SW Engineering CSC648/848 Fall 2019
State Exchange App
Section: 01 | Team #07

Global Team

Roderic Kong - Team Lead
Email: rkong1@mail.sfsu.edu
Chris Jol - Front End Lead
Kevin Venegas - Back End Lead
Deep Bhuller - Git Master

Milestone 4

12/11/2019

Date	Submission	Revisions
12/11/2019	01	0

Product Summary

Product Name: State Exchange

Functionality List :

- P1 Functions(SFSU FE & BE):
    - Home page with recent posts
    - Search bar using %like
    - Create New Post page
    - User Dashboard to view posts
- P1 Functions(Fulda FE & BE):
    - User Login
    - User Registration
    - Messaging & Contract Seller

Unique To Our Product:

- Designed for SFSU students

Link to Product: http://ec2-18-224-39-11.us-east-2.compute.amazonaws.com:3000/index

Usability Test Plan

Function to be tested: Uploading Post

Test Objectives:

- The objective of our usability test plan is to test the usability of our Upload Function. The reason for conducting usability testing on our Upload Function is to ensure users can upload files onto our server with ease and stability. This also tests our use cases along with search function to see if it correctly populates with new posts.

Test Background & Setup:

- System Setup:
    - The background setup requires access to our MySQL database, browsers such as Mozilla, Safari, Chrome, or Edge, our website link, photos and login credentials.
- Starting Point:
    - The start point of our website testing will be starting from our homepage. Once in the home screen, create your account by clicking “Sign Up”.
    - Input all of your desired credentials.
    - Once you are done, press the “Login” button which is located next to the Sign Up button.
    - Input your credentials.
    - On the menu button, select “New Post”.
    - Complete the entire form.
- Intended Users:
    - The intended Users are SFSU Students.
- What is measured:
    - The design and ease of access of website.
    - Reliably bring up the posts page which is updated dynamically and fetched via search.
    - Each post will also dynamically create a detail page which loads all the information of the post, which is chained via Primary ID linked in the MySQL database.

Usability Task Description:

- **Input:**
    - Creation of the post.
- **Output:**
    - Check the list of posts that is similar in the name of what the product is posted. The results will be just the recently posted item. The user is also able to click on the button which routes itself into a details page. The details page will list all the information such as price, and comments.
- **Effectiveness:**
    - Effectiveness would be measured if users are able to continue to see what they recently posted along with other parameters such as Title, Price, and clickable button.

- **Efficiency:**
    - How would we measure effectiveness:
        - We would list in couple of posts, and check our MySQL database and see if all the information passed into the would exist.
    - How we would measure effectiveness:
        - We would make 20 posts and see if it breaks any of our code.

Input: Search for ‘couch’
Output: Check that a list of items is returned containing term %like ’ccouch’

Input: Search without a search term
Output: Check that a list of recent posts is returned

Lickert Subjective Test:
Question - 1:
Was searching for the insert page simple to find? (Legend: 1 = Bad, 5 = Great)
Circle:

1	2	3	4	5

Question - 2:
How difficult is the posting page intuitive to use? (Legend: 1 = Bad, 5 = Great)
Circle:

1	2	3	4	5

Question - 3:
Would you return and reuse the website again? (Legend: 1 = Bad, 5 = Great)
Circle:

1	2	3	4	5

QA Test Plan

Test Objectives:

- New Posting Functions and related functionalities within our Website

Hardware Setup:

- A mobile device, PC, or laptop capable to run updated versions of Chrome, Safari, or Edge is sufficient.

