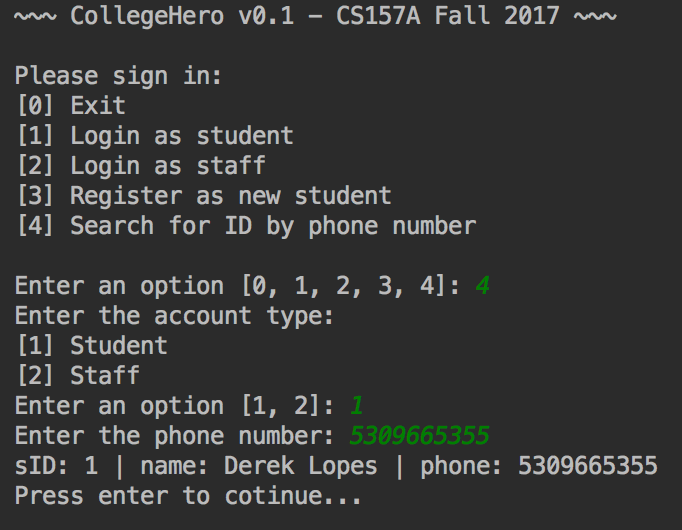
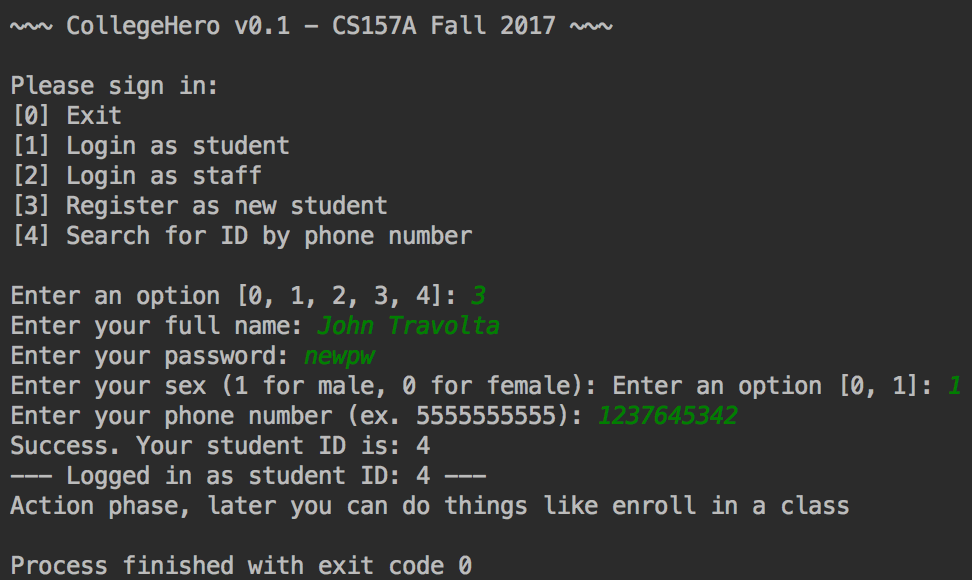
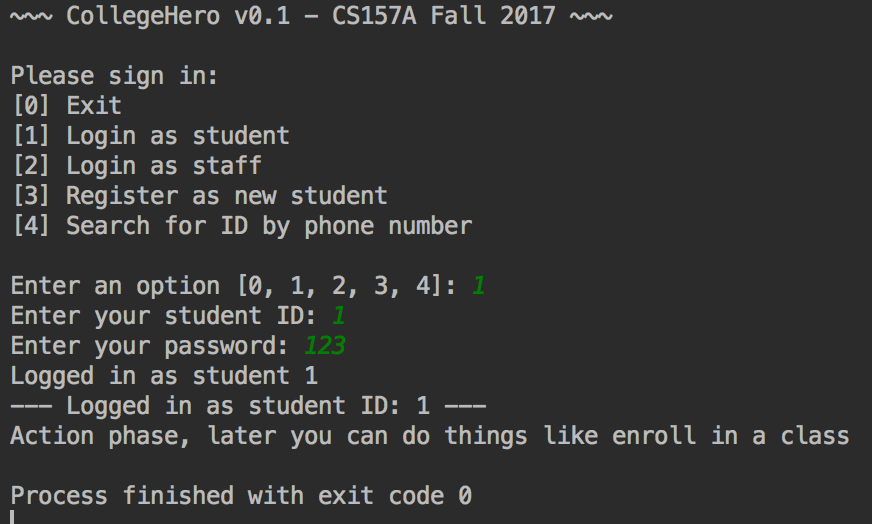
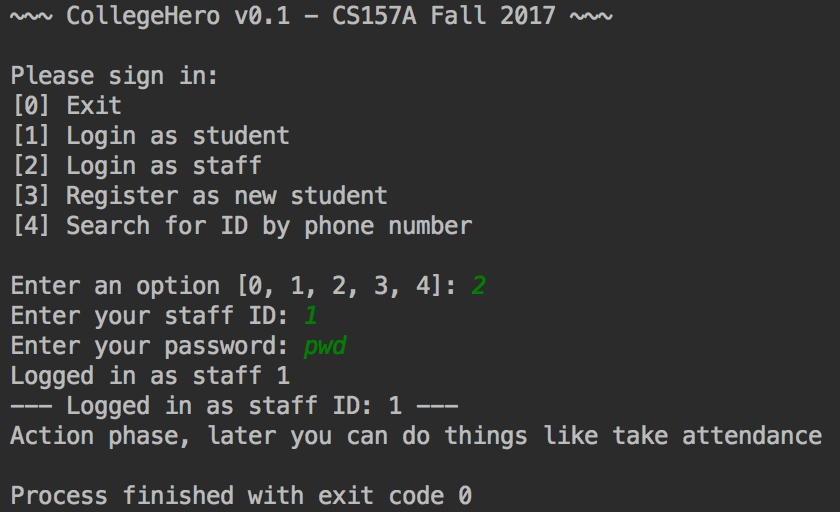
Team Name: Coursera

