**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. |
| 11/4/19 | Data definitions have been refined, added scenarios and use cases, refined high-level functional and non-functional requirements, refined high-level system architecture and added database organization details, added risks and actions. |

**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.

This idea of a public forum for campus activities could be extended beyond safety applications. Having an efficient method of communication across a student population allows for information to spread across campus quickly and increase student organization activity across campus.

**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. With Campus Snapshots, when someone submits an incident, the information is available to the entire user-base.

A similar ticket system is one used by Comcast to track service outages. Comcast customers can document their internet outages, and view a map which highlights areas in which other users are reporting service outages. The result is a service which allows for Comcast to quickly locate and repair service outages in affected areas. Similar systems are used by other internet service providers, and power companies as well to track power outages.

**Data Definition**

|  |  |
| --- | --- |
| Guest | Privileges: Can only browse posts. |
| Registered User | Privileges: Can browse, comment, create new posts  Elements: Username, Password, UID (PK) |
| Moderator | A moderator is a subset of a registered user.  Privileges: Delete *any* post |
| Post | A user-created section (card) containing a picture, description, comments, and an active/inactive indicator.  Elements: UID(Foreign Key), PID (PK), Picture, postText, commentList |
| Incident | A post containing information regarding a school-related event; such as a power outage, hazardous scene, etc. |
| Incident Feed | A centralized, scrollable feed which displays all posts made to the website in chronological order, most recent at the top.  Elements: A List of Posts |
| Comment | A string of roughly 200 characters. Users can comment on both active, and inactive incidents.  Elements: UID (Foreign Key), PID(Foreign Key), CID(PK), commentText |

**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. Posts may have comments and likes associated with them which are made by any user. An option to sort the posts between most liked and most recent is also available.

**Scenarios and Use Cases:**

|  |  |
| --- | --- |
| Case 1 | Add a Post |
| Actor | Registered User |
| 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 | Registered 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 | Registered User |
| Flow | A registered user has the ability to mark his incident active/inactive. To avoid abuse, a post will be marked inactive automatically in 30 days unless it is confirmed with the registered user that the incident is still active. |

|  |  |
| --- | --- |
| Case 4 | Mark A Post as Spam/Inappropriate |
| Actor | Moderator |
| Flow | A moderator logs into the platform. The moderator then navigates to a page titled "Spam Cases"--this page displays a list of reported spam/inappropriate incidents and comments (this will be able to be sorted). The moderator is then able to delete any post/comment he deems is spam/inappropriate. |

|  |  |
| --- | --- |
| Case 5 | Viewing Posts and Comments |
| Actor | Guest |
| Flow | A user (unregistered/registered) navigates to the *Incident Feed* page. The user then is able to browse through incidents based off of the current school they have chosen (which will be an option at the top of the feed). |

|  |  |
| --- | --- |
| Case 6 | Deleting a Comment |
| Actor | Registered User |
| Flow | A registered user navigates to an incident with a comment they have posted. The registered user chooses "Delete Comment" to remove their comment from the post. |

|  |  |
| --- | --- |
| Case 7 | Deleting a Post |
| Actor | Registered user |
| Flow | A registered user navigates to an incident they created. The registered user chooses the "Delete Incident" option from a dropdown menu. |

**Functional Requirements:**

* The system should display the login and sign up option as well as the post feed on the homepage.
* A user should be allowed to post, comment, like (posts and comments), report and share entries.
* Administrators should constantly maintain the platform, i.e marking any spam or inappropriate posts.
* Administrators should constantly check for any listings that have been resolved and can be set for “Mark Inactive.”
* The system should not allow a user to go over the 200 word limit.
* The system should not allow a user to post/comment unless they have signed into their account.
* The system should not allow two users to have the same username.
* If a user has entered invalid credentials, 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.
* A user should have a “Report” option for any posts they see that are considered spam or inappropriate
* On the feed, the system should display the title, date, a picture relating to the article, the number of likes, and the top 3 comments for every post
* The system should give the user an option to sort their feed between “Most Likes” and “Most Recent”
* The system should not allow any generic user to have administrator rights, these are only offered to university employees who are administrators/moderators of the website.
* When a user selects the ‘Most Likes’ option, the feed should be arranged from the most liked to the least liked posts
* When a user selects the ‘Most Recent’ option, the feed should be arranged from the most recent to the oldest dated post.

