CEN 4010 Principles of Software Engineering Fall 2019

Milestone 4 Beta Launch

Team Name: Code Exterminators
Project Name: Campus Snapshots
Team Number: 15
Students: Bryan Way:
Front-end Developer - bway2019@fau.edu

Joseph Burton:
Back-end Developer, Product Owner - josephburton2017@fau.edu

Corey Jones:
Back-end Developer, Back-end Team Lead, Scrum Master - coreyjones2016@fau.edu

Dominic Morocco:
Front-end Developer, Front-end Team Lead
dmorocco2016@fau.edu

Documentation Date: November 18, 2019

History Table

Revision Number	Revision Date	Description
1	November 4, 2019	Initial Version
2	November 18, 2019	Beta Launch

Product Summary

Many students whether they live on campus or commute rely on the facilities provided on campus including dining locations, the library, recreational facilities, dormitories, parking areas, and classrooms. Additionally, campus administration must respond to the needs of students and ensure that these facilities adequately provide for the students and are properly maintained. When these services are not maintained well it can negatively impact students and leave them dissatisfied with the operation of the university.

In order to provide for the needs of both students and administration we propose Campus Snapshots as a platform for providing up to date information regarding issues and events on campus. The purpose of this platform will be to keep university administration up to date on the current status of the campus and problems affecting students. It will also provide a means for students to point out problems that they notice on campus and seek a resolution to them. Problems intended to be reported on the platform include issues such as broken AC, clogged toilets, full trash bins, or other utilities on campus in disrepair.

The intention of Campus Snapshots is to improve the quality of service provided by the campus and allow for issues to be quickly and accurately identified. We also hope that this service will create a greater sense of community between students and administration by allowing them to cooperate on creating a better environment on campus. Students should be able to see real results when they report issues that they encounter by having them resolved in a timely manner and campus administration can be confident that they have an accurate view of the current state of the campus.

Features of Campus Snapshots

- Users on Campus Snapshots can create a profile after registering with an email, username, and password
- Users create posts made up of text and images that other users can see.
- Users can create special posts called reports that consist of an issue around their campus with an image of the associated issue. After being created, these reports enter a queue where staff users can see and address them, alerting the sending user of when the status of a report has been updated.
- Users can also create public events that other users can join and participate in until its time frame is over.
- If a user sees an inappropriate post, they can report it and have a staff user look at it and determine whether or not to punish the poster.

Link to Campus Snapshots:
http://lamp.cse.fau.edu/~cen4010fal19_g15/CampusSnapshots/index.php

