SW Engineering CSC648/848 Spring 2022

PurpleMarket

TEAM 8

Name	Role	Email
Mark Kim	Team Lead	mkim22@mail.sfsu.edu
Khushboo Gandhi	Front End Lead	
Cody Huang	Back End Lead	whuang17@mail.sfsu.edu
Ernesto Diaz	GitHub Master	ediaz8@mail.sfsu.edu
Jesus Cervantes Fajardo		
Vivian Kuang		
Jiasheng Li		

Milestone 4

May x, 2022

Milestone/Version	Date
M4v1	05/15/2022

1) Product summary

The name of our product is Purple Market. It is a web application that connects students, staff, and faculty from San Francisco State University. It is a marketplace where students and staff can buy and sell products to others at SFSU.We create a safe way to shop by manually verifying each post before publishing it for sale. We provide a custom tailored experience for people associated with SFSU. Our website is simple to navigate and easy to use. There is no clutter with our clean design. We sort all of our products into general and easy to understand categories. Users are able to search for the items they want in the search bar or filter out the ones they are looking for by categories. Once a user finds their product they can sign up on our site and send the seller a message.

Our application allows users to:

- Search the site for certain posts.
- Filter search results.
- Register for an account

Once registered, users will be able to:

- Create posts.
- Get in contact with poster/seller
- Edit and correct and/or delete their own post.
- Login/out of their account.
- Reset their password.
- Inspect their transaction history.

Finally admins will use MySQL workbench to:

- Do everything a registered user does.
- Approve or decline an item post.

Website url: https://csc648.mskim.dev

2) Usability test plan – max 2 pages

Function: Post an item

- Test objective - Posting is a priority one functional requirement of the app and related to one key user case of the app, so it is necessary to test if users can easily post an item and try to identify problems with usability with the app. Testing allows developers to revise, iterate and retest the app based on the analytical results and the report from the test plan.

- Test background and setup

  The system is running on an Amazon EC2 instance. Usability testers can use Chrome and Safari for browsers to participate in this usability test.
  The starting point of this usability test is the home page with the user already logged into the system. Then the testers are to navigate to the post item page and proceed to post an item on the system.
  The intended users of the system are students, staff, and faculty of San Francisco State University.
  The URL of the system to be test is the page for posting an item , https://csc648.mskim.dev/createpost
  The first thing to be measured is the task completion success rate which is the percentage of users who can complete the usability task in the defined time. The second thing to be measured is the number of errors for completing the usability task which are to be reported by the testers. The third thing to be measured is the average time it takes users to complete posting an item. The fourth thing to be measured is the average number of clicks and average number of screens to complete posting an item. Finally, the usability test will implement a Likert questionnaire to measure their comfort and confidence at posting an item on the system.

- Usability Task description

  As a usability tester, you will need to follow the requirements of system setup before entering the start point. Once logged in, and on the home page, please attempt to post an item to the system. You will also be the test monitor, so please record the number of screens you visit, the number of clicks you use, any errors you encounter, and the time it takes to complete the entire process of posting an item (if budget and/or time allows, this data will be gathered from action recording software on the system that the tester uses). After that, the testers will finish an evaluation of efficiency form and an evaluation of user satisfaction form.

Task	Description
Task	Post an item on PurpleMarket
Machine State	The prioritized functionalities of PurpleMarket have been implemented
Successful Completion criteria	The post is loaded on the user's post history of PurpleMarket
Benchmark	Completed in1 minute

- Evaluation of Effectiveness
  We measure effectiveness of posting by measuring the percentage of usability testers who completed the posting task one minute. Also, we measure the number of errors incurred during their task completions.

Test/Use case	% Completed	Errors	Comments
Post	0%	1. Receiving an error after clicking a post item button	The function for sending the posting data from front-end to back-end fails, so all testers could not post an item successfully.

- Evaluation of efficiency
  We measure efficiency in time which includes average time of those who complete the task, efficiency in effort which is the number of clicks, and efficiency in content which include the number of screens and number of pages of instructions to complete posting.

