

SW Engineering CSC648/848 Fall 2019

Gator-Aid

Milestone 4

Section 02

Team 09

TEAM LEAD

Nayan Pandey

npandey1@mail.sfsu.edu

Nikhil Mohan – Front-end Lead

George Pernov – Back-end Lead

Stephanie Sechrist – Github Master

Ali Nasralla

Jackie Huang

Revision	Date	Description
-	12/13/2019	Initial Submission

Table of Contents

1. Product Summary	2
2. Usability Test Plan.....	2
2.1 Test Objectives.....	2
2.2 Test Background and Setup.....	3
2.3 Usability Task Description	3
2.4 Lickert Subjective Test	4
3. QA Test Plan.....	5
3.1 Test Objectives.....	5
3.2 Hardware and Software Setup	5
3.3 Feature To Be Tested	5
3.4 QA Test Plan.....	5
3.5 QA Test Results Summary	6
4. Code Review	6
5. Self-Check on Best Practices for Security	8
6. Self-Check: Adherence to Original Non-Functional Specs.....	9

1. Product Summary

Gator-Aid is an online application exclusive to San Francisco State University students. Our aim is to help college students at our school by letting them sell or buy products as soon as possible by making the school itself an ideal spot to meet up for transactions.

Major Committed Functions

Unregistered Users

- Shall be able to browse items by category.
- Shall be able to view listings.
- Shall be able to register an account with an SFSU email address.
- Shall be able to search items.
- Shall be able to sort listings by price.
- Shall be able to sort listings by date created.

Registered Users (includes above)

- Shall be able to log in.
- Shall be able to post items for sale as listings.
- Shall be able to contact sellers.

Administrator (includes above)

- Shall be required to approve or reject listings.
- Shall be able to delete listings.

2. Usability Test Plan

2.1 Test Objectives

The search function will be tested, as we feel it is critical to the function of Gator-Aid. Included in the search function is the ability to search based on the following categories: All (default search category), Books, Electronics, Furniture, Tutoring, and Other. The user may or may not input a search query in order for the function to work. Once the user clicks the search button, the listings will be displayed as image thumbnails above the title, price, and Buy Now button. By default, the listings will be ordered by date posted, with the most recent appearing first.

The objectives of the test are to determine the effectiveness, efficiency, and satisfaction of our application. The following questions will be answered:

- Are users able to find the search bar?
- Does the function work as we expect?
- Is it easy to use?
- Is it straightforward for the user how to use it?

- Are users able to find the items they are looking for?
- Are the listings displayed to the user in a useful way?

2.2 Test Background and Setup

System Setup

The evaluators will meet with our front-end lead and tests will be performed on the user's device, which can be either a mobile device, laptop, or desktop, while the lead looks on without offering guidance or support. Using the latest version of Chrome or Firefox, each evaluator will access our website and perform the test. After performing a task, they will each be given a questionnaire designed to answer the questions posed in Test Objections.

Starting Point

The user will begin the test from the Gator-Aid home page: <http://ec2-13-52-181-0.us-west-1.compute.amazonaws.com:8000/>. They will attempt to complete the test on their device, while taking mental notes along the way so as best to answer our questionnaire.

Intended Users

The intended users of Gator-Aid will be students of San Francisco State University. We expect computer-literate users between 18 and 29 years of age, the most common ages of students at SFSU. Because there is a wide range of items that will be available for sale, users may live on- or off-campus; for example, on-campus users might be more likely to search for furniture than off-campus users. Our target demographic is looking to save money on goods, especially textbooks and electronics. It is expected that most users will access the Gator-Aid on their mobile devices.

URL of System to be Tested

<http://ec2-13-52-181-0.us-west-1.compute.amazonaws.com:8000/>

What Is to be Measured

In this usability test, we are focused on measuring user satisfaction. In other words, we want to know if the user was comfortable using Gator-Aid. This will be evaluated through the use of Licker subjective test questions.

2.3 Usability Task Description

To measure effectiveness, we will look at:

- the percentage of users who completed the task.
- the number of errors made per task.
- the number of tasks users completed versus the number of tasks given.

To measure efficiency, we will look at:

- the number of clicks the user made to complete the task.

- the time each user took to attempt the task.
- the average time all users took to attempt the task.

2.4 Lickert Subjective Test

1. From the home page, find the least expensive television available for purchase.

It was easy to find the category drop-down menu.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

It was easy to sort by price after getting search results.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

The results were displayed in a way that helped me see what options were available for purchase.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

2. You're interested in listing yourself as a Calculus tutor, but you're not sure what price you should charge. Find out the current range of prices for tutors in this subject.

I was easily able to find what I was looking for based on my search input.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

It was easy to view the most and least expensive listings based on my search result.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

The prices of tutors were easily viewed.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

3. You're new to SFSU, and you need many things, including furniture, electronics, and books. Browse all available listings.

