Course ID: CPS 5995-01

The Daily Roar

Software Design Document (SDD)
Version 1.1

Prepared by: Uko Ebreso, Alex Fisher, Kevin Parra

Instructor: Dr. Jing-Chiou Liou

Date: February , 2023

Table of Contents

1.0 Version History

Version	Revised By	Revised Date	Changes
1.0	Team	2/28/23	Baseline

2.0 Introduction

2.1 Purpose

The Software Design Document is a written report of the Daily Roars system design. This report goes into detail about the software design of the Daily Roar system after a concatenation of meetings with the client and group members.

2.2 Goals and Objectives

The Daily Roar System was created in order to consolidate emails, and minimize them to topics related to career development. Consolidation will increase email viewing rate and email interactions. Once implemented, the application will able to accomplish these objectives:

- Increase the viewing rate of emails by 66%
- Increase job posting link click-through rate by 100%.

This SDD focuses on the system design. The following are the details provided in this document

- Database Schema Diagram
- Entity Relationship Diagram
- Use - Case Diagram
- System Architecture

2.3 Statement of Scope

A web application will be developed for the CS/IT department that allows faculty to broadcast to students about career development opportunities. The system will allow faculty to reach out to targeted groups of students, based on specific criteria such as class level, major, and degree program. The system will record statistics of engagement and interaction of students with the emails through tracking analytics. If a student begins the semester after the initial upload of students to the database, the system will allow faculty to add additional students. Previous emails will be archived with a backlog to allow for later review.

2.4 Acronyms and Definitions

- CS - Computer Science
- IT - Information Technology

- SDD - Software Design Document
- ERD - Entity Relationship Diagram
- HTML - Hypertext Markup Language
- CSS - Cascading Style Sheets
- PHP - Hypertext Preprocessor
- jQuery - Javascript library that makes HTML traversal and manipulation simpler
- JS - Javascript, scripting language to dynamically update web browser content
- GA - Graduate Assistant

3.0 System Design

3.1 System Architecture

As a web application, this system follows a client-server model with three-tier architecture. The three layers within the architecture are the Presentation layer (Client), the Application layer (Server) and the Data layer (Database).

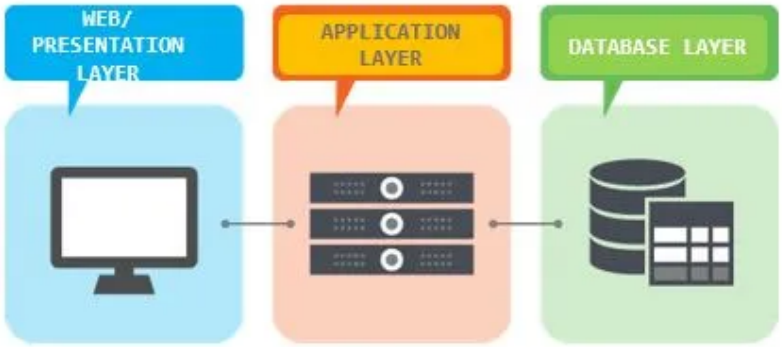


Figure 1: Three Tier Architecture

3.2 Architecture of Daily Roar Application

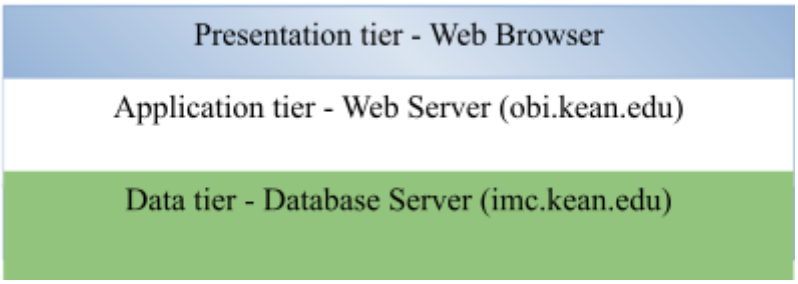


Figure 2: Architecture of Daily Roar Application

3.3 Technical Details Behind the Layers

1. Presentation layer (web pages)
    a. Built using: HTML, CSS, Bootstrap, JavaScript, jQuery
2. Application layer
    a. Built using: PHP, Apache Web Server (hosted on obi.kean.edu)
3. Data layer
    a. Built using: MySQL (hosted on imc.kean.edu)