Group Members: Derek Lopes, Tuan Nguyen, Adam Houseman

CollegeHero: Report 2

Basic Overview: Our project is to create a school management system. Users are either a student or a staff member. Staff members may either be teachers or admins (which changes privileges). Every student and staff is given an ID and can sign in using that. They must also provide a phone number for verification. If a user forgets their ID they can find it with their phone number. Students can create enroll in classes, which automatically raises their tuition. They can then pay that tuition. Admins can create classes and assign teachers to classes. A class cannot be created if there is a time conflict for the room it will be held in. Classes have a maximum capacity and once that has been reached, no students can add that class to their schedule. Teachers can take attendance for a class. Schedules for students, teachers, and rooms can also be viewed.

Screenshots:



Database Schema, Procedures, and Triggers:

-- Tables  
  
**DROP DATABASE IF EXISTS** CollegeHero;  
**CREATE DATABASE** CollegeHero;  
**USE** CollegeHero;  
  
**DROP TABLE IF EXISTS** student;  
**CREATE TABLE** student  
(  
 sID **INT PRIMARY KEY AUTO\_INCREMENT**,  
 **name VARCHAR**(255) **NOT NULL**,  
 **password VARCHAR**(60) **NOT NULL**,  
 sex **BOOLEAN**,  
 phone **VARCHAR**(10) **UNIQUE**,  
 tuition **INT DEFAULT** 0 **NOT NULL**,  
 updatedAt **TIMESTAMP DEFAULT** *CURRENT\_TIMESTAMP*()  
);  
  
**DROP TABLE IF EXISTS** room;  
**CREATE TABLE** room  
(  
 rID **INT PRIMARY KEY AUTO\_INCREMENT**,  
 roomNumber **INT**,  
 building **VARCHAR**(45)  
);  
  
**DROP TABLE IF EXISTS** department;  
**CREATE TABLE** department  
(  
 dID **INT PRIMARY KEY AUTO\_INCREMENT**,  
 office **INT**,  
 title **VARCHAR**(45) **NOT NULL**,  
 **FOREIGN KEY** (office) **REFERENCES** room (rID)  
);  
  
**DROP TABLE IF EXISTS** staff;  
**CREATE TABLE** staff  
(  
 tID **INT PRIMARY KEY AUTO\_INCREMENT**,  
 **name VARCHAR**(255) **NOT NULL**,  
 **password VARCHAR**(60) **NOT NULL**,  
 department **INT**,  
 staffTypeID **BOOLEAN**,  
 phone **VARCHAR**(10),  
 **FOREIGN KEY** (department) **REFERENCES** department (dID)  
);  
  
