San Francisco State University

SW Engineering CSC648/848 Fall 2020

GatorGoods

Team 08 Global

Keith Eastman, Team Lead || keastman@mail.sfsu.edu

Trenton Smith, GitHub Master | Editor

Zhuozhuo (Joy) Liu, Front-end Lead

Yugyeong (YG) Lee, Back-end Lead

In coordination with Fulda University, Germany

Milestone 4

December 8, 2020

Date Submitted	Dates Revised
12/08/2020	12/10/2020

1. Product Summary

- **Name:** GatorGoods
- **Coordination:** GatorGoods is developed by a joint Global team comprised of students from San Francisco State University, CA, and Fulda University, Germany.
- **URL:** http://ec2-54-153-71-183.us-west-1.compute.amazonaws.com:5000/
- **Committed Function List:**
  - **Unregistered Users**
    1. Users shall be able to browse items by category
    2. Users shall be able to search items by category and/or text
    3. Users shall be able to filter browse/search results
    4. Users shall be able to register accounts by providing their username, password, and San Francisco State University email address
  - **Registered Users**
    1. Users shall have complete functionality of an Unregistered User plus the following:
    2. Users shall be able to log into their unique accounts by providing their username and password
    3. Users shall be able to create product listings
    4. Users shall be able to select from a listing of safe meeting places around the San Francisco State University campus in order to meet with a buyer or seller (unique to GatorGoods)
    5. Users shall be able to contact the selling parties of product listings
    6. Users shall be able to access their dashboard where they can view all their product listings, read messages from prospective buyers, and/or choose to delist a product listing
  - **Admin Users**
    1. Users must moderate content on the site through the following means:
       1.1. Deleting a pending product listing which violates the terms and agreements (Fulda team responsible)
       1.2. Approving a pending product listing which conforms to the terms and agreements will allow it to be published (Fulda team responsible)

2. Usability Test Plan

2.1. Test Objectives

The feature that we selected to test is search. The goal of the usability testing is assessment - to evaluate the usability of a basic but fundamental feature. In specific, the purpose of the usability testing is to measure the effectiveness, efficiency, and

satisfaction of the users regarding the feature. The results will help us gain knowledge on how well the search feature is implemented and if needed, insight into how to potentially replan our product road map.

2.2. Test Background and Setup

- **System setup:** The website shall be able to run on browsers including Google Chrome and Safari.
- **Starting point:** Users should start from the homepage of GatorGoods while on Chrome or Safari. We will then ask the user to search for “used chairs” without giving them any further instruction.
- **Intended users:** We shall recruit 10 participants comprised of SFSU students, faculties and staff. The number of participants in each category (student, faculty, and staff) must be no less than 3.
- **URL:** http://ec2-54-153-71-183.us-west-1.compute.amazonaws.com:5000/ Since we are starting from the home page, this is the starting URL as well.
- **What is to be measured:** We will measure the time duration in which users complete the task, the number of users who completed or didn’t complete the task, and user satisfaction.

2.3. Usability Task Description

- **Scenario:** You’ve heard that GatorGoods is an online marketplace for SFSU students, so you open your web browser of choice (either Chrome or Safari) and navigate to its homepage. Once there, you decide to search for used chairs…

Task:	Search GatorGoods for used chairs.
Prompt:	Given the above task, what would your action be? Please take your action.
Machine state:	Search results properly loaded.
Successful completion criteria:	Return all 4 product listings for chairs on the search results page.
Benchmark:	Completed in 10 seconds.

- **Effectiveness:**
  The criteria of successful completion is to “return all 4 product listings for chairs on the search results page”. We have two metrics to measure the effectiveness of the search feature.
  1. The percentage of participants who complete the task in 10 seconds.
  2. The average number of errors each participant makes.
- **Efficiency:**
  The criteria of successful completion is to “return all 4 product listings for chairs on the search results page”. We have two metrics to measure the efficiency of the search feature.
  1. Average number of clicks it took users who successfully completed the task.
  2. Amongst those who successfully completed the task, measure the average time it took for completion.
- **Satisfaction:**
  To measure user satisfaction, we have a Likert scale with 3 survey questions, and an additional open-ended question to collect qualitative feedback.
  1. To what extent do you agree with the following statement:

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I was able to locate the searchbar easily.					
I was able to search for the desired listings.					
I would have no trouble using the searchbar to find further listings.					

  2. (Optional) Was there anything in your search process we could improve? If so, what?