Situation two that a usability user has not had an account:

1) Average time of those tester who complete the task: 100 seconds
2) Number of screens: 7 (Homepage, Post Items, Login, Register, Login, Homepage, and Post Items)
3) Number of clicks : 30 clicks
   - Post Items
   - Item Name
   - Price
   - Category
   - Choose File
   - Select File
   - Select OK

- Description
- Post Item
- Select OK for error
- Login
- Register here
- First Name
- Last Name Email
- Password
- Confirm Password
- Show Password (not a necessary step but I wanted to see if it worked)
- Register
- Email
- Password
- Login
- Post Items
- Item Name
- Price
- Category
- Choose File
- Select File
- Select OK
- Description
- Post Item
- Select Ok on the error I got a 500 error and not the 24 hours approve

4) Number of pages of instructions: 2 (register page posting page instructions)
   Situation one that a usability user already had an account:
   1) Average time of those tester who complete the task: 42 seconds
   2) Number of screens: 2 (Homepage and Post Items)
   3) Number of clicks : 14 clicks
   - Login
   - Email
   - Password
   - Login
   - Post Items
   - Item Name
   - Price
   - Category
   - Choose File
   - Select File
   - Select OK
   - Description
   - Post Item

- Select Ok on the error I got a 500 error and not the 24 hours approve

4) Number of pages of instructions: 1 posting page instructions

- Evaluation of user satisfaction

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
I found the website easy to use.	2				
It is easy to post an item.	2				
It is easy to know when posting an item is successful.			2		

3) QA test plan - max 2 pages

Function: Post an item

Test objectives:

Test whether our post function is usable. This means that the user is able to create a post through our post page by inserting certain information within the data fields. Testing shall also show whether the process contains any bugs. The function shall also be tested to see if it works as per specs, and that the output of the tests match the expected output defined in the following defined test cases.

Hardware used:

System Architecture:

- Server Host: Amazon EC2 2vCPU 4GB RAM
- Amazon Linux 2 Operating System
- MySQL v8.0.19 Database
- Nginx v1.19 Web Server
- Web Analytics: Google Analytics

User System Specs:

PC specs:

- Intel i7 11700KF
- 16GB RAM

- Nvidia 3080 10gb graphics card
- ASRock motherboard
- 500 gb SSD

MacBook specs:

- Intel Core I5
- 16 GB RAM
- 512 GB SSD

Software Used:

First Set of Tests:

- Windows 11 Operating System
- Chrome Web Browser
- Link to post page: https://csc648.mskim.dev/createpost

Second Set of Test:

- Mac Operating System: macOS Monterey v12.3
- Safari Web Browser
- Link to post page: https://csc648.mskim.dev/createpost

Feature to be tested:

Our feature to be tested is our post function. As stated above, this function shall allow a user to create a new post within our website by inserting a description, price, image, and other information into the data fields listed within our post page.

Test cases with Windows 11 and chrome web browser

Test #	Test Title	Description	Input	Expected Output	Result
1.	Posting an item	Create a new post.	Click on Post Items. Input for form fields: Macbook Air, 950, Electronics, Picture of macbook (item to be posted), Selling a used macbook air.	Message “Your post has been uploaded. It will take upto 24 hours for the post to be reviewed and approved.”receive d when the button “Post Item” is clicked	PASSED
2.	Check if data is recorded in the database	Check whether the data used is correctly inserted into our database.	After clicking “Post Item” and being approved, open MySQL	Check that the following is recorded in the database: item_id = 14,	PASSED

			Workbench and run the following query: “SELECT * from csc648.item;”	item_seller_id = 10, item category = 4, item_name = Macbook Air	
3.	Check item is not live	Check if the item is posted on the home page	Open Chrome web browser and input “https://csc648.mskim.dev/” into the search bar. Find the posted item	Item posted is the top left item, since it is the latest item	PASSED

