GATOR JOE'S

Team 13

Milestone 4

Software Engineering CSC 648-848
Fall 2019 - Section 02

Piero Calenzani (Team Lead): ptcalenzani@gmail.com

Grant Kennedy (Front-end Lead)

Mark Hammond (Back-end Lead)

Chuhao 'William' Yang (Front-end)

Varun Sura (Front-end)

Zhuoxin Tan (Back-end)

Date Submitted	Date Revised	Description
12/12/2019		First draft

1. Product Summary

Gator Joe's Website – Gator Joe's is the online marketplace for SFSU students by SFSU students. GJ's is a local site for students in San Francisco State University that allows for buying and selling of various types of items including books, electronics, furniture and more.

Committed Functions

Unregistered Users:

- Users can view, search, and browse through all active listings
- Registration and login will be available to all new users, implementing lazy registration
- Website's unique feature is that it can specify an SFSU course and search for books specifically through that course
- Website will display the 12 most recent listings on the homepage
- Listings will display the uploaded image on its page and thumbnails in condensed views
- Users will be able to fill out a sell form for a listing and only be prompted to register after submitting their listing.
- Users can sort search results by price and category

Registered Users:

- Registered users can post listings and are prompted for the proper information in all categories
- Registered users can purchase items
- Registered users will be able to log in and log out
- Registered users can send and receive messages about listings
- Registered users can see their active listings as well as all recent listings on the homepage

Admin Users:

- Administrators will perform all their tasks through workbench
- Administrators can approve or remove listings
- Administrators can delete users

2. Usability Test Plan

Test Objectives

This usability test will focus on the search and results functionality of the website project. The search feature of the website is the most important aside from browsing because it is the quickest way to allow users to view items. Searching cuts browsing time significantly by allowing users to specifically check if an item they want is available. Website also allows search by category which further narrows the scope of a given search. In addition to correct functionality, the search and category fields must be persistent for ease of use. Correctly functioning search and results are vital to the project and must be thoroughly tested.

Test Background and Setup

System Setup: Project website supports Google Chrome and Mozilla Firefox browsers. No extensions or additional software required to use the website.

Starting Point: The search bar functionality is present in the navbar at the top of every webpage. In addition, it functions the same regardless of which page the user is searching from. Home page of the site reachable at http://3.17.190.122/.

Intended Users: The search bar is implemented for all users looking to purchase. The userbase for this website is focused towards students at SFSU ages 18-24. Expected computer skills are average but should not be technically demanding. Searching is intended to be the primary way for users to find items they need or want. Users looking to sell an item can also search for similar items to use as an example when creating their listings.

Evaluation: Efficiency in effort and design will be tested in this usability plan. Task should be done with minimal clicks and instructions. User satisfaction will be surveyed after the usability tests to measure ease of use and website clarity.

Usability Task Description

Open a Google Chrome or Firefox browser and navigate to the project page: http://3.17.190.122/. Perform the following on the website's search:

1. A search for all listings
2. A search for only Furniture listings
3. A search for listings related to phones
4. A search for furniture listings that are new (listed in title as new)

Review the search results for each search and verify that only relevant results are being shown. Correct results indicate successful completion of task criteria. Expected benchmark for full completion of task is 1.5 minutes.

Effectiveness would be measured by the amount of clicks the tester takes to perform each search in the task. By reviewing the click count of search, we can examine how effective each action is in our search functionality.

Efficiency would be measured by the amount of time taken and confusion caused by the usability task. A large amount of time taken would indicate that the interface for searching is unclear for first-time users to perform the given searches. Furthermore, if any one of those tasks takes more time than the others, we can pinpoint which functionality in our search needs to be reviewed.

Satisfaction would be measured by deploying the following survey after the task is completed:

Question	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
It was easy to use the search bar and category dropdown.					
The search results and listings were clearly displayed.					
It was preferable to use the search over browsing.					

3. QA Test Plan

Test Objectives

The functionality of using the website's search and results will be tested for bugs. Input validation on the search bar will be tested to only take up to 40 characters. The search and category selected after a search should remain consistent on the results page. Search and category menu should be fully functional from any page in the website. Only correct and relevant results should be displayed for each tested search. Empty searches should return all found listings.

