JUNGLE

SW Engineering
CSC 648/848
Fall 2022
Milestone 4
Team 2

TEAM LEAD	Nanda Pandian (npandian@mail.sfsu.edu)
GITHUB LEAD	Troy Carloni
FRONT-END LEAD	Farid Mehdipour
BACK-END LEAD	Richard Aguilar
SOFTWARE DEVELOPER (1)	Hugo Moreno
SOFTWARE DEVELOPER (2)	Yuwei Liu
DATE SUBMITTED	12/09/22
DATE REVISED	TBD

Product Summary

Name: Jungle

Description of product:

Jungle is a platform used to sell San Francisco State University graphic media, eTextbooks, and a few miscellaneous items. SFSU media includes videos, photos, and other visual content. The purpose of our application is to bring convenience to the customer's finger tips. Oftentimes SFSU students and faculty are struggling to find time between their class or work schedules to get certain media. Our application allows customers to easily purchase and get access to SFSU's media as they are needed.

Itemized list of ALL major committed functions:

1. Unregistered users are able to create temp registration account
2. Unregistered users are able to search items
3. Unregistered users are able to perform general browse
4. Unregistered users are able to check product details
5. Unregistered user are able to browse by category
6. Registered users are able to log in their account
7. Registered users are able to create posts
8. Registered user are able to change email
9. Registered user are able to change password
10. Registered user are able to message other registered users in regards to their posting

What is unique in your product:

Jungle's unique feature is its ability to filter the product search by SFSU class and professor names. Oftentimes SFSU students and faculty have a tough time trying to find out which media belongs to which class and professor. Therefore, Jungle helps narrow down SFSU students and faculty' search to find what they need by adding another filter option.

URL to your product:

http://35.233.185.244:8080/

Usability Test Plan

Test Objectives:

For our usability test plan we decided to use our create post function feature. In this feature we will be testing the ease of creating a post. Specifically how fast it is to create the post, how easy the create a post tool is to use and how clear the feature itself is to use.

Test Objectives and Setup:

Starting Point: Users will begin at the home page of our program and will need to find and navigate to create a post page themselves.

Intended Users: The intended users for this page are San Francisco State University students who want to sell their digital media on our website. This feature is also only available for users who have already registered on our website. If you do not have an account you will be sent to the registration page.

URL to Create Post Page for students who want to sell: http://35.233.185.244:8080/createPost

What is to be measured: For this usability test we are going to have the user's answer a survey with a series of statements and give them multiple answer options including Strongly Agree, Agree, Neutral, Disagree, and Strongly Disagree.

Usability Task Description:

For this task you will create and publish a post on the Jungle website which you will then later evaluate. Creating a post requires you to set a title, description and give a price along with including an image of your product

Evaluation of Effectiveness:

To evaluate effectiveness we will need to see if the user was successfully able to create a post with a title, description price and photo. If the user was able to successfully create the post we would consider the feature effective.

Evaluation of Efficiency:

To evaluate efficiency we would first need to know if the user was able to complete the task in a timely manner and without any complication. We would also need to know if the was able to

complete the task easily with a few amount of clicks. Lastly we would need to know if the task itself was easy and simple to understand.

Evaluation of User Satisfaction:

Prompt 1:

The task of creating and publishing a post was simple and easy to complete. *

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly agree

Prompt 2:

Creating a post was quick and efficient. *

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly agree

Prompt 3:

The instructions for the task were clear and easy to understand. *

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly agree

QA Test Plan

Using the same function as the usability test, below is the QA test plan.

