

Club utils: Final report

Craigory Coppola • Joshua Webb • Demitrios White



November 30, 2018

CS 380

Term Project

NOTE:

All Source Code Revisions Available on GitHub (http://www.github.com/AgentEnder /ClubUtils)

Contents

[Project Description: 1](#_Toc531378093)

[Software Requirements Specification: 2](#_Toc531378094)

[Introduction 2](#_Toc531378095)

[1.1 Purpose of the Product 2](#_Toc531378096)

[1.2 Scope of the Product 2](#_Toc531378097)

[1.3 Acronyms, Abbreviations, Definitions 2](#_Toc531378098)

[1.4 Overview 3](#_Toc531378099)

[2.0 Overall Description 4](#_Toc531378100)

[2.1 Product Perspective 4](#_Toc531378101)

[3.0 Specific Requirements 4](#_Toc531378102)

[3.1 External Interface Requirements 4](#_Toc531378103)

[Use Case Diagram: 6](#_Toc531378104)

[Software Development and Implementation Plan: 7](#_Toc531378105)

[SDLC Choice and Explanation: 8](#_Toc531378106)

[UML Diagrams: 9](#_Toc531378107)

[Statechart: 9](#_Toc531378108)

[Class Diagram: 10](#_Toc531378109)

[Sequence Diagram: 11](#_Toc531378110)

[Testing Plan: 12](#_Toc531378111)

[Architecture View: 14](#_Toc531378112)

[Execution View: 14](#_Toc531378113)

[Dependency View: 15](#_Toc531378114)

[Testing Results: 16](#_Toc531378115)

# Project Description:

1. Team Members
   1. Craigory Coppola
   2. Joshua Webb
   3. Dimitrios White
2. Project Title

ClubUtils

1. Project Description

Since two of our group members are leadership in student organizations, we decided to write a collection of utilities that would make club management run more smoothly. ClubUtils is meant to be a set of tools that can be used by any organization, not just catering to the needs of the clubs we are involved in. As such, each utility needs to be generic and easy to use by those who do not know how the software works. The current list of utilities is as follows:

* Membership tracker: Track active club members, and their role in the club. This should allow leadership to view member info, such as emails for each member.
* Attendance tracker: Track attendance for each member, as well as statistics about meeting attendance (Average meeting attendance, attendance trends).
* Mass Email: Send emails out to each member, or filter by club role.

# Software Requirements Specification:

## Introduction

ClubUtils is a collection of utilities that would make club management run more smoothly. ClubUtils is meant to be a set of tools that can be used by any organization, not just catering to the needs of the clubs we are involved in. As such, each utility needs to be generic and easy to use by those who do not know how the software works.

### 1.1 Purpose of the Product

This requirements document serves several purposes. First, it serves to establish the needs of various clubs that the ClubUtils application will fill. Second, the document provides descriptions of the various functions (and the relationships among them) that make up the application. Any constraints on the software design are included. The goal is to provide to the software developers the information necessary to design and implement the system. Finally, the document provides a baseline for verification and validation activities and for later enhancements.

### 1.2 Scope of the Product

The objective of the ClubUtils program is to provide useful tools to be applied by club officers in managing and tracking their duties and members. The products of this project includes a membership tracker for use in keeping track of all members of a club, an attendance tracker for use in tracking the attendance of all club members, and a mass email function.

Further scope includes the ability for users to sign up for events, as well as the ability for the President or Cabinet to set up a tournament.

The system is to provide the President and Cabinet with a menu of choices that include the option to add and remove users, see current users, see users attendance record, and send mass email to users.

### 1.3 Acronyms, Abbreviations, Definitions

Advisor:

The Faculty Advisor/Sponsor of the club utilizing ClubUtils

President:

The president of the club. This user should have extra powers including moderation of content, full email access, and the power to remove other cabinet members or shift the presidential role.

Cabinet:

The leadership for the club utilizing ClubUtils. This includes the president and advisor, as well as any other leadership figures (Secretary, treasurer, etc.).