HW and SW setup

Website should function on any machine connected to the internet that can run Google Chrome or Firefox browsers.

Website must support the latest two versions of Google Chrome and Firefox. Browsers must be updated to ensure this. No further software required for testing.

Feature to be tested

In the browser's address field, navigate to http://3.17.190.122/. At the top left of all webpages in the site, the search bar will be used for four tests. The following example test searches are to be tested:

1. Search with Category 'All' and empty search field. This must yield all 12 listings in the results.
2. Search with Category 'Furniture' and empty search field. This must yield all 4 results in the furniture category.
3. Search with Category 'All' and search field with text 'phone'. This must yield a single result for a used iphone.
4. Search with Category 'Furniture' and search field with text 'new'. This must yield two furniture results with titles including 'new'.

Expected time for completion of this test is 1 minute.

QA Test plan (Chrome):

Test #	Description	Input	Expected Output	Pass/Fail
1	Empty search under 'All' categories	▪ None	All 12 listings	
2	Empty search under 'Furniture' category	▪ 'Furniture' category	All 4 furniture listings	
3	'Phone' search under 'All' categories	▪ 'Phone' search text	One listing for 'iphone'	
4	'New' search under 'Furniture' category	▪ 'New' search text ▪ 'Furniture' category	Two listings for 'new' under furniture	

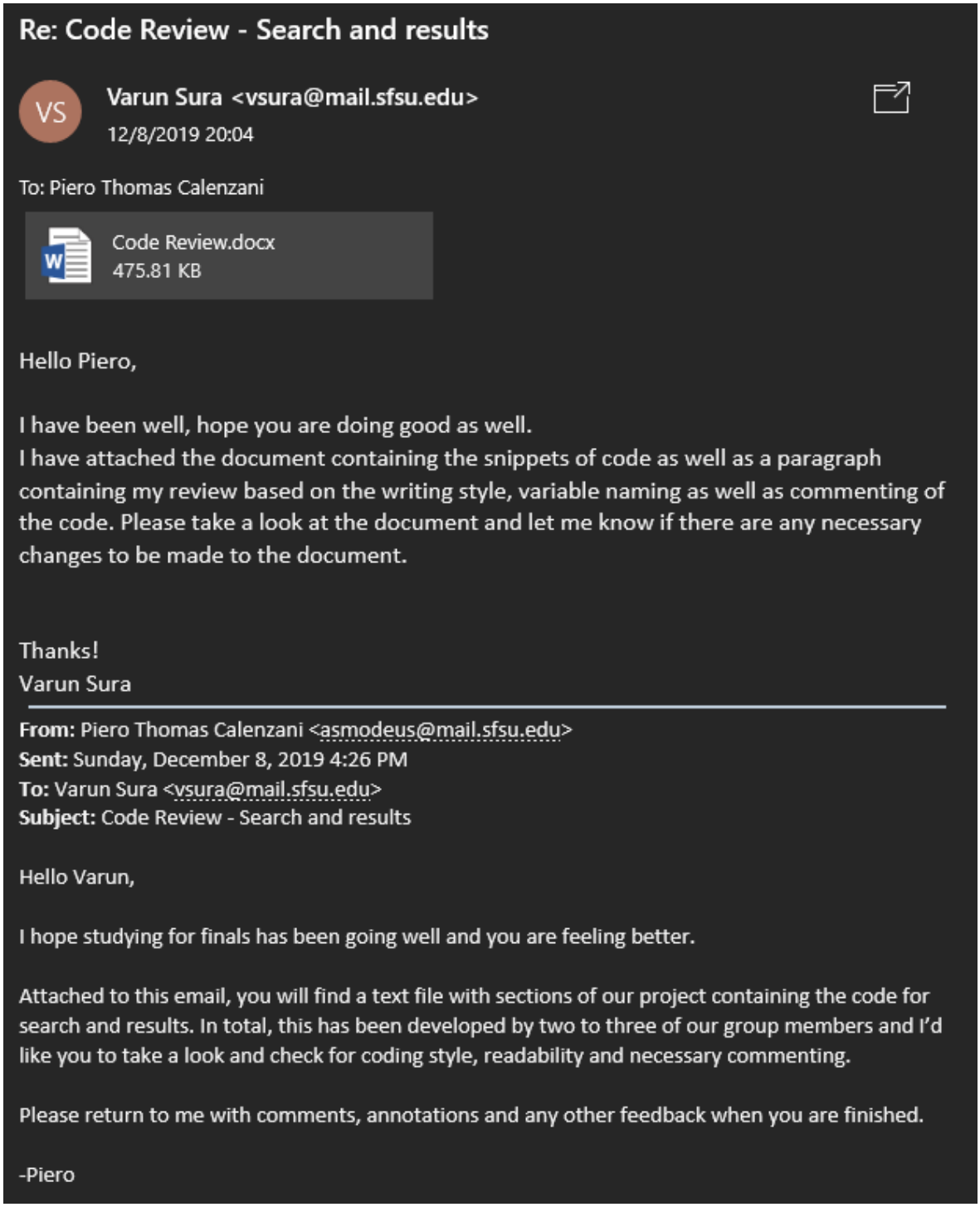
QA Test plan (Firefox):