Usability Test Plan

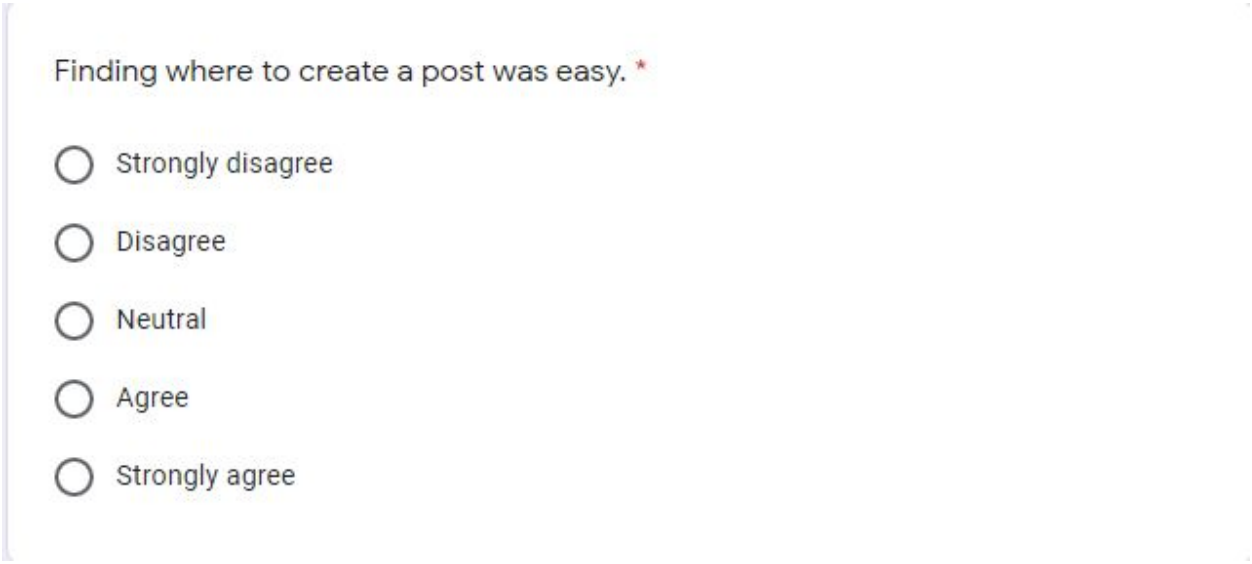
Test Objectives

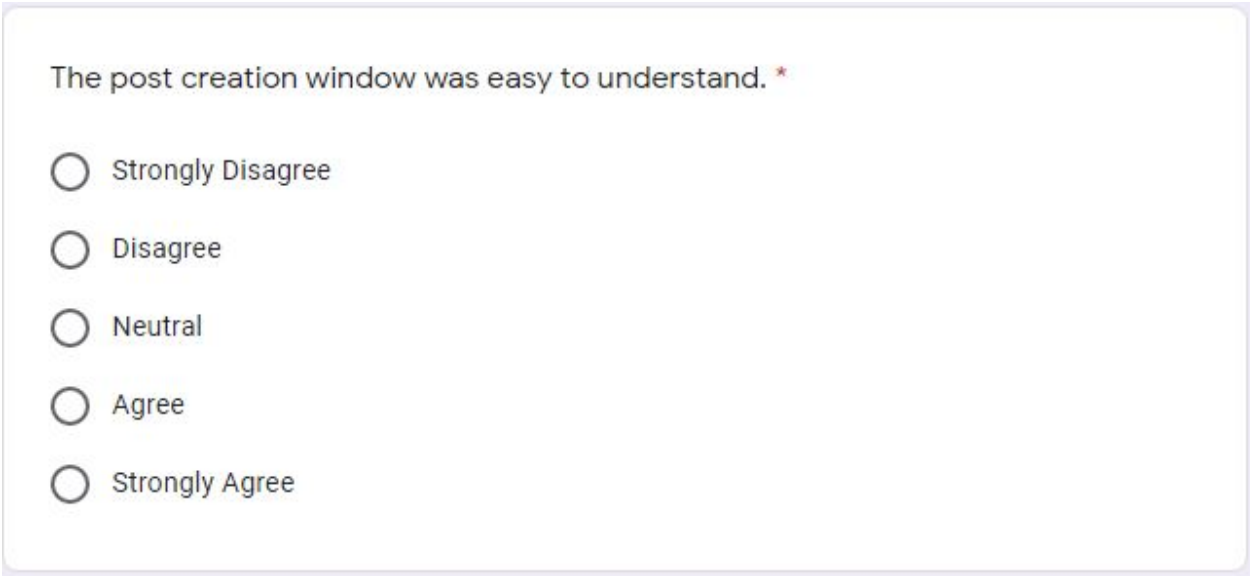
The goal of this usability test is to test how users feel about the execution of the post creation feature in Campus Snapshots. We will be aiming to record how users felt about certain aspects of the post system after using Campus Snapshots by having them fill out a Questionnaire created with three Lickert scale formatted questions. Based on their answers, we will change or modify features in the future.