**Development** **Priority:**

**Must Have**

* The system should display the login and sign up option as well as the post feed on the homepage.
* A user should be allowed to post, comment, like (posts and comments), report and share entries.
* Administrators should constantly maintain the platform, i.e marking/deleting any spam or inappropriate posts and comments.
* The system should not allow a user to go over the 200 word limit.
* The system should not allow a user to post/comment unless they have signed into their account.
* The system should not allow two users to have the same username.
* If a user has entered invalid credentials, 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.
* On the feed, the system should display the title, date, a picture relating to the article, the number of likes, and the top 3 comments for every post
* The system should not allow any generic user to have administrator rights, these are only offered to university employees who are administrators/moderators of the website.

**Desired**

* Administrators should constantly check for any listings that have been resolved and can be set for “Mark Inactive.”
* A user should have a “Report” option for any posts they see that are considered spam or inappropriate

**Opportunistic**

* The system should give the user an option to sort their feed between “Most Likes” and “Most Recent”
* When a user selects the ‘Most Likes’ option, the feed should be arranged from the most liked to the least liked posts
* When a user selects the ‘Most Recent’ option, the feed should be arranged from the most recent to the oldest dated post.

**Non-Functional Requirements**

* + - * Performance – A user should be able to access any page on the website within 15-seconds, and should be able to post an incident (including an image upload) within 25 seconds.
      * Usability – A user should be able to gain an intuitive understanding of how to use the website within a minute of interacting with the site.
      * Accessibility – The website should be accessible to anyone who requests it.
      * Expected Load - The website (platform) should be able to serve at least five-hundred active users concurrently.
      * Security Requirements – The website will be secured from front-end (e.g. scripting attacks) and back-end (e.g. SQL-injection) concerns.
      * Storage – A post will contain no more than five-hundred characters, and a maximum image size of 3Mb.
      * Availability – The website shall be available 24/7.

# High-level system architecture and database organization

**High level Architecture of the code must be consistent with UML class diagram (see below).**

The architecture of our system will include components developed using HTML, Javascript, and CSS (these will all be used for the front-end); PHP and SQL (for the backend).

The backend will be developed using an OOP approach with the following classes:

* + - * + User (this class will handle items such as verifying user credentials, retrieving user information, etc.)
        + Incident (this class will handle items such as creating incidents, deleting incidents, retrieving incident information, etc.)
        + Comment (this class will handle items such as adding, removing, and retrieving comments on incidents.)

**DB organization: Describe the main database schema/organization (high level), e.g. list main DB tables and items in each DB table**

Database Schema (High-level):

* + - * + A *users* table will contain information regarding registered users such as the username, password, and school\_id (school\_id will indicate which school the user attends)
        + A *schools* table will contain information regarding schools such as an id (that will uniquely identify a university) and a school name.
        + A *incidents* table will contain information regarding incidents such as an incident id (that will uniquely identify an incident), incident title, incident image (just the filename), incident description, a timestamp, and whether or not an incident is active or not.
        + A *comments* table will contain information regarding incident comments such as an incident id (which will indicate which incident a comment is made on), a user id (which will indicate which user submitted the comment), the comment, and a timestamp.

**Media storage: Decide if images and video/audio will be kept in file systems or in DB. Describe any other special data format requirements like for video/audio/GPS etc.**

* + - * + Registered users will be able to upload images (as they relate to an incident). The images will be saved to a *user images folder* and their filenames will be saved in the database.

**Search/filter architecture and implementation: what will be the algorithm for search; what DB terms will be searched, how it will be coded and organized in the DB. Similarly, say what DB items will be filtered/sorted**

* + - * + Registered users will have the ability to search incidents (both active and inactive). There will be no custom algorithm used to arrange search. The system will capitalize on MySQL to return search results (i.e. LIKE).
        + Incidents will be sorted on the feed-page by timestamp (most recent appear at the top).
        + Comments will be sorted on an incident by timestamp (most recent appear at the bottom).

