**CEN4010 Principles of Software Engineering**

**Fall 2019**

**Milestone 1 Project Proposal and High-Level Description**

**Campus Snapshot**

By G10 Software (Group 10)

Proposed on September 23, 2019

**Revision Table:**

|  |  |
| --- | --- |
| **Date** | **Revision** |
| **10/2/19** | **Non-Functional Requirements have been further detailed.** |
|  |  |
|  |  |

**Joshua Cidoni-Walker** ([jcidoniwalke2017@fau.edu](mailto:jcidoniwalke2017@fau.edu))

**George Porte** (gporte2018@fau.edu)

**Bea Montilla** (bmontilla2015@fau.edu)

**Tashika Williams** (tashikawilli2015@fau.edu)

**Executive Summary**

Campus Snapshot is an online platform that provides users the ability to report, comment on, and view campus incidents. The platform is a modernized approach to a ticket-based system, incorporating a social aspect to the ticket paradigm.

The advent of social-based platforms have greatly improved public safety by increasing public awareness. By incorporating a social aspect to legacy ticket paradigms, we aim to increase campus participants’ awareness of campus incidents. This increased awareness will allow all participants of the campus to interact in a more efficient, transparent way.

**Competitive Analysis**

**Campus Snapshot Legacy Ticket Systems**

|  |  |
| --- | --- |
| Browse (both active and inactive) incidents reported by others | The ability to browse incidents submitted by others is not allowed. |
| Comment on (both active and inactive) incidents reported by others | The ability to comment on incidents submitted by others is not allowed. |

Campus Snapshot competes with legacy ticket systems (such as *Zendesk*) by modernizing the classic ticket-based approach. By using Campus Snapshot (and our paradigm) information is shared with the public, rather than just the organization’s administration--promoting efficiency and transparency.

**Data Definition**

|  |  |
| --- | --- |
| User | Either a visitor (unregistered or registered) or a campus official (administration). |
| Comment | A string of roughly 200 characters. Users can comment on both active, and inactive incidents. |
| Post | A user-created section (card) containing a picture, description, comments, and an active/inactive indicator. |
| Picture | An image that is uploaded to a post (as it relates to a particular incident) |
| Incident Feed | A centralized, scrollable feed which displays all posts made to the website in chronological order, most recent at the top. |

**Project Overview:**

Campus Snapshot is meant to be used by both campus officials and the public. The application will be designed such that a scrollable feed of incidents can be viewed by users. These incidents are marked active by default. After resolving a posted incident, administrators are able to set the incident as inactive. Posts may also have comments associated with them which are made by any User.

**Scenarios and Use Cases**:

|  |  |
| --- | --- |
| Case 1 | Add a Post |
| Actor | University Student |
| Flow | A student walking around campus sees a broken water fountain. They take a picture of the water fountain. They then go to the Campus Snapshot website, and click on the log in link to go to a separate login page. After entering their credentials (correctly) they are redirected to the index page. Here they see a feed and also a prompt to start a new post. They upload the photo via a form and can also add a text description of the incident. Once the student clicks Post, the data is then added to the website’s feed. |

|  |  |
| --- | --- |
| Case 2 | Comment on a Post |
| Actor | Generic User |
| Flow | The user wants to comment on a post on the website’s main feed. If the user is not already signed in, they click on the log in link to go to a separate login page. After correctly entering their credentials, they are redirected to the index page. Once on the index page, the user can scroll to the post they want to comment, and under the post there will be a comment section. This section displays each comment made by Users on the post, in chronological order of posting. At the bottom of the comment section is a text box form where the user types their comment and submits it. The comment is then added to the comment section of the post. |

|  |  |
| --- | --- |
| Case 3 | Mark A Post Inactive |
| Actor | University Employee |
| Flow | Once a posted incident has been addressed, a representative of the school must sign in to an account which has Admin privileges. The Admin can then scroll down to the specific post as required, and click a ‘Mark Inactive’ button which will set the Post’s state to Inactive, expressing that the incident has been resolved. |

**Functional Requirements**

1. The system should display the login and sign up option as well as the posting feed on the homepage.
2. A user should be allowed to post, comment and share entries.
3. Administrators should constantly maintain the platform, i.e marking any spam or inappropriate posts.
4. Administrators should constantly check for any listings that have been resolved and can be set for “Mark Inactive.”
5. The system should not allow a user to go over the 200 word limit.
6. The system should not allow a user to post/comment unless they have signed into their account.
7. The system should not allow two users to have the same username.
8. If a user has inputted their credentials incorrectly, the system should prompt a message asking them to try again. The system should also give them a “Forgot Password” option, where a code would be sent to their email, enabling them to access their account.

**Non-Functional Requirements**

1. Performance
2. Usability
3. Accessibility
4. Expected Load
5. Security Requirements
6. Storage
7. Availability

**System Architecture**

1. Main Software Products:
2. GitHub
3. Google Drive
4. Trello
5. Tools:
6. TBD
7. Languages:
8. HTML
9. CSS
10. JAVASCRIPT
11. List of Core APIs:
12. TBD
13. Supported Browsers:
14. Google Chrome
15. Mozilla Firefox
16. Safari
17. Framework:
18. Laravel

**Team**

**Group name:** G10

**Members (with Roles):**

* Joshua Cidoni-Walker (Product Owner)
* George Porte (Scrum Master)
* Bea Montilla (Developer)
* Tashika Williams (Developer)

**Checklist**

1. Team decided on basic means of communications **(Done)**
2. Team found a time slot to meet outside of class (**On Track**)
3. Front and back end team leads chosen **(Done)**
4. Github master chosen **(Done)**
5. Team ready and able to use the chosen back and front-end frameworks **(Done)**
6. Skills of each team member defined and known to all **(Done)**
7. Team lead ensured that all team members read the final M1 and agree/understand it before submission **(Done)**