Case Study

Due: 12 Jan 2018

Name	**Student ID**
Parteek Dheri	991454985
Ishan Anand	991466218
Nikola Petrovski	991466135

Professor: Dr. Joe Ganczarski

Approach taken: Bottom-up

Case Study

PLEASE READ THE FOLLOWING NOTES CAREFULLY BEFORE YOU START:

- This case study is worth 20% of the course grade, and must be completed in **groups of 1 to 5 by the end of week 13.** You must submit a Word document containing all code and screenshots to the SLATE drop box by the deadline posted on SLATE.
    - **All email submissions will be given the grade of zero.**
    - **No late submissions will be allowed.**
- You must demo your table designs, data and queries to the instructor in class during **week 13 (or 14). No late demos will be allowed.**

In this case study, you and your team members are asked to design and implement the database component of a software system that could be used by a Registrar's Office of a College. The various screens of the systems are shown in the screen mock-ups in the Appendix below.

More specifically, your tasks are:

1. Design all the necessary tables where all the relevant data can be efficiently stored
2. Write all the SQL code to support the application

Pay attention to the following requirements:

- A course might be offered in multiple sections (Note that on the mock-ups, a Section is called a "Course Offering" – They are just synonyms).
- A course (eg DBAS27198) can be offered in multiple sections (eg., 1145_24828, 1145_24123, etc.). Each section is associated with a specific semester (e.g. Fall 2013, Winter 2014), and an instructor.
- Students are registered with a Section, never with a course.

SQL code:

- You must submit the SQL code to create all the tables in proper order using MySQL or MS SQL Server standards (please specify at the beginning of your project which software platform you are using).
- You must write the SQL code to populate the tables in correct order.

- You must also write the SQL queries to pull the data from the various tables into the grid formats shown on the mock-up pages. (Please note that you might have to join or sub-query your tables together to produce the grids exactly as shown.)

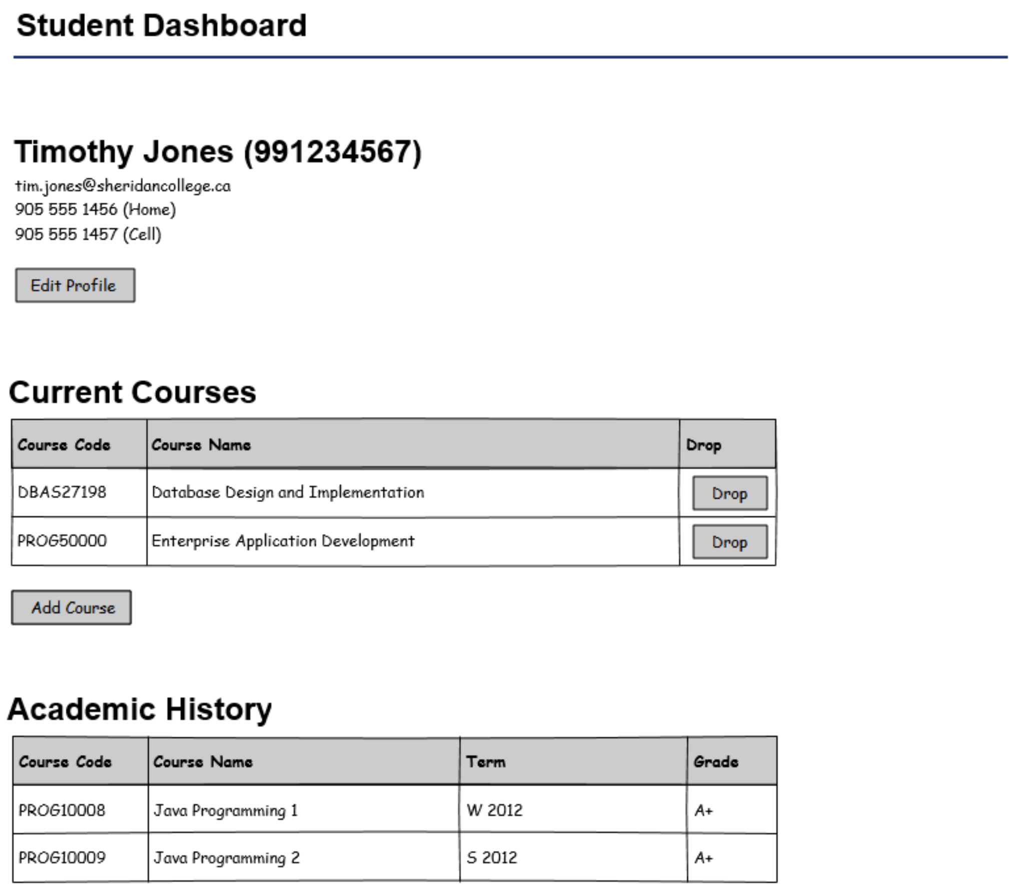
Please feel free to approach the instructor and clarify anything that is unclear about this case study.