**Your own APIs: Describe and define at high level any major APIs that you will create.**

There will be several APIs allowing for the access of data. They are as follows:

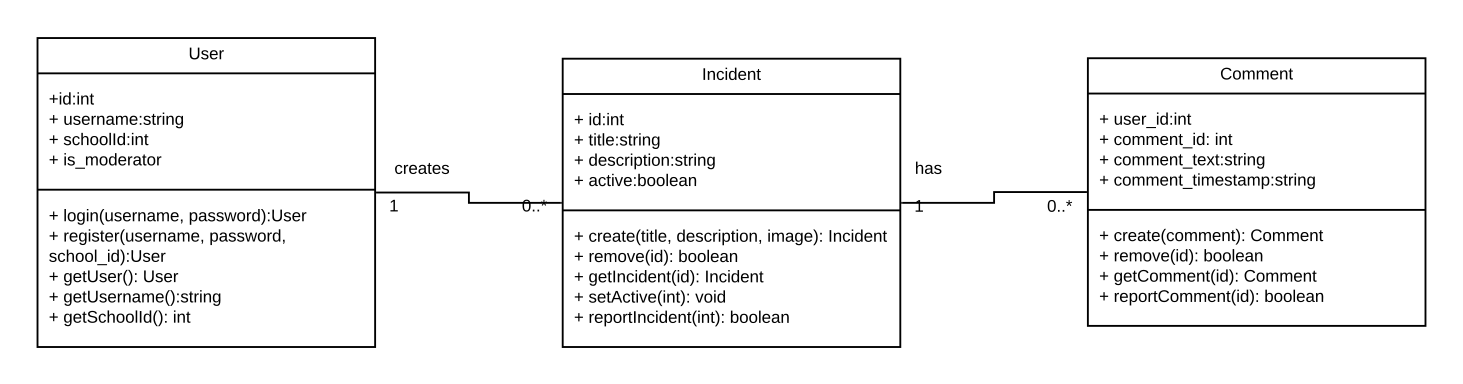
* + - * + An API will be developed that will return a list of incidents (with associated information, such as title, description, image path). These incidents will be able to be filtered through parameters--such as filtering for school, whether they are active or not; sorted by date/time.
        + An API will be developed that will return comments of an incident. These comments will be able to filtered by the incident they correspond to, and date/time.

**Describe any significant non-trivial algorithm or process (like rating, ranking, automatic prioritizing of items etc.)**

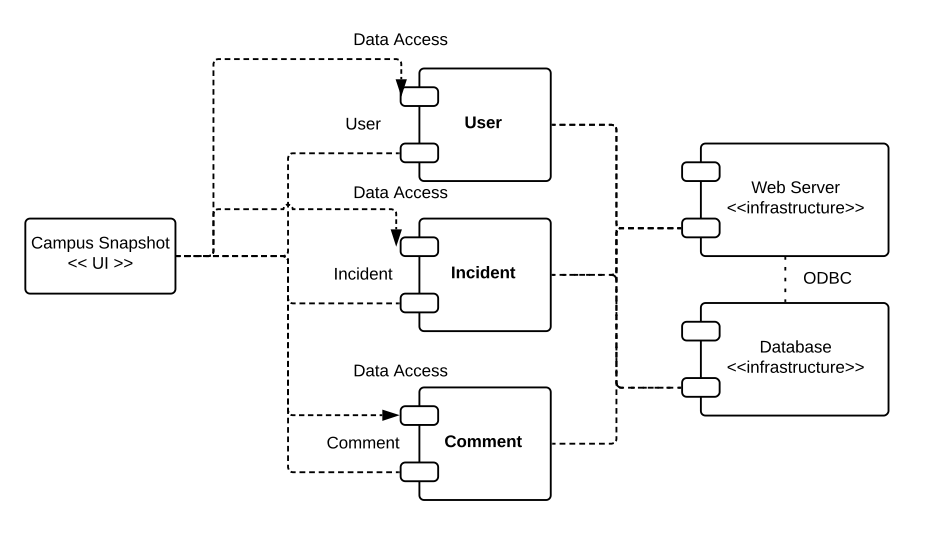
The system will use internal MSQL algorithms to perform various functions such as sorting/filtering.