Default:

A general user of the software. This could be leadership that has not signed in, or any other user which does not sign in. General club membership would also fall under this role.

Username:

An identification used by a User to access the program.

### 1.4 Overview

The remainder of the document is organized as follows. Section 2 describes general factors that affect the ClubUtils software requirements. Section 3 lists specific software requirements for the ClubUtils application.

## 2.0 Overall Description

### 2.1 Product Perspective

ClubUtils is a self-contained program meant to work independently of any other program.

## 3.0 Specific Requirements

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces

Admin/President

The Admin/President Role should be able to:

•Transfer club roles to users (president, cabinet)

Cabinet

Cabinet members should be able to:

•View club information

•Update attendance logs

•Can create events

Default

The Default Role should be able to:

•The user may create an account with the program using their Morehead state email as a username.

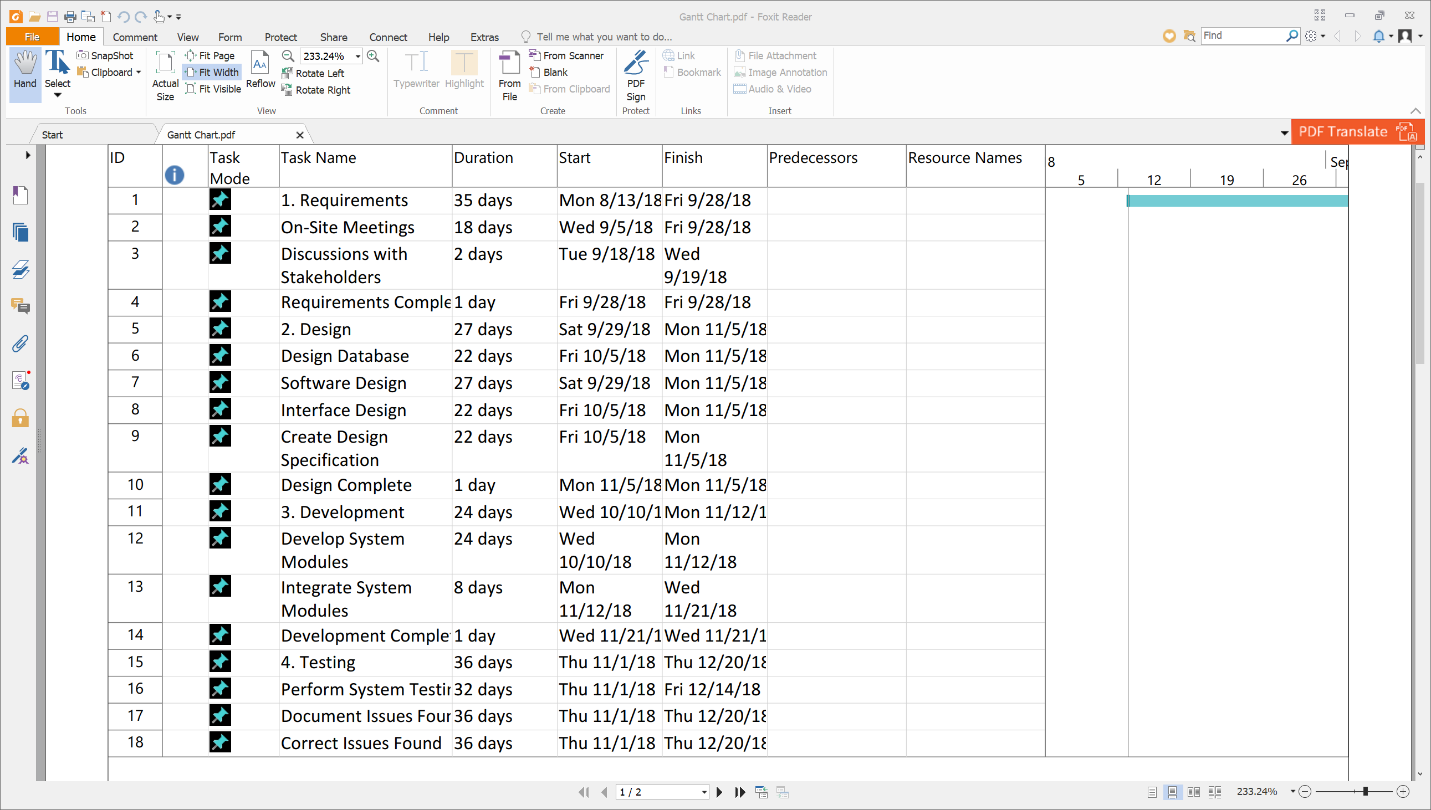
•View events and prior announcements

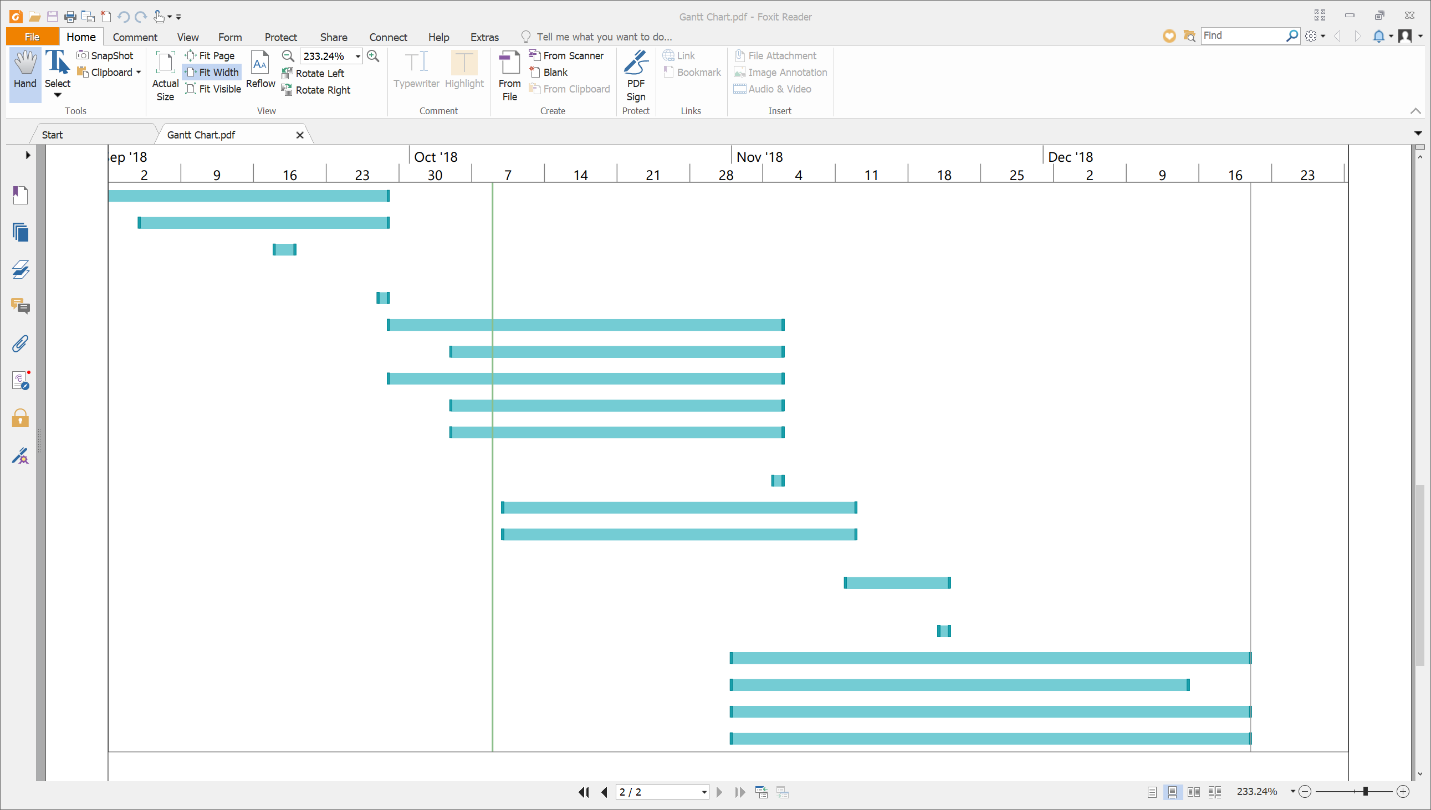
•Sign up for events

# Use Case Diagram:



# Software Development and Implementation Plan:





# SDLC Choice and Explanation:

We chose the Agile methodology for our development due to the changing needs of our project.

There is uncertainty in the requirements, which comes from the fact that different functionality has differing priority. We need to communicate regularly with our customers. This is necessary for keeping on track with the requirements. Due to the nature of our customers (MSU Game Club specifically, along with the pre-vet club), there are certain requirements which could change. For example, we considered adding some club-specific functionality such as a tournament manager for game club which would not be as general purpose. This would be a