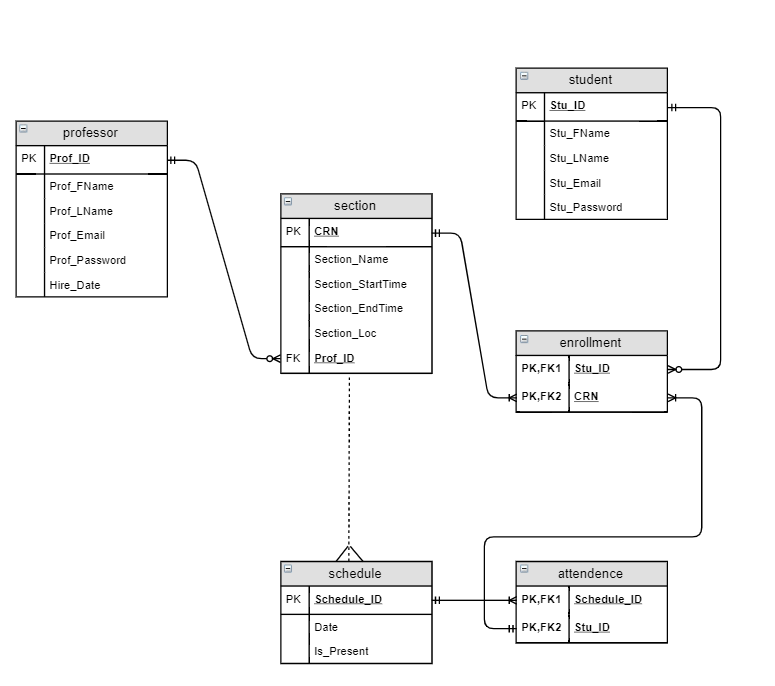
Team name: Lemonade

Project title: Attendance tracking database

Group Members: Patrick Horne, Erik Stroud, Lukas Paradiso, and Nicholas Thompson

It is part of another project in CSC 450 with the same group members



We will be working to create a web and mobile based application of attendance tracking software. This software will be useful in tracking classroom attendance in schools or universities, or any other situation where attendance needs to be tracked. The web application will be accessible to an administrator/teacher who is wanting to keep a record of attendance. The students will also have access to the web application in the case that they have problems with the application on their phones. The mobile application will be used by a user/student to “check in” to a classroom or meeting.

This application relies on a few major functions. The first one being, to generate a class or meeting code for the student/user to log into a class initially. Also, this software will generate a code each day the administrator wants to log attendance. This code will be inputted by the student/user on their mobile application to log their attendance for the meeting. To ensure that a person is truly in attendance, we will use the location services on the mobile phone or specific subnets depending on the university’s implementation of their network. Another function that we will be implementing is for the administrator to view and manage their class records.

Throughout this project we expect to run into some problems including connectivity issues from the mobile application and website to the database, and performance issues with the user’s phone network. At the moment, we are unsure of how to implement that the student/user is actually in their class or meeting.

Professor will be an entity in our database and will serve as our table of professors. Our Professor table will have the following attributes: Prof\_ID (the employee ID number of the professor), Prof\_FName (first name), Prof\_LName (last name), Prof\_Email (employee email address), and Prof\_password (the professor’s password which will be encrypted), Hire\_Date (the date which the professor was hired).

Section will be an entity in our database and will serve as the table containing a section of a class. We will not have a class entity because we only care about each class’ individual sections. Section will have the following attributes: CRN (the course registration number of the section), Section\_Name (the name of the section), Section\_StartTime (the time the section will start), Section\_EndTime (the time the section will end), Section\_Loc (the location the section will be taught), and the foreign key Prof\_ID from the professor table.

Student will be an entity in our database and will serve as our table of students. Our student table will have the following attributes: Stu\_ID (the student ID number of the student), Stu\_FName (first name), Stu\_LName (last name), Stu\_Email (student email address), and Stu\_password (the student’s password which will be encrypted).

Schedule will be an entity in our database and will serve as the table containing the schedule of the Section entity. Schedule will have the following attributes: Date (the date of the section on a given day), Is\_Present (the boolean value representing if the student was present in the section on a given day). Schedule has a weak many-to-one relationship with Section.