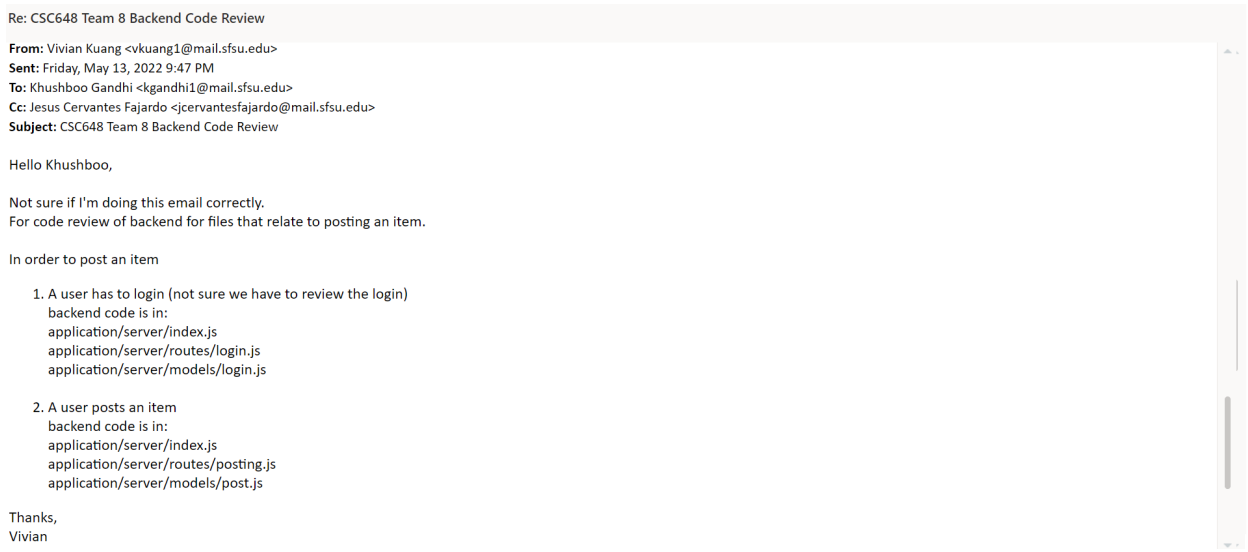
Test cases with macOS and Safari web browser

Test #	Test Title	Description	Input	Expected Output	Result
1.	Posting an item	Create a new post.	Click on Post Items. Input for form fields: Futon Sofa, 200, Furniture, Picture of futon sofa (item to be posted), Moving out soon and need to get rid of this sofa.	Message received when the button “Post Item” is clicked	FAILED
2.	Check if data is recorded in the database	Check whether the data used is correctly inserted into our database.	After clicking “Post Item” and being approved, open MySQL Workbench and run the following query: “SELECT *	Check that the following is recorded in the database: item_id = 12, item_seller_id = 9, item category = 4, item_name =	FAILED

			from csc648.item;”	Macbook Air	
3.	Check item is not live	Check that the item is not posted on the home page	Open Safari web browser and input “https://csc648.mskim.dev/” into the search bar. Find the posted item	Item posted is the top left item, since it is the latest item	PASSED

4) Code Review:

Email Screenshots:



Re: CSC648 Team 8 Backend Code Review

From: Vivian Kuang <vkuang1@mail.sfsu.edu>
Sent: Friday, May 13, 2022 9:47 PM
To: Khushboo Gandhi <kgandhi1@mail.sfsu.edu>
Cc: Jesus Cervantes Fajardo <jcervantesfajardo@mail.sfsu.edu>
Subject: CSC648 Team 8 Backend Code Review

Hello Khushboo,

Not sure if I'm doing this email correctly.
For code review of backend for files that relate to posting an item.

