Design Document

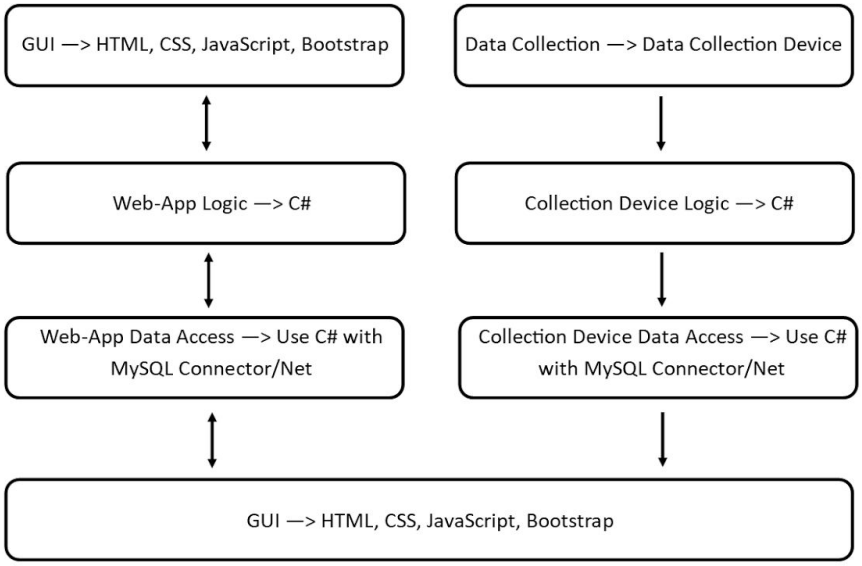
CSCE 361 - Fall 2017
Group 6

1 Introduction

This document provides a high level overview of the architecture and entity relations for the ACT (Attendance Collection & Tracking) system. This document contains descriptions of each architectural layer as well as relationships between different parts of the system and a detailed database table schema. The audience of this document is the system engineers and software architects who will be creating and maintaining this system.

2 Architecture

2.1 Introduction



The high level architectural design of this system will be layered, with two columns. One will feed data into the database and one will send and receive data from the GUI layer to the database layer and back. Both will utilize Entity frameworks to simplify the implementation. The collection device logic layer will read in student and TA exit and entry data and transmit that data to the collection device data access layer which will add the information to the SQL database. The GUI layer will pull information from the database using the web app logic layer and the web app data access layer. The web app logic layer, also using the Entity framework, will perform any data manipulation necessary before the data is displayed on the GUI layer.

2.2 Modules

2.2.1 GUI Layer

This is the primary layer the user will interact with, it will host the user interface. Users will be able to see the current number of students and TAs having office hours in the SRC and the schedule of TA office hours. Professors will be able to login to see detailed analytics based off of the data collected from student and TA SRC usage.
A combination of HTML, CSS, JavaScript, and Bootstrap framework will be used to generate the web pages from the data and allow for intuitive user interaction. Professors will be able to login to view the data analytics generated with Chart.js, change TA hours, and export data for their own analysis. Changes made in the GUI layer will be sent down the layers to update the database.

2.2.2 Web App Logic Layer

The web app logic layer will handle data requests made by the GUI layer. The web app logic layer will use C# objects provided by the web app data access layer to perform operations on the data stored in the database. Some operations include deciding which TAs are present during their TA hours, and creating the appropriate numbers It will pass the final formatted data to the GUI layer. The web app logic layer will also pass data to change from the GUI layer to the web app data access layer.

2.2.3 Web App Data Access Layer

The web app data access layer will query the database for information used by the web app logic layer. The web app data access layer will establish connection using the Entity framework. Data retrieved from the database will be formatted into C# objects which will be passed to the web app logic layer. The web app data access layer will also take in changes to data from the web app logic layer, and make the corresponding queries.

2.2.4 Database

The database will be implemented using an SQL database, hosted locally. Eventually the database will be hosted on the cse server. Access to the database will be restricted to the two data access layers.

2.2.5 Collection Device Data Access Layer

The device data access layer will query the database to insert information passed in from the collection device logic layer. It will be written in C#, and will also use Entity frameworks to

establish database connection. Unlike the web app data access layer, the collection device data access layer will not perform queries to read data from the database.