Software Setup:

- Internet connection
- Connection to MySQL workbench OR accessing MySQL tables within AWS.
- Access browsers such as Chrome, Mozilla, or Safari.
- Accessing into website: http://ec2-18-224-39-11.us-east-2.compute.amazonaws.com:3000/index

Features To Be Tested:

- New Postings
- Displaying of the New Postings in:
    - Front Page
    - Details
    - Dashboard

QA Testing Plan: - Table Format:

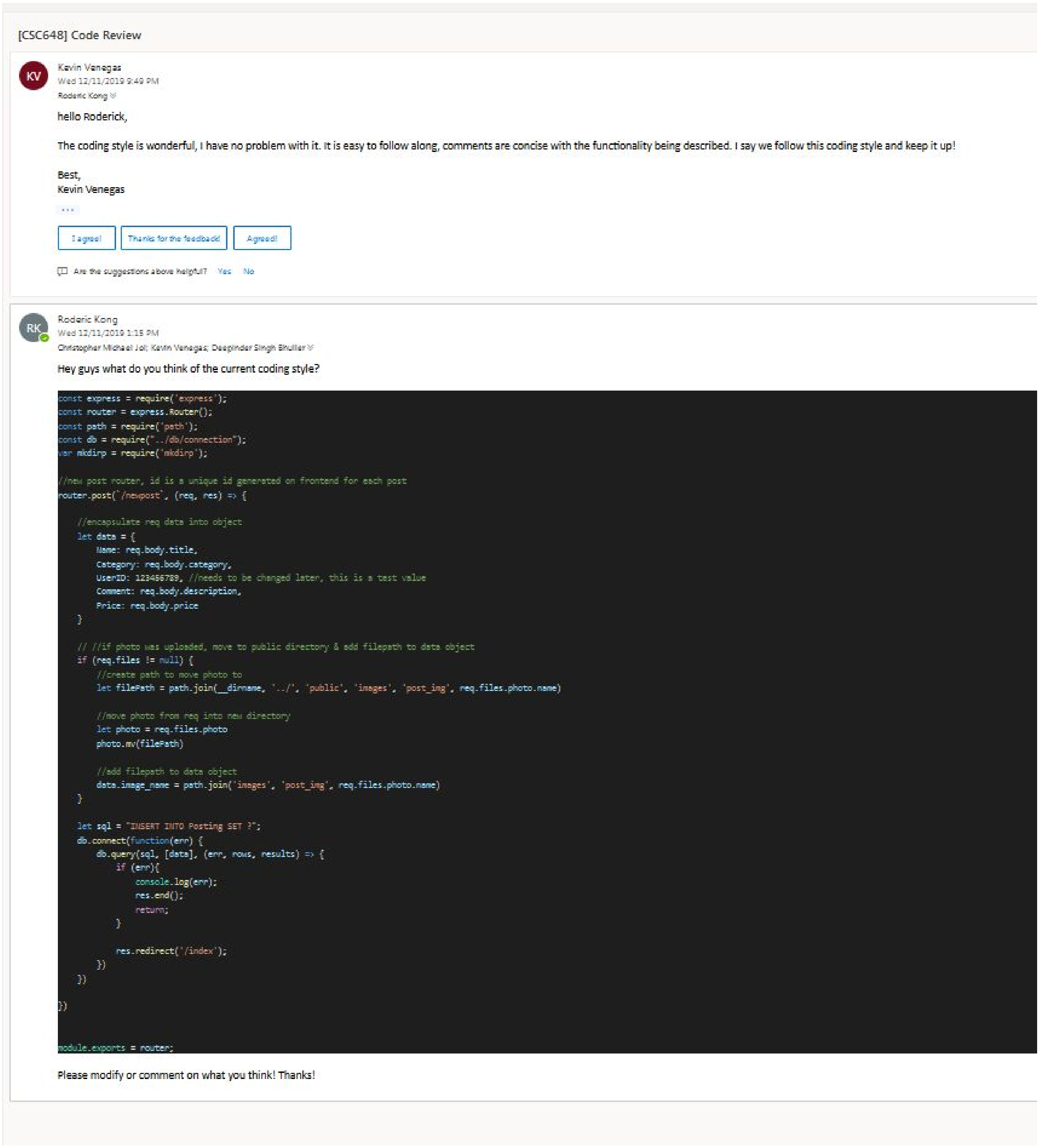
Number	Description	Test Input	Expected Results	PASS/FAIL	Browser Used:
1	Insert Post 9	Title: Test 9 Image: image_9 Details: Book Price: 99 Description: This is Test for Post 9	Post Passes: Website responses with "Success"	Pass	Mozilla
2	Insert Post 10: This is to test the MySQL query which is also needed to test Price Column and see if String Passes into INT	Title: Test 10 Image: image_10 Details: Electronics Price: asdf Description: This is Test for Post 9	Post Passes: Website Response with "Success": Price = NULL in MySQL	Fail	Mozilla
3	Insert Post 11: This is to test if insert MySQL is able to take no input.	Title: Image: Details: Price: Description:	Post Passes: Website Response with "Success": Everything aside from ID column is NULL	Fail	Mozilla