**DROP TABLE IF EXISTS** class;  
**CREATE TABLE** class  
(  
 cID **INT**,  
 section **INT NOT NULL**,  
 **subject VARCHAR**(45) **NOT NULL**,  
 tID **INT**,  
 rID **INT**,  
 days **VARCHAR**(15),  
 start\_at **TIME**,  
 end\_at **TIME**,  
 capacity **INT NOT NULL**,  
 cost **INT NOT NULL**,  
 **FOREIGN KEY** (tID) **REFERENCES** staff (tID),  
 **FOREIGN KEY** (rID) **REFERENCES** room (rID),  
 **PRIMARY KEY** (cID, section)  
);  
  
**DROP TABLE IF EXISTS** enrolled;  
**CREATE TABLE** enrolled  
(  
 sID **INT**,  
 cID **INT**,  
 section **INT**,  
 **PRIMARY KEY** (sID, cID, section),  
 **FOREIGN KEY** (sID) **REFERENCES** student (sID),  
 **FOREIGN KEY** (cID, section) **REFERENCES** class (cID, section)  
);  
  
**DROP TABLE IF EXISTS** attendance;  
**CREATE TABLE** attendance  
(  
 sID **INT**,  
 cID **INT**,  
 section **INT**,  
 **day DATE**,  
 **PRIMARY KEY** (sID, cID, **day**),  
 **FOREIGN KEY** (sID) **REFERENCES** student (sID),  
 **FOREIGN KEY** (cID, section) **REFERENCES** class (cID, section)  
);  
  
**DROP TABLE IF EXISTS** studentArchive;  
**CREATE TABLE** studentArchive  
(  
 ID **INT PRIMARY KEY AUTO\_INCREMENT**,  
 sID **INT**,  
 **name VARCHAR**(255) **NOT NULL**,  
 **password VARCHAR**(60) **NOT NULL**,  
 sex **BOOLEAN**,  
 phone **VARCHAR**(10) **UNIQUE**,  
 tuition **INT DEFAULT** 0 **NOT NULL**,  
 updatedAt **TIMESTAMP**,  
 **UNIQUE** (sID)  
);  
  
-- Procedures  
  
**DELIMITER** //  
  
**DROP PROCEDURE IF EXISTS** *getStudentByID*//  
**CREATE PROCEDURE** *getStudentByID*(**IN** ID **INT**)  
 **BEGIN  
 SELECT** \*  
 **FROM** student  
 **WHERE** sID = ID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStudentPasswordByID*//  
**CREATE PROCEDURE** *getStudentPasswordByID*(**IN** ID **INT**, **OUT** pass **VARCHAR**(60))  
 **BEGIN  
 SELECT password  
 INTO** pass  
 **FROM** student  
 **WHERE** sID = ID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStaffByID*//  
**CREATE PROCEDURE** *getStaffByID*(**IN** ID **INT**)  
 **BEGIN  
 SELECT** \*  
 **FROM** staff  
 **WHERE** tID = ID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStaffPasswordByID*//  
**CREATE PROCEDURE** *getStaffPasswordByID*(**IN** ID **INT**, **OUT** pass **VARCHAR**(60))  
 **BEGIN  
 SELECT password  
 INTO** pass  
 **FROM** staff  
 **WHERE** tID = ID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStudentIDByPhone*//  
**CREATE PROCEDURE** *getStudentIDByPhone*(**IN** phone **VARCHAR**(10))  
 **BEGIN  
 SELECT** sID,  
 student.name,  
 student.phone  
 **FROM** student  
 **WHERE** student.phone = phone;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStaffIDByPhone*//  
**CREATE PROCEDURE** *getStaffIDByPhone*(**IN** phone **VARCHAR**(10))  
 **BEGIN  
 SELECT** tID,  
 staff.name,  
 staff.phone  
 **FROM** staff  
 **WHERE** staff.phone = phone;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *createStudent*//  
**CREATE PROCEDURE** *createStudent*(**IN name VARCHAR**(255), **IN password VARCHAR**(60), **IN** sex **BOOLEAN**, **IN** phone **VARCHAR**(10),  
 **OUT** ID **VARCHAR**(255))  
 **BEGIN  
 INSERT INTO** student **VALUES** (**NULL**, **name**, **password**, sex, phone, 0, **NULL**);  
 **SELECT** student.sID  
 **INTO** ID  
 **FROM** student  
 **WHERE** phone = student.phone;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *createStaff*//  
**CREATE PROCEDURE** *createStaff* (**IN name VARCHAR**(255), **IN password VARCHAR**(60), **IN** department **INT**, **IN** staffType **INT**, **IN** phone **VARCHAR**(10))  
 **BEGIN  
 INSERT INTO** staff **VALUES** (**NULL**, **name**, **password**, department, staffType, phone);  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getAllSectionInfoByClassID*;  
**CREATE PROCEDURE** *getAllSectionInfoByClassID*(**IN** cID **INT**)  
 **BEGIN  
 SELECT** \*  
 **FROM** class  
 **WHERE** class.cID = cID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getAllSectionInfoBySubject*//  
**CREATE PROCEDURE** *getAllSectionInfoBySubject*(**IN subject VARCHAR**(45))  
 **BEGIN  
 SELECT** \*  
 **FROM** class  
 **WHERE** class.subject = **subject**;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStudentSchedule*//  
**CREATE PROCEDURE** *getStudentSchedule*(**IN** sID **INT**)  
 **BEGIN  
 SELECT** \*  
 **FROM** class  
 **WHERE** class.cID **IN** (**SELECT** enrolled.cID  
 **FROM** enrolled  
 **WHERE** enrolled.sID = sID);  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStaffSchedule*//  
**CREATE PROCEDURE** *getStaffSchedule*(**IN** tID **INT**)  
 **BEGIN  
 SELECT** \*  
 **FROM** class  
 **WHERE** class.tID = tID;  
 **END** //  
  
