SW Engineering CSC 648[01] Fall 2019

GatorTrader

Team 2
Sean Darryanto - Team Lead, Backend Lead | darryanto@mail.sfsu.edu
Aaditya Chokshi - Frontend Lead
Momo Sun - Frontend SWE
David Chung - Frontend SWE, Github Master
Christopher Smith - Backend SWE
Adam Bea - Backend SWE
Kam Khai - Fullstack SWE

Milestone 4
12/11/2019

Revisions

December 11, 2019	Initial Draft

Table of Contents

Executive Summary

GatorTrader is a classified advertisement and marketplace web application that aims to provide the easiest way for San Francisco State University students to buy and sell items within the context of the college community. Through simple yet elegant interfaces, students will be able to connect and together over a variety of shared networks.

Gator Trader's unique feature is that users can obtain helpful recommendations based on the major they register under when they create an account.

GatorTrader has a variety of features:

- Guest users shall be able to browse through the website without logging in.
- Guest users who want to register shall need to be verified by SFSU email.
- Guest users shall be able to contact the sellers only after being redirected to login.
- Guest users shall be able to contact the administrator.
- Guest users shall be able to create an account.
- Guest users shall be able to create post data but must be logged in first.
- Registered users shall be able to contact sellers.
- Registered users shall be able to log in their account by username/ email and password.
- Registered users shall be able to reset their password if they forget.
- Registered users shall be able to create listings and posts on the website.
- Registered users shall be able to upload pictures of the items they want to sell.
- The administrator shall be able to postpone the posting if they violate the policy of the website.
- The administrator shall be able to access the database.
- The administrator shall be able to contact the sellers directly.
- The administrator shall be able to reply to messages from Guest users.
- The site shall present the Terms, Conditions and Privacy Policy during the registration.

Usability Test Plan

Test Objective:

- The objective of this test is to expose the efficiency of *GatorTrader*'s search functionality.
- The search functionality of *GatorTrader* is one of the most important functions of the *GatorTrader* web application. It allows the user to easily search for sellers with their particular item in mind.

Test Background and Setup:

System Setup

- The tester will need access to a computer with access to both the internet and one of the following internet browsers: Google Chrome or Mozilla Firefox.

Starting Point

- The tester's entry point for the test will be *GatorTrader*'s homepage, where the search function to be tested will be promptly displayed.

Tasks to be Accomplished

- The tester will attempt to search for all listings available.
- The tester will attempt to search for all listings within a category of their choosing.
- The tester will attempt to search for a specific listing from all listings available.

Intended Users

- The intended user is any individual who is a part of the current student or alumni community at San Francisco State University.
- It is not assumed that the intended user has any technical skills or any advanced knowledge or experience in the context of web browsing.

Completion Criteria

- The tester will indicate whether or not they accomplished the intended search query by completing a questionnaire after the attempt.

URL of System

- http://13.52.236.160/

Usability Questionnaire

1. **I found the Search function easy to use (Choose one)**

____Strongly Disagree ____Disagree ____Neutral

____Agree ____Strongly Agree

2. **I found all the listings available (Choose one)**

____Strongly Disagree ____Disagree ____Neutral

____Agree ____Strongly Agree

3. **I found the listings for a specific category I searched for (Choose one)**

____Strongly Disagree ____Disagree ____Neutral

____Agree ____Strongly Agree

4. **I found the specific listing I was searching for (Choose one)**

____Strongly Disagree ____Disagree ____Neutral

____Agree ____Strongly Agree

QA Test Plan

Test Objective

1. The goal of this test is to validate the functioning of the *get_listings* function, one of the core features of the website.

2. The function will be validated through repeated testing, and comparing the outcomes against expected results.

3. The tester shall be given a set of specific instructions to follow, and the result of each set shall be recorded as a PASS or FAIL based on set-specific conditions.

Hardware and Software Requirements

1. The QA tester shall have access to a desktop or laptop computer that supports the current versions of two web browsers; Mozilla Firefox and Google Chrome.

2. The computer shall be connected to the internet via Wi-Fi, Ethernet cable, Cellular connection, etc.

3. The internet connection download speed should be adequate for browsing the internet, and checking emails (at least 5.0 Mbps).