3. QA Test Plan

- **Test objectives:** Assessing if search performs as per specs.
- **HW and SW setup (including URL):**
  - Hardware: MacOS
  - Software:
    - AWS EC2 instance running Ubuntu Server 18.04

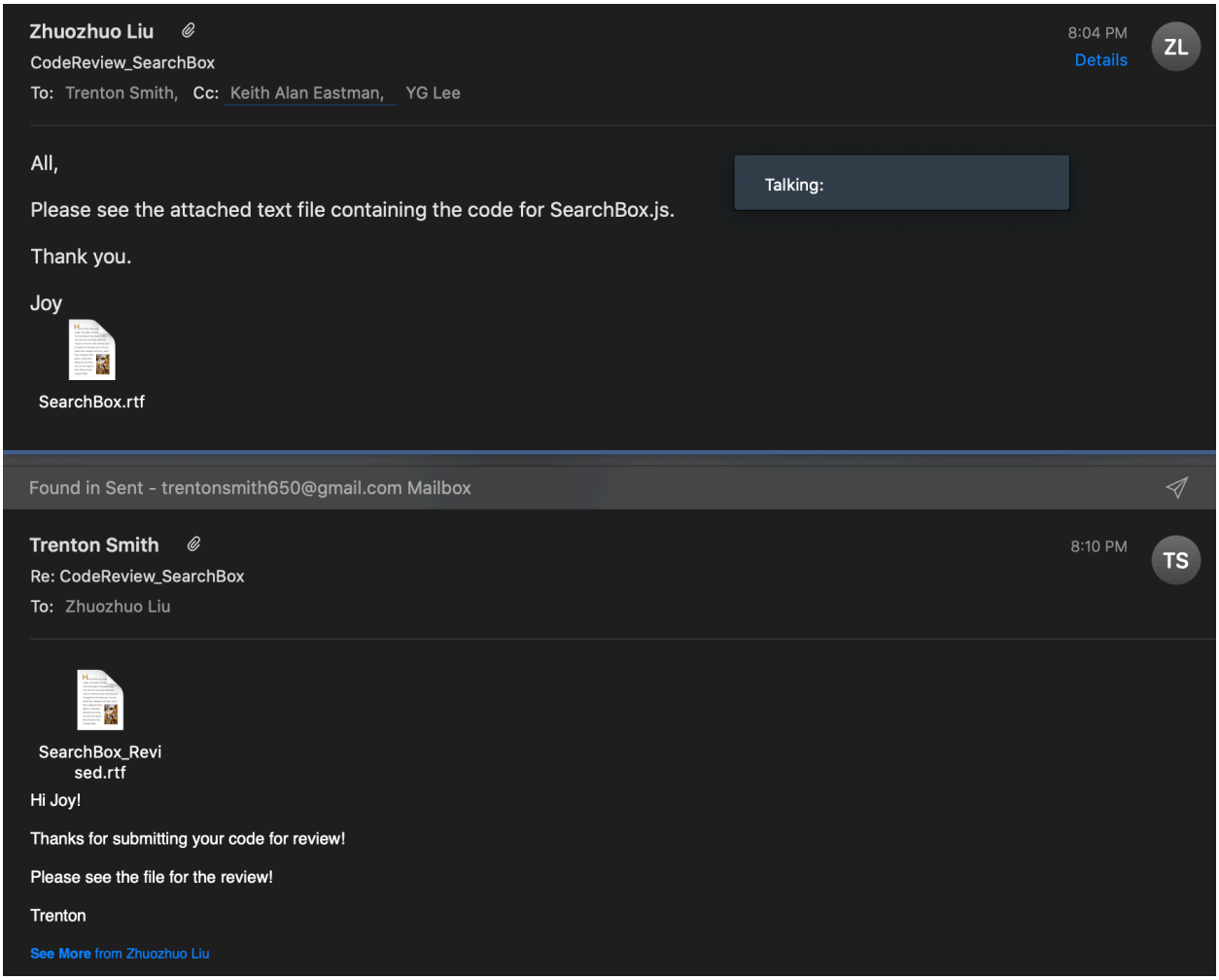
    - Browsers: Chrome, Safari
  - URL: http://ec2-54-153-71-183.us-west-1.compute.amazonaws.com:5000/
- **Feature to be tested:** Search