**DROP PROCEDURE IF EXISTS** *logAttendance*//  
**CREATE PROCEDURE** *logAttendance*(**IN** sID **INT**, **IN** cID **INT**, **IN** section **INT**, **IN** classDay **DATE**)  
 **BEGIN  
 INSERT INTO** attendance  
 **VALUES** (sID, cID, section, classDay);  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *enrollInClass*//  
**CREATE PROCEDURE** *enrollInClass*(**IN** sID **INT**, **IN** cID **INT**, **IN** section **INT**)  
 **BEGIN  
 INSERT INTO** enrolled  
 **VALUES** (sID, cID, section);  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *unEnrollInClass*//  
**CREATE PROCEDURE** *unEnrollInClass*(**IN** sID **INT**, **IN** cID **INT**, **IN** section **INT**)  
 **BEGIN  
 DELETE FROM** enrolled  
 **WHERE** enrolled.sID = sID  
 **AND** enrolled.cID = cID  
 **AND** enrolled.section = section;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *payTuition*//  
**CREATE PROCEDURE** *payTuition*(**IN** sID **INT**, **IN** amount **INT**)  
 **BEGIN  
 UPDATE** student  
 **SET** tuition = tuition - amount  
 **WHERE** student.sID = sID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStaffByName*//  
**CREATE PROCEDURE** *getStaffByName*(**IN** staffName **VARCHAR**(255))  
 **BEGIN  
 SELECT** staff.tID,  
 staff.name,  
 staff.department,  
 staff.phone,  
 staff.staffTypeID  
 **FROM** staff  
 **WHERE** staff.name = staffName;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStaffByDepartment*//  
**CREATE PROCEDURE** *getStaffByDepartment*(**IN** department **VARCHAR**(255))  
 **BEGIN  
 SELECT** staff.tID,  
 staff.name,  
 staff.department,  
 staff.phone,  
 staff.staffTypeID  
 **FROM** staff  
 **WHERE** staff.department = department;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStaffByType*//  
**CREATE PROCEDURE** *getStaffByType*(**IN** type **BOOLEAN**)  
 **BEGIN  
 SELECT** staff.tID,  
 staff.name,  
 staff.department,  
 staff.phone,  
 staff.staffTypeID  
 **FROM** staff  
 **WHERE** staff.staffTypeID = type;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *changeStudentPhone*//  
**CREATE PROCEDURE** *changeStudentPhone*(**IN** sID **INT**, **IN** phone **VARCHAR**(10))  
 **BEGIN  
 UPDATE** student  
 **SET** student.phone = phone  
 **WHERE** student.sID = sID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *changeStudentPassword*//  
**CREATE PROCEDURE** *changeStudentPassword*(**IN** sID **INT**, **IN** pwd **VARCHAR**(60))  
 **BEGIN  
 UPDATE** student  
 **SET** student.password = pwd  
 **WHERE** student.sID = sID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *changeClassInstructor*//  
**CREATE PROCEDURE** *changeClassInstructor*(**IN** cID **INT**, **IN** tID **INT**)  
 **BEGIN  
 UPDATE** class  
 **SET** class.tID = tID  
 **WHERE** class.cID = cID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getRoomSchedule*//  
**CREATE PROCEDURE** *getRoomSchedule*(**IN** rID **INT**)  
 **BEGIN  
 SELECT** \*  
 **FROM** class  
 **WHERE** class.rID = rID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *createClass*//  
**CREATE PROCEDURE** *createClass*(**IN** cID **INT**, **IN** section **INT**, **IN** subjct **VARCHAR**(45), **IN** tID **INT**, **IN** rID **INT**,  
 **IN** days **VARCHAR**(15), **IN** start\_at **TIME**, **IN** end\_at **TIME**, **IN** capacity **INT**, **IN** cost **INT**)  
 **BEGIN  
 INSERT INTO** class  
 **VALUES** (cID, section, subjct, tID, rID, days, start\_at, end\_at, capacity, cost);  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStudentsEnrolled*//  
**CREATE PROCEDURE** *getStudentsEnrolled*(**IN** cID **INT**, **IN** section **INT**)  
 **BEGIN  
 SELECT** student.name,  
 student.phone  
 **FROM** student  
 **WHERE** student.sID **IN** (**SELECT** sID  
 **FROM** enrolled  
 **WHERE** enrolled.cID = cID  
 **AND** enrolled.section = section);  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *getStudentAttendance*//  
**CREATE PROCEDURE** *getStudentAttendance*(**IN** sID **INT**)  
 **BEGIN  
 SELECT** \*  
 **FROM** attendance  
 **WHERE** attendance.sID = sID;  
 **END**//  
  
