**Milestone 1 Project Proposal and High-level description**

**OWL Alerts**

**Panda Development Team (Group 7)**

**Project Members**

Product Owner: Paul Birns

[pbirns2016@fau.edu](mailto:pbirns2016@fau.edu)

Scrum Master: Danica Brozowski

[dbrozows@fau.edu](mailto:dbrozows@fau.edu)

Back End Lead: Akiva Green

[greena2015@fau.edu](mailto:greena2015@fau.edu)

Front End Lead: Aaron Zygala

[azygala2016@fau.edu](mailto:azygala2016@fau.edu)

**Advisor**

Shihong Huang

[shihong@fau.edu](mailto:shihong@fau.edu)

**Documentation Date: 22 September 2019**

**Florida Atlantic University**

**CEN 4010 Principles of Software Engineering, Fall 2019**

**History Table**

|  |  |
| --- | --- |
| **Revision** | **Revision Date** |
| Paul wrote Data Definitions | 9/18/2019 |
| Danica wrote functional/Non Functional Requirements | 9/17/2019 |
| Aaron wrote Executive Summary, Competitive Analysis | 9/20/2019 |
| Akiva wrote Overview of Use Cases, and High-level System Architecture | 9/20/2019 - 9/21/2019 |

**NOTE:** For this milestone, our team decided to work with Google Docs. Instead of utilizing GitHub, our team decided it was best to work through Google Docs to avoid constant merge conflicts or delayed editing. Google Docs is much faster, and allows for simultaneous edits. While working together through a phone call this document allows for cohesive teamwork.

You can view/comment on this document with the following link:

https://docs.google.com/document/d/115sGGiPsbWnDBep9K0yiqyyt4ePj7y5YtbL\_ZWZHboY/edit?usp=sharing

1. Executive Summary

The Florida Atlantic University Alerts System (shortened to OWL Alerts) will give students the ability to alert the campus staff whenever an issue has occurred as well as alert students to events taking place around campus. On campus, students can let the campus know that there is a water leakage at certain locations of the campus, so that the staff can quickly fix the issue. Additionally, commuting students can receive alerts about traffic and campus parking before travelling to their classes. The Alerts system can help make a student’s life simpler, so they do not have to worry about minor issues and can focus more on their classes and studies. To use the system, students will need to create an account, at which time they will be able to view a dashboard about upcoming campus events and issues. They can choose to receive daily or weekly emails to keep them updated about school-wide events, or any issues occurring on campus. When students create an account, they will specify whether they live on campus or not. Students that live on campus will receive notices and messages about their specific buildings, and off-campus students will receive traffic alerts and parking notifications. Students also have the ability to report specific issue posts if they are inappropriate or untrue. The system is simple and easy to use, with buttons and functions easily laid out on a user’s cell-phone or computer.

We have discussed the user’s view of the software. Administrators, such as faculty in charge of cleaning the campus, or members of the IT department, will be able to create accounts as well. An administrator’s account will be able to view the current list of issues, including the date the issue was created and resolved, the description of the issue, and the creator of the issue. They can also see the specific administrator who is designated to fix the issue. Administrator’s also have the ability to remove any inappropriate/untrue issue reports from the backlog. The administrator view of the application is useful for faculty members who need guidance about the location and description of specific issues.

1. Competitive Analysis

The competitors focused on in this competitive analysis are other systems to allow students

and campus faculty to communicate. This includes the current alert system used by FAU and

systems used by other colleges.

|  |  |
| --- | --- |
| **OWL Alerts’ Features** | **Competitors’ Features(FAU current system)** |
| Can receive campus alerts, parking notifications, and school event reminders all in the same place. | Spreads these alerts across multiple different systems. |
| Allows students and campus faculty to communicate through an interactive platform. | Allows students and campus faculty to communicate via email notifications. |
| Allows students to receive email notifications for alerts. | Allows students to receive email notifications for alerts. |
| Gives students a nice user experience, with a dashboard outlining upcoming events, current issues, and parking information. | Sends email alerts and phone calls for different issues. No real user experience. |

The above table displays the competitive relationship between our proposed software and the current systems used by FAU and other schools. Clearly, our system is focused on a great user experience and system that allows for clear communication between the faculty and students at FAU. This can allow students to feel more welcome and calm during a stressful school year. This system also gives students an immediate answer to their issues, as administrators will be issued to aid their problems almost immediately. Additionally, this system is constantly updated with parking information, giving off-campus students help as well. This system aims to serve as an all-encompassing FAU information and campus issue hub, to help and aid students in any way it can.