4. After the above requirements have been met, the tester shall open the following link on either Mozilla Firefox, or Google Chrome to begin the test

   - http://13.52.236.160

Features to be Tested

The test will attempt to validate the *get_listing* feature of the site. If done properly, the tester shall be able to view all listings, and check to see that their information is accurate.

As of December 10th, 2019, the database contains 4 tables with sufficient data to run all tests below.

Test #	Test Title	Description	Test Input	Expected Result	Result (Firefox)	Result (Chrome)
1	Display all Listings	The tester shall click the yellow search button in the upper right corner of the page, and count the results.	NA	- 9 Listings should be displayed	**PASS**	**PASS**
2	Display Book Listings	The tester shall click the dropdown tab to the right of the search bar, and select “Books”. They will then click the yellow search button and count the results.	NA	- 2 Listings should be displayed - Math Textbook - Physics Textbook	**PASS**	**PASS**
3	Display Tool Lisings	The tester shall click the dropdown tab to the right of the search bar, and select “Tools”. They will then click the yellow search button and count the results.	NA	- 1 Listing should be displayed - Screwdriver	**PASS**	**PASS**

Code Review

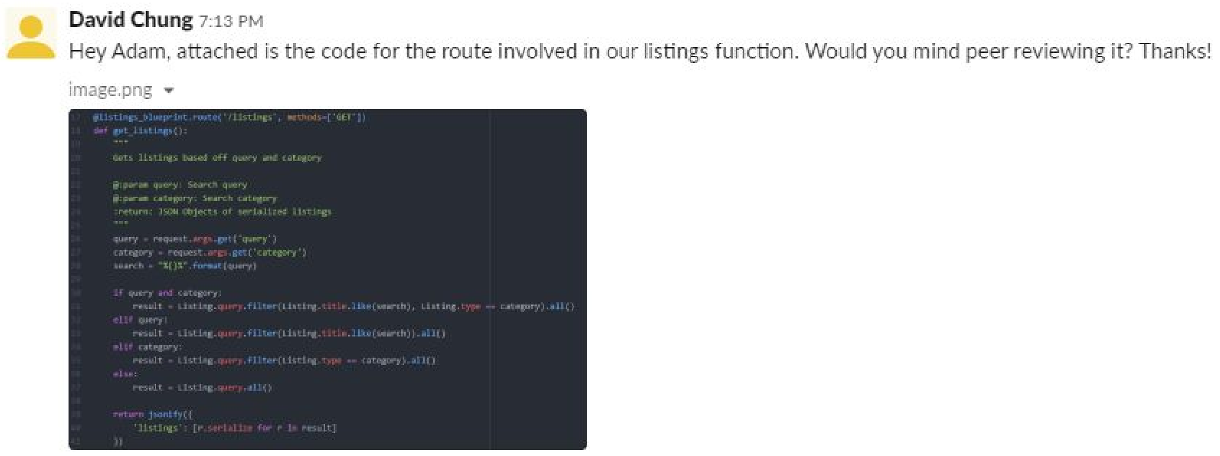
Coding Style:

- Use of underscores (_) to represent spaces in multi-word file and variable names in Python, use of camelCase in Javascript.
- Consistent spacing to represent indentation within code blocks:
    - 2 spaces for Frontend
    - 4 spaces for Python
- All files housed in proper folders pertaining to their use within our project
- Consistent capitalization for HTML element names, and proper closing of all said elements
- Defined pixel height/width for images
- Language and character encoding defined at the beginning of documents
- Descriptive Python file names, with underscores to represent spaces
- GitHub commits pushed with clear description of changes made
- Input checking for both length and alphanumeric validity

Code Review:

We chose to do a code review on our listings function. This function allows us to retrieve listings that match the parameters given to us by the user and return them.

Function: Get Listings
Location: application/server/routes/listings.py



David Chung 7:13 PM
Hey Adam, attached is the code for the route involved in our listings function. Would you mind peer reviewing it? Thanks!