In order to post an item

1. A user has to login (not sure we have to review the login)
   backend code is in:
   application/server/index.js
   application/server/routes/login.js
   application/server/models/login.js

2. A user posts an item
   backend code is in:
   application/server/index.js
   application/server/routes/posting.js
   application/server/models/post.js

Thanks,
Vivian

Re: CSC648 Team 8 Backend Code Review

docs.google.com

Thanks,
Vivian

From: Khushboo Gandhi <kgandhi1@mail.sfsu.edu>
Sent: Friday, May 13, 2022 11:41 PM
To: Vivian Kuang <vkuang1@mail.sfsu.edu>; Jesus Cervantes Fajardo <jcervantesfajardo@mail.sfsu.edu>
Subject: Re: CSC648 Team 8 Backend Code Review

Hello!
I just realised I have to inform you about the files too.

Posting an item!
application/web/src/components/ItemPost

The QA mentions about deleting and updating post but our website does not allow us to do that lol.
so idk what we are supposed to do there. We will ask mark tomorrow.

I am not fully sure about this mail either. Just let me know if you want the locations of some specific pages.

Thanks,
Khushboo

Re: CSC648 Team 8 Backend Code Review

From: Vivian Kuang <vkuang1@mail.sfsu.edu>
Sent: Saturday, May 14, 2022 2:43 PM
To: Khushboo Gandhi <kgandhi1@mail.sfsu.edu>; Jesus Cervantes Fajardo <jcervantesfajardo@mail.sfsu.edu>
Subject: Re: CSC648 Team 8 Backend Code Review

Hello,

Here's my summary:

Yes, there is a header for the code maybe it could be more descriptive.
Yes, there are inline comments for the code maybe there should be an inline comment for the main functions.
The functions are named appropriately so that someone would know what it does without the need of a comment.
The code is also organized well in the order of imports, functions/variables, then html.
Yes, there are proper and consistent class/methods/variable names.
No, there is no consistency with naming established in the DATA section of Milestone 2 but I don't think that's for frontend. Those naming is more for backend mySQL.
The commit comments are okay.

And a link to doc I commented on. I copy and paste the code onto a Google Doc.
https://docs.google.com/document/d/11YBckz1rCBk1pOMagqcsFix0jl1GesNv-BaH7LHs--E/edit?usp=sharing

Frontend Code Review

// HEADER:Create A Post Code import React, { useState, useEffect } from "react"; import { Row, Col, Container, Button, Dropdown, ButtonGroup, Form } from "react-bootstrap"; import styles from "./index.module.css"; import image from "../../images/image.png"; import axios from "axios"...

Re: CSC648 Team 8 Backend Code Review

KG Khushboo Gandhi
To: Vivian Kuang; Jesus Cervantes Fajardo

Sat 5/14/2022 7:21 PM

Hello,
Here's my summary for the backend code review.

The pages which I am reviewing are linked here:

https://github.com/CSC-648-SFSU/csc648-03-sp22-team8/blob/devm3-route-testing/application/server/routes/posting.js

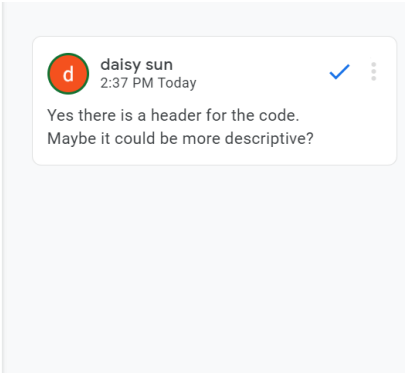
https://github.com/CSC-648-SFSU/csc648-03-sp22-team8/blob/devm3-route-testing/application/server/models/post.js

Summary:
There is a header present for the code that provides a clear description of what it is.
There are proper and consistent class names, variable names are well defined, and it does what is needed without comments.
A bit more description on what the storage function is doing and how multer works. That part seems confusing to understand.
The code is also organized in terms of functions, variables and the SQL functions in the model is also easy to understand.
There is consistency with the class names from the DATA section of the milestone 2.
Git commits are good with explanation.

