if the selected user is a typical end user of the application, or are matched as closely to the criteria as possible. The major characteristics of the test user are a San Francisco State University student with some experience making online purchases and having any range of college experience.

2. System Setup

The usability test for the application will take place on the SFSU campus to simulate the typical environment a student would be in. This location could potentially be a café, library, or classroom setting. The test user may be on a PC laptop or Macintosh laptop and will be required to have internet access using either of the following web browsers: Google Chrome (Version 57.0.2987 and 56.0.2924), Mozilla Firefox (Version 54.0a2 and 53.0), and Safari (Version 10.1 and 10).

3. Starting Point

The test user will begin their session from the landing page http://sfsuse.com/~sp17g05 of the web application, as that is the initial page all users will begin from. This is a great starting point as this will give us the ability to see number of clicks needed while balancing the time spent on the task.

4. Task to be Accomplished

Create a new item listing and confirm it exists.

5. Completion Criteria

Each session will require that the test user acts and inputs specific data that would be used in course the task. The scenario is completed when the test user signals the scenario's goal has been obtained. This may be whether the session was successful or unsuccessful. The test user may also signal for necessary guidance as to authorize determining the situation as a critical error. Individual evaluations regarding ease of use and satisfaction will be gathered via Lickert scale questionnaires at the end of the session.

Questionnaire form

1. It was easy to post an item for sale.

○	○	○	○	○
Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

Comments:

2. There were too many steps to take to post an item.

○	○	○	○	○
Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

Comments:

3. You found the website simple to use.

○	○	○	○	○
Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

Comments:

3. QA Test Plan

Test Objectives

The purpose of Quality Assurance testing is finding defects which may get created by the programmer while developing the software or improper handling of any types of potential errors. Gaining confidence in and providing information about the level of quality and stability of the web application as well as persistence of relevant data and to prevent defects. To make sure that the end product meets the developers, stakeholders and user requirements. To ensure that the end product satisfies the documented specifications without defects or anomalies in the user interface. To gain the confidence of the customers by providing them with a properly tested quality product that is also stable and secure.

Hardware and Software Setup

Quality Assurance testing must be conducted on a variety of hardware and software configurations. Ideally, QA testing involves all of the major hardware platforms: Windows based PC, Linux based PC, Macintosh Computer, Android based Mobile and Apple mobile. Of those hardware platforms the web application should be tested on the major browsers. For our purposes the main platforms are Windows based PC and Apple Computer hardware and will be required to have internet access using the following web browsers: Google Chrome (Version 57.0.2987 and 56.0.2924), Mozilla Firefox (Version 54.0a2 and 53.0), and Safari (Version 10.1 and 10).

Test Feature

The feature to be tested is the sell item process. The process involves navigating to the sell item page where they will be asked to create a new account with sfsu.edu email address. Then the tester will create a new item post with specified inputs for the text fields after verifying visual element placements. They will upload any photo file type and create the post. They will then verify that the post exists and can be viewed without errors. The tester will then create another post where specified text is intended to create errors or will be rejected by validation security protocols. They will also attempt to upload a single none image file type to assure that the web application will handle such anomalies without error or loss of service. Additionally, the tester will be asked to notice any performance issues such as slowdowns or response lag.

QA Test Form:

Project Name: Gator 2 Gator	
QA Test Form	
Test Case ID: 0001	**Test Designed by:** Andrew Lesondak
Test Priority (Low/Medium/High): High	**Test Designed date:** 5/1/2017
Module Name: New Item Post	**Test Executed by:**
Test Title: Sell Item Quality Assurance	**Test Execution date:**
Description: Test the ability to sell an item and quantify the quality of this service.	
Pre-conditions: User has navigated in web browser to http://sfsuse.com/~sp17g05	
Dependencies: A stable operating environment and proper software versions.	

