**CEN4010 Principles of Software Engineering Summer 2018**

Milestone 3: Detailed Requirements, Architecture, and Vertical Software Prototype

DAFJ Ninjas, Owl-iView

Team 1

Daniel Gross

Dgross13@fau.edu

Austin Newland

Anewland2015@fau.edu

Filipe Catarcione

Fcatarcione2012@fau.edu

Jonathan Ortiz Collazo

Jortizcollaz2016@fau.edu

[Owl-iView CLICK HERE](http://owl-iview.azurewebsites.net/)

http://owl-iview.azurewebsites.net/

Login Information:

guest@mail.com

Guest1234

**Documentation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Description** | **Date** |
| 1.0 | Daniel Gross | Document Creation (M1) | 6-18-2018 |
| 2.0 | Daniel Gross | Document Update (M3) | 7-23-2018 |

Table of Contents

[1 Executive Summary 3](#_Toc520143556)

[2 Competitive Analysis 5](#_Toc520143557)

[3 Data Definition 5](#_Toc520143558)

[4 Overview, Scenarios and Use Cases 6](#_Toc520143559)

[5 High Level Functional Requirements 7](#_Toc520143560)

[6 Non Functional Requirements 8](#_Toc520143561)

[7 High Level System Architecture 8](#_Toc520143562)

[8 High-Level UML diagrams 9](#_Toc520143563)

[8.1 Team and Check List 10](#_Toc520143564)

[9 Identification of key risks 11](#_Toc520143565)

# Executive Summary

Florida Atlantic University (FAU) requires a web application to be assembled with the goal of delivering a real time snapshot of FAU’s University grounds and happenings. Specifically, we will permit society to observe an instance of the campus in real time, which provides information about events happening around campus and details the reported issues that need attention on the campus grounds.

The application is a private to members only, where members can view upcoming extracurricular activities/ campus events and each members reported issues. Campus administration will have a snapshot containing active issues reported by all application members. The advantage of the application is that a member who sees an issue occur in real-time has the ability to notify the appropriate maintenance personnel to prevent the issue from escalating and a disaster from happening. This application is valuable because if maintenance receives an early notification of an issue it can be repaired, which saves the entity time and money from performing a major repairs. This application will fit all market sectors because staff, the first responder, can report any issues or events that arise, in turn enabling a great real-time notification system providing the maintenance and members a snapshot of campus grounds and events.

DAFJ Ninjas consists of full-stack developers, a scrum master, and a product owner. The application named, “Owl-iView”, will be a custom built and designed developed in the C# language and will use asp.net core for its framework. The needs for Owl-iView are:

* Members to report issues through the app via a form uploading necessary information for administration to diagnose the situation.
* FAU’s maintenance to check the application for any reported issues.
* Secure login page for a secure member account, including an optional two factor authentication (Google/Microsoft authenticator app required) and a registration page for new members.
* Quick data storage into the application’s database.
* Ease of use, intuitive, attractive, and media rich system. View status of issues (reported, in progress, and completed), app required to be in English, compatible on major browsers, searchable by major search engines.
* View and comment on existing posts and make a report thread.
* FAU’s maintenance team to update the status of a reported issue so members will be notified of status (Reported, In Progress, Resolved).

System constraints:

* System is usability is limited to the English language, while performance and speed depends on the type of browser used.
* System storage is limited to the allocated space on the server.
* System usability is limited reports cannot be exported.
* System security requires just a username and password.
* System accessibility is limited to account holders only.

Top 3 requirements or design objectives (in order of importance) is:

* Deliver product to customer specification of creating an easy and attractive web system for members to report an issue through a form.
* Members can have a snapshot of all events and issues on campus.
* Secure login page.

The strategy initially is to build an attractive web application with the imposed constraints. DAFJ ninjas will design and develop the application to satisfy all of the needs and objectives. The design will be evaluated by the customer after providing releases of the site in order to gain feedback. Members will use the applications upload and reporting feature from any device because it is mobile friendly. Key features are that our framework creates a dynamic web page since asp and html work well together, better application security with asp’s built in windows authentication, faster because asp is a server side technology where the code is executed before sending to browser. Another key feature is that Asp.Net Core is cross-platform so it can run on Windows, Linux, and macOS. A novelty of our project is the reporting feature because it shows the good and the bad issues and events, respectively, on campus grounds. The values of the project are that it will improve the quality of life for students with the ability to communicate issues fast on campus.