3.4 Context Diagram

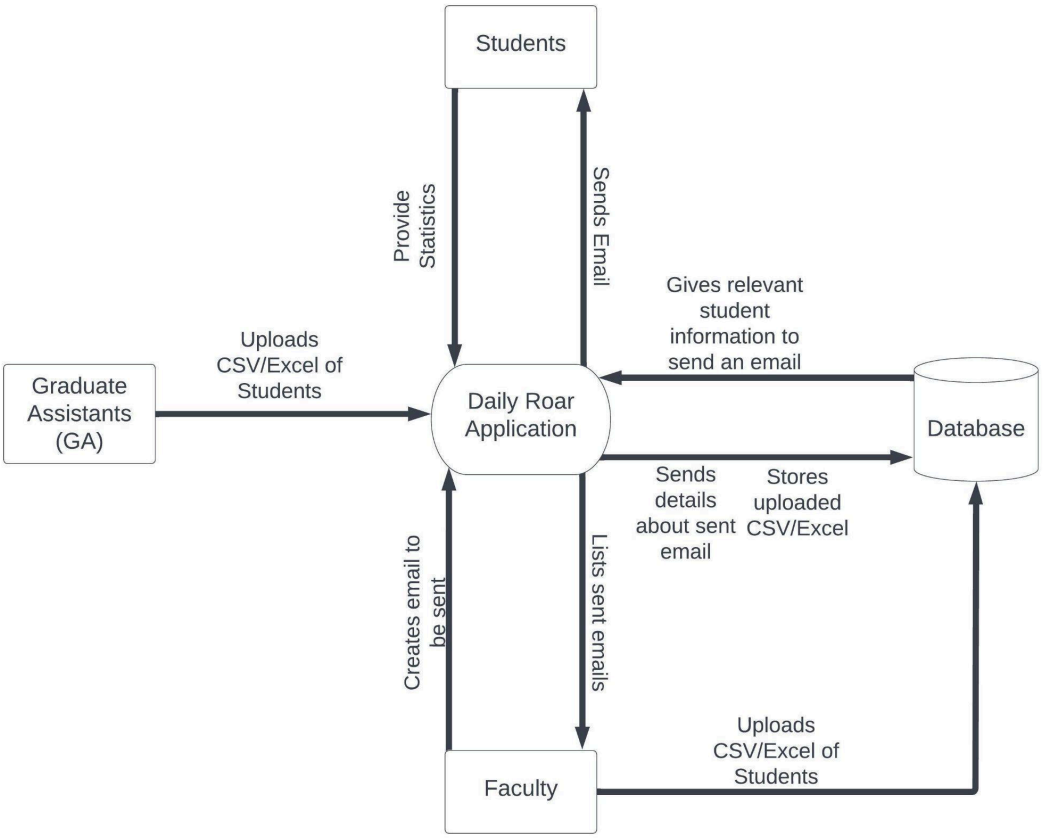


Figure 3: Context Diagram

3.5 Use Case Diagram

Use case diagrams are created to provide a visual representation of the system. They are also used to visualize how the system will be used by its users being the students at Kean

University and for this particular project, by the faculty and GA's. Individual diagrams will also be used to represent each type of user and how they interact with the system.

Actors are used to represent the users that will interact with the system. The tasks that each user could perform are demonstrated by arrows that point to the ovals which represent the specific tasks that could be performed. The system itself is represented as the large rectangle where the tasks are within.