Enrollment serves as a junction table between Section and Student. Attendance serves as a junction table between Enrollment and Schedule.

|  |  |  |
| --- | --- | --- |
| Schedule\_ID | INT(11) | Primary Key, Foreign Key |
| Stu\_ID | INT(11) | Primary Key, Foreign Key |

|  |
| --- |
| ATTENDANCE |

|  |
| --- |
| ENROLLMENT |

|  |  |  |
| --- | --- | --- |
| Stu\_ID | INT(11) | Primary Key, Foreign Key |
| CRN | INT(11) | Primary Key, Foreign Key |

|  |
| --- |
| PROFESSOR |

|  |  |  |
| --- | --- | --- |
| Prof\_ID | INT(11) | Primary Key |
| Prof\_FName | VARCHAR(45) |  |
| Prof\_LName | VARCHAR(45) |  |
| Prof\_Email | VARCHAR(45) |  |
| Prof\_Password | VARCHAR(45) | Foreign Key |
| Hire\_Date | Date | NOT NULL |

|  |
| --- |
| SCHEDULE |

|  |  |  |
| --- | --- | --- |
| Schedule\_ID | INT(11) | Primary Key |
| Date | datetime |  |
| Is\_Present | Tinyint(4) |  |

|  |
| --- |
| SECTION |

|  |  |  |
| --- | --- | --- |
| CRN | INT(11) | Primary Key |
| Section\_Name | VARCHAR(45) |  |
| Section\_StartTime | VARCHAR(45) |  |
| Section\_EndTime | VARCHAR(45) |  |
| Section\_Loc | VARCHAR(45) |  |
| Prof\_ID | INT(11) | Foreign Key |

|  |
| --- |
| STUDENT |

|  |  |  |
| --- | --- | --- |
| Stu\_ID | INT(11) | Primary Key |
| Stu\_FName | VARCHAR(45) |  |
| Stu\_LName | VARCHAR(45) |  |
| Stu\_Email | VARCHAR(45) |  |
| Stu\_Password | VARCHAR(45) |  |

To learn how to navigate our web app, click “Learn More” on our homepage!

**Queries:**

Two-table join

SELECT p.Prof\_FName, p.Prof\_LName, se.Section\_Name, se.Section\_StartTime, se.Section\_EndTime, se.Section\_Loc

FROM professor p

INNER JOIN section se

ON p.Prof\_ID = se.Prof\_ID;

Three-table join

SELECT st.Stu\_FName, st.Stu\_LName, e.CRN, se.Section\_Name

FROM enrollment e

INNER JOIN student st

ON st.Stu\_ID = e.Stu\_ID

INNER JOIN section se

ON se.CRN = e.CRN

Self Join

SELECT st1.Stu\_ID, st1.Stu\_FName, st2.Stu\_FName, st2.Stu\_LName "Common\_LName"

FROM student st1

INNER JOIN student st2

ON st2.Stu\_LName = st1.Stu\_LName

WHERE st2.Stu\_ID <> st1.Stu\_ID

ORDER BY st2.Stu\_LName

Aggregate

SELECT CRN, COUNT(Stu\_ID) AS Class\_Count

FROM enrollment

WHERE CRN = '300';

Aggregate using GROUP BY and HAVING

SELECT CRN, COUNT(Stu\_ID)

FROM enrollment

GROUP BY CRN

HAVING COUNT(Stu\_ID) >= 2

Text-based-search using LIKE

SELECT \*

FROM professor

WHERE Prof\_FName like 'J%';

Subquery

SELECT p.Prof\_ID, p.Prof\_FName, p.Prof\_LName, se.Section\_Name, se.Section\_StartTime, se.Section\_EndTime, se.Section\_Loc

FROM professor p

JOIN section se

ON p.Prof\_ID = se.Prof\_ID

WHERE se.Section\_StartTime != (SELECT Section\_StartTime

FROM section

WHERE Prof\_ID = 3 AND Section\_StartTime = "14:00")

Stored Procedure

CREATE PROCEDURE show\_students

BEGIN

SELECT Stud\_ID, Stu\_FName, Stu\_LName;

END;

Trigger

CREATE TRIGGER logDeleteStudent AFTER DELETE ON student

FOR EACH ROW

BEGIN

DELETE FROM enrollment

    WHERE enrollment.Stu\_ID = old.id

END;