Remember to treat this case study as a learning challenge and not just a “burden” that you need to get over with. You will learn a lot of useful skills from this project if you treat it as a learning opportunity.

Have fun!

VIEWS

Appendix: Screen Mockups



Student Dashboard

Timothy Jones (991234567)

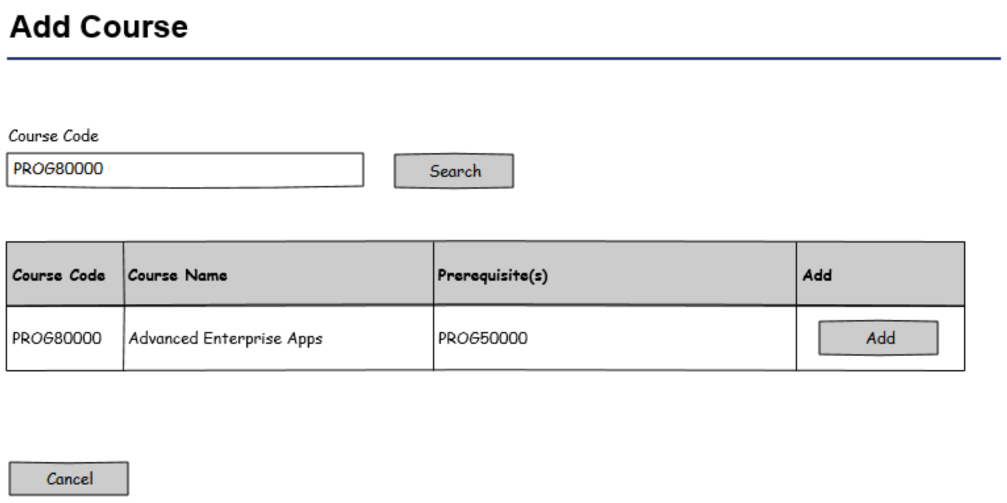
tim.jones@sheridancollege.ca
905 555 1456 (Home)
905 555 1457 (Cell)

Current Courses

Course Code	Course Name	Drop
DBAS27198	Database Design and Implementation	Drop
PROG50000	Enterprise Application Development	Drop

Academic History

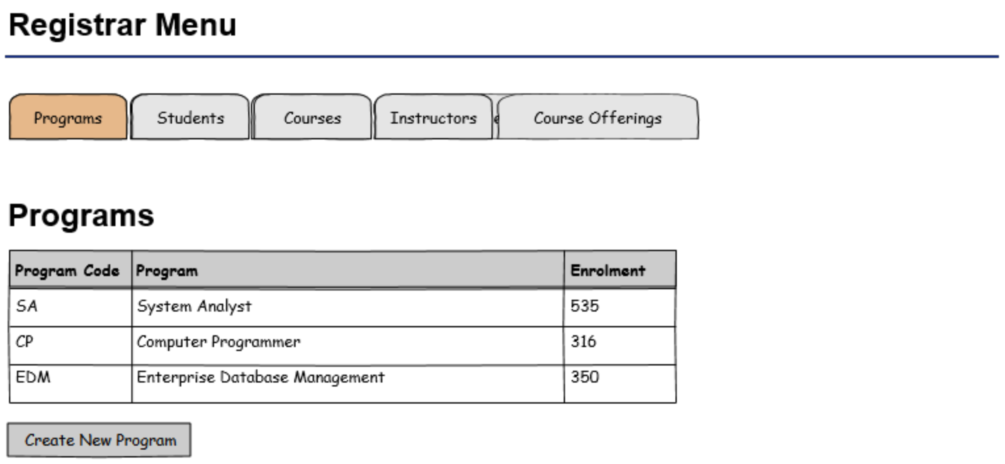
Course Code	Course Name	Term	Grade
PROG10008	Java Programming 1	W 2012	A+
PROG10009	Java Programming 2	S 2012	A+