It was easy to view all listings.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

I would use Gator-Aid again.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

I would recommend Gator-Aid to a fellow student.

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
----------------	-------	---------	----------	-------------------

3. QA Test Plan

3.1 Test Objectives

Our objective is to ensure that the key feature of search is functional and meeting/exceeding requirements. We will use white-box testing on the requirements in order to ensure that the search feature is implemented into the application and is working from base level without needing any changes.

3.2 Hardware and Software Setup

The user will be using the most recent development build of Gator-Aid, using the latest build of Chrome (79.0.3945.66) and Firefox (70) on MacOS Catalina.

3.3 Feature to Be Tested

An SFSU student will be able to locate the search bar easily, select the category of items they want to search for and then be able to see all of the results displayed correctly based on the category.

3.4 QA Test Plan

The user should be able to select a category from All, Books, Electronics, Furniture, Tutoring, and other. When the user selects All, it will default to show listings from all of the categories. If the user selects a specific category, they will be able to view all of the items from that category, and they are able to sort by date (default) posted or price.

Test Cases:

- Students cannot select a category and all results will show up.
- Students can select each category individually.
- Students will be able to sort by date most recently posted.
- Students will be able to sort by price.

Test Number	Description	Input	Output	Result
1	User is on the main page and all the results will be displaying below the search bar.	none	User will be able to view all of the posts from most recent posted on the homepage.	PASS

2	User will be on main page and selecting between Books, Electronics, Furniture, Tutoring, and other.	User will select whichever category they want.	User will be redirected to proper category page.	PASS
3	User is on specified category that they selected.	User will see on the right side they can sort by most recent and selects that.	All of the posts will be changed from their previous organization to most recent in front.	PASS
4	User is on specified category that they selected.	User will see on the right side they can sort by price (low to high) and selects that.	All of the posts will be changed from their previous organization to lowest to highest price-wise.	PASS



3.5 QA Test Results Summary

The test student user was able to search properly and extensively with the proper categories. They were successful in using the different features such as sorting by category and from that category sorting by most recent and price.

4. Code Review

To make our code readable and easier to maintain the future, we have used best practice coding styles for JavaScript, CSS, and HTML, such as

- using correct indentation and alignment (2 spaces).
- using explanatory variable names that help describe what is being represented.
- in-line comments.
- correct naming conventions according to the language.
- following the model-view-controller structure for organizing our code.

 Reply  Reply All  Forward



Fri 12/13/2019 4:43 PM

Stephanie Llanes Sechrist

RE: Code Review

To Jackie Huang

Hi Jackie,

Code looks good and everything works fine, but you're missing a lot of comments. It would be helpful to see a header with your name, the purpose of the code, and the input/output. Some in-line comments would be nice as well; as it is, it's a little hard to understand the code. Formatting looks great, though!

Keep up the good work :)