Code Review

Coding Style:

- Front end:
  - Heavily reliant on formatting:
  - HTML are converted into .pug files
  - JS files connected used by pug files are separated into javascript files, and inserted into Vue Folder
  - Intended code functionalities are managed by the Vue Folder.
  - Allows for easier modifications
- Back end:
  - Many routers that send back pages are done within the the files.js
  - Routers that are meant with functionalities are in their separate js files.
  - Routers with extra functionalities are also named with their intended functions.
  - Coding practices that attempts on MySQL Queries:
    - Each member is accessed with req.body.[name that is managed by Vue folder] (body-parser middle-ware)
    - Data member is then fitted into whichever SQL query.
    - Query is then set into a database connection alongside its query function.

Code Review by Peer:



[CSC648] Code Review

Kevin Venegas
Wed 12/11/2019 9:49 PM
Roderic Kong

hello Roderick,

The coding style is wonderful, I have no problem with it. It is easy to follow along, comments are concise with the functionality being described. I say we follow this coding style and keep it up!

Best,
Kevin Venegas

Roderic Kong
Wed 12/11/2019 1:15 PM
Christopher Michael Jol; Kevin Venegas; Deepinder Singh Bhuller

Hey guys what do you think of the current coding style?

```
const express = require('express');
const router = express.Router();
const path = require('path');
const db = require("../db/connection");
var mkdirp = require('mkdirp');

//new post router, id is a unique id generated on frontend for each post
router.post(`/newpost`, (req, res) => {

    //encapsulate req data into object
    let data = {
        Name: req.body.title,
        Category: req.body.category,
        UserID: 123456789, //needs to be changed later, this is a test value
        Comment: req.body.description,
        Price: req.body.price
    }

    // //if photo was uploaded, move to public directory & add filepath to data object
    if (req.files != null) {
        //create path to move photo to
        let filePath = path.join(__dirname, '../', 'public', 'images', 'post_img', req.files.photo.name)

        //move photo from req into new directory
        let photo = req.files.photo
        photo.mv(filePath)

        //add filepath to data object
        data.image_name = path.join('images', 'post_img', req.files.photo.name)
    }

    let sql = "INSERT INTO Posting SET ?";
    db.connect(function(err) {
        db.query(sql, [data], (err, rows, results) => {
            if (err){
                console.log(err);
                res.end();
                return;
            }

            res.redirect('/index');
        })
    })

})


module.exports = router;
```

Please modify or comment on what you think! Thanks!

Self Check on Best Practices-Security

Major Assets Protection:

- Users password.
    - Done by using MySQL innate encoding method

Passwords in Database:

- Is encrypted.

Input data validation:

- Pricing on insert post.

Self Check: Adhere 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).
    a. 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.
    a. Done! Safari & Mozilla - ok!
3. Selected application functions must render well on mobile devices.
    a. Done!
4. Data shall be stored in the team's chosen database technology on the team's deployment server.
    a. Done!
5. No more than 50 concurrent users shall be accessing the application at any time.
    a. On Track….
6. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.
    a. Done!
7. The language used shall be English.
    a. Done!
8. Application shall be very easy to use and intuitive.
    a. Done!
9. Google analytics shall be added.
    a. On Track…. -Added but no stats are registering
10. No email clients shall be allowed
    a. Done!
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.
    a. On Track…..

12. Site security: basic best practices shall be applied (as covered in the class)
    a. On Track…..
13. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
    a. 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).
    a. Done!