Add Course

Course Code: PROG80000

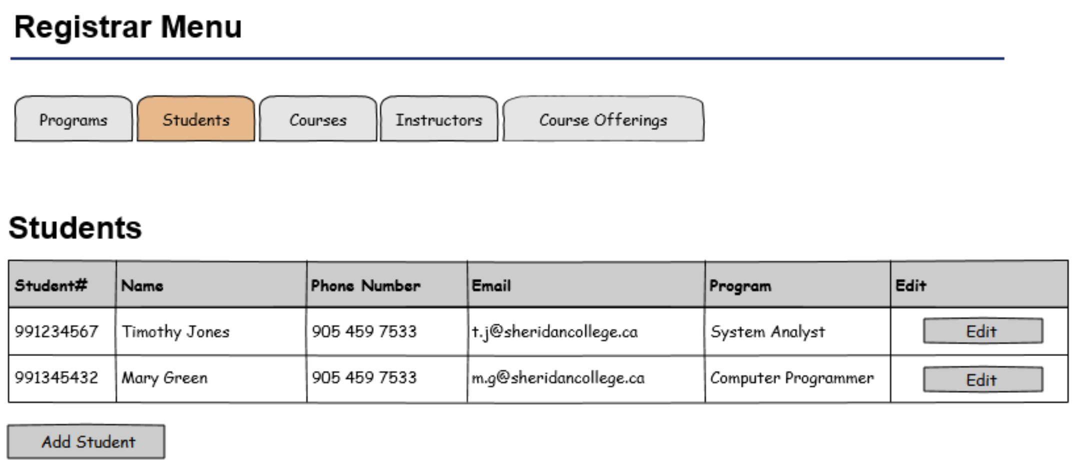
Course Code	Course Name	Prerequisite(s)	Add
PROG80000	Advanced Enterprise Apps	PROG50000	Add



Registrar Menu

Programs

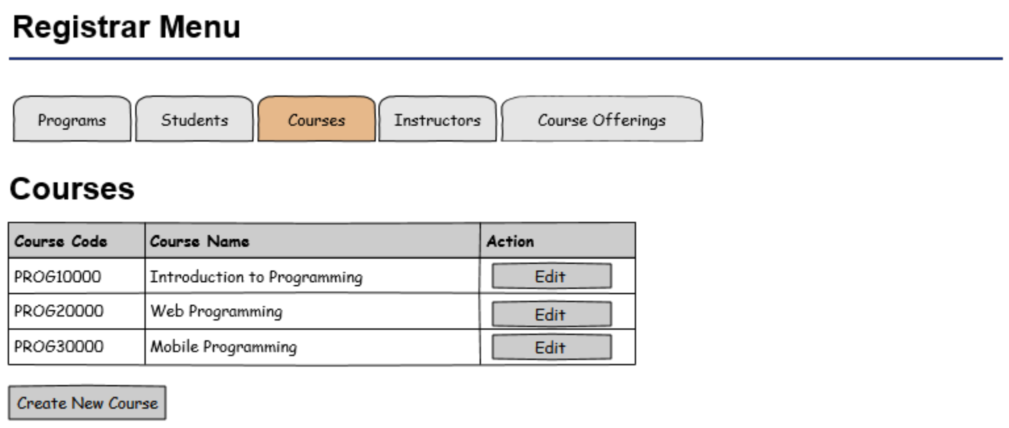
Program Code	Program	Enrolment
SA	System Analyst	535
CP	Computer Programmer	316
EDM	Enterprise Database Management	350



Registrar Menu

Students

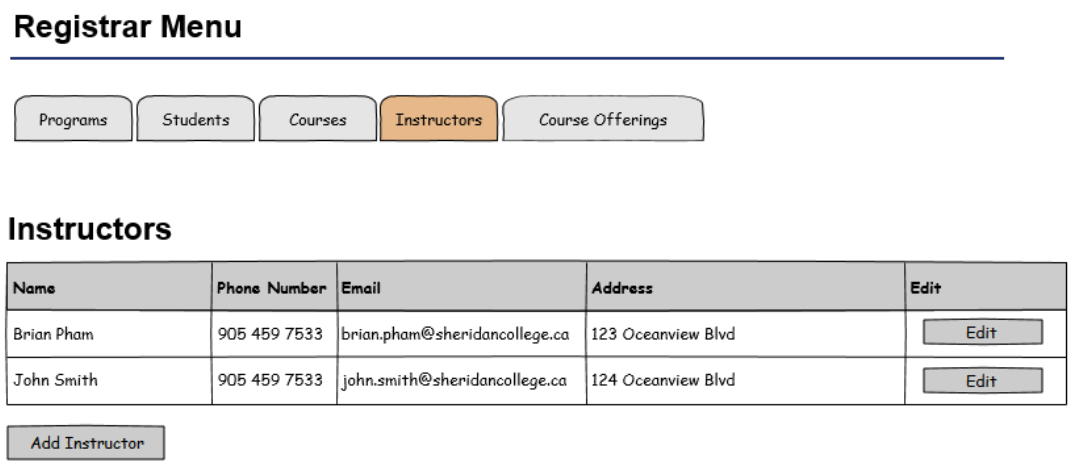
Student#	Name	Phone Number	Email	Program	Edit
991234567	Timothy Jones	905 459 7533	t.j@sheridancollege.ca	System Analyst	Edit
991345432	Mary Green	905 459 7533	m.g@sheridancollege.ca	Computer Programmer	Edit