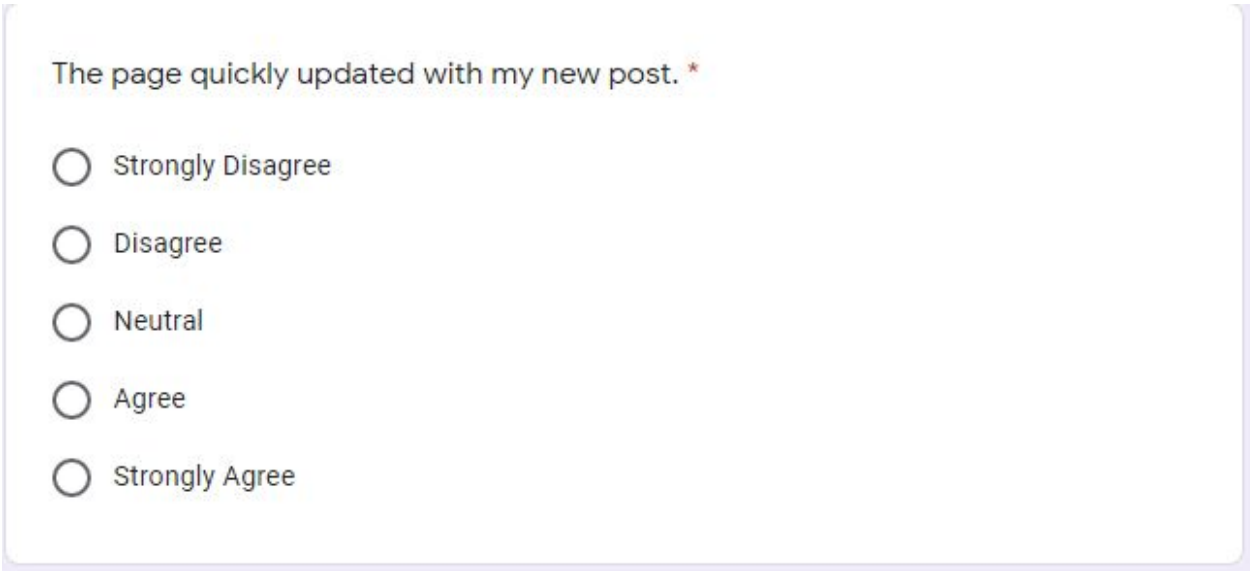
Test Plan

System Setup: Test Users will only need to create an account on Campus Snapshots and attempt to create a post.
Starting Point: The test begins from the post window in Campus Snapshots.
Task to be accomplished: The task to be accomplished is that a post is successfully pushed to that user's database and page.
Intended User: All users of Campus Snapshots are intended to use the post feature.
Completion Criteria: When the test user has completely entered a post and submitted it.
URL to be tested: http://lamp.cse.fau.edu/~cen4010fal19_g15/CampusSnapshots/index.php

Questionnaire Form







https://docs.google.com/forms/d/1XuBs8SskRfh7AbnxfOSa0YjcMcRtpuQKNEgJ1-AwjTA/

QA Test Plan

Test Objectives

The goal of this QA test is to test the execution of the post creation feature in Campus Snapshots. We will be testing to find bugs and non-desirable behavior within the posting function by entering various types of data into the post window and testing whether or not the system reacts the way we expect it to.

Hardware and software setup

For testing, the hardware necessary is a working computer with internet access. The software necessary is just a modern browser (such as Google Chrome, Firefox, etc.). Testing users will be directed to the Campus Snapshots URL and create an account. After signing up, they will log in and attempt to create to create multiple posts.

Feature to be tested

The feature to be tested is full post creation, which involves the following features:

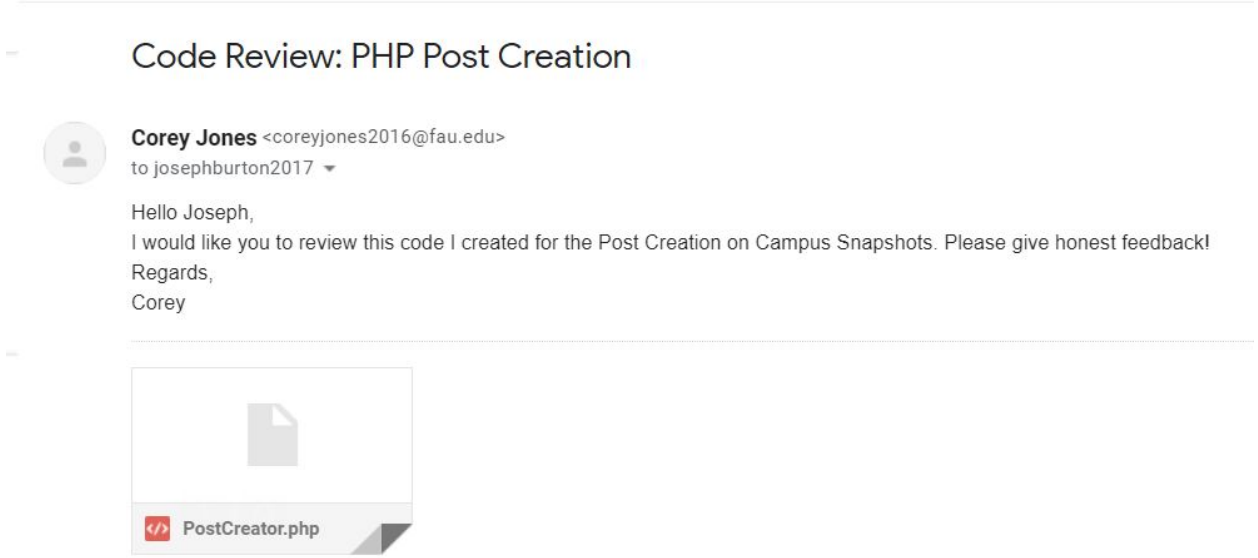
- Opening the post creation interface and creating a post
- Publishing a created post and saving it to the user's database
- Check that published posts are correctly displayed on the user's page