- **QA Test plan - table format:**

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Fuzzy Search	Testing if correct results display after a text search without selecting a category	Type “human” into the searchbar without selecting a category	A single product listing for “Humans” will be returned	Chrome: PASS Safari: PASS
2	Max Character	Testing if the searchbar will allow a text string over 40 characters to be entered	Attempt to type a string longer than 40 characters into the searchbar	The string will not be accepted as a valid search parameter	Chrome: PASS Safari: PASS
3	Category Search	Testing if a specific text AND category search will return results when there are no matches	Type “human” into the searchbar after selecting “Furniture” from the category dropdown	No product listings will be returned and the view will specify “0 Results” for the search	Chrome: PASS Safari: PASS

4. Code Review

4.1. Email Chain



4.2. Initial Document

4.2.1. Part A

```
import React, { useState, useEffect } from "react";
import { useHistory } from "react-router-dom";
import axios from "axios";
import { Button, InputGroup, FormControl, Form } from "react-bootstrap";



export default function Searchbox() {
  const [searchTerm, setSearchTerm] = useState("");
  const [category, setCategory] = useState("");
  const [productListings, setProductListings] = useState([]);
  const history = useHistory();

  // useEffect(() => {
  //   const getListings = () => {
  //     axios
  //       .post("/api/search/searchProducts", {
  //         searchTerm: searchTerm,
  //         category: category,
  //       })
  //       .then((response) => {
  //         // console.log(response.data);
  //         setProductListings(response.data);
  //       });
  //   };
  //   getListings();
  //   // console.log(searchTerm);
  // }, [searchTerm, category]);

  const handleCategoryChange = (e) => {
    setCategory(e.target.value);
  };

  const handleInputChange = (e) => {
    setSearchTerm(e.target.value);
  };

  // onSearch -> redirect to result page
  const onSearch = () => {
    history.push("/searchresults", {
      // productListings: productListings,
      category: category.valueOf(),
      searchTerm: searchTerm,
    });
  };

  return (
    <div>
      {/* Search Bar */}
      <InputGroup className="mx-auto" >
        <Form.Group style={{ width: "8rem" }}>
          <Form.Control
            as="select"
            name="category"
            value={category}
            onChange={handleCategoryChange}
          >
            <option value={""}>Category</option>
            <option value={"1"}>Books</option>
            <option value={"2"}>Furniture</option>
```

4.2.2. Part B

```
const handleCategoryChange = (e) => {
  setCategory(e.target.value);
};

const handleInputChange = (e) => {
  setSearchTerm(e.target.value);
};

// onSearch -> redirect to result page
const onSearch = () => {
  history.push("/searchresults", {
    // productListings: productListings,
    category: category.valueOf(),
    searchTerm: searchTerm,
  });
};

return (
  <div>
    {/* Search Bar */}
    <InputGroup className="mx-auto" >
      <Form.Group style={{ width: "8rem" }}>
        <Form.Control
          as="select"
          name="category"
          value={category}
          onChange={handleCategoryChange}
        >
          <option value={""}>Category</option>
          <option value={"1"}>Books</option>
          <option value={"2"}>Furniture</option>
          <option value={"3"}>Electronics</option>
        </Form.Control>
      </Form.Group>

      <Form.Group inline style={{ width: "15rem" }}>
        <FormControl
          placeholder="Search.."
          className="mr-sm-2"
          maxLength="30"
          value={searchTerm}
          onChange={handleInputChange}
        />
      </Form.Group>

      <Form.Group>
        <InputGroup.Append>
          <Button variant="outline-secondary" onClick={onSearch}>
            Search
          </Button>
        </InputGroup.Append>
      </Form.Group>
    </InputGroup>
  </div>
);
}
```

4.3. Revised Document

4.3.1. Part A

```
import React, { useState, useEffect } from "react";
import { useHistory } from "react-router-dom";
import axios from "axios";
import { Button, InputGroup, FormControl, Form } from "react-bootstrap";


//Don't forget to include the header comment - see our discord for the specified format
//at the least it should contain the file name, purpose, and author.

export default function Searchbox() {
//good job at initializing the constants at the top of the code
//also nice consistency in naming
  const [searchTerm, setSearchTerm] = useState("");
  const [category, setCategory] = useState("");
  const [productListings, setProductListings] = useState([]);
  const history = useHistory();

//If you submit commented-out code, please provide a brief description of…
//why it is still needed or what its purpose is e.g. "testing purposes"
  // useEffect(() => {
  //   const getListings = () => {
  //     axios
  //       .post("/api/search/searchProducts", {
  //         searchTerm: searchTerm,
  //         category: category,
  //       })
  //       .then((response) => {
  //         // console.log(response.data);
  //         setProductListings(response.data);
  //       });
  //   };
  //   getListings();
  //   // console.log(searchTerm);
  // }, [searchTerm, category]);

  const handleCategoryChange = (e) => {
    setCategory(e.target.value);
  };

//Suggestion: provide a more specific function title consistent with declared state e.g. handle"SearchTerm"Change
  const handleInputChange = (e) =>
    setSearchTerm(e.target.value);
  };

  // onSearch -> redirect to result page //like the use of comment explaining feature
  const onSearch = () => {
    history.push("/searchresults", {
      // productListings: productListings,
      category: category.valueOf(),
      searchTerm: searchTerm,
    });
  };

  return (
    <div>
      {/* Search Bar */} //like the use of comments labeling sections of code
      <InputGroup className="mx-auto" >
        <Form.Group style={{ width: "8rem" }}>
          <Form.Control
            as="select"
```