Registrar Menu

Courses

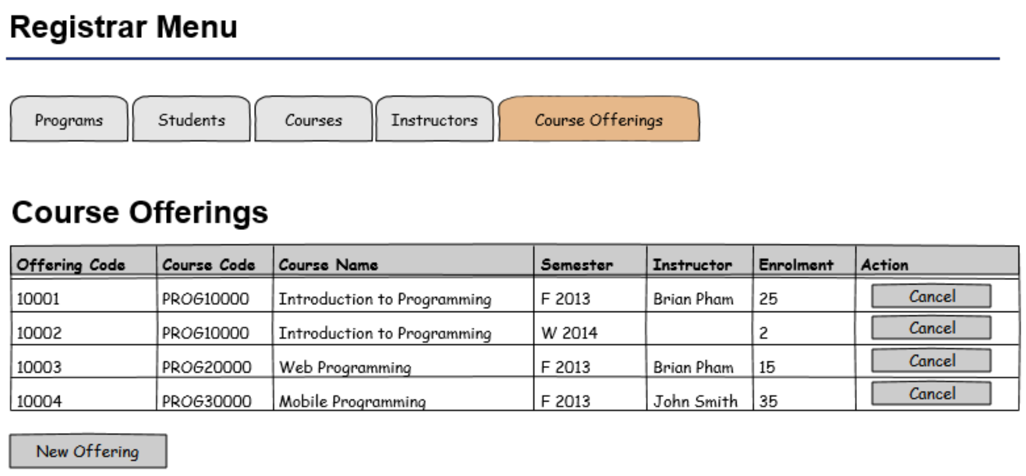
Course Code	Course Name	Action
PROG10000	Introduction to Programming	Edit
PROG20000	Web Programming	Edit
PROG30000	Mobile Programming	Edit



Registrar Menu

Instructors

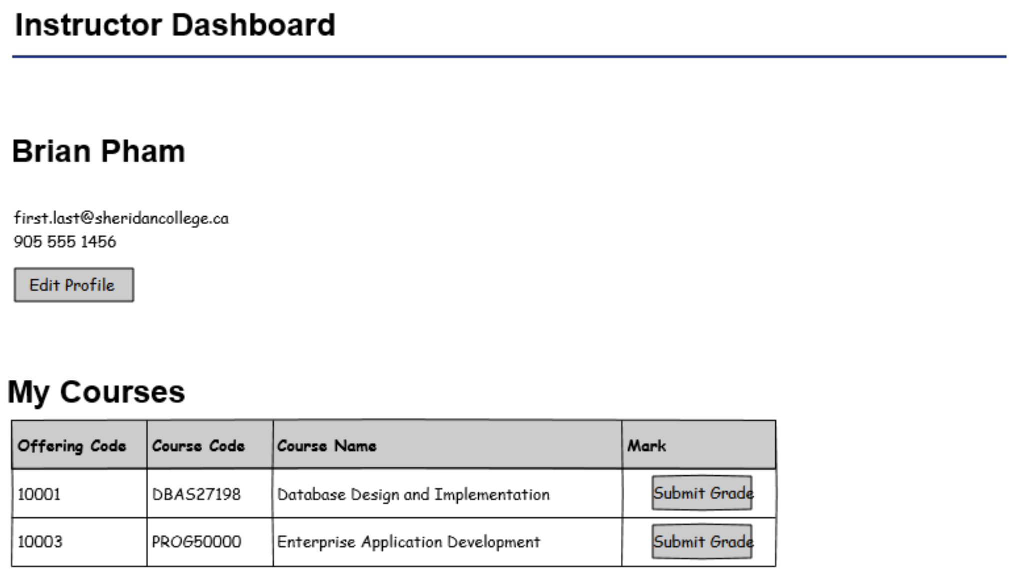
Name	Phone Number	Email	Address	Edit
Brian Pham	905 459 7533	brian.pham@sheridancollege.ca	123 Oceanview Blvd	Edit
John Smith	905 459 7533	john.smith@sheridancollege.ca	124 Oceanview Blvd	Edit



Registrar Menu

Course Offerings

Offering Code	Course Code	Course Name	Semester	Instructor	Enrolment	Action
10001	PROG10000	Introduction to Programming	F 2013	Brian Pham	25	Cancel
10002	PROG10000	Introduction to Programming	W 2014		2	Cancel
10003	PROG20000	Web Programming	F 2013	Brian Pham	15	Cancel
10004	PROG30000	Mobile Programming	F 2013	John Smith	35	Cancel