# Competitive Analysis

|  |  |
| --- | --- |
| Owl-iView | CitySourced App |
| Real-time snapshot of issues and events of university grounds. | Snapshot with events and issues. |
| Commenting on posts, like and threads | No comment on post. |
| Secure login and two factor authentication. | Username and password login. |
| Cross Platform (Linux, Windows, Apple) and mobile friendly. | Mobile friendly. |
| Attractive and dynamic. | Attractive. |

To summarize, our planned advantages compared to what we have will give us an edge over competitors because our site will have better performance, ease of use, and very attractive. The system will prove useful because of the real time snapshot and event features.

# Data Definition

Database=System for storing all application data.

SQL = Structured Query Language to request specific data from the database.

Administrators= University Campus grounds maintenance and management who will be alerted of issues to fix.

Members= Application users who sign up for account in application to post events or issues.

UML= Unified Modeling Language.

Language-Integrated Query (LINQ) = general purpose standard query operators that allow traversal, filter, and projection operations.

API=Application Programmable Interface.

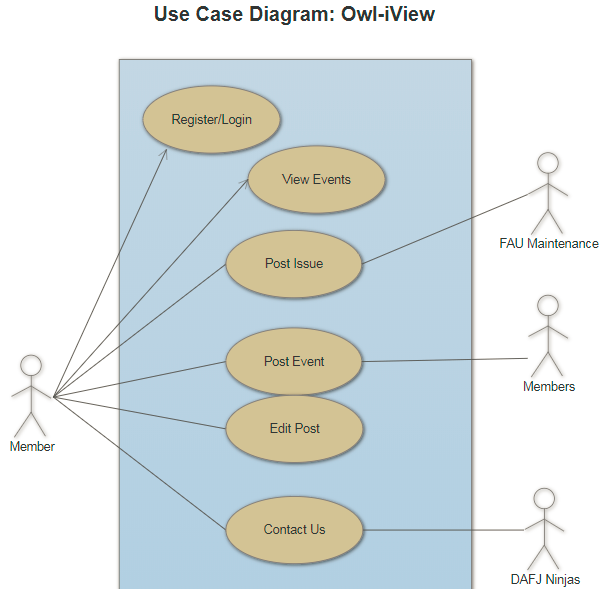
Entity Framework (EF) Core = Object-relational mapper that enables us to work with relational data using domain-specific objects.

# Overview, Scenarios and Use Cases

FAU may use this application for uploading issues for the campus administrators to address and repair as needed. The web application is beneficial because it is fast and provides a direct report to the correct person to be identified of the report.

Members using the application can easily make a report to system through a form online to report a good or potentially broken service that requires fixing.

Users will be uploading a report of an issue they find fit to be directed to maintenance for fixing. Members and administrators on campus may receive a report for campus ground happenings and events.



# High Level Functional Requirements

High level functionalities are:

* 1. Create an Events page to display posted Events.
  2. Create an Issues page to display posted Issues.
  3. Create a New Posts page for members to make a post.
  4. Create login page for members to login.
  5. Setup database to store application information.
  6. Host application on Azure.
  7. Create registration page for new member accounts.
  8. Ability to comment on a post.
  9. Ability to upload an image and assign a severity to a post.
  10. Make application mobile friendly.
  11. Setup member account page for user profile settings (profile picture, email, phone number, change password, setup multifactor authentication).
  12. Create forgot password link for members who forgot credentials.
  13. Setup remember this computer option on login page.
  14. Add feature of pop-up after a click.
  15. Allow admins/managers to modify the status of an issue.
  16. Allow admins/managers to view/manage a table of all registered users.
  17. Allow members to specify a campus on the post.

# Non Functional Requirements

System performance and speed will be an advantage with asp.net core. Storage will be on an Azure server. System usability is limited reports cannot be exported. The web system protects user accounts by a password. System accessibility is great with cross platform capabilities. By using the Azure services, we can expect a heavy load of data easily.

# High Level System Architecture

Our main software product includes visual studios 2017 for development and Azure to host our web system. Also, asp.net core and C# for our framework provide fast user interface and fast storage of data in the database located in Azure. The framework is cross platform and supports all major browsers (Chrome, Mozilla, Safari). For any external code we plan to use, its source and license will be listed.

**Database** **name**: Snapshots\_Dev

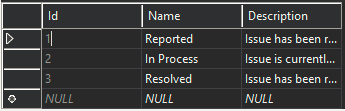
**Table names**: Posts, Status, Comments, Members, Campus

Posts Table



* Primary Key: Id
* Foreign Key: CampusId
  + Has a relationship with the Campus table
* Foreign Key: StatusId
* Has a relationship with the Status table

Status



* Primary Key: Id

Comments



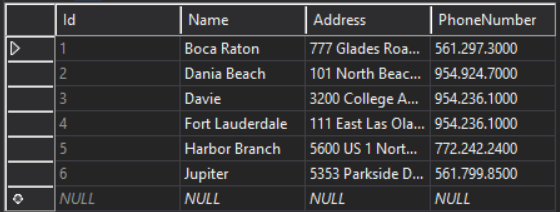
* Primary Key: Id
* Foreign Key: PostId
* Has a relationship with the Posts table

Members



* Primary Key: Id

Campus



* Primary Key: Id

When a member makes a new post and uploads a photo, the photo is going to be stored in a local folder to the application and the path to the photo is stored in the DB (url field in the Posts table).