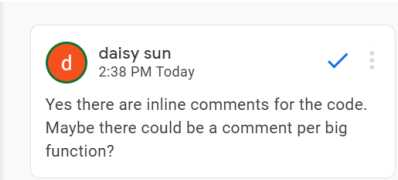
Thanks,
Khushboo

Backend reviewing Frontend screenshots:

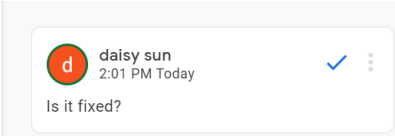
```
// HEADER:Create A Post Code
import React, { useState, useEffect } from "react";
import {
  Row,
  Col,
  Container,
  Button,
  Dropdown,
  ButtonGroup,
  Form
} from "react-bootstrap";
import styles from "./index.module.css";
import image from "../../images/image.png";
import axios from "axios";
import { useNavigate } from "react-router-dom";
```




```
/**
 * Load Post Item Page Component
 * @returns HTML of ItemPost Component
 */
const ItemPost = () => {
  const [itemname, setItemname] = React.useState("");
  const [price, setPrice] = React.useState("");
  const [categories, setCategories] = React.useState([]);
  const [course, setCourse] = React.useState("");
  const [description, setDescription] = React.useState("");
  const [uploaded_pic, setUploadedPic] = useState();
  const [imageToShow, setImageToShow] = useState(image);
  const [selectedCategoryId, setSelectedCategoryId] = useState(0);
  const [selectedCategory, setSelectedCategory] = useState("Select Category");
  const [toggleCourse, setToggleCourse] = useState(false);

  const navigate = useNavigate();
  const userInformation = localStorage.getItem("user_login_information");
```

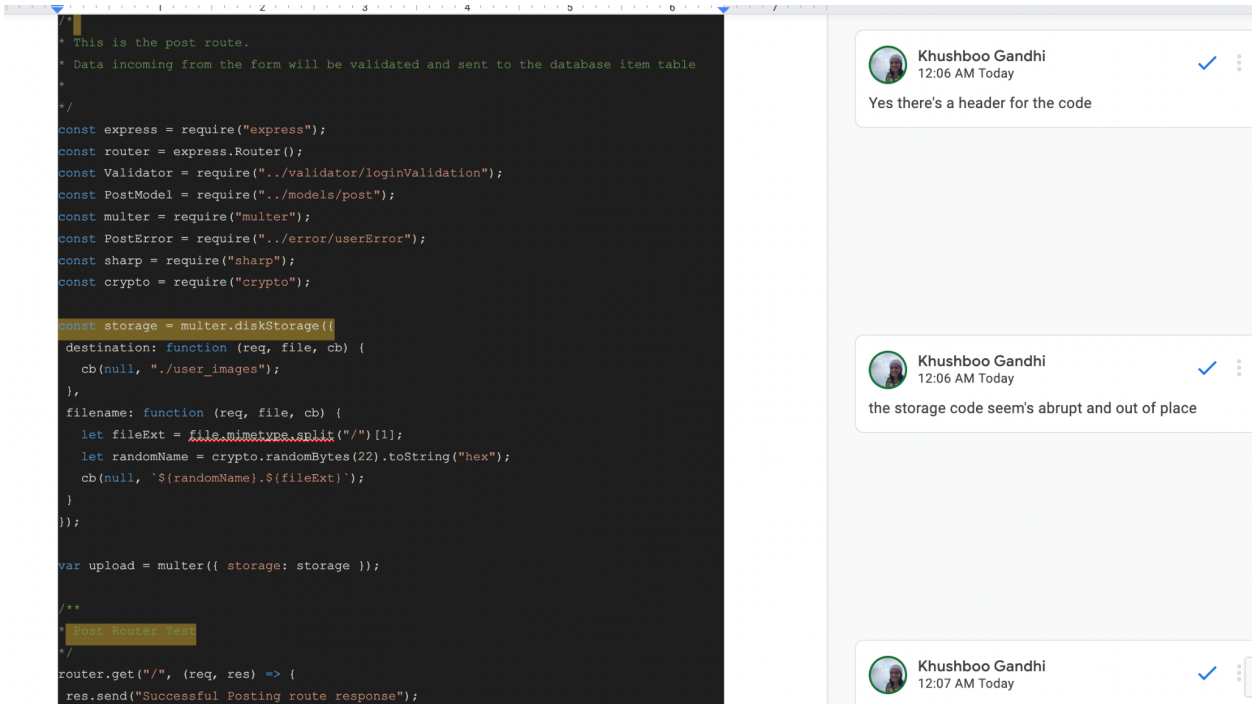



```
  //TODO: Fix this
  const checkValidity = () => {
    console.log("In validity");
    if (
      uploaded_pic === null ||
      price === "" ||
      itemname === "" ||
      description === "" ||
      selectedCategoryId === null
    ) {
```



