CSC 648/848 Summer 2019

Green World
Team 06

Austin Tsang (atsang3@mail.sfsu.edu)
Audrey Wong
Han Huang
Alexander Caley
Gebrezgabiher Mengis
Jesus Valdes
Syed Abidi

Milestone 4
8/6/19

1) Product summary (e.g. how would you market and sell your product)

- Green World
- P1 List.
    - Homepage
        - Show List of all items sorted by recently added
        - Each item can be clicked to get more details about item
        - Detailed item will show data about the clicked item as well as the location on the map
    - Register
        - Users can register for an account
        - If account is already being used, show error message
        - Passwords are encrypted
    - Login
        - Users can sign in with existing accounts
        - If password or username is incorrect show error message
        - Utilizes Json Web Token(JWT) to validate user session.
    - Logout
        - Kill users current session.
    - Post
        - Form that takes in users image, type of hazard, user location, and a description
        - Inserts into database.
    - Search
        - Can search based on items in database.
        - 3 different kinds of searches based on:
            - Any match in all items
            - Type of Issue
            - Date, most recent and oldest
    - Styling
        - Original Styling on all pages, including navigation bar and footer.
        - Responsiveness

- URL: http://ec2-18-144-30-72.us-west-1.compute.amazonaws.com/home

2) Usability test plan

- Test Objectives:
    - Posting
        - Posting is important because without users posting items or issues, we don’t have any data to display
        - Without any data to display, we no longer have a reason why users should visit our site.
        - This needs to be tested to ensure that items can be posted without having errors or issues to ensure a good user experience.
        - If a user is having issues inserting items, then you will probably lose that user almost instantly
- Test background and setup
    - To measure effectiveness
        - Fill in the fields and check if there's any issue filling them out, including adding a single image(don’t add more than one).
        - After clicking submit check the database and see if it has uploaded to the database
        - Check the homepage and see if item is being display after posting.
    - To measure efficiency
        - Does everything work properly?
        - Does it take long for the user to post and check page?
        - Are there any bugs?
    - Lickert subjective test

Question	Very bad	Bad	Ok	Good	Very Good
How easy was it to post the issue?					
Did the pages look appealing?					
Experience with adding items?					

3) QA test plan - max 2 pages

Test objectives:

- Testing to see if items get inserted properly
- Upload form works
- After inserting into post it appears on homepage
- Items are inserted into database

HW and SW setup(including URL):

- Tested on PC: mac and window all works as expected
- Mobile Iphone and android all works as expected
- Works on multiple browsers like chrome, safari and firefox
- Link: **http://ec2-18-144-30-72.us-west-1.compute.amazonaws.com/post** (only works if the AWS is on)

Feature to be test:

- Posting

QA Test plan:

- test #: 1 tested on multiple browsers
- test title: Post type
- test description: check if the post type that is inserting is correct
- test input: "Garbage"
- expected correct output: it inserted to the database
- test results (PASS or FAIL for each tested browser): PASS

- test #: 2
- test title: Post status
- test description: check if the status that is inserting is ether in progress or complete only
- test input: "in progress"
- expected correct output: it inserted to the database
- test results (PASS or FAIL for each tested browser): PASS

- test #: 3
- test title: Location
- test description: check if the location that is number, symbol or letter for the address, letter for city and only numerical for the zipcode.
- test input: "94114"
- expected correct output: it inserted to the database
- test results (PASS or FAIL for each tested browser): PASS

4) Code Review:

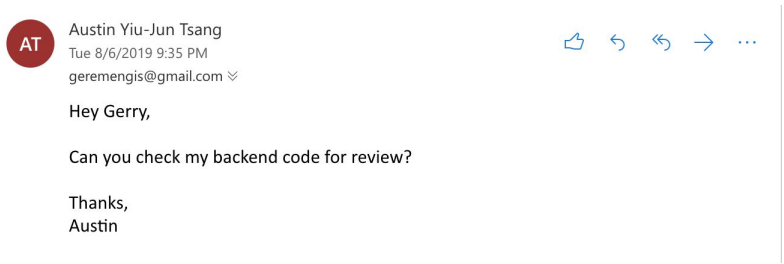
a. Code style:
   i. Variables
      1. Messages and errors are declared using using caps with underscore
      2. Function names are used by camel casing.
      3. Prettier extension in Visual Studio Code to handle indentations and spacing
      4. Comments above function to explain what the function is used for.

```
// Postings into database from post form
router.post('/', upload.single('image'), function (req, res, next) {
   var {id, location, postType, postStatus, picture} = req.body;Î

   let query = "INSERT INTO postings (location, postType, postStatus, picture) VALUES
(?,?,?,?)"

   connection.query(query, [location, postType, postStatus, picture],
       function (err, result) {
           if (err) {
               console.log(err);
               res.send('Error');
           }
           else {
               console.log('success');
               res.send('Success');
           }
       });
});
```

b. Code review:



```
// Postings into database from post form
router.post('/', upload.single('image'), function (req, res, next) {

   // Gerry M.
   // ID variable does not have value when posting
   // Picture also does not have value
   var {id, location, postType, postStatus, picture} = req.body;î

   let query = "INSERT INTO postings (location, postType, postStatus, picture) VALUES
(?,?,?,?)"

   connection.query(query, [location, postType, postStatus, picture],
       function (err, result) {
           if (err) {
               console.log(err);

               // Gerry M.
               // Messages must be in caps
               res.send('Error');
           }
           else {
               console.log('success');

               // Gerry M.
               // Messages must be in caps
               res.send('Success');
           }
       });
});
module.exports = router;
```

5) Self-check on best practices for security – ½ page

- Major assets we are protecting
    - Users password and information that they posted
    - Users session data, including JWT
- We are encrypting passwords using Bcrypt API and we are encrypting users JWT using with a random hash. We prevent SQL injections
- Text up to 50 in text bar is being used. We have characters up to 255 characters to insert in database.

6) Self-check: Adherence to original Non-functional specs

List of 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. 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. **ISSUE. UNCERTAIN ON THIS ONE**
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. **ON-TRACK**
9. Google analytics shall be added. **DONE**
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. **DONE**
12. Site security: basic best practices shall be applied (as covered in the class). **DONE**
13. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development. **ON-TRACK**
14. The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Summer 2019. For Demonstration

Only” at the top of the WWW page. (Important so as to not confuse this with a real application).**ON-TRACK**