Actual test cases

Test Case #	Test Case Description	Test Steps	Test Data	Expected Result	Test Results
1	We are going to enter expected data into the window, namely, plain text.	1) Open a new post window 2) Enter text into the post window and finish creating the rest of the post 3) Press "Submit Post"	Input = "Hello World!"	Post should successfully appear on User's page	Google Chrome: Firefox:
2	We are going to enter nothing in the window.	1. Open a new post window 2. Press "Submit Post"	Input = ""	Post should be rejected	Google Chrome: Firefox:
3	We are going to enter -1-1-1-1 into the window.	1. Enter text into the post window and finish creating the rest of the post 2. Press "Submit Post"	Input = "-1-1-1-1"	Post should successfully appear on User's page	Google Chrome: Firefox:

Code Review

The coding style we have chosen is standard protocol for the languages used (for example, PHP states are tabbed out under if statements and the curly braces begin on the if statement line and end on the tab under the initial if keyword).



Initial code review email

Joseph Burton
to me

```
<?php
   require 'PHP/connection.php';
   if (empty($_POST['title']) == true or empty($_POST['location']) == true or empty($_POST['description']) == true {
        $errorMessage .= <li>"You are missing either a title, location, or description. Please enter this information and try again."</li>
   }
   elseif (empty($_POST['title']) == false and empty($_POST['location']) == false and empty($_POST['description']) == false ) {
               chdir('../Files/Events/');//make sure folder is created
               if(!file_exists($_SESSION['UID'])) {
                       mkdir($_SESSION['UID']);//add proper permissions to signature
               }
               if(!chdir($_SESSION['UID'])) {
                       die("Error changing Directory");
               }
               $posttype = $_POST["type"];
               if ($posttype == 'event') {
                       $conn = connect();//should be moved to above the check
                       $title = $_POST['title'];//move like variable out of check
                       $location = $_POST['location'];
                       $startdate = $_POST['startDate'];
                       $enddate = $_POST['endDate'];
                       $description = $_POST['description'];
                       $title = htmlspecialchars($title);//not needed
                       $location = htmlspecialchars($location);//not needed
                       $description = htmlspecialchars($description);//not needed
                       $addevent = $conn->prepare("INSERT INTO Event VALUES( DEFAULT, ?, ?, ?, ?, ?, ?)");
                       $addevent->bindParam(1, $_SESSION['UID']));
                       $addevent->bindParam(2, $title));
                       if(isset($_FILES["fileToUpload"])){
                               include 'fileUpload.php';//awkward will need to turn into function
                               $addevent->bindParam(3, $_COOKIE['CurrentFile']); //given null on no picture
                       }
```

Joseph's reply (1): code came back packed with helpful comments and fair critique

```
                       if(isset($_FILES["fileToUpload"])){
                               include 'fileUpload.php';//awkward will need to turn into function
                               $addevent->bindParam(3, $_COOKIE['CurrentFile']); //given null on no picture
                       }
                       $addevent->bindParam(4, $_POST['startDate']));
                       $addevent->bindParam(5, $_POST['endDate']));
                       $addevent->bindParam(6, $_POST['description']));
                       $addevent->execute();
               }
               elseif ($posttype == "report") {
                       $conn = connect();//not needed if first note is followed
                       $title = $_POST['title'];//move like variables out of check
                       $location = $_POST['location'];
                       $description = $_POST['description'];
                       $type = $_POST['reporttype']; //if we choose to add in selecting a type, edit this line with the correct name of the field!
                       $title = htmlspecialchars($title);//not needed
                       $location = htmlspecialchars($location);//not needed
                       $description = htmlspecialchars($description);//not needed
                       $type = trim($type);//not needed
                       $type = stripslashes($type;// error and not needed
                       $type = htmlspecialchars($type);//not needed
                       $addreport = $conn->prepare("INSERT INTO Reports VALUES(DEFAULT, ?, ?, ?, DEFAULT, ?, ?, ?)");
                       $addreport->bindParam(1, $_SESSION['UID']));
                       $addreport->bindParam(2, $title));
                       if(isset($_FILES["fileToUpload"])){
                               include 'fileUpload.php';//turn into function and push include to top
                               $login->bindParam(3, $_COOKIE['CurrentFile']);//given null on no picture
                       }
                       $addreport->bindParam(4, $type);
                       $addreport->bindParam(5, 1));//wrong function need bindValue
                       $addreport->bindParam(6, $description));
                       $addreport->execute();
               }
   }//anything to echo?
?>
Verbose but logic is sound make sure the top if statement works and is written correctly bottom one is needed only if there is more than one option that causes the top to fail.
Up to you for readability.
```