1. Data Definition

* User (Implements Residency Status)
  + Properties
    - User ID - a unique ID to identify a user
    - User Password - a secure password created by the user to login
    - Phone Number
    - Mailing Address - additional legal/security feature
    - Email Address - the email address for the user to receive notifications
  + Functions:
    - Create an account
    - Post issues/events on campus
    - Receive information (via email or dashboard) about issues/events
  + **User Types** 
    - Student - A User Who Takes Classes on Campus
      * Properties:
        + ZNumber - FAU’s student number
        + Class Status - freshman, sophomore, junior, or senior
        + Residency Type (Commuter or Dormer)
    - Admin - A User Responsible For Handling Issues
      * Properties:
        + A list of issues that admin responsible for handling
        + Backlog - Able to View Backlog of Issues
      * Functions
        + Handle issues occuring around campus
      * Note: Admins **can** be a dormer
    - Faculty - A person who works on campus
    - Note: A separate account will be required for each user type
      * e.g. Person A has both Student and Faculty Account
* Residency Status
  + Commuter
    - Properties
      * Home Address
    - Functions
      * Receive Status Notification About Campus Parking
        + e.g. Lots Closed
  + Dormer
    - Properties
      * Dorm (IRT, UVA, etc)
      * Room Number
    - Functions
      * Receive Status Notifications About their Building
* Dorm
  + Properties
    - Building Name
    - Physical Location
    - Admin Assignee - Admin to Call About Issues
* Issue Backlog
  + Property
    - List of Issue - A list of all issues reported on website
    - Date Last Updated
    - Admin - Who is responsible for maintaining this backlog
* Issue
  + Property:
    - Reference ID
    - Date Created - When was this Issue Created?
    - Date Resolved (“Unresolved” if not)
    - Description
      * What is the issue?
      * How did it happen?
      * When did it happen?
    - Originator - User Who Created this Issue?
    - Admin Assignee - The Admin Responsible for Handling this Issue

1. Overview, scenarios, and use cases

The Florida Atlantic University Alerts System, OWL Alerts, is developed for use by the average FAU student and faculty.

Use Case 0: Log-In:

1. The user accesses the website.
2. The system displays the message:
   1. Please enter your username and password to continue.
3. The user enters their account information
4. The system displays the menu:
   1. Choose “Post an Issue” to communicate an issue with the campus.
   2. Choose “View your Dashboard” to see current events, issues, and notifications.
   3. Choose “Log-Out” to leave your account.

Variation #1: Admin Log-In:

1.1 In Step 3, the admin enters their account information.

1.2 The system displays the menu:

1.2a Choose “View Verified Issues” to look at all faculty verified issues

1.2b Choose “View Unverified Issues” to look at posted issues from students that have yet to be verified by faculty.

1.2c Choose “View Resolved Issues” to look at issues that have been resolved.

1.2d Choose “Log-out” to leave your account.

Use Case 1: Post an Issue

1. The user carries out Use Case: Log-In.
2. The user chooses “Post an Issue.”
3. The system displays an empty textbox, with an option to “Post the Issue.”
4. The user enters their issue, and any images that correspond with the problem.
5. The user posts their issue to the system, and they receive a confirmation message saying “Thank you! Your post has been received.”
6. The system saves the post to the Issue Backlog.

Use Case 2: View the Dashboard

1. The user carries out Use Case: Log-In.
2. The user chooses “View Your Dashboard.”
3. The system displays a menu:
   1. Press “View school wide events” to see events occurring on campus.
   2. Press “View your dorm’s issues” to view recent problems in your dorm building.
   3. Press “View major issues” to see the major problems across campus.
4. The user chooses “View your dorm’s issues.”
5. A neat list is displayed on the screen, showing the recent issue posts at the user’s dorm building.

Variation #1: Commuter Student:

1. Start from Step 3.
2. The system displays a menu:
   1. Press “View school wide events” to see events occurring on campus.
   2. Press “View parking information” to find open parking spaces on campus.
   3. Press “View major issues” to see the major problems across campus.
3. Since the user commutes to school, they can view parking information instead of dorm issues.