Test objectives - what is being tested:

- Search
- Ensure all methods of search queries function correctly
- Ensure search is protected against sql injection

HW and SW setup (including URL):

- Browsers: Microsoft Edge, Google Chrome
- URL: http://35.233.185.244:8080/

Feature to be tested:

- Search

QA Test plan: Search

#	Title	Description	Test Input	Expected Output	PASS/FAIL
1	All products	Query DB for all posts through search bar	Leave search bar input field empty and category should be set to “All”	Get 5 results. Name of expected items; “test”, “horror”, “literature”, “movietime”, “America”	Edge: PASS Chrome: PASS
2	Input field	Use search bar input field to search for post	In search input field type “test” and hit enter	Get 1 result. Result should be “test” with category “video” and price “$80.00”	Edge: PASS Chrome: PASS
3	category option	Use category option to search for posts	Leave search bar input field empty and category should be set to “audio”	Get 1 result. Result should be “america” with category “Audio” and price “$80.00”	Edge: PASS Chrome: PASS
4	Input field and category option	Use search bar input field and category option to search for posts	Set category option to “Video” and type “test” in search bar input field	Get 1 result. Result should be “test” with category “video” and price “$80.00”	Edge: PASS Chrome: PASS
5	Sql Injection	Attempt to manipulate DB with sql injection in search bar	Set category option to “All” and type “SELECT * FROM Users” in search bar input field	Get “No Post Found, Showing All Posts”, and will show all 5 posts.	Edge: PASS Chrome: PASS
6	Character limit	Attempt a search with more than 45 characters	In search bar input field type “1” 41 times.	Get zero results. “exceeded character search length” should show on page	Edge: PASS Chrome: PASS
7	No post found	Search for a post that does not exist	In the search bar input field type “helloworld”	Get “No Post Found, Showing All Posts”, and will show all 5 posts.	Edge: PASS Chrome:PASS

Code Review

Email sent to teammate for code review

CSC 648-848 Fall 2022 Milestone 4 Code Review

NS

Nanda Sivagnanam Pandian <npandian@mail.sfsu.edu> Today at 10:21 42AM

To: Farid Mehdipour

Hi Farid,

Hope all is well.

I was hoping you could take a look at a portion of code that I'd like to get some feedback on. Due to security reasons, I will not be sending the code directly through email. So, please go to the repo link and follow the code path as stated below. The code highlighted is what I'd like to get reviewed on. Please let me know if there are any questions or issues finding the specific code.

Repo link: https://github.com/CSC-648-SFSU/csc648-03-fa22-team02

A. **application >> myapp >> routes >> index.js**
    a. inside index.js, please find the `router.get(\search`
B. **application >> my app >> public >> javascript >> frontend.js**
    a. inside of frontend.js, please find `searcNav`

Thank you for your time and help!

Best regards,
Nanda

Email response (1): Code review performed by teammate

Re: CSC 648-848 Fall 2022 Milestone 4 Code Review

FM

Farid Mehdipour <fmehdipour@mail.sfsu.edu>

Today at 10:27 25AM

To: Nanda Sivagnanam Pandian

Hi Nanda,

Hope all is well too.

Thanks for reaching out, I would be happy to do the code review, and please let me know if you have any questions in this process.

The search in the Navigation bar has three sections:

1. The first part is categories, where the users choose what kind of element they are looking for, these categories are:
   - ebook
   - Video
   - Image
   - Audio

2. The middle part or the second part is where the user enters the name of the product.

3. The third part is the button that executes the search.

The following functions are what make the searchNav execute accordingly.

Repository: application/myapp/routes/index.js
The function **router.get('/search', async (req, res, next)** is the get function that receives the search_term that the user enters:

In order to determine whether the search term and category are valid inputs, it first accepts them. If they are, it then takes the category input and search term, retrieves the result from the database, and passes it back.

This is how the query Prompt for the DB would appear.

1)If a user requests all the components in a particular search name:

`SELECT * FROM Posts WHERE name LIKE '%${search_term}%' AND approved = 1`;

2) If the users want all the elements inside of DB with no specification on the category or the search term:

`SELECT * FROM Posts WHERE approved = 1`

3) Output all the elements in that category if the user only specified the category and not the search term:

`SELECT * FROM Posts WHERE category = "${category}" AND approved = 1`; }

4)Take all the outputs and pass them to the DB if neither of these options is chosen:

`SELECT * FROM Posts WHERE category = '${category}' AND (name LIKE '%${search_term}%') AND approved = 1; `