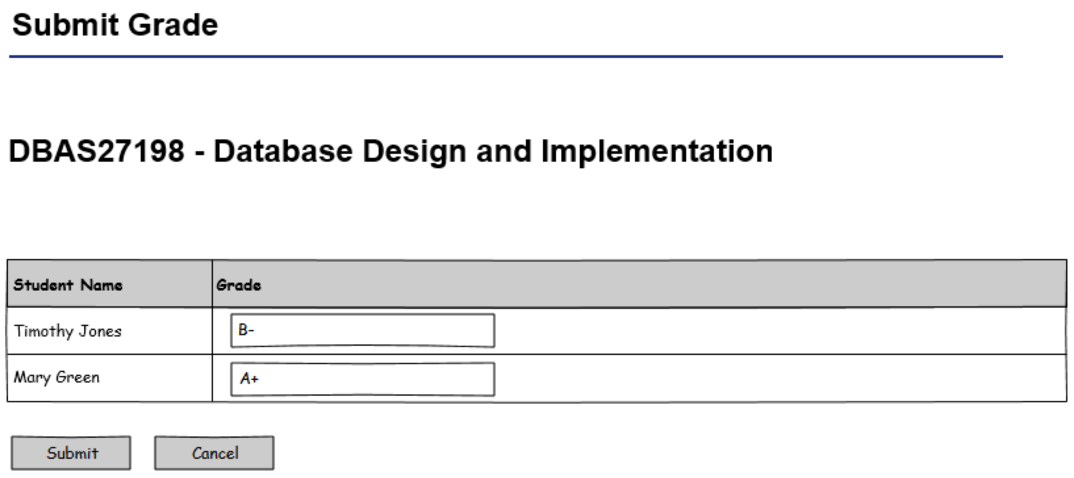
Instructor Dashboard

Brian Pham

first.last@sheridancollege.ca
905 555 1456

My Courses

Offering Code	Course Code	Course Name	Mark
10001	DBAS27198	Database Design and Implementation	Submit Grade
10003	PROG50000	Enterprise Application Development	Submit Grade



Submit Grade

DBAS27198 - Database Design and Implementation

Student Name	Grade
Timothy Jones	B-
Mary Green	A+

Normalisation

0NF

Student (name, stID, email, homePhn, cellPhn,
[courseCode, courseName]
[courseCode, courseName, term, grade])

Course (courseCode, courseName, preq)

RegP(progCode, prog, enrol)

RegS(Student#, name, phn, email, prog)

RegC(courseCode, courseName)

RegI(name, phn, email, address)

RegCO(offCode, courseCode, courseName, sem, instructor, enrolments)

Instructor(name, phn, email,
[offCodce, courseCode, courseName])

SubmitGrade(course, [Sname, grade])

1NF

Student (name, stID, email, homePhn, cellPhn)

StudentCourse (stID, courseCode, courseName)

AcadimicHistory(stID, courseCode, courseName, term, grade)

Course (courseCode, courseName, preq)

RegP(progCode, prog, enrol)

RegS(Student#, name, phn, email, prog)

RegC(courseCode, courseName)

RegI(name, phn, email, address)

RegCO(offCode, courseCode, courseName, sem, instructor, enrolments)

Instructor (name, phn, email)

InstructorCourse (name, offCodce, courseCode, courseName)

SubmitGrade(courseCode, courseName)

CourseGrade (courseCode, Sname, grade)

2NF

Student (name, stID, email, homePhn, cellPhn)

StudentCourse (stID, courseCode)

CourseS(courseCode, courseName)

AcadimicHistory (stID, courseCode, term, grade)

CourseAH (courseCode, courseName)

Course (courseCode, courseName, preq)

RegP(progCode, prog, enrol)

RegS(Student#, name, phn, email, prog)

RegC(courseCode, courseName)

RegI(name, phn, email, address)

RegCO(offCode, courseCode, courseName, sem, instructor, enrolments)

Instructor (name, phn, email)

InstructorCourse (name, offCodce, courseCode, courseName)

SubmitGrade(courseCode, courseName,)

CourseGrade(courseCode, Sname, grade)

3NF

Student (name, stID, email, homePhn, cellPhn)

StudentCourse (stID, courseCode)

CourseS (courseCode, courseName)

AcadimicHistory (stID, courseCode, term, grade)

CourseAH (courseCode, courseName)

Course (courseCode, courseName, preq)

RegP (progCode, prog, enrol)

RegS (Student#, name, phn, email, prog)

