**CEN 4010 Principles of Software Engineering**

**Fall 2019**

**Milestone 4: Beta Launch and Reviews**

**Project Name: Owl Eyes**

**Team Number: 13**

**Names and roles of team members:** Benjamin Blakely (Product Owner), Said Blanc (Development team), Sola Fatade (Development Team), Caren Youssef (Scrum Master)

**Date of the document:** 11/18/2019

**Revision history table**

|  |  |
| --- | --- |
| **Revision Dates** | **Documented Changes** |
| 9/23/2019 | Initial Submission |
| 10/2/2019 | Updated data definition and non-functional requirements sections to meet suggestions made by TA. |
| 10/26/2019 | Updated the functionality on the GUI and resubmitted to meet the suggestions made by the TA. |
| 11/4/2019 | Milestone 3 Update |
| 11/8/2019 | Re-submission of Milestone 3 with updated Risk Assessment and UML diagrams |

**Product Summary**

1. Name of product: Owl Eyes
2. All major committed functions (FINAL priority list from Milestone 3, the “1”)

* Easy to navigate feed
* Students can upload reports to the website using text and a picture
* Organizations can upload events and advertise their activities
* Campus staff has privileges to control the website and send out important alerts/messages

1. Our product has various unique features, such as a live feed of events on campus posted by organizations, students’ ability to see updates from organizations that they are part of on campus, students’ ability to report issues and problems happening on campus through a report with text and visuals. In addition, our product will also include a search function, allowing for easy search of reports and locations.
2. <http://lamp.cse.fau.edu/~cen4010fal19_g13/public_html/>

**Usability test plan**

**Test Objectives:**

For our product, we were tasked with the objective of creating a web system to display Campus Snapshots.The Campus Snapshots would be a current visual description of the state of campus. By using our system, users on campus would be able to report instantaneous issues that need to be fixed or that require attention. For example, students walking on the breezeway would be able to submit reports of dumpsters overflowing, or a broken lamp pole on the sidewalk. In addition, users would also be able to report on the good things happening on campus. Such events can include exciting games, outdoor concerts, students’ activities, etc. For organizations and clubs, this privilege will be in the form of uploading pictures and text to our website in order to advertise for their events quickly and efficiently. In this usability test plan, we will demonstrate the ability of a user (student, staff, or organization) to upload a report to our website including a representative image and text. The test will clearly show the system setup, the starting point, the task to be accomplished, the intended user, and the completion criteria. It will also walk the user through the process of logging in or signing up before they are able to complete the above test.

**Test plan:**

* System setup:

o The system will be setup so that when the user first reaches our website, they will be prompted to either log in or create an account before they are able to proceed. Before becoming a member of our website, users are restricted from viewing the reports, feed, news, etc. This is to maintain the security and protection of our university’s information.

* Starting point:

o The starting point of uploading a post or report to our website, Owl Eyes, will be to either create an account or log in as a returning user.

* Task to be accomplished:

o The task to be accomplished here is to successfully upload and view a report on Owl Eyes. To begin, the user will need to navigate either to reports if they are a student reporting an incident or to feed if they are an organization advertising their event on our website. Afterwards, they will reach the part of the page where there will be a “form” that they will need to fill out and submit. The form will take the location, description, image, and priority of the request. Once filled out, the report can be submitted to our website simply by clicking “submit”.

* Intended user:

o For this test plan, the intended user is the student; however, other user-types, such as organization or campus staff will also have the right to upload and submit reports.

* Completion criteria:

o The test is complete when the user clicks “submit” after filling out the respective form for their request and they are able to see the confirmation message displayed after. The message should read “Your request has been submitted successfully”. For further ensuring that the test was completed correctly, the user can refresh the feed/reports page to see their report displayed there.

* URL of the system to be tested:

o <http://lamp.cse.fau.edu/~cen4010fal19_g13/public_html/>

**Three Lickert scale questionnaire:**

For each of the questions below, please circle the response that best characterizes how you feel about the statement.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Strongly Disagree** | **Disagree** | **Neutral** | **Agree** | **Strongly Agree** |
| The feed was user-friendly and easy to use | 1 | 2 | 3 | 4 | 5 |
| The upload process was simple and easy to follow | 1 | 2 | 3 | 4 | 5 |
| I could see my submitted report successfully on the page | 1 | 2 | 3 | 4 | 5 |

**QA test plan- maximum 2.5 pages**

For the same function you chose for the usability test, write AND execute a QA test plan (class slides)