Joseph's reply (2)

Self-check on best practices for security

Major assets to be protected: Usernames, Passwords, E-mail addresses, First Names, Last Names

Passwords are encrypted using the standard hashing algorithm provided by PHP (PASSWORD_DEFAULT）

Password
$2y$10$pwC5W0E3z.DB.5BboWlRSu5G5JjN8dbAn3R5BnsiQuZ...
$2y$10$eFM4FGjFOVWNC3BTd5o2C.LzhFFOa7fyBHyVRv7vfb0...
$2y$10$jbnGQHKriDepTbUwQffPxOpu1lOb90gayd.zvQWZ6GL...
$2y$10$lPvh18Hlj4sYUEskhUocFuRgGjjxwlkJ1Dczwkyc9zJ...
$2y$10$z7yJoCCDjJDM1Qc9yixnzem5XyUc7i/lipw/tyh/4DE...

An example of encrypted passwords

Input data that is validated: Log-in information, Post data

```php
$f = $_POST["Username"];
$r = $_POST["Password"];

$conn = connect();

$data = trim($r);
$data = stripslashes($data);
$data = htmlspecialchars($data);

//$r = password_hash($r, PASSWORD_DEFAULT);

$login = $conn->prepare("SELECT UID, Password FROM Person
WHERE UserName = ? LIMIT 1");
$login->bindParam(1, $f);

$login->execute();
$login->setFetchMode(PDO::FETCH_ASSOC);
$result = $login->fetch();

if(password_verify($r, $result["Password"])){
```

A portion of the log-in PHP. It formats the username given before attempting to search for it. The given password is then compared using password_verify() to the password stored at that UID.

```php
if (empty($_POST['title']) == true or empty($_POST['location']) == true or empty($_POST['description']) == true {
$errorMessage .= <li>"You are missing either a title, location, or description. Please enter this information and try again."</li>
}
elseif (empty($_POST['title']) == false and empty($_POST['location']) == false and empty($_POST['description']) == false ) {
    chdir('../Files/Events/');
    if(!file_exists($_SESSION['UID'])) {
        mkdir($_SESSION['UID']);
    }
    if(!chdir($_SESSION['UID'])) {
        die("Error changing Directory");
    }
    $posttype = $_POST["type"];
    if ($posttype == 'event') {
        $conn = connect();
        $title = $_POST['title'];
        $location = $_POST['location'];
        $startdate = $_POST['startDate'];
        $enddate = $_POST['endDate'];
        $description = $_POST['description'];
        $title = htmlspecialchars($title);
        $location = htmlspecialchars($location);
        $description = htmlspecialchars($description);
```

A portion of the post creation PHP. It will not continue if any required input field is empty. It also addresses special characters in title, location, and description.

Self-check: Adherence to original Non-functional specs

- All Users can create profiles through a username and password. **DONE**
- Normal Users will register by providing a username, password, email address, and full name. **DONE**
- Staff Users will have their accounts created by the site administrators. **DONE**
- All Users can create public posts (consisting of text and images) that are visible to all other users. **ON TRACK**
- Normal Users can create report posts (consisting of text and images) that are visible to Staff Users in the report queue. **ON TRACK**
- Users must provide an image related to the issue they are reporting along with the location of the issues, a title, and a description. **DONE**
- A User can see the status of their report (reported, in process, and resolved). **ON TRACK**
- All Users can create public events, which other Users can join, that end after a certain period of time or end after being closed by a Staff User. **ON TRACK**
- Users must specify a time and date range for when their event will occur along with an image, title, location, and description. **DONE**
- Staff Users can view and modify a report queue which consists of pending and in process Normal Users reports. **ON TRACK**

Peer Review

Team Member	Points
Joseph Burton	35/25
Corey Jones	30/25
Dominic Morocco	0/25
Bryan Way	35/25
Total	100