flexible requirement, and we needed to be able to accommodate this flexibility. Agile allows the requirements to flex throughout the project and was thus chosen.

We need to be able to rapidly iterate. Being able to take our initial iterations and run them by the clients will allow us to stay within the bounds of the actual requirements and ensure we deliver the product they need, rather than the one we built.

Time is a very important factor. Because we need to be able to finish before the end of the semester, we need to keep track of when things are due to start and finish. Agile is effective at implementing a schedule and keeping on top of it, which fits our needs perfectly.

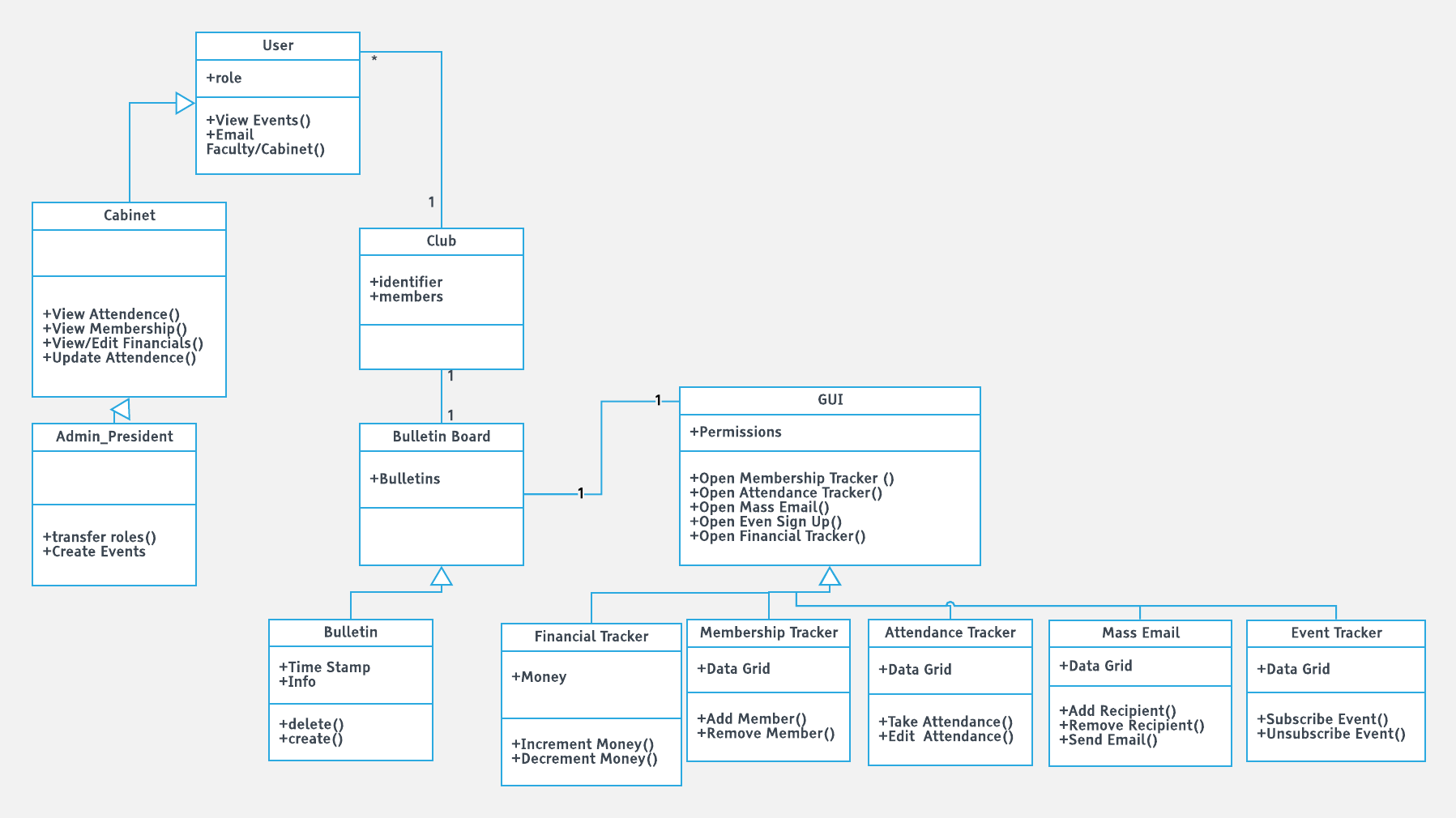
Due to the complexity of the project and scheduling conflicts, we need a SDLC that helps to manage communication between project participants as well as with our users. Agile helps keep the conversation open and is flexible enough to keep up with our changing schedules.