2.2.6 Collection Device Logic Layer

The Collection Device Logic Layer reads in data from the collection device and performs any necessary formatting before passing it onto the collection device data access layer. It will be written in C#, and will primarily deal with creating timestamps to pass to the collection device data access layer.

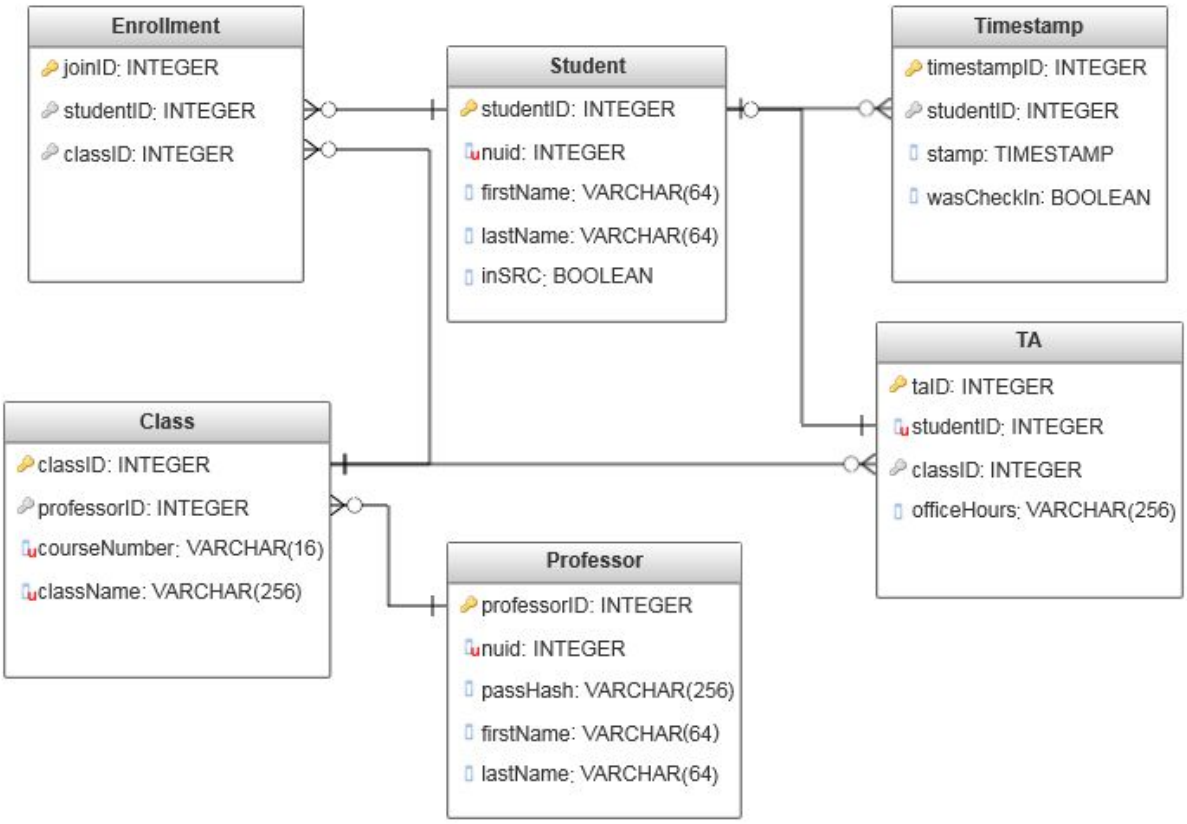
2.2.7 Collection Device

This is the hardware component that will perform the scanning of students' Ncards and generate the timestamps that will be passed to the collection device logic layer.