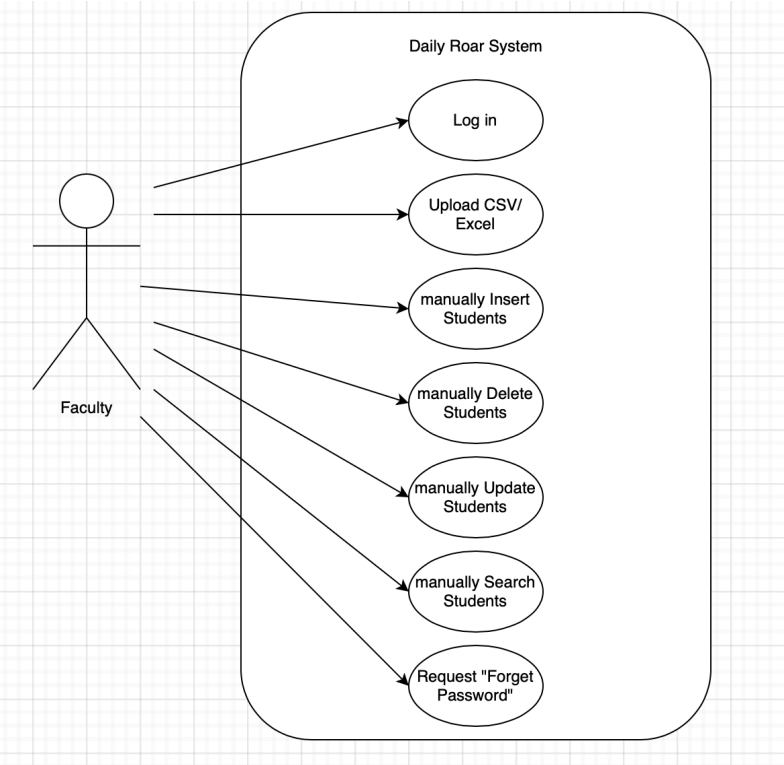


Figure 4: Faculty Use Case

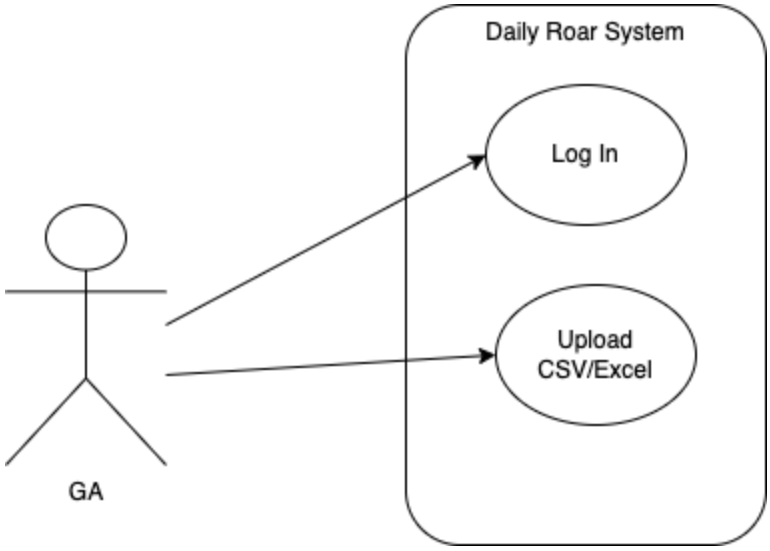


Figure 5: GA Use Case

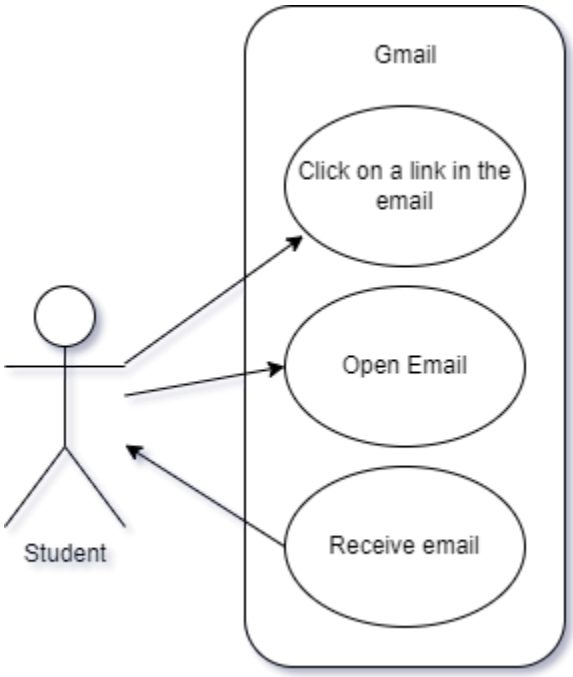


Figure 6: Student Use Case

4.0 System Functional Model

4.1 Representation of Functional Modules

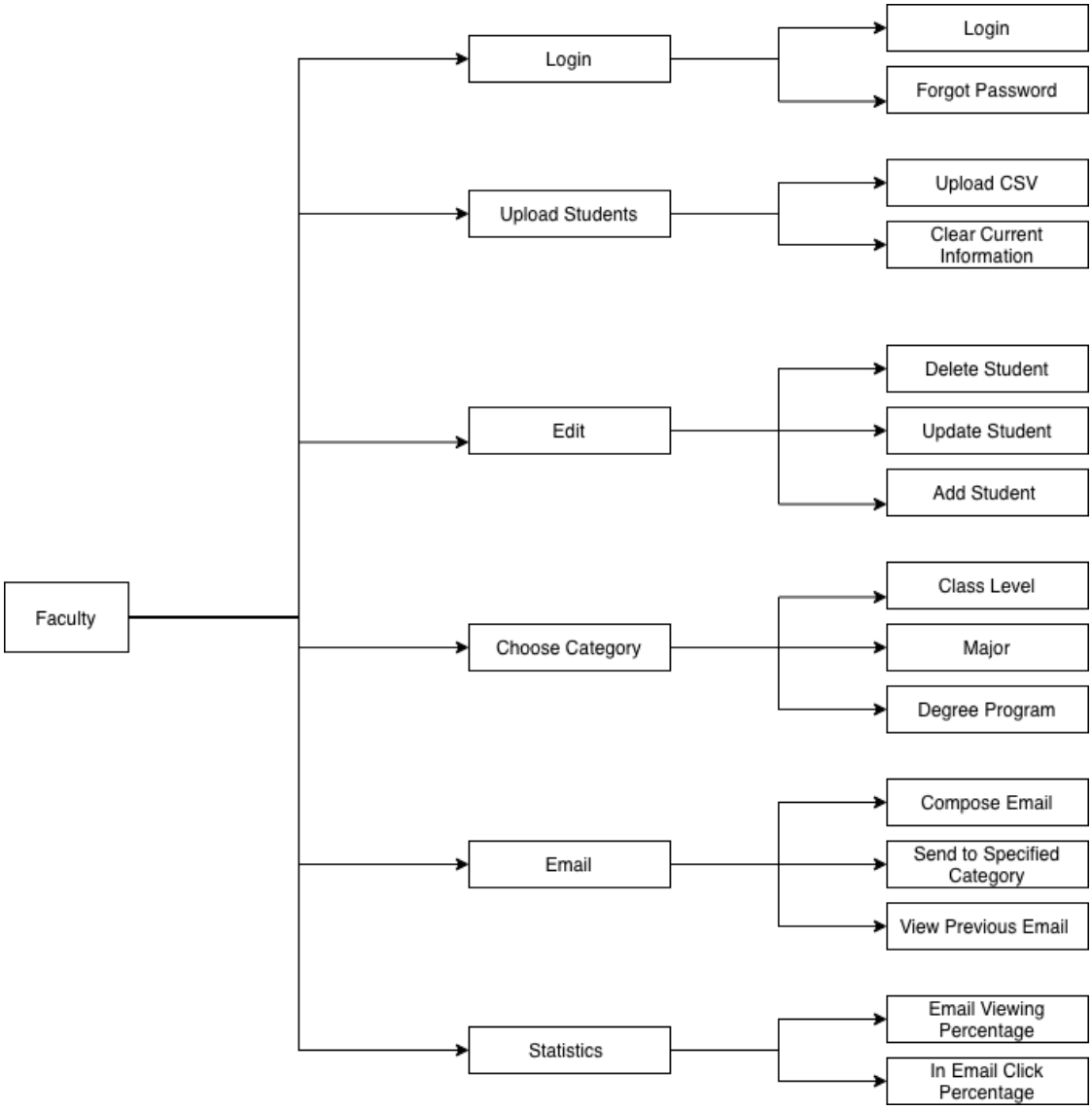