1. Create formal QA test plan to contain:
   1. Test objectives: 0.5 pages
      1. As our primary objective, it’s important to us that our main function - the reading and writing of user reports and feed items - is not only working as intended, but is designed to be as foolproof as possible.
      2. We recognize that without proper testing, it’s possible that some oversight in development creates a problem with deployment that could negatively impact the performance of the site or the user experience, so we’ll be seeking to make sure our system is robust as possible.
   2. Hardware and software setup:
      1. Our hardware setup for QA testing will include at least one desktop computer running Windows 10 as our primary vector for testing.
      2. Our site is designed with mobile in mind, as a secondary objective, so we’ll also be testing everything on a smartphone as well.
      3. It’s our goal that our website is functional on all major browser - however, for testing purposes, we’ll be accessing and testing the site using Google Chrome and Mozilla Firefox.
   3. Feature to be tested:
      1. Reading/Writing of Report Items
      2. Can reports be submitted by any user and then be accessed by any other user?
      3. Can only higher level users post items to the feed page?
      4. Can the list of reports be searched through by keyword in order to further filter the list?
   4. Actual test cases: 3 test cases and results of testing them in your system: 1 page
      1. Test Case 1: Create an account and submit a report.
         1. Result: Account created successfully and report submitted
      2. Test Case 2: Logout and Log into a different account and look at the report page
         1. Result: New account created and report page accurately displays the previously submitted report
      3. Test Case 3: Login as a regular student and attempt to upload feed. Then, Login as user:admin/pass:testadmin, and attempt to upload to feed.
         1. Result: Feed items uploaded both times and displayed accordingly.
      4. Test Case 4: Enter an item in the search bar
         1. Result: Items in the Report Inbox are filtered and only terms with matching keywords are returned.
   5. Present test plan and summary in table format

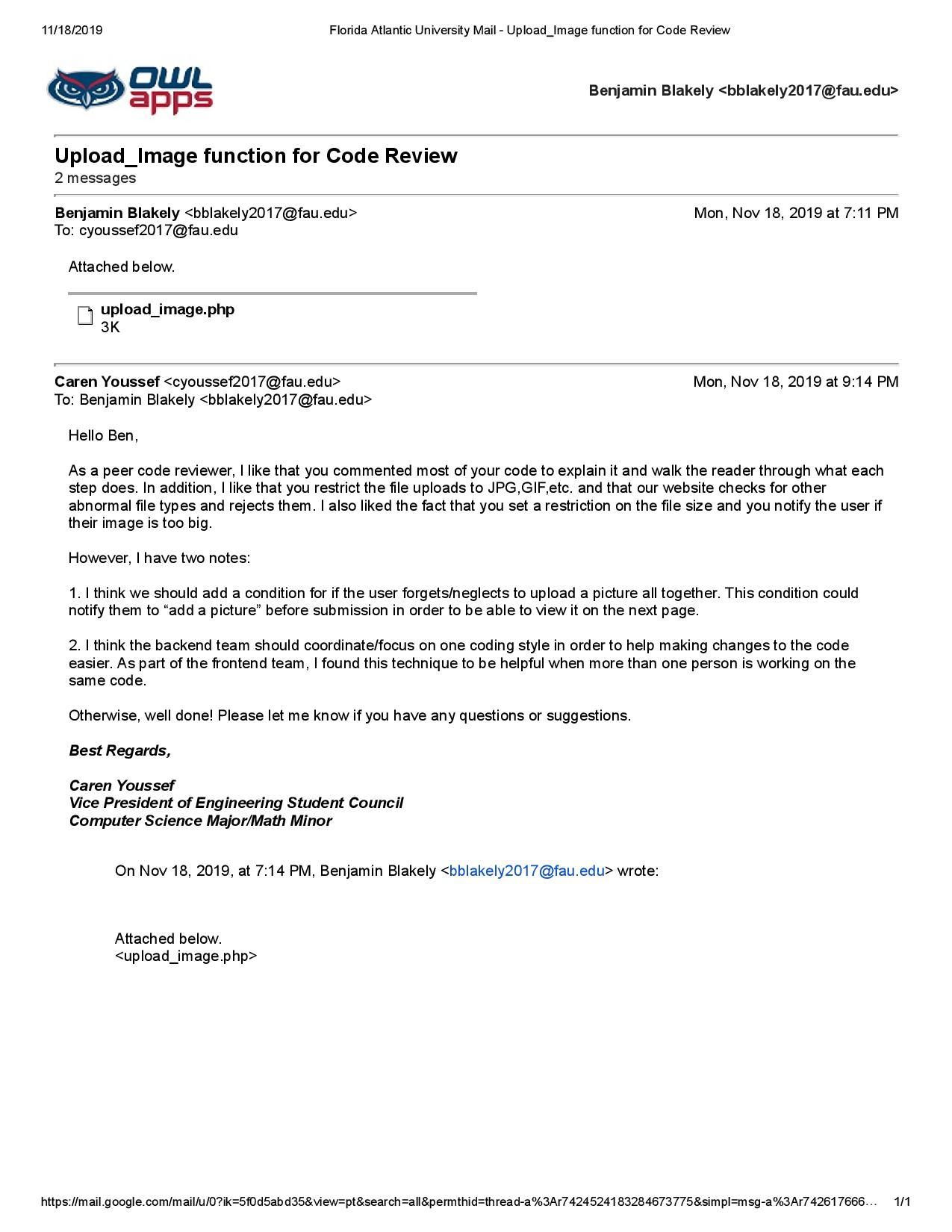
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | Test Title | Test Description | Test  Input | Expected  Correct Output | Test Results  (Chrome/Firefox) |
| 1 | Basic User Function Test | Testing Basic User Functions: Logging in and Filing Reports | Username/password  Report text fields + image | Logged in account session  Recorded report in database | Working in both browsers. |
| 2 | Cross-account Report Sharing | Testing that different accounts can correctly access reports filed by other users | Report text fields + uploaded image filed on initial account | Report text fields + uploaded image able to be read from other account | Working in both browsers |
| 3 | Normal User and Admin Account Privileges | Testing that normal users and admin accounts have different privilege levels | Differentiated user and admin account credentials | User and admin account correctly differentiated by different session privilege token. | Working in both browsers |
| 4 | Report Inbox Search Functionality | Testing that when a search term is entered, the report inbox is correctly filtered to only display the items that have a matching key. | Search terms: FAU, somewhere, abcdefg | Filtered inbox correctly returned: multiple objects for FAU, one object for somewhere, and no objects for abcdefg | Working in both browsers |