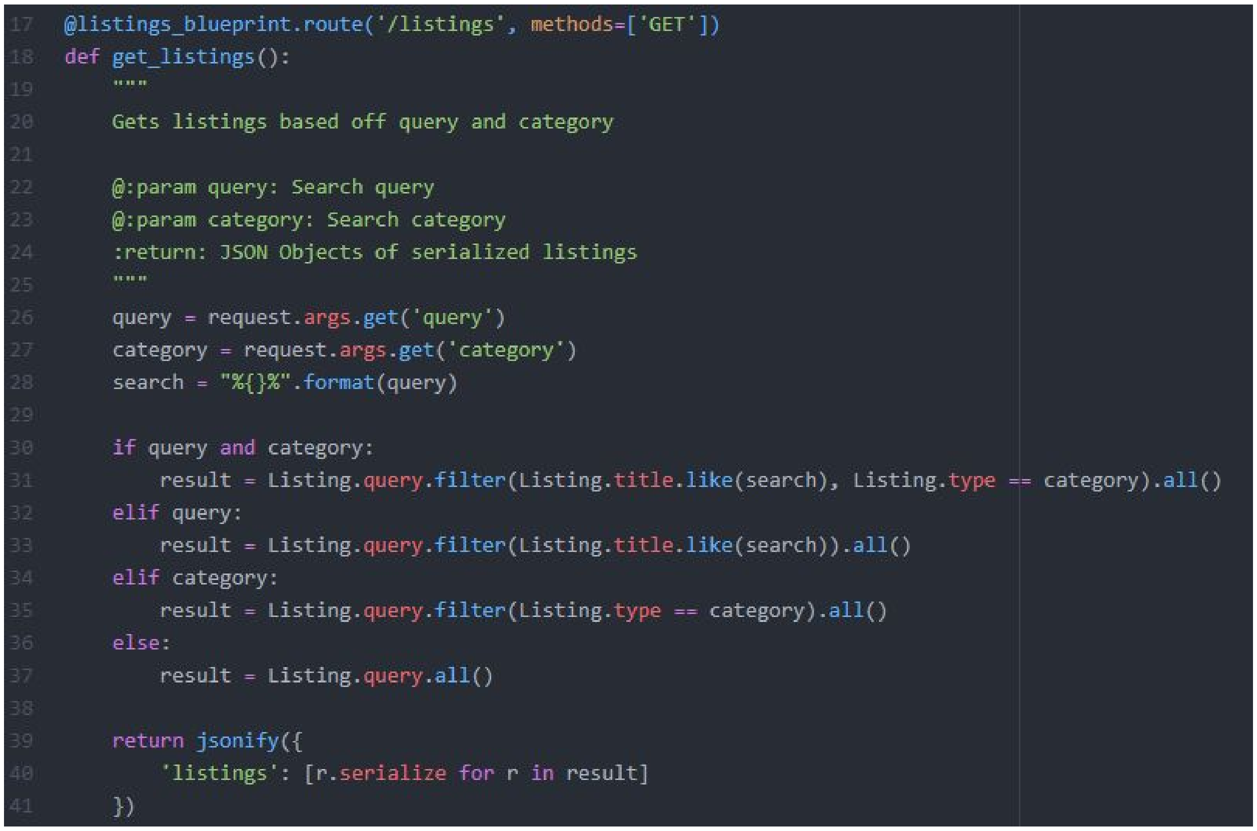
image.png

```python
@listings_blueprint.route('/listings', methods=['GET'])
def get_listings():
    """
    Gets listings based off query and category

    @:param query: Search query
    @:param category: Search category
    :return: JSON Objects of serialized listings
    """
    query = request.args.get('query')
    category = request.args.get('category')
    search = "%{}%".format(query)

    if query and category:
        result = Listing.query.filter(Listing.title.like(search), Listing.type == category).all()
    elif query:
        result = Listing.query.filter(Listing.title.like(search)).all()
    elif category:
        result = Listing.query.filter(Listing.type == category).all()
    else:
        result = Listing.query.all()

    return jsonify({
        'listings': [r.serialize for r in result]
    })
```



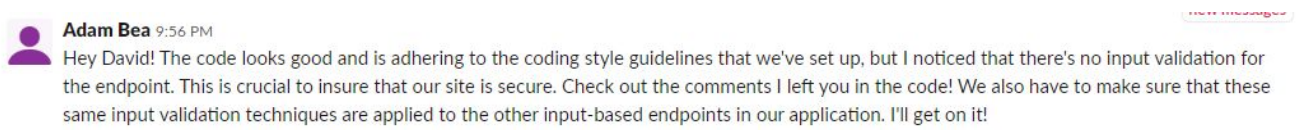
```python
@listings_blueprint.route('/listings', methods=['GET'])
def get_listings():
    """
    Gets listings based off query and category

    @:param query: Search query
    @:param category: Search category
    :return: JSON Objects of serialized listings
    """
    query = request.args.get('query')
    category = request.args.get('category')
    search = "%{}%".format(query)

    if query and category:
        result = Listing.query.filter(Listing.title.like(search), Listing.type == category).all()
    elif query:
        result = Listing.query.filter(Listing.title.like(search)).all()
    elif category:
        result = Listing.query.filter(Listing.type == category).all()
    else:
        result = Listing.query.all()

    return jsonify({
        'listings': [r.serialize for r in result]
    })
```