Figure 8: Faculty Functional Module

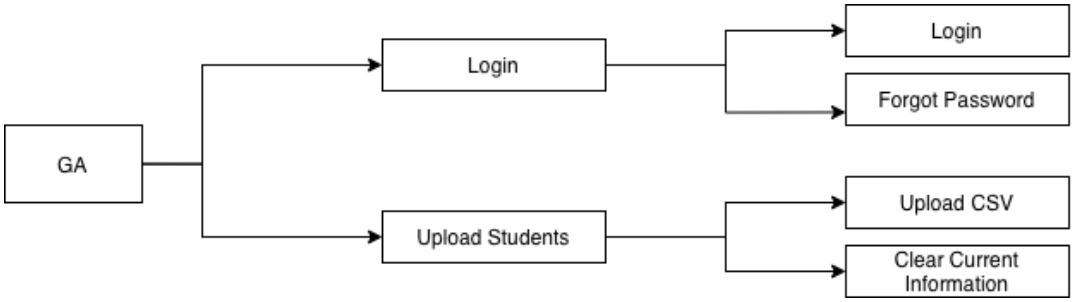


Figure 9: GA Functional Module

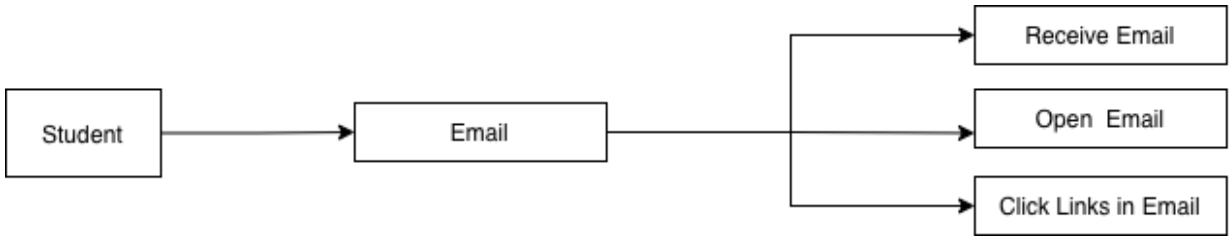
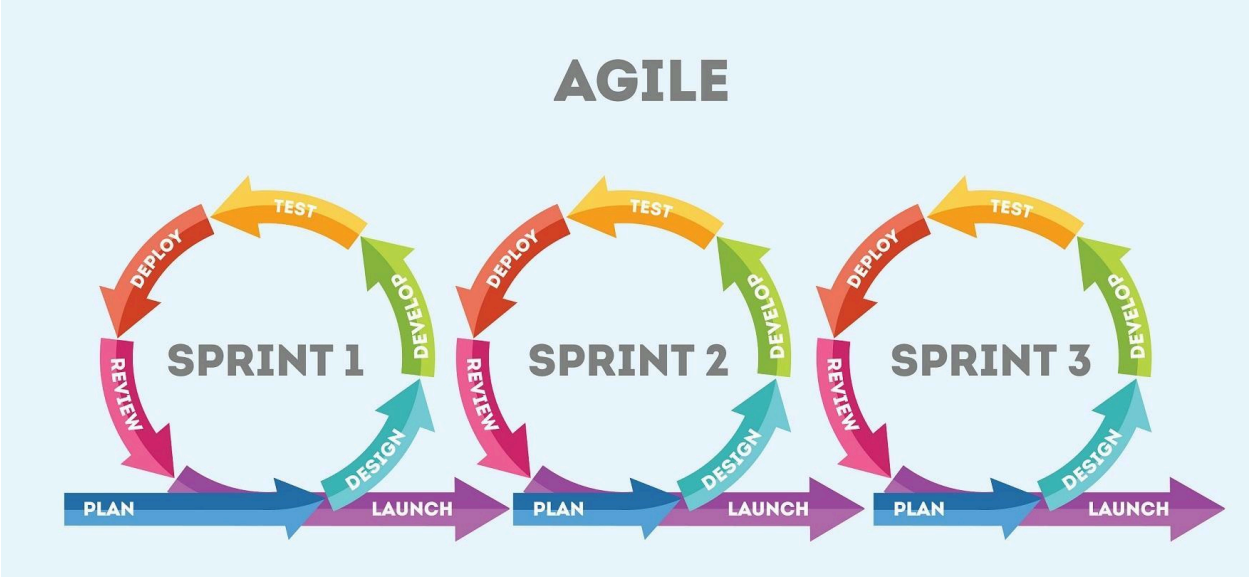


Figure 10: Student Functional Module

4.2 Software Process Model

Agile methodology is being used to create and manage this project and its software. Unlike the waterfall method, Agile is better suited for the job making the development of it more efficient and organized. The Agile method allows the developers to break up the project into multiple deliverables and sprints as well as also providing much room for changes and improvements throughout those deliverables and sprints.