* + 1. Table columns: test #, test title, test description, test input, expected correct output, test results (PASS or FAIL for each tested browser)
  1. Apply above test on 2 browsers and record in table

**Code Review**

State what coding style you chose. Choose the code related to the feature you used for QA and usability test and submit an example of it (2 pages max). Document it as follows:

1. One team member submits the code to another team members for peer review
2. Peer review is done by e-mail and comments are to be included in the code
3. Submit listing containing the peer review and commented code



**Self-check on best practices for security- ½ page**

1. List major assets you are protecting
   1. Currently, we interact with and store the user credentials of any members of the site. This simply includes a username and password.
   2. However, further development of this system would probably mean that more of users’ personal information would be collected for validation and analytical reasons. For this reason, it’s important that we’re careful about exactly what information we need to keep and how we transmit that information, so as to keep user’s privacy protected at all cost.
2. Confirm that you encrypt password in the DB
   1. Passwords are never stored in our database in an unencrypted form - our code hashes the user’s password and that hash is stored in the database for future account validation.
3. Confirm input data validation (list what is being validated and what code you used)- request that we validate search bar input
   1. Client-side validation exists for some simple things - such as prompting the user that they have invalid characters in their input field.
   2. Furthermore, all user requests that are handled by the server are scrubbed to make sure that no external commands can be executed via an injection based attack.

**Self-check: Adherence to original Non-functional specs**

Copy all original non-functional specs as in high level application document published at the beginning of the class (milestone 1 I think) and for each, say DONE, ON TRACK, or ISSUE and explain it.

* System must be able to process data and handle requests from a large number of users distributed across a wide network without impacting the service's reliability - ideally, without impacting the user experience either.
  + DONE
* System is only meant to be accessible to the students and faculty of FAU, which means that an authentication protocol must be in place that can verify a potential user's connection to the school before their account is created.
  + ON TRACK
* Our service must have some means of effective administration - so that possible abuse of the system can be easily and actively mitigated.
  + DONE
* Security is of utmost importance, as unsecure accounts leave students vulnerable to a wide range of possible attacks. As such, it's highly advisable that all traffic is encrypted and user accounts are by default at least protected by some sort of 2FA (two-factor authentication).
  + ON TRACK

**Project Peer Evaluation:**

Caren: 25%

Said: 25%

Sola: 25%

Ben: 25%