3 Class Diagrams

3.1 Database

3.1.2 Database Schema



3.1.2 Schema Description

3.1.2.1 Student Table

The Student table represents students enrolled in CSCE classes. The table has five entries:

- studentID - The primary key for the entry
- nuid - The 8 digit NUID of the student
- firstName - A string of the student's first name
- lastName - A string of the student's last name
- inSRC - A boolean value storing if a student is currently in the SRC

3.1.2.2 Timestamp Table

The Timestamp table stores the times when a student has checked in or out. The table has four entries:

- timestampID - The primary key for the entry
- studentID - The foreign key for the student who is associated with the timestamp
- stamp - The timestamp when the event occurred
- wasCheckIn - boolean value sotoring if the event was a check in (false is check out)

3.1.2.3 Professor Table

The Professor Table represents the instructors of courses. The table has four entris:

- professorID - The primary key for the entry
- nuid - The 8 digit NUID of the professor
- passHash - The hash of the professor's password
- firstName - The first name of the professor
- lastName - The last name of the professor

3.1.2.4 Class Table

The Class table represents courses students can enroll in and that professors can teach. The table four entries:

- classID - The primary key for the entry
- professorID - The foreign key for the professor who teaches the class
- courseNumber - The UNL assigned course number (ex "CSCE 361")
- className - The name of the class (ex "Software Engineering")

3.1.2.5 TA table

The TA table represents teaching assistants (TAs), and links the student to the class the student TAs for. The table has four entries:

- taID - The primary key for the entry
- studentID - The student entry for the TA
- classID - The class the TA is helping with
- officeHours - Information about the date and time of the TA's office hours.

3.1.2.6 Enrollment table

The Enrollment represents each instance of a student enrolling in a class. The table has three entries:

- joinID - The primary key for the entry
- studentID - The student enrolling
- classID - The class being enrolled in

3.2 Class Information

Data in the database will be represented in the logic layers as C# objects. An object will represent an entry in the table. If an entry has a many to one relationship with another entry, the relationships will be implemented with a data structure storing many objects of the desired type (I.E. a Student who is in more than one Class will have a list of Classes to store a schedule in). While most data structures show minimal inheritance, the TA will be a child of a Student object.

3.3 GUI Layer

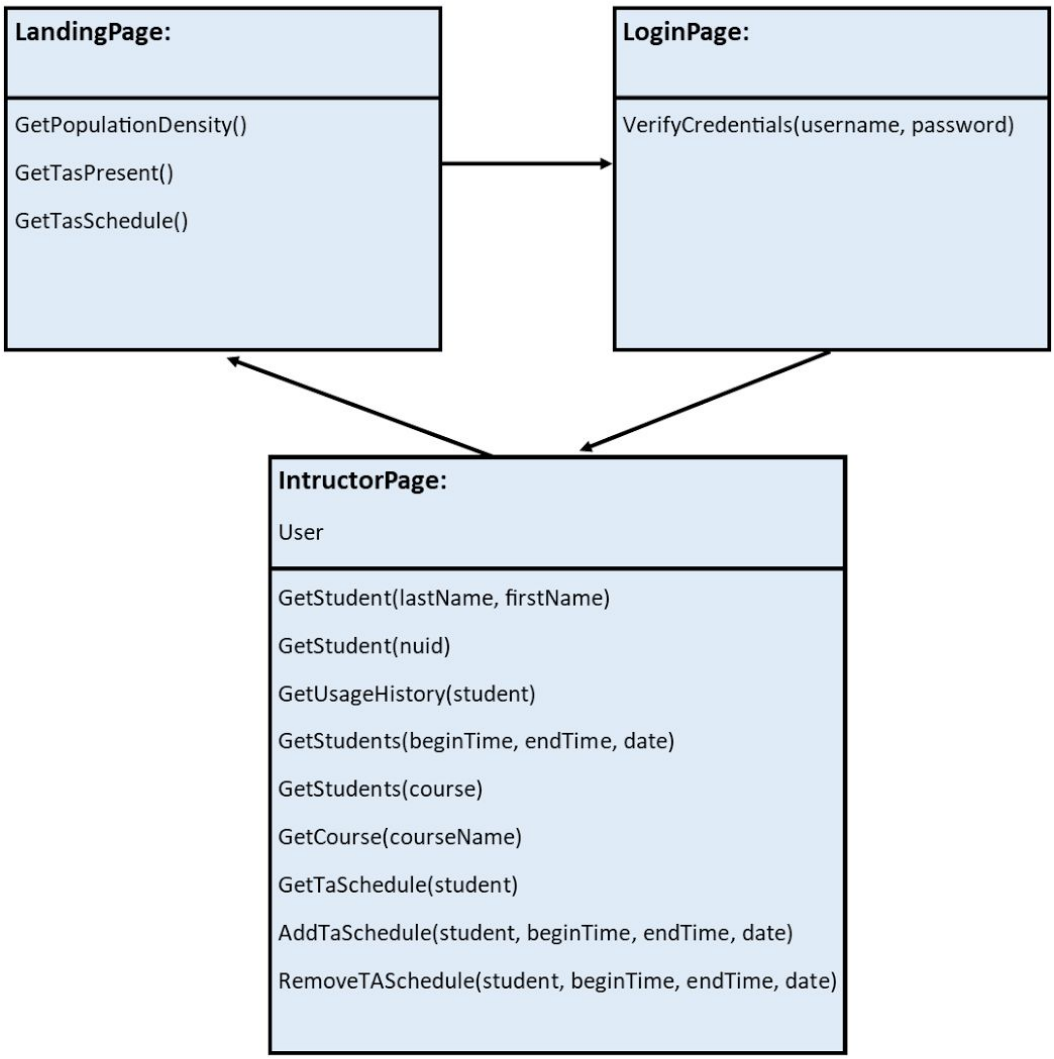
Three pages make up the graphical user interface of the system which includes the landing page, the login page, and the instructor page. The landing page will appear when users point their web browser to the website. The landing page will display information about the population density of students in the room, which TAs are present, and a schedule for the TAs. This will be a static webpage containing all of the aforementioned data. The landing page will contain two buttons: the refresh button or the login button. The information on the webpage may be updated by refreshing the browser or by pressing the refresh button located on the webpage. Users with special access may begin the login process by pressing the login button. The functions required for this page include retrieving the number of students in the room, the names of the TA's present, and all of the TA's schedules.