RegC (courseCode, courseName)

RegI (name, phn, email, address)

RegCO (offCode, ***courseCode***, sem, instructor, enrolments)

RecCOC (courseCode, courseName)

Instructor (name, phn, email)

InstructorCourse (name, offCodce, courseCode)

InstructorCourseC (courseCode, courseName)

SubmitGrade (courseCode, courseName)

CourseGrade (courseCode, Sname, grade)

Integration

Writing all tables together

Student (name, stID, email, homePhn, cellPhn)

StudentCourse (stID, courseCode)

CourseS(courseCode, courseName)

AcadimicHistory (stID, courseCode, term, grade)

CourseAH (courseCode, courseName)

Course (courseCode, courseName, preq)

RegP (progCode, prog, enrol)

RegS (Student#, name, phn, email, prog)

RegC (courseCode, courseName)

RegI (name, phn, email, address)

RegCO (offCode, ***courseCode***, sem, instructor, enrolments)

RecCOC (courseCode, courseName)

Instructor (name, phn, email)

InstructorCourse (name, offCodce, courseCode)

InstructorCourseC (courseCode, courseName)

SubmitGrade (courseCode, courseName)

CourseGrade (courseCode, Sname, grade)

Removing synonyms and homonyms

Student (Sname, stID, email, homePhn, cellPhn)

StudentCourse (stID, courseCode)

CourseS (courseCode, courseName)

AcadimicHistory (stID, courseCode, sem, grade)

CourseAH (courseCode, courseName)

Course (courseCode, courseName, preq)

RegP(progCode, prog, enrol)

RegS(stID , Sname, cellPhn, email, prog)

RegC(courseCode, courseName)

RegI(instName, instPhn, instEmail, address)

RegCO (offCode, ***courseCode***, sem, instName, enrolments)

RecCOC (courseCode, courseName)

Instructor (instName, instPhn, instEmail)

InstructorCourse (name, offCodce, courseCode)

InstructorCourseC (courseCode, courseName)

SubmitGrade (courseCode, courseName)

CourseGrade (courseCode, Sname, grade)

Join relations with same primary key.

Student (Sname, stID, email, homePhn, cellPhn, ***prog***)

StudentCourse (stID, courseCode, sem, grade)

Course (courseCode, courseName, preq)

RegP (progCode, prog, enrol)

RegCO (offCode, ***courseCode***, sem, ***instName***, enrolments)

Instructor (instName, instPhn, instEmail, address)

CourseGrade (courseCode, Sname, grade)

New foreign keys

Student (Sname, stID, email, homePhn, cellPhn, ***prog***)

StudentCourse (stID, courseCode, sem, grade)

Course (courseCode, courseName, preq)

RegP (progCode, prog, enrol)

RegCO (offCode, ***courseCode***, sem, ***instName***, enrolments)

Instructor (instName, instPhn, instEmail, address)

CourseGrade (courseCode, Sname, grade)

Remove unneeded (derived) fields

Student (Sname, stID, email, homePhn, cellPhn, ***prog***)

StudentCourse (stID, courseCode, sem, grade)

Course (courseCode, courseName, preq)

RegP (progCode, prog)

RegCO (offCode, ***courseCode***, sem, ***instName***)

Instructor (instName, instPhn, instEmail, address)

CourseGrade (courseCode, Sname, grade)

Remove unneeded relation

Student (Sname, stID, email, homePhn, cellPhn, ***prog***)

StudentCourse (stID, courseCode, sem, grade)

Course (courseCode, courseName, preq)

RegP (progCode, prog)

RegCO (offCode, ***courseCode***, sem, ***instName***)

Instructor (instName, instPhn, instEmail, address)

CourseGrade (courseCode, Sname, grade)

Improved table names

Student (Sname, stID, email, homePhn, cellPhn, ***prog***)

StudentCourse (stID, courseCode, sem, grade)

Course (courseCode, courseName, preq)

Program (progCode, prog)

CourseOffering (offCode, ***courseCode***, sem, ***instName***)

Instructor (instName, instPhn, instEmail, address)

CourseGrade (courseCode, Sname, grade)

Extra steps:

1. Changed primary key of Student course as course code from course offering is to be picked
2. Same done with CourseGrade Table and Sname changed to StID as they refer to same entity by different attributes.
3. Changed primary key of Program and foreign key of Student table

Student (Sname, stID, email, homePhn, cellPhn, ***progCode***)
StudentCourse (stID, offCode, sem, grade)
Course (courseCode, courseName, preq)
Program (progCode, prog)
CourseOffering (offCode, ***courseCode***, sem, ***instName***)
Instructor (instName, instPhn, instEmail, address)
CourseGrade (offCode, stID, grade)