Email response (2): Continuation of code review performed by teammate

Re: CSC 648-848 Fall 2022 Milestone 4 Code Review

FM

Farid Mehdipour <fmehdipour@mail.sfsu.edu>

Today at 10:28 29AM

To: Nanda Sivagnanam Pandian

Repository: application/myapp/public/javascript/frontendjs.js

There are three functions inside of the frontendjs.js that are components of SearchNav's front end.
The submit button on the search initiates the execution of this function initially.
This function is the first one to receive the search phrase and category inputs before passing them on to the following function.
By using getElementById, it retrieves the elements from the HTML (user interface), examines their value, and saves them.
Additionally, it develops searchURL using the values in the
`/search?search=${searchTerm}&category=${categoryText}`;

After that, it fetches the data on that URL and creates the cards which are going to be the product thumbnails in the search result.

executeSearchNavbar():
This function takes the search term and the category inputs and passes them to the backend in the HTTP format of `/result-page?search=${searchTerm}&category=${categoryText}`
Where we navigate the location of the window to that URL to it.
Before that, it also checks the URL to see if it is validated.

executeSearchNavbar_onPage():

This function gets the URL parameter on the location of the window tab, then takes the two variable of search term and category from that URL and pass it back as a new URL:

`/search?search=${searchTerm}&category=${categoryText}`

The category value is then examined to determine its length and the number of post products that can be returned to the user.
With that knowledge, we can determine how many card products to display, so we execute a retrieve function, display the number of results, and then feed the remaining product information back to the build card method (this function creates the product post thumbnails on the screen)

Self check on best practices for security

Asset to be protected	Types of possible/expected attacks	Your strategy to mitigate/protect the asset
User Login Credentials	• SQL Injection • Leaking user credentials from malicious parties	• To prevent a SQL Injection attack the queries executed use SQL Parameters to prevent the possibility of a malicious person to ruin the database . • To prevent major leaks of user credential data the password is hashed once going to the database. The reason it isn't encrypted is because an encryption can be decrypted. To assure full safety hashing the password is best.
Search bar	• SQL Injection • XSS Attack	• Similar to the login credential and SQL Injection takes advantage of how SQL queries or executes within the code. Similar to the login credentials I will use SQL parameters because they are known to prevent these attacks.
Post Creation	• XSS Attack • Users post malicious content or spam the website with posts that violate the website's terms and conditions they agreed to.	• XSS is usually a foreign party trying too input a script in an input area within the website to access information regularly not allowed. To prevent this any post a user wants to add will be

		validated before added to the website which will prevent bad actors from accessing information
Changing user credentials	• Bad actor would want to change the password or email of victims account	• To prevent the attempt to change the users password they will have to have access to their account. When changing the password you will be asked to enter the old password, new password and to confirm the new password. With the changing of email or password when forgotten the user will email the website support for the request

Self-check original Non-functional specs *(performed by team lead)*

NON-FUNC SPEC	DONE	ON TRACK	ISSUE
Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0	YES		
Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers		YES All the pages render their sizes correctly, except the CSS is a little off. So the company banner or the feed page will not be centered perfectly. Front end team is on the fix!	
All or selected application functions must render well on mobile devices		YES All the pages are rendering their sizes proportionally to the mobile device, except About & Profile pages. Front end team is debugging the code to find the minor error.	
Data shall be stored in the database on the team's deployment server.	YES		
No more than 50 concurrent users shall be accessing the application at any time	YES		
Privacy of users shall be protected	YES		

The language used shall be English (no localization needed)	YES		
Application shall be very easy to use and intuitive	YES		
Application should follow established architecture patterns	YES		
Application code and its repository shall be easy to inspect and maintain	YES		
Google analytics shall be used	YES		
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	YES		
Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	YES		
Site security: basic best practices shall be applied (as covered in the class) for main data items	YES		
Media formats shall be standard as used in the market today	YES		
Modern SE processes and practices shall be used as specified in the class,	YES		

including collaborative and continuous SW development			
The application UI (WWW and mobile) shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Fall 2022. For Demonstration Only” at the top of the WWW page nav bar. (Important so as to not confuse this with a real application).	YES		