This project is projected to have sprints throughout the life of the project development, Specifically the project will have 2 large sprints with multiple deliverables in each sprint.

Sprint 1: Create and design the front end of the web page as well as set up tables for the database. Allowing the user to be able to use some of the functionality of the system. This includes creating a login page with an authentication system and allowing them into their user dashboard whether they are Faculty or GA. Some of the functionality available will be to log in, Upload student information files (CSV/Excel), and send out emails.

Sprint 2: Create a “Forgot Password” function so if the Faculty or GA forget their information they can easily reset their account. The faculty should be able to send emails out to the students and the students should be able to view those emails. The system should be able to track the interactions of emails and links within emails from the students. Connect the front-end of the system to the back-end. Pass tests cases with a 95% passing rate to determine system efficiency, functionality, and project accomplishment.

4.3 Entity-Relationship Structure

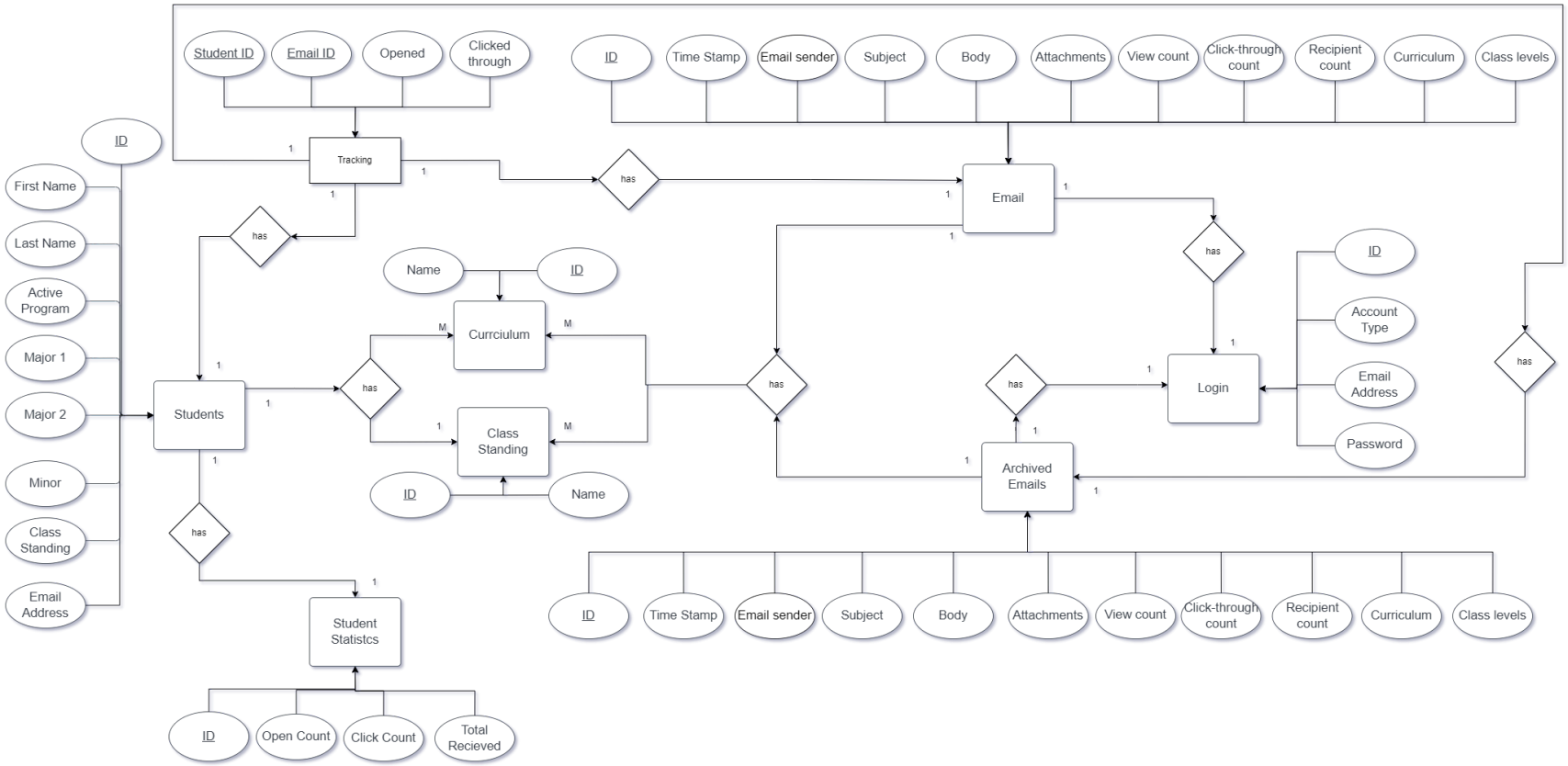


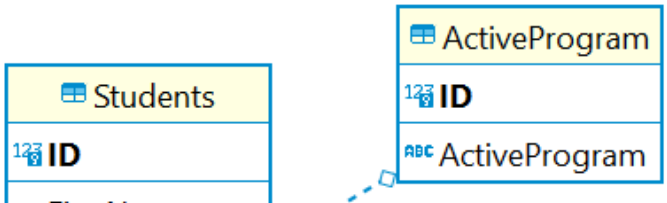
Figure 8: Entity-Relationship Diagram

The ERD above has the following **entities**:

- **Login** (ID, Account type, Email Address, Password)
- **Students** (ID, First name, Last name, Active program, Major 1, Major 2, Minor, Class Standing, Email address)
    - Has relationship with:
        - **Student Statistics** (1 to 1)
        - **Curriculum** (1 to many)
        - **Class Standing** (1 to 1)
- **Student Statistics** (ID, Open Count, Click Count, Total Received)
- **Tracking** (StudentID, EmailID, Opened, Click through)
    - Has relationship with:
        - **Students** (1 to 1)
        - **Email** (1 to 1)
- **Curriculum** (ID, Name)
- **Class Standing** (ID, Name)
- **Email** (ID, Time Stamp, Email sender, Subject, Body, Attachments, View count, Click-through count, Recipient count, Curriculum, Class levels)
    - Has relationship with:
        - **Curriculum** (1 to many)
        - **Class standing** (1 to many)
        - **Login** (1 to 1)
- **Archived Email** (ID, Time Stamp, Email sender, Subject, Body, Attachments, View count, Click-through count, Recipient count, Curriculum, Class levels)
    - Has relationship with:
        - **Curriculum** (1 to many)
        - **Class standing** (1 to many)
        - **Login** (1 to 1)

5.0 Data Design

5.1 Database Schema Diagram



6.0 Design Constraints, Restrictions and Limitations