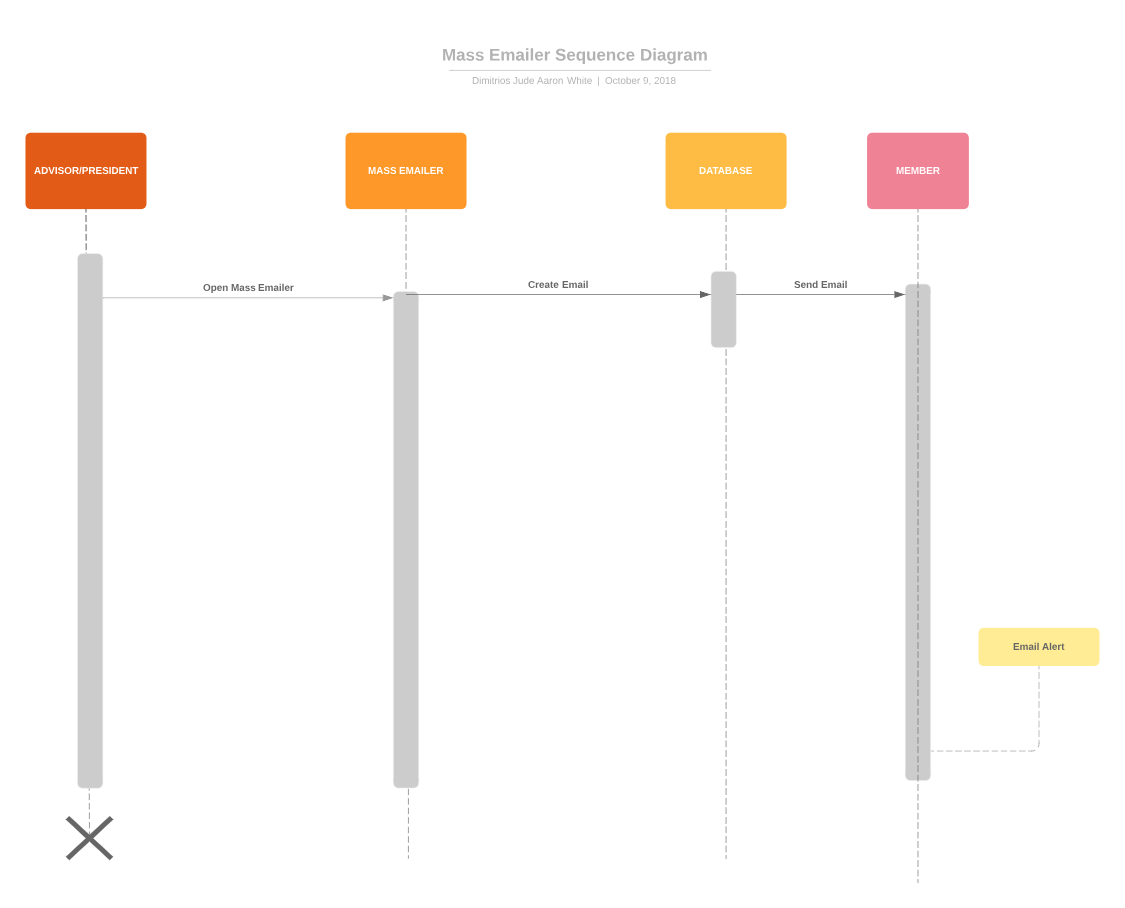
# UML Diagrams:

## Statechart:



## Class Diagram:



Sequence Diagram:

# Testing Plan:

The testing for ClubUtils will be white box. The reason for this is the code base is fairly small and easily covered by three people. Further, white box testing allows for a very thorough examination of the code. Time is also a factor, and white box testing allows for more easy analysis of the code once a bug is found.

ClubUtils uses a Top Down approach. The units will be integrated early, before they are complete, requiring that we regressively test the program as each unit is added to the main interface.

The testing objectives for each module are as follows:

**Attendance Tracker**: correctly access attendance information from database; correctly update attendance information; correctly update database with attendance information

**Calendar**: correctly display calendar; correctly access events from database; correctly display days with events; correctly allow events to be added; correctly update database with events

**Bulletin**: correctly access announcements from database; correctly allow announcements to be created; correctly allow database to be updated with announcements; correctly display announcements

**Financial Tracker**: needs to correctly draw financial information from database; needs to correctly compute new financial information; needs to correctly update database with new financial information

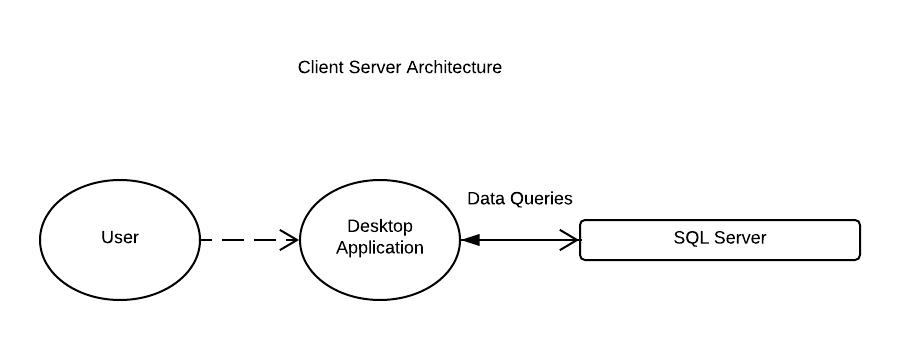
**Login Menu**: correctly allow users to be created; correctly add users to database; correctly access users from database; correctly allow access to the club’s main menu once signed on

**Main Menu**: correctly show the club’s bulletin and calendar; correctly grant access to features depending on users’ access level; correctly allow users to sign ou

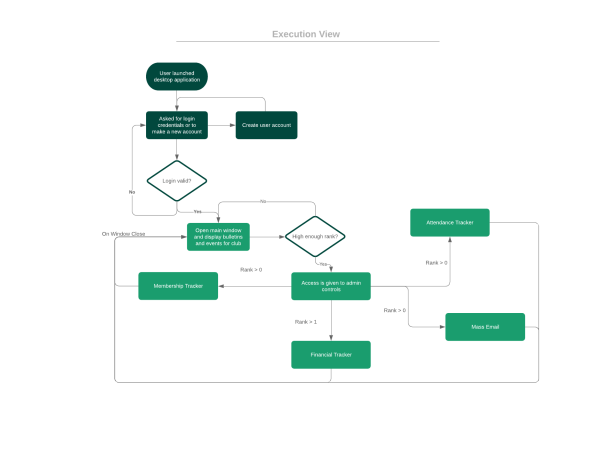
**Mass Emailer:** Needs to correctly access club users’ emails from database; needs to correctly read user’s message; needs to correctly send message to all club users’ emails.