Backend reviewing Frontend summary:
Yes, there is a header for the code, maybe it could be more descriptive.
Yes, there are inline comments for the code, maybe there should be an inline comment for the main functions.
The functions are named appropriately so that someone would know what it does without the need of a comment.
The code is also organized well in the order of imports, functions/variables, then html.
Yes, there are proper and consistent class/methods/variable names.
No, there is no consistency with naming established in the DATA section of Milestone 2 but I don't think that's for frontend. Those naming is more for backend mySQL.
The commit comments are okay.

Frontend reviewing backend screenshots:

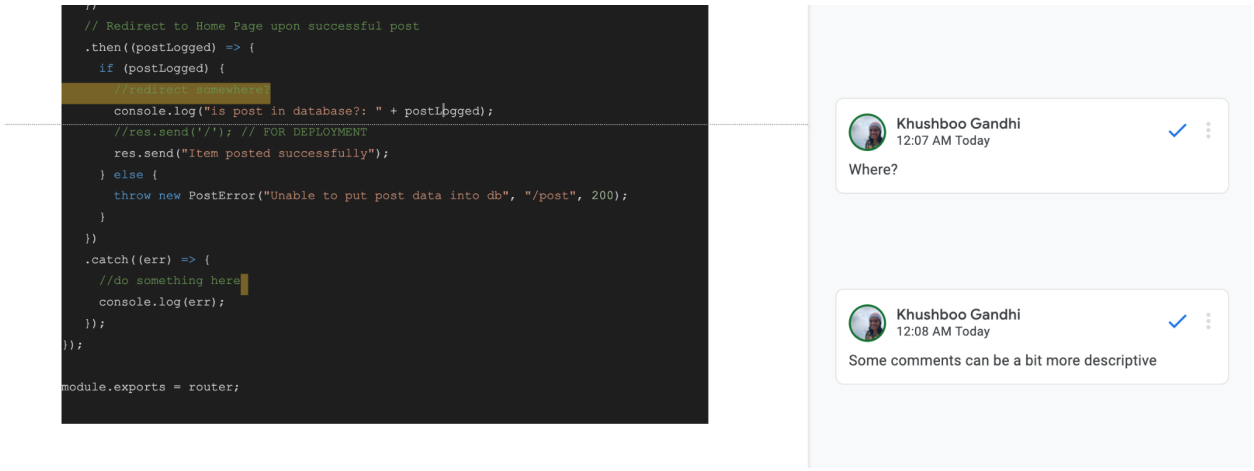


```
/**
* This is the post route.
* Data incoming from the form will be validated and sent to the database item table
*
*/
const express = require("express");
const router = express.Router();
const Validator = require("../validator/loginValidation");
const PostModel = require("../models/post");
const multer = require("multer");
const PostError = require("../error/userError");
const sharp = require("sharp");
const crypto = require("crypto");

const storage = multer.diskStorage({
 destination: function (req, file, cb) {
   cb(null, "./user_images");
 },
 filename: function (req, file, cb) {
   let fileExt = file.mimetype.split("/")[1];
   let randomName = crypto.randomBytes(22).toString("hex");
   cb(null, `${randomName}.${fileExt}`);
 }
});

var upload = multer({ storage: storage });

/**
* Post Router Test
*/
router.get("/", (req, res) => {
 res.send("Successful Posting route response");
```