Test #	Description	Input	Expected Output	Pass/Fail
1	Empty search under 'All' categories	▪ None	All 12 listings	
2	Empty search under 'Furniture' category	▪ 'Furniture' category	All 4 furniture listings	
3	'Phone' search under 'All' categories	▪ 'Phone' search text	One listing for 'iphone'	
4	'New' search under 'Furniture' category	▪ 'New' search text ▪ 'Furniture' category	Two listings for 'new' under furniture	

4. Code Review

Coding Style for project implements PEP 8 (Style for Python Code). More information can be found here: https://www.python.org/dev/peps/pep-0008/.

Code Review for search and results done by Varun Sura (front-end developer) via email:



Re: Code Review - Search and results

VS Varun Sura <vsura@mail.sfsu.edu>
12/8/2019 20:04

To: Piero Thomas Calenzani

Code Review.docx
475.81 KB

Hello Piero,

I have been well, hope you are doing good as well.
I have attached the document containing the snippets of code as well as a paragraph containing my review based on the writing style, variable naming as well as commenting of the code. Please take a look at the document and let me know if there are any necessary changes to be made to the document.

Thanks!
Varun Sura

From: Piero Thomas Calenzani <asmodeus@mail.sfsu.edu>
Sent: Sunday, December 8, 2019 4:26 PM
To: Varun Sura <vsura@mail.sfsu.edu>
Subject: Code Review - Search and results

Hello Varun,

I hope studying for finals has been going well and you are feeling better.

Attached to this email, you will find a text file with sections of our project containing the code for search and results. In total, this has been developed by two to three of our group members and I'd like you to take a look and check for coding style, readability and necessary commenting.

Please return to me with comments, annotations and any other feedback when you are finished.

-Piero

Attached Code Review file:

```
****************** Webpage to handle search results (routing.py) ******************
@app.route('/results', methods=["GET", "POST"])
def results():
    if request.method == 'POST':
        if request.form['btn'] == 'search':
            searchTerm = request.form['search']
            cat = request.form['category']
            searchResults = Listing.get_results(searchTerm, cat)

            return render_template("results.html",
                                   title="Gator Joe's - {} results".format(searchTerm),
                                   category_names=Category.get_all_categories(),
                                   auth=current_user.is_authenticated,
                                   searchquery=searchTerm, searchcat=cat, results=searchResults,
                                   num_results=len(searchResults))
        else:
            return postRouting()
    elif request.args:
        searchTerm = request.args.get('search')
        cat = request.args.get('cat')
        searchResults = Listing.get_results(searchTerm, cat)

        return render_template("results.html",
                               title="Gator Joe's - {} results".format(searchTerm),
                               category_names=Category.get_all_categories(),
                               auth=current_user.is_authenticated,
                               searchquery=searchTerm, searchcat=cat, results=searchResults,
                               num_results=len(searchResults))
    else:
        return redirect(url_for('homepage'))
```