We are using Entity Framework with a Code-First approach to work with the database. Entity Framework is an object-relational mapper that enables us to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that you would need to write. We focus on the domain of our application and start creating classes our domain entity. The Entity Framework API will create the database based on our domain classes.

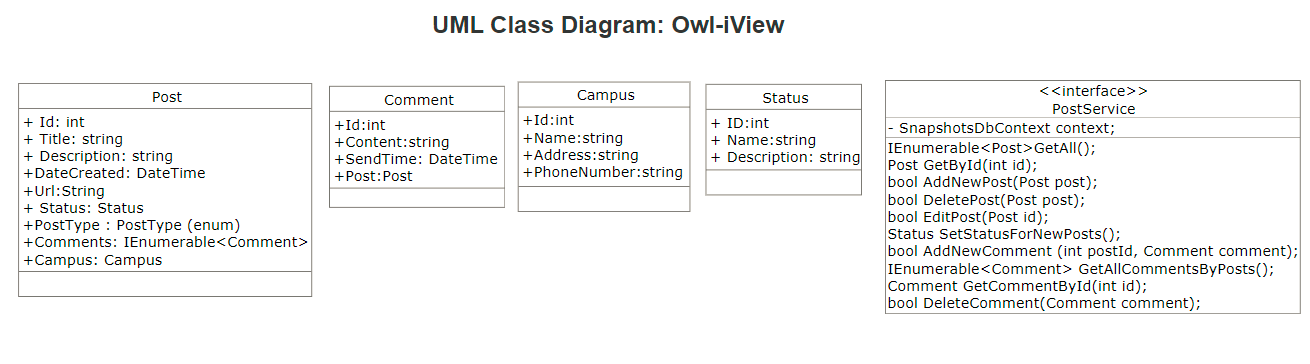
We have a created a Library with a “PostService” class. All searches implemented within the PostService class are accessing the ID field of the tables. The SnapshotsDbContext class represents our database. It contains DbSet objects that represent the tables in our database. We can use the DbSet object for create, read, update, and delete operations. So to get all posts in the database for example, we can simply use the following code: return context.Posts;

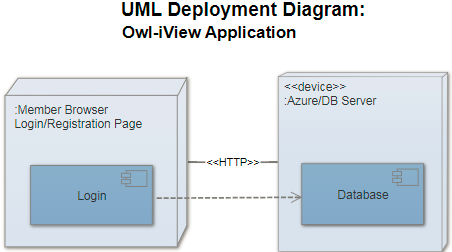
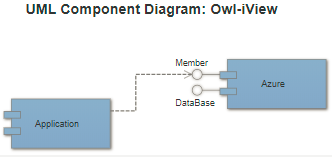
To Access the database, we use .NET’s Language-Integrated Query (LINQ) which defines a set of general purpose standard query operators that allow traversal, filter, and projection operations to be expressed in a direct yet declarative way.

This application is going to be built using an API called Entity Framework (See definition section).

The application’s algorithm may sort posts by Id or status for quick retrieval of data.

# High-Level UML diagrams





# 8.1 Team and Check List

Group 1 (DAFJ Ninjas) consists of Daniel Gross (Product Owner and front end developer), Austin Newman (Front/Back end developer and Scrum Master), Filipe Catarcione (lead developer), and Jonathan Ortiz Collazo (developer).

a) Team decided on basic means of communications DONE

b) Team found time to meet outside of 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 frameworks

DONE

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

g) Team lead ensured team read M3 and agree for its submission DONE

# Identification of key risks

1) Skills risks = DAFJ ninjas are skilled to perform dedicated tasks specified above to fulfill application requirements within the assigned developers and given time frame.

2) Schedule risks= All high level functional requirements will be completed in time frame.

3) Technical risks= DAFJ ninjas are technically inclined to complete assigned task with chosen framework.

4) Teamwork risks= Team is working together and able to communicate about tasks through online resources.

5) Legal/content risks= Development of the application will be completed without any concerns of legal, license, or copyright issues.

The team plans to resolve risks through communicating the problem and researching a solution online through reputable and legal sources. Also, by managing tasks appropriately with Trello and assigning team members the tasks needed to meet deadlines, which is the key is to resolving any risk encountered in the development process. Using Trello provides risk management and control by assigning a level of severity to a task, prioritizing that task.