**Membership Tracker**: Correctly access member information from database; correctly display member information; correctly remove members access from the club.

# Architecture View:



# Execution View:



# Dependency View:



# Testing Results:

Login Screen:

•Tested security class to ensure sha-256 operating properly.

•Generated hashes match externally generated hashes, pass.

•Tested salting passwords by running account creation for multiple accounts with same password.

•Hashes different, pass.

•Test Exit button by clicking it,

•Program closes, pass.

•Testing multiple accounts with same email and same club

•Creates multiple logins, each their own user. FAIL

•Disallow multiple logins per email per club, Pass

•Attempt to create account with non-.edu email

•Account created, FAIL.

•Check last 3 characters in email against EDU, prevents .coms, .orgs etc. PASS

Main Screen/Dashboard:

Calendar:

•Tested events on current day

•Display correctly, pass.

•Tested events in far past

•Display correctly, pass.

•Tested events in far future

•Display correctly, pass.

•Tested recurring events, with no end date

•Program bogged down due to adding event until max DateTime instance

•Cap recurring events to only display for next year.

•Tested recurring events with end date prior to beginning date.

•Doesn't display at all, valid behavior, pass.

Bulletins:

•Tested retrieval of RTF from db.

•Parses properly and loads bulletins, pass.

•Tested retrieval of multiple bulletins

•Sorted backwards, fail.

•Fix sort, pass.

•Tested no bulletins

•Nothing displays, pass.

Admin Toolbar:

•Tested logging in with various roles

•Menu options only show when they should, pass.

•Tested clicking on each option

•Applicable windows open, pass.

Admin Icons (+Bulletin, +Event):

•Tested by clicking on each icon

•Applicable window opens.

•Tested while logged in with various roles

•Always visible, FAIL.

•Fix visibility, collapsed when not available. Pass.

New Bulletin Screen:

•Tested with plaintext first

•Bulletins are saved properly, pass.

•Switched to Rich Text formatting

•Serialization of rtf into db string fails, FAIL.

•Fixed serialization, pass.

•Tested against SQL Injection

•New bulletin saved, but empty. Reveals SQL Injection weakness, FAIL.

•Future Work.

New Event Screen:

•Tested with multiple dats (far past, present, far future):

•All save properly, pass.

•Test Event name field against SQL Injection

•Vulnerable, Fail.

•Future Work

Membership Tracker:

•Tested if only president and advisor could edit member info

•Only president and advisor could edit member info, pass.

•Tested if all cabinet members could view member info

•All cabinet members could view member info, pass.

•Tested if edits made by the president and advisor saved to database correctly

•Edits saved properly to database, pass.

•Tested SQL Injection on all data fields in member data

•Saved the sql statements in the database as strings and did not run them, pass.

•Tested if changing a user's email in the member data would change their login credentials

•The user's email properly updated and changed their login credentials to the new email, pass.

Attendance Tracker:

•Tested if all cabinet members could take attendance for a selected day from the date picker

•All cabinet members could properly take attendance, pass.

•Tested if check boxs worked properly for taking attendance

•Check boxs would not appear in the datagrid, FAIL.

•Left field as an integer until further work could be done.

•Tested if the save button would only show up for the dates that have not already taken attendance

•Save buttons was always showing up, FAIL.

•Forced button to always be invisible unless a new date was selected

•Tested SQL Injection on all data fields in attendance data

•Saved the sql statements in the database as strings and did not run them, pass.

•Tested if the data field "Present" would accept non integer values

•The data field would not allow non integer values to be inputed, pass.