Reviewed By: Adam Bea



Adam Bea 9:56 PM
Hey David! The code looks good and is adhering to the coding style guidelines that we've set up, but I noticed that there's no input validation for the endpoint. This is crucial to insure that our site is secure. Check out the comments I left you in the code! We also have to make sure that these same input validation techniques are applied to the other input-based endpoints in our application. I'll get on it!

Security Practices

The major assets we are protecting all pertain to user information, which include the following:

- Passwords

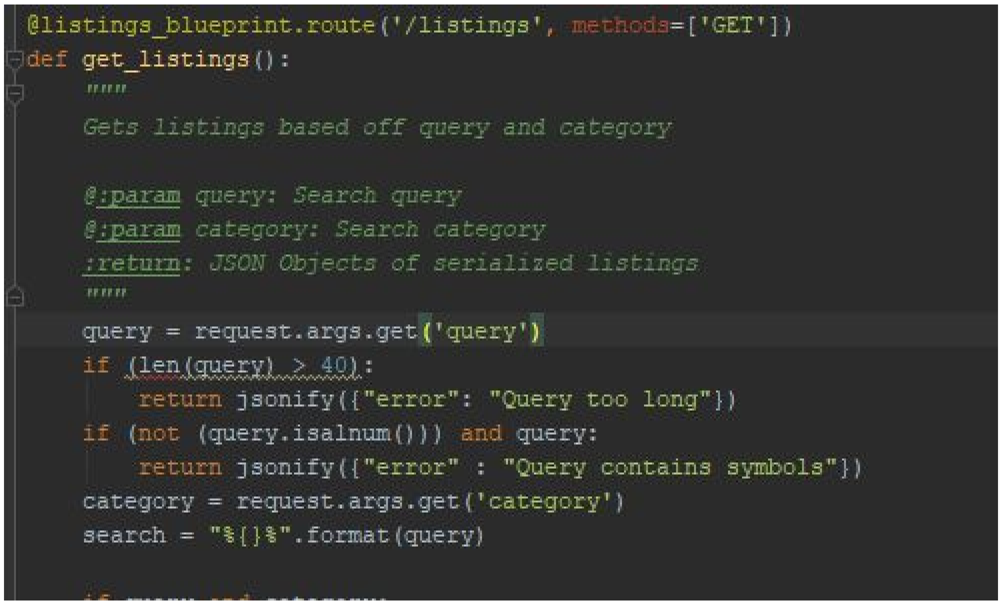
Password Protection:

We are using the **Werkzeug python library** to hash user passwords.

Regarding SQL Injection attacks, we are utilizing flask-SQLAlchemy libraries such that we never explicitly send SQL queries.

Passwords **are** encrypted within our database.

Input data **is** validated (see below).



```
@listings_blueprint.route('/listings', methods=['GET'])
def get_listings():
    """
    Gets listings based off query and category

    @:param query: Search query
    @:param category: Search category
    :return: JSON Objects of serialized listings
    """
    query = request.args.get('query')
    if (len(query) > 40):
        return jsonify({"error": "Query too long"})
    if (not (query.isalnum())) and query:
        return jsonify({"error" : "Query contains symbols"})
    category = request.args.get('category')
    search = "%{}%".format(query)
```

Adherence to Non-functional Specs

Non Functional Requirement	Done	On Track	Issue
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).		X	
Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers	X		
Selected application functions must render well on mobile devices		X	
Data shall be stored in the team's chosen database technology on the team's deployment server.	X		
No more than 50 concurrent users shall be accessing the application at any time	X		
Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.	X		
The language used shall be English.	X		
Application shall be very easy to use and intuitive.	X		
Google analytics shall be added	X		
No email clients shall be allowed	X		

Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	X		
Site security: basic best practices shall be applied (as covered in the class)	X		
Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development	X		
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).	X		