Merging tables with same primary key

Student (Sname, stID, email, homePhn, cellPhn, ***progCode***)

StudentCourse (stID, offCode, sem, grade)

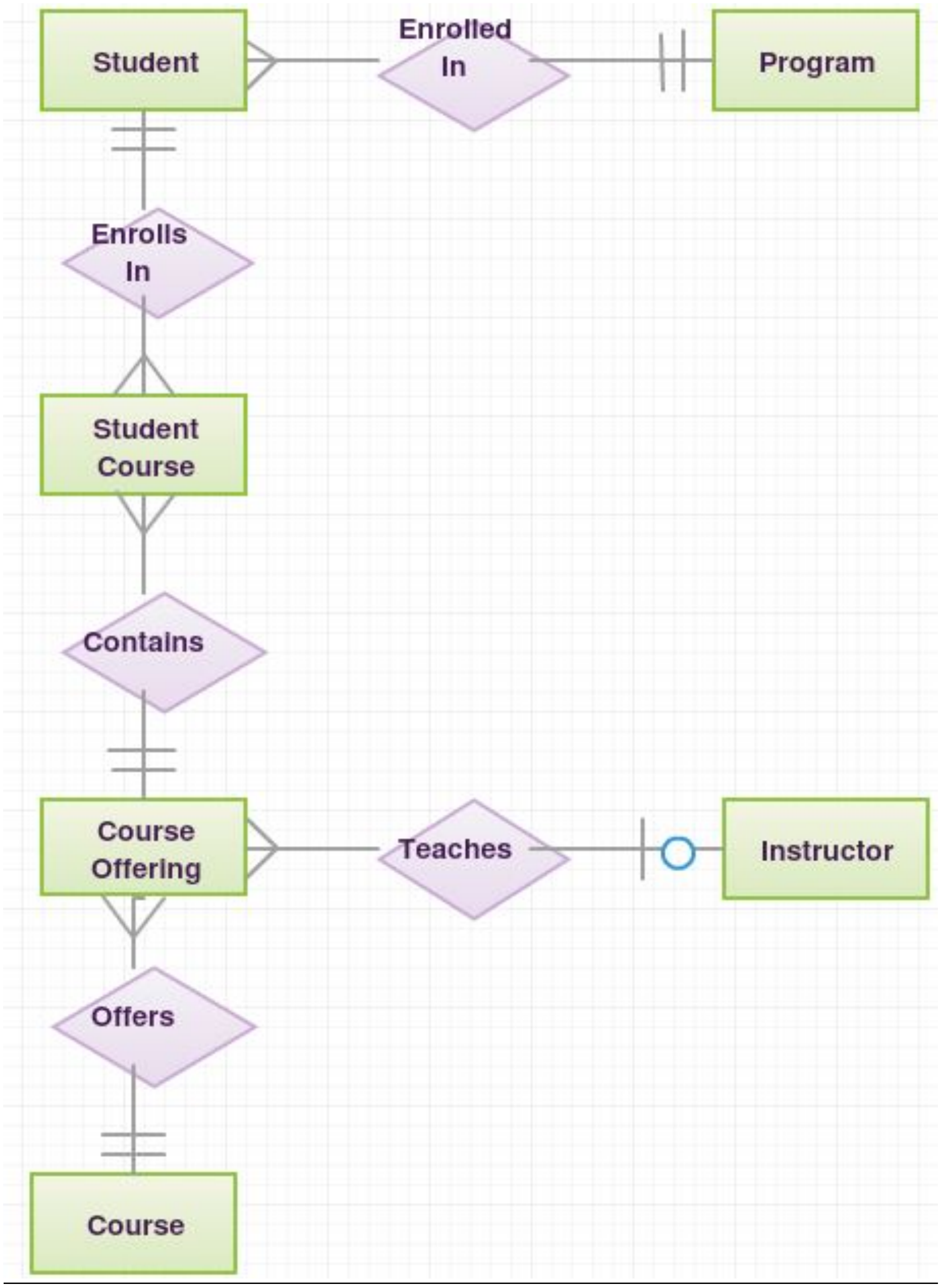
Course (courseCode, courseName, preq)

Program (progCode, prog)

CourseOffering (offCode, ***courseCode***, sem, ***instName***)

Instructor (instName, instPhn, instEmail, address)

<u>ER Diagram</u>


Student
Enrolled In
Program
Enrolls In
Student Course
Contains
Course Offering
Teaches
Instructor
Offers
Course