Khushboo Gandhi
12:06 AM Today
Yes there's a header for the code

Khushboo Gandhi
12:06 AM Today
the storage code seem's abrupt and out of place

Khushboo Gandhi
12:07 AM Today



```
  // Redirect to Home Page upon successful post
  .then((postLogged) => {
    if (postLogged) {
      //redirect somewhere?
      console.log("is post in database?: " + postLogged);
      //res.send('/'); // FOR DEPLOYMENT
      res.send("Item posted successfully");
    } else {
      throw new PostError("Unable to put post data into db", "/post", 200);
    }
  })
  .catch((err) => {
    //do something here
    console.log(err);
  });
});

module.exports = router;
```

Khushboo Gandhi
12:07 AM Today
Where?

Khushboo Gandhi
12:08 AM Today
Some comments can be a bit more descriptive

Summary:

- There is a header present for the code that provides a clear description of what it is.
- There are proper and consistent class names, variable names are well defined, and it does what is needed without comments.
- A bit more description on what the storage function is doing and how multer works. That part seems confusing to understand.

- The code is also organized in terms of functions, variables and the SQL functions in the model is also easy to understand.
- There is consistency with the class names from the DATA section of the milestone 2.
- Git commits are good with explanation.

5) Self-check on best practices for security – 1⁄2 page

- List major assets you are protecting
- List major threats for each asset above
- For each asset above say how you are protecting it (1-2 lines of text per each)
- Confirm that you encrypt PW in the DB
- Confirm Input data validation (list what is being validated and what code you used) / We require that at minimum you validate
    - search bar input for up to 40 alphanumeric characters
    - registration e-mail to include “sfsu.edu” at the end;

How to present: Best is to present this in the table below, include ALL assets and methods you plan to implement

Asset to be protected	**Types of possible/expected attacks**	**Your strategy to mitigate/protect the asset**
Passwords of Users	Hackers can brute force attack and hash collision to try to solve the password. Also attacks can be done through phishing	We’re using Bcrypt to hash the passwords. Bcrypt can slow down the attackers and is adaptive
Messages of Users	Phishing, SQL code injection.	Message route only made available to registered users; unregistered users cannot message. When you login you only have access to your messages.
Posts of Users	Phishing, SQL code injection	Post route only made available to registered users; unregistered users cannot post or see their post history until they login or register. Keep your login information secured in your notes.
Input Data Validation: Search Input limit (40 char)	Code injection	Show the error in the frontend when you try to search over 40 characters. The 40 character limit reduces the

		number of characters available for code injection.
Input Data Validation: Registration Page	They can do SQL injection to get information. If user input is incorrectly filtered, SQL statements can be executed by the application.	Show the errors in the frontend when you incorrectly put the password, email,name. Validation prevents code from being injected into the form
Input Data Validation: Login Page	They can do SQL injection to get information. If user input is incorrectly filtered, SQL statements can be executed by the application.	Show the errors in the frontend when you incorrectly put the password, email,name. Validation prevents code from being injected into the form

6) Self-check of the adherence to original Non-functional specs – performed by team Leads

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0
   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
   ON TRACK (Safari, Chrome)
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.
   DONE
5. No more than 50 concurrent users shall be accessing the application at any time
   DONE
6. Privacy of users shall be protected
   DONE
7. The language used shall be English (no localization needed)
   DONE
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
    DONE
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
    DONE
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
    DONE
15. Media formats shall be standard as used in the market today
    DONE
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, Spring 2022. For Demonstration Only” at the top of the WWW page nav bar. (Important so as to not confuse this with a real application).
    DONE