Use Case 3 - Faculty Verifying Issue:

1. The faculty member carries out Use Case: Log-In (Variation 1: Admin Login)
2. The faculty member chooses the “View Unverified Issues” option
3. The faculty member clicks on the issue that has been verified by themselves and hits the verify button.
4. The faculty member can click the edit button to edit the post for when it is viewed on the “Verified Issues” page.

Use Case 4 - Faculty Resolve an Issue:

1. The faculty member carries out Use Case: Log-In (Variation 1: Admin Login)
2. The faculty member chooses the “View Verified Issues” option
3. The faculty member clicks on the issue that has been resolved.
4. The faculty member clicks the “Resolved” button to mark the issue as resolved.
   1. The issue can now be viewed on the “Resolved Issues” page.
   2. The faculty member can choose to notify individuals that are affected (users in the affected dorm, affects a large body of users, and etc.)

An FAU faculty member who is a part of the campus IT team has been briefed on a new web application that his department will be using to track student reported technology issues. He creates his account via the verified faculty sign-up page. After account creation he logs in and sees the list of issues (unverified), verified issues, and resolved issues. Starting with unverified issues they will look through them and based upon their judgement they will verify the issue physically. Once they have verified the issue they will mark the issue as verified and delete all the duplicates. Upon verification the faculty member will move to have the issue resolved. Once the issue has been resolved it will be marked as resolved.

1. Initial list of high-level functional requirements
   1. Usable from major browsers such as Chrome, Firefox, and Safari
   2. Read and write information about events and problems with text and photos
      1. Example: 9/16/19 water tank explosion 3rd floor IRT
      2. Software must be able to access files created by external tools and programs.
   3. Edit status of existing posts
      1. Example : 9/16/19 water tank explosion 3rd floor IRT(resolved 9/17/19)
   4. Verify events and problems and status
      1. Example : 9/16/19 water tank explosion 3rd floor IRT(resolved 9/17/19) \*verified by (# of users)
   5. Ability to add a report to the incident
      1. Software must be able to access files created by external tools and programs.
   6. Archive of past reports by section/type
      1. Events archive
      2. Maintenance Archive
      3. Miscellaneous Archive
   7. Shareability on user social media
      1. Software should be able to share content to an external social media platform
   8. Users CANNOT direct message each other
      1. Communication between users is strictly limited to comments on user posts.
   9. Users will not have access to other user data
      1. No profile page specified for individual users.
2. List of non-functional requirements
3. Product Requirements:
   1. Available 24/7 with database updates in real time.
   2. Required photo with every post for validation of event/issue.
   3. Backlog up to a years worth of data.
   4. Post reporting capability to report unsafe and inaccurate content to be removed.
   5. Informative error messages.
   6. Fast refresh time.
4. Eternal Requirements:
   1. Secure login/registration interface
   2. Required agreement to privacy policy and user conduct agreement.
   3. Privacy agreement at registration and link available at all times on app.
   4. User agreement and code of conduct available upon registration and linked on site to be

accessible at all times

* 1. Unauthorized use is not allowed

1. Organizational Requirements:
   1. Secure data collection
   2. Simple user interface developed for users
      1. User can learn the in’s and out’s of the app in under 30 minutes
   3. Delivered by Final project due date
2. High-level system architecture

Language(s) Used: HTML, CSS, Javascript, Typescript

Framework Used: NodeJS, jQuery

Tool(s) Used: Visual Studio Code, Brackets, Apache

Supported Browsers: Chrome, FireFox, and Safari

API Used: Google Maps, Google Oauth

1. Team

**Panda Development Team**

Danica Brozowski (Scrum Master)

Paul Birns (Product Owner)

Aaron Zygala (Front End Lead)

Akiva Green (Back End Lead)

1. Checklist

a) Team decided on basic means of communications

DONE

b) Team found a time slot to meet outside of the class

DONE

c) Front and back end team leads chosen

DONE

d) Github master chosen

DONE

e) Team ready and able to use the chosen back and front-end frameworks

DONE

f) Skills of each team member defined and known to all

DONE

g) Team lead ensured that all team members read the final M1 and agree/understand it

before submission

DONE