4.3.2. Part B

```
    setCategory(e.target.value);
  };

//Suggestion: provide a more specific function title consistent with declared state e.g. handle"SearchTerm"Change
  const handleInputChange = (e) =>
    setSearchTerm(e.target.value);
  };

  // onSearch -> redirect to result page //like the use of comment explaining feature
  const onSearch = () => {
    history.push("/searchresults", {
      // productListings: productListings,
      category: category.valueOf(),
      searchTerm: searchTerm,
    });
  };

  return (
    <div>
      {/* Search Bar */} //like the use of comments labeling sections of code
      <InputGroup className="mx-auto" >
        <Form.Group style={{ width: "8rem" }}>
          <Form.Control
            as="select"
            name="category"
            value={category}
            onChange={handleCategoryChange}
          >
            <option value={""}>Category</option>
            <option value={"1"}>Books</option>
            <option value={"2"}>Furniture</option>
            <option value={"3"}>Electronics</option>
           //be sure to include an option for "other" w/value 4
          </Form.Control>
        </Form.Group>

        <Form.Group inline style={{ width: "15rem" }}>
          <FormControl
            placeholder="Search.."
            className="mr-sm-2"
            maxLength="30" //should be 40 per reqs.
            value={searchTerm}
            onChange={handleInputChange}
          />
        </Form.Group>

        <Form.Group>
          <InputGroup.Append>
            <Button variant="outline-secondary" onClick={onSearch}>
              Search
            </Button>
          </InputGroup.Append>
        </Form.Group>
      </InputGroup>
    </div>
  );
}
```

5. Self-check: Security Best Practices

Asset	Value	Exposure
User Data	High	High, harm to reputation
Database	High	High, Loss of data, Compromise of server
Server	High	High, harm to reputation, loss of customers due to downtime

Threat and Control Analysis

Threat	Probability	Control	Feasibility
Unauthorized user gains access to confidential data	High	1. Input validation 2. Limit search queries to 40 characters 3. Authenticate users with encrypted tokens 4. Encrypt passwords in database 5. Images are stored as BLOBS in database	Moderate cost, moderately difficult implementation
Unauthorized user makes system unavailable	Low	1. Host server on virtual machine	Low cost, simple
Unauthorized content is uploaded by users	High	1. Require product listings to be vetted by administrator	Low cost, simple
Fake Accounts	High	1. Require a valid SFSU email address for new users	Low cost, simple

6. Self-check: Adherence to Original Non-Functional Specs

1. 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)
   ON TRACK
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers
   ON TRACK
3. All or selected application functions must render well on mobile devices
   ON TRACK
4. Data shall be stored in the database on the team's deployment server
   ON TRACK
5. No more than 50 concurrent users shall be accessing the application at any time
   ON TRACK
6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users
   ON TRACK
7. The language used shall be English (no localization needed)
   ON TRACK
8. Application shall be very easy to use and intuitive
   ON TRACK
9. Application should follow established architecture patterns
   ON TRACK
10. Application code and its repository shall be easy to inspect and maintain
    ON TRACK
11. Google analytics shall be used
    ON TRACK
12. No e-mail clients shall be allowed. Interested users can only message to sellers via in-site messaging. One round of messaging (from user to seller) is enough for this application
    ON TRACK
13. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI
    DONE
14. Site security: basic best practices shall be applied (as covered in the class) for main data items
    ON TRACK

15. Media formats shall be standard as used in the market today
    ON TRACK
16. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
    ON TRACK
17. The application UI (WWW and mobile) shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Fall 2020. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application)
    DONE