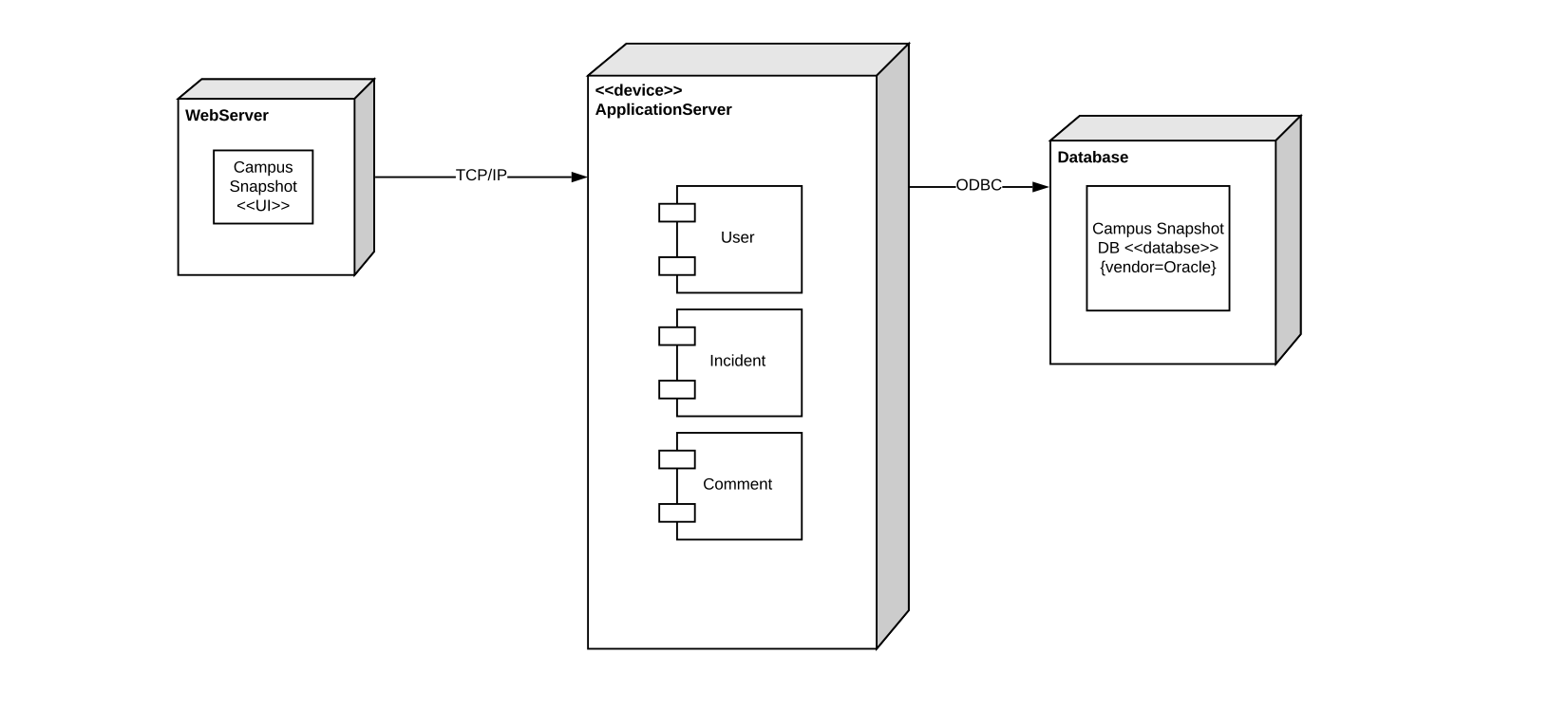
# High-Level UML diagrams

**UML Class Diagram**

****

**UML Component and Deployment Diagrams**





**Key Risks**

The risks posed below will be addressed upon further development of our platform.

**Skill risks:** Each team member has the necessary skills required to complete this project.

**Schedule risks:** There are no schedule risks at this time.

**Technical risks:** The only risk at the present time is ensuring the platform is mobile- friendly.

**Teamwork risks** There are no teamwork related risks at this time.

**Legal/content risks** There are no legal/content risks during the development of this project