**DROP PROCEDURE IF EXISTS** *archiveStudents*//  
**CREATE PROCEDURE** *archiveStudents*(**IN** updatedBy **DATE**)  
 **BEGIN  
 INSERT INTO** studentArchive  
 (sID, **name**, **password**, sex, phone, tuition, updatedAt)  
 (**SELECT** \*  
 **FROM** student  
 **WHERE** student.updatedAt < updatedBy);  
 **DELETE FROM** student  
 **WHERE** student.updatedAt < updatedBy;  
 **END**//  
  
-- Triggers  
  
**DROP TRIGGER IF EXISTS** increaseTuition//  
**CREATE TRIGGER** increaseTuition  
**BEFORE INSERT ON** enrolled  
**FOR EACH ROW  
 BEGIN  
 UPDATE** student  
 **SET** tuition = tuition + (**SELECT** cost  
 **FROM** class  
 **WHERE** cID = new.cID  
 **AND** section = new.section)  
 **WHERE** student.sID = new.sID;  
 **END**//  
  
**DROP TRIGGER IF EXISTS** lowerTuition//  
**CREATE TRIGGER** lowerTuition  
**BEFORE DELETE ON** enrolled  
**FOR EACH ROW  
 BEGIN  
 UPDATE** student  
 **SET** tuition = tuition - (**SELECT** cost  
 **FROM** class  
 **WHERE** cID = old.cID  
 **AND** section = old.section)  
 **WHERE** student.sID = old.sID;  
 **END**//  
  
**DROP TRIGGER IF EXISTS** checkCapacity//  
**CREATE TRIGGER** checkCapacity  
**BEFORE INSERT ON** enrolled  
**FOR EACH ROW  
 BEGIN  
 IF** (**SELECT** *COUNT*(enrolled.cID)  
 **FROM** enrolled  
 **WHERE** enrolled.cID = new.cID  
 **GROUP BY** cID)  
 =  
 (**SELECT** capacity  
 **FROM** class  
 **WHERE** class.cID = new.cID)  
 **THEN SIGNAL SQLSTATE** '45000'  
 **SET MESSAGE\_TEXT** = 'That class is at full capacity';  
 **END IF**;  
 **END**//  
  
**DROP TRIGGER IF EXISTS** checkClassTimeConflict//  
**CREATE TRIGGER** checkClassTimeConflict  
**BEFORE INSERT ON** class  
**FOR EACH ROW  
 BEGIN  
 IF** (**SELECT** cID  
 **FROM** class  
 **WHERE** rID = new.rID  
 **AND** days **LIKE** *CONCAT*('%', new.days, '%')  
 **AND** ((end\_at >= new.start\_at **AND** start\_at <= new.start\_at)  
 **OR** (end\_at >= new.end\_at **AND** start\_at <= new.end\_at)))  
 **THEN SIGNAL SQLSTATE** '45000'  
 **SET MESSAGE\_TEXT** = 'There is a class scheduled in that room at that time';  
 **END IF**;  
 **END** //  
  
**DELIMITER** ;