MySQL Script

Creating database and tables.

```
DROP DATABASE IF EXISTS CaseStudy;
CREATE DATABASE CaseStudy;

USE CaseStudy;

DROP TABLE IF EXISTS  StudentCourse;
DROP TABLE IF EXISTS  CourseOffering;
DROP TABLE IF EXISTS  Course;
DROP TABLE IF EXISTS  Instructor;
DROP TABLE IF EXISTS  Student;
DROP TABLE IF EXISTS  Program;


CREATE TABLE Program(
                        progCode varchar(50),
                        prog varchar(80),
                        PRIMARY KEY (progCode)
);

CREATE TABLE Student(
                        stID VARCHAR(15) NOT NULL,
                        Sname varchar(50),
                        email varchar(80),
                        homePhn VARCHAR(15),
                        cellPhn VARCHAR(15),
                        progCode varchar(30),
                        PRIMARY KEY (stID),
   FOREIGN KEY (progCode) REFERENCES Program(progCode)
);

CREATE TABLE Instructor (
                        instName varchar(55),
   instPhn VARCHAR(15),
   instEmail varchar(55),
   address varchar(222),
   PRIMARY KEY (instName)

);
```

```
CREATE TABLE Course (
                              courseCode varchar(22),
   courseName varchar(50),
   preq varchar(50),
   PRIMARY KEY (courseCode)
);

CREATE TABLE CourseOffering (

                              offCode varchar(50),
                              courseCode varchar(55),
                              sem varchar(30),
                              instName varchar(30),
                              PRIMARY KEY (offCode),
   FOREIGN KEY (courseCode) REFERENCES Course(courseCode),
   FOREIGN KEY (instName) REFERENCES Instructor(instName)
);

CREATE TABLE StudentCourse (
                              stID VARCHAR(15),
                              offCode varchar(55),
                              sem varchar(30),
                              grade varchar(10),
                              PRIMARY KEY (stID, offCode)
);
```

Inserting records

```
INSERT INTO Program
VALUES ("SA","System Analyst");

INSERT INTO Program
VALUES ("CP","Computer Programmer");

INSERT INTO Program
VALUES ("EDM","Enterprise Database Management");


INSERT INTO Student
VALUES (991234567,"Timothy Jones","t.j@sheridancollege.ca","9054597533", "9054597533","SA");

INSERT INTO Student
VALUES (991454685,"Parteek Dheri","dherip@sheridancollege.ca","1111111111", "2222222222","SA");

INSERT INTO Student
VALUES (991111222,"Joe Ganczarski","joeg@sheridancollege.ca","1111111111", "2222222222","EDM");


INSERT INTO Student
VALUES (991345432,"Marry Green","m.g@sheridancollege.ca","9054597533", "9054597533","CP");



INSERT INTO Instructor
VALUES ("Brian Pham","9054597533","brian.pham@sheridancolelge.ca", "123 Oceanview Blvd");

INSERT INTO Instructor
VALUES ("John Smith","9054597533","john.smith@sheridancolelge.ca", "124 Oceanview Blvd");



INSERT INTO Course
VALUES ("DBAS27198","Database Design And Implementation", null);

INSERT INTO Course
VALUES ("PROG50000","Enterprise Application Development", "PROG10000");

INSERT INTO Course
VALUES ("PROG10000","Introduction to programming", null);

INSERT INTO Course
VALUES ("PROG10008","Java programming 1", null);

INSERT INTO Course
VALUES ("PROG10009","Java programming 2", "PROG10008");
```

```
INSERT INTO Course
VALUES ("PROG20000","Web Programming", null);

INSERT INTO Course
VALUES ("PROG30000","Mobile Programming", null);

INSERT INTO Course
VALUES ("PROG80000","Advanced Enterprise Apps", "PROG50000");



INSERT INTO CourseOffering
VALUES ("10001","DBAS27198","F 2013", "Brian Pham");

INSERT INTO CourseOffering
VALUES ("10002","PROG10000","W 2014", null);

INSERT INTO CourseOffering
VALUES ("10003","PROG50000","F 2013", "Brian Pham");

INSERT INTO CourseOffering
VALUES ("10004","PROG30000","F 2013", "John Smith");

INSERT INTO CourseOffering
VALUES ("10005","PROG30000","F 2013", "John Smith");

INSERT INTO CourseOffering
VALUES ("10006","PROG50000","F 2013", "John Smith");

INSERT INTO CourseOffering
VALUES ("10009","PROG10008","F 2013", "John Smith");

INSERT INTO CourseOffering
VALUES ("10010","PROG10009","F 2013", "John Smith");



INSERT INTO StudentCourse
VALUES ("991234567","10005","F 2017", null);

INSERT INTO StudentCourse
VALUES ("991454685","10005","F 2017", null);

INSERT INTO StudentCourse
VALUES ("991234567