Best,
Stephanie

From: Jackie Huang <jackie19601970@gmail.com>

Sent: Friday, December 13, 2019 3:52 PM

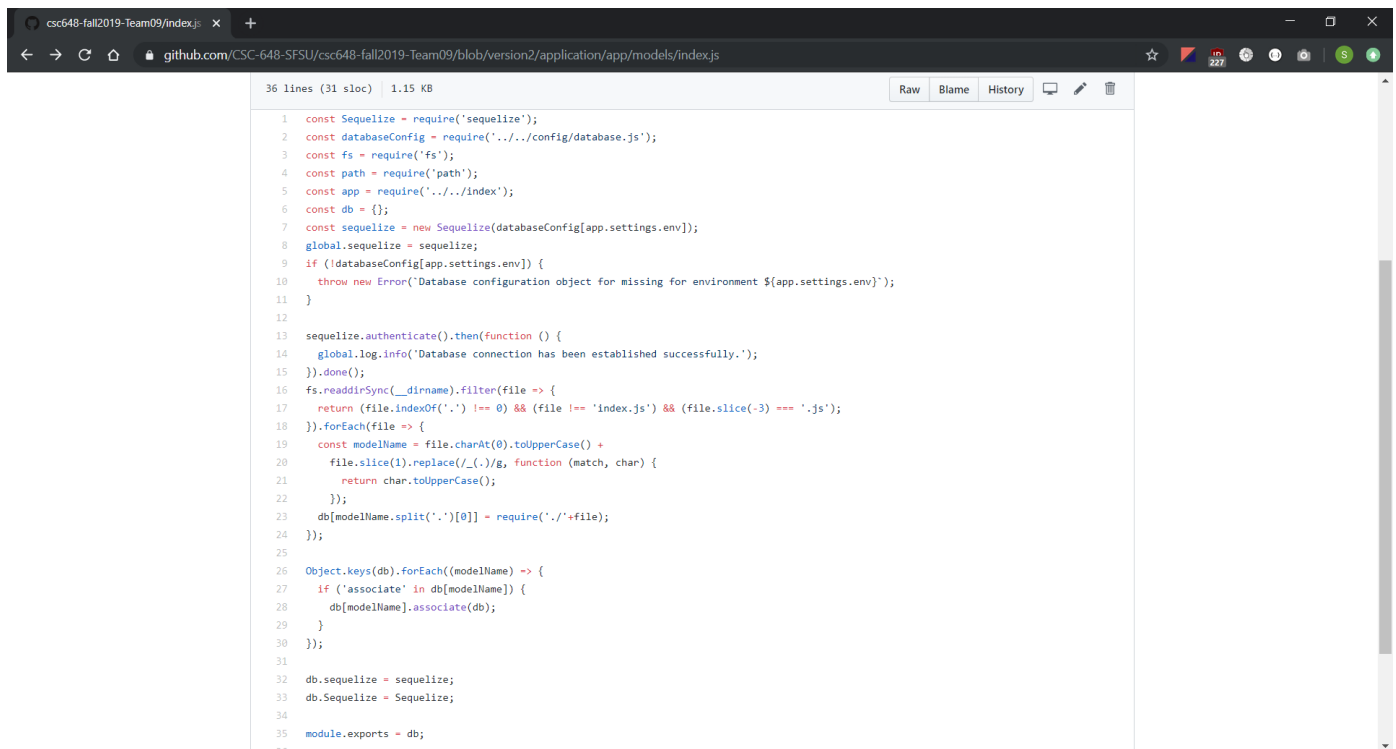
To: Stephanie Llanes Sechrist <ssechrist@mail.sfsu.edu>

Subject: Code Review

Hello there,

I recently pushed some code with the file name /application/apps/models/index.js onto Github version2 branch. I would love it if you can take a look at it to see if I can improve it in any way.

Thank you!

A screenshot of a web browser displaying a GitHub file viewer for the file 'index.js' in the repository 'csc648-fall2019-Team09'. The browser's address bar shows the GitHub URL. The file viewer interface includes a header with '36 lines (31 sloc)' and '1.15 KB', and buttons for 'Raw', 'Blame', and 'History'. The main content area shows the JavaScript code for 'index.js', which includes database configuration, file reading, and model association logic. The code is syntax-highlighted and line-numbered from 1 to 36.

```
1 const Sequelize = require('sequelize');
2 const databaseConfig = require('../../config/database.js');
3 const fs = require('fs');
4 const path = require('path');
5 const app = require('../../index');
6 const db = {};
7 const sequelize = new Sequelize(databaseConfig[app.settings.env]);
8 global.sequelize = sequelize;
9 if (!databaseConfig[app.settings.env]) {
10   throw new Error('Database configuration object for missing for environment ${app.settings.env}');
11 }
12
13 sequelize.authenticate().then(function () {
14   global.log.info('Database connection has been established successfully.');
```

5. Self-Check on Best Practices for Security

The major assets we are protecting include the database, the password, and the input data since it requires a validation check.

We are protecting the assets through different methods. For the password we are using an encryption check. For the input data we require a validation test. For the https we are using https security. The entire password is protected using crypto and hash. The email for our project is protected via Kraken jwt which is essentially another hash. The website is protected using the same https security.

We had confirmed the input data validation that it was a max of 40 characters by searching different phrases that were greater than or equal to 40. The 40 characters had worked as it was the limit. The next test we had done was through characters less than 40. All of these inputs had worked properly.

Sensitive Information

- Passwords, input data were hashed before stored into the database.
- Caching the authentication tokens for those who are currently logged in.

Authorizations

- Admin account is allowed to review posts.
- In addition to reviewing posts, admin is also able to delete posts that have been there too long, or simply approve all of the posts.

Input Validation Layers

Front-end layer validation on login/register used for user experience to lessen the bad requests towards the backend.

- Required fields
- Email formatting
- Max Characters (40)

Back-end layer validation to ensure that the database checks are correct and in order to prevent SQL attacks.

- Required fields
- Email formatting
- Mac characters (40)

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). **DONE**
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers. **DONE**
3. Selected application functions must render well on mobile devices. **ON TRACK**
4. Data shall be stored in the team's chosen database technology on the team's deployment server. **DONE**
5. No more than 50 concurrent users shall be accessing the application at any time. **DONE**
6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. **DONE**
7. The language used shall be English. **DONE**
8. Application shall be very easy to use and intuitive. **DONE**
9. Google analytics shall be added. **ON TRACK**
10. No e-mail clients shall be allowed. **DONE**
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI. **DONE**
12. Site security: basic best practices shall be applied (as covered in the class). **ON TRACK**

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