6.1 Design Constraints

A design constraint that the team experienced is that the project is 3 months long so we had to come up with the design rather quickly.

Another constraint the team experienced during design was figuring out how to follow proper database structure when deciding how to keep track of the categories of students that an email was sent to.

6.2 System Restrictions and Limitations

The database necessitates that we follow third normal form; however the data we plan on occupying the database does not follow this criteria and would be unconventional to construct in this format. Data being stored for which recipients received emails needs to be stored in one column with multiple entries.

7.0 Tools and References

8.0 Tools Used to Create Diagrams

Tool used to create the diagrams is called “Draw.io”. Draw.io is a web application that allows users to create different types of diagrams. Draw.io provides hundreds of different symbols and characters. The application allows for multiple people to work together on singular diagrams at the same time from different computers and share them into a google drive. The symbols and characters make it easy for anyone to create diagrams, the ER diagram and Use Case diagrams for this project were created using this web application.

Another tool that was briefly used to create the context diagram was LucidChart. Lucid chart is another web application that allows users to create diagrams. The difference between the two is that for LucidChart the user has to create an account in order to use it as well. Another difference is that LucidChart is limited to characters and symbols limiting its capacity to make Use Case and ER Diagrams.

Github was another tool that was used to keep track of all the codes and to keep all the documents together. Github was also used to create tasks for each sprint. For example, in sprint 1 there are different tasks like update/delete student, manually insert student and creating a search function. All the tasks are assigned to certain members of the group and they are able to switch the task from “Todo”, “In progress” and “Done”.

8.1 Documentation Tool

Google Docs is being used to document this project. Google Docs is a Google platform that allows users to create documents and folders that can be placed inside google drive which is a system used to store documents and other folders in one place to be shared with other people. In these drives multiple users can work on the same document at the same time from different computers.

8.2 Reference Materials

- https://app.diagrams.net/
- https://docs.google.com/
- https://github.com/