Test #	Testing	Test Procedure	Test Input	Expected Correct Output	Result (Pass/Fail)
1.	Registration	Navigate to user register page and enter user name.	User = Test1234	No Warnings or Errors	
2.	Registration	Enter email address.	email = test1234@sfsu.edu	No Warnings or Errors	
3.		Enter password.	password = test1234	No Warnings or Errors	
4.	Registration	Click Register Button.	N/A	Account successfully created without errors.	
5.	New Item	Navigate to Sell Item Page and enter Item Title	title = Test Item	No Warnings or Errors	
6.	New Item	Enter item description.	description= Test Description	No Warnings or Errors	
7.	New Item	Select Item Category.	category = Furniture	No Warnings or Errors	
8.	New Item	Select Photo to Upload.	Select any Image file type	No Warnings or Errors	
9.	New Item/ Upload	Click the Submit.	N/A	Redirects to Item Index page.	
10.	Persistence	Navigate to bottom of index page.	N/A	Item exists at bottom of index	
11.	Show Item	Click on the test item in index to view the post.	N/A	Item page is displayed correctly	
12.	Validation	Navigate to new sell item page and enter title.	title = Another Test *@#$^	No Warnings or Errors	
13.	Validation	Enter description.	description = !@#$%^&*()	No Warnings or Errors	
14.	Validation	Enter Category.	category = Books	No Warnings or Errors	
15.	Validation/ Upload	Upload File.	Upload any none image file type	No Warnings or Errors	

16.	Validation	Click The Submit button to create the new test post	N/A	Redirects to Index Page without Errors or Warnings	
17.	Validation	Navigate to the bottom of the index page and verify that the new post exists.	N/A	The new post should be at the bottom.	
18.	Show Item/ Validation	Click the item that was just created to view it.	N/A	The item page should load without Warnings or Errors.	
19.	Validation	Verify the photo element is simply a broken link.	N/A	The photo element did not load and is a broken image symbol.	
20.	Logout	Logout	N/A	Logs out of account without Warnings or errors	
21	Rendering	Reflect on speed of application response.	N/A	Application responded quickly with no noticeable slowdown.	

QA Test Results

Test #	Testing	Test Procedure	Test Input	Expected Correct Output	Firefox (Pass/Fail)	Chrome (Pass/Fail)	Safari (Pass/Fail)
1.	Registration	Navigate to user register page and enter user name.	User = Test1234	No Warnings or Errors	Pass	Pass	Pass
2.	Registration	Enter email address.	email = test1234@sfsu.edu	No Warnings or Errors	Pass	Pass	Pass
3.		Enter password.	password = test1234	No Warnings or Errors	Pass	Pass	Pass
4.	Registration	Click Register Button.	N/A	Account successfully Created without errors.	Pass	Pass	Pass
5.	New Item	Navigate to Sell Item Page and enter Item Title	title = Test Item	No Warnings or Errors	Pass	Pass	Pass
6.	New Item	Enter item description.	description= Test Description	No Warnings or Errors	Pass	Pass	Pass
7.	New Item	Select Item Category.	category = Furniture	No Warnings or Errors	Pass	Pass	Pass
8.	New Item	Select Photo to Upload.	Select any Image file type	No Warnings or Errors	Pass	Pass	Pass
9.	New Item/ Upload	Click the Submit.	N/A	Redirects to Item Index page.	Pass	Pass	Pass
10.	Persistence	Navigate to bottom of index page.	N/A	Item exists at bottom of index	Pass	Pass	Pass
11.	Show Item	Click on the test item in index to view the post.	N/A	Item page is displayed correctly	Pass	Pass	Pass
12.	Validation	Navigate to new sell item page and enter title.	title = Another Test *@#$^	No Warnings or Errors	Pass	Pass	Pass
13.	Validation	Enter description.	description = !@#$%^&*()	No Warnings or Errors	Pass	Pass	Pass
14.	Validation	Enter Category.	category = Books	No Warnings or Errors	Pass	Pass	Pass
15.	Validation/ Upload	Upload File.	Upload any none image file type	No Warnings or Errors	Pass	Pass	Pass
16.	Validation	Click The Submit button to create the new test post	N/A	Redirects to Index Page without Errors or Warnings	Pass	Pass	Pass
17.	Validation	Navigate to the bottom of the index page and verify that the new post exists.	N/A	The new post should be at the bottom.	Pass	Pass	Pass
18.	Show Item/ Validation	Click the item that was just created to view it.	N/A	The item page should load without Warnings or Errors.	Pass	Pass	Pass
19.	Validation	Verify the photo element is simply a broken link.	N/A	The photo element did not load and is a broken image symbol.	Pass	Pass	Pass
20.	Logout	Logout	N/A	Logs out of account without Warnings or errors	Pass	Pass	Pass
21	Rendering	Reflect on speed of application response.	N/A	Application responded quickly without slowdown.	Pass	Pass	Pass

4. Code Review

We use the Standard php coding style Psr-1.

Team member, James Lee, submitted code to be reviewed by team member, Tim Bauer. The code review is regarding having James reduce the size of site's forms. The code review is performed for the Registration page as it was an example to use for moving forward with updating the Post an Item form after. The code review is performed through Github with comments included.

James:

Submits Pull Request

-Shrink the box fields in registration form …
-Changed "Terms Accepted" check box to "I have read and accept the terms of service"
-Changed the error message if checkbox is not checked, from "This value must be true" to "please check the box before continuing"

Tim:

Good work. Please see comment in web/css/main.css for requested tweaks. :)

Tim's comments in web/css/main.css:

Nicely done. That was definitely easier than I was anticipating.
That being said, we really need a better name than 'myclass'.

On further thought, the right way to do this would be to make this the default for all forms in our app. That way, if we decide we need another form in the future or decide our forms need to look slightly different, we're only altering code in one place rather than trying to go around and fix every form in the app.

Case in point, there are several forms in our app that need this change and, ideally, we should make one change that fixes all the forms instead of doing each one individually. The way to do this would be to modify/override the bootstrap template we're already using.

Further along those lines, can we get the smaller form to be centered in its container by default?

app/Resources/views/user/register.html.twig

```
{% extends 'base.html.twig' %}

{% block body %}
   <div class="container">
      <div class="row">
         <div class="col-xs-12">
            <h1>Register!</h1>

            {{ form_start(form, {'attr':{'class':'myclass'}}) }}
            {{ form_row(form.username) }}
            {{ form_row(form.email)}}
            {{ form_row(form.plainPassword.first, {
               'label': 'Password'
            }) }}

            {{ form_row(form.plainPassword.second, {
               'label': 'Repeat Password'
            }) }}
            <a href="https://termsfeed.com/terms-
service/2f805086c3002ea1b07084f032c3bc5e">Terms of service</a>
            {{ form_row(form.IHaveReadAndAcceptTheTermsOfService) }}

            <a href="http://sfsuse.com/~sp17g05/classProject/web/">cancel</a>

            <button type="submit" class="btn btn-primary" formnovalidate>
               Register
            </button>

            <br></br>

            <p><a href="https://termsfeed.com/privacy-
policy/c455b06cbdd957a654630e5cd5ef93b6">Privacy Policy</a>


            {{ form_end(form) }}

         </div>
      </div>
   </div>
{% endblock %}
```

src/AppBundle/Form/UserRegistrationForm.php

```php
<?php

namespace AppBundle\Form;

use AppBundle\Entity\User;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\CheckboxType;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\Form\Extension\Core\Type\PasswordType;
use Symfony\Component\Form\Extension\Core\Type\RepeatedType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Validator\Constraints\IsTrue;

class UserRegistrationForm extends AbstractType
{
   public function buildForm(FormBuilderInterface $builder, array $options)
   {
      $builder
         ->add('email', EmailType::class)
         ->add('username', TextType::class)
         ->add('plainPassword', RepeatedType::class, ['type' => PasswordType::class])
         ->add('IHaveReadAndAcceptTheTermsOfService', CheckboxType::class, array(
      'mapped' => false,
      'constraints' => new IsTrue(),));
   }

   public function configureOptions(OptionsResolver $resolver)
   {
      $resolver->setDefaults([
         'data_class' => User::class,
         'validation_groups' => ['Default', 'Registration']
      ]);
   }
}
```

web/css/main.css

```css
.myclass {
   width: 200px;
}
```

5. Self-Check on Security Best Practices

Major Protected Assets

1. User Name
2. Email Address
3. User Password

Security Practices

As part of the Symfony framework all passwords are encrypted by default in the database and have been verified. Validation is in place to restrict registration to mail.sfsu.edu domain only. Symfony also validates all text fields to prevent various common web attacks. The application and database is deployed on Amazon Web Services and can only be accessed through Secure Shell. Access through Secure Shell is protected by a unique pre-shared key. Database access is restricted to local users meaning that it can only be reached through Secure Shell remotely. The Login and User Name is only known to the development team.

6. Adherence to Original Non-Functional Specs

1. Application shall be developed using class provided LAMP stack - **ON TRACK**
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 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, and Chrome. – **ON TRACK**
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 TRACK**
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 – **ON TRACK**
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. - **ON TRACK**
12. Google analytics shall be added - **DONE**
13. Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services. – **ON TRACK**
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**
18. Payment exchanges will not be done by the site and all exchanges shall be done by the user's choosing. - **DONE**
19. Browser versions that will be supported are Google Chrome (Version 57.0.2987 and 56.0.2924), Mozilla Firefox (Version 54.0a2 and 53.0), and Safari (Version 10.1 and 10). - **DONE**