After pressing the login button on the landing page, the user will be prompted to enter a valid username and password. When the user enters their credentials the submitted information references the credentials stored in the database. If the credentials match an existing user's credentials exactly, the user will be directed to the instructor page. If the credentials do not exactly match any existing users' credentials in the database, then the user is prompted with an error message.

The instructor page has several additional functionalities. Upon loading the instructor page, the instructor may obtain specific data stored in the database regarding room usage and TA schedules. More specifically, the instructor may search for a student, a class, and/or a time/day. The instructor can search for a student by name or NUID. The resultant output will include a list of all students from the database that match the search. A student function will get student SRC usage history which will retrieve a list of the timestamps associated with the student. The instructor may also search for a course. The functions associated with a course include retrieving students in the course and/or specific section. Instructors may also submit a query for the time/day. The functions associated with a temporal query include retrieving students in the SRC at a given time. Another additional function is that each of the queries can be given constraints. For example, a search for any student who was in the SRC on Oct. 13th, 2017 and is in the CSCE 361 course.

The instructor page will also have an option to edit, or remove TA schedules. To edit a TA schedule the instructor will search the student who is a TA. Instructors can select the TA. The pertinent TA information will then be retrieved by the logic layer and displayed on the webpage for the instructor. The instructor can then remove the TA or change the schedule information associated with the TA and submit it. The TA scheduling functionality requires functions to retrieve TA scheduling information, update, and remove TA scheduling information.

There will be an option to view the data analytics of the query results. The functions required for data analytics include passing the the queried data to a chart.js object. Only two-variable visualizations will be supported. The instructor will have the option to choose from the available function options listed above that they would like for the x and the y variables for a two-dimensional plane. If there is an input mismatch, an error notification will be displayed to the instructor. Furthermore, there will be an option to export the query results into a JSON format. Therefore, there will need to be a function that converts the query results into a JSON formated output. The instructor will also have the option to navigate back to the landing page.


LandingPage:
GetPopulationDensity()
GetTasPresent()
GetTasSchedule()
LoginPage:
VerifyCredentials(username, password)
IntructorPage:
User
GetStudent(lastName, firstName)
GetStudent(nuid)
GetUsageHistory(student)
GetStudents(beginTime, endTime, date)
GetStudents(course)
GetCourse(courseName)
GetTaSchedule(student)
AddTaSchedule(student, beginTime, endTime, date)
RemoveTASchedule(student, beginTime, endTime, date)