Fig: Code snippet for search

```
****************** Html search bar with category dropdown (header.html) ******************
<form action="/results" class="row form-inline mr-auto" method="post" name="search">

    <select class="custom-select my-1 mr-1" id="inlineFormCustomSelectPref"
            name="category">

        <!-- Following jinja logic will check given form category and-->
        <!-- make dropdown menu persistent-->
        <option
        {% if not request.form['category'] %}selected{% endif %}>All</option>

        {% for cat in category_names %}

        <option value="{{ cat }}"
        {% if request.form['category'] == cat %}selected{% endif %}>{{cat}}</option>

        {% endfor %}


    </select>
    <input class="form-control mr-1" type="text" placeholder="Enter Search Here"
           name="search" value="{{request.form.search}}", maxlength="40",
           oninvalid="alert('Please enter upto 40 characters only!')">
    <button class="btn btn-outline-warning my-0" type="submit" name="btn" value="search"><span
            class="oi oi-magnifying-glass"></span></button>

</form>
```

Fig: Code snippet for Search bar

```
****************** Html for displaying search results (results.html) ******************
<!--Renders this block if there are any results found. -->
{% if num_results > 0 %}
<div class="row">
    <div class="col-sm-12">
    <h2>Showing {{ num_results }} Results: {{ searchquery }}</h2>
    </div>
</div>

<div class="row py-4">
    {% for listing in results %}
    <div class="col-sm-6 col-md-4 col-lg-3">
        <div class="row h-100">
            <div class="col-sm-12 mb-3">
                <div class="card h-100 py-4">
                    <img src="/static/resources/thumbnotfound.jpg" class="rounded mx-auto d-block" width="200px">
                    <div class="card-body">
                        <h5><a href="{{ '/listing/%s'%listing.id }}" class="card-title" target="_blank">{{ listing.title }}</a>
</h5>
                        <b class="card-text">${{ listing.sell_price }}</b>
                        <p class="card-text">{{ listing.description }}</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
    {% endfor %}
</div>
<!-- Renders this block if there are no results for the search -->
{% elif num_results == 0 %}
<div class="row">
    <div class="col-sm-12">
    <h2 align="center">Sorry! No Results found for: "{{ searchquery }}"</h2>
    <h3 align="center">Please try another search</h3>
    </div>
</div>
{% endif %}
```

Fig: Code snippet for displaying the results.

Review written by Varun:

Overall the code is neatly written and and indentations also improved the readability greatly. It follows the coding style and the variables are also named so that the purpose of the variables is easily understood by looking at the code which makes it easy to skim through the code effectively and implement any new functionalities with ease. Some of the sections of the code are also well commented which helps with maintenance process as well as the debugging process.

The code is also follows the writing style so that new functionalities can be implemented and the existing ones can also be changed with minimal effort and difficulties.

5. Self-Check on Best Practices for Security

User Passwords

Passwords will be encrypted, and only the hashed strings will be kept in the database.

Registration Validation

All fields in the registration will be properly validated and unwanted characters **(),<>\|-;** will be rejected.

Search Bar Validation

Search bar text field will only accept 40 characters maximum.

Listing Validation

Titles for listings are limited to 40 characters and are restricted from unwanted characters **(),<>\|-;** .

Book Selling Validation

The field for course number in the book sell form will only accept up to 3 numbers.

Messaging Validation

User messages will reject unwanted characters **(),<>\|-;** when attempting to send text.

6. Adherence to Non-Functional Specs

1. **DONE** Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO).

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

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

4. **DONE** Data shall be stored in the team's chosen database technology on the team's deployment server.

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

6. **ON TRACK** Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.

7. **DONE** The language used shall be English.

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

9. **DONE** Google analytics shall be added

10. **DONE** No e-mail clients shall be allowed

11. **DONE** Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.

12. **DONE** Site security: basic best practices shall be applied (as covered in the class)

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

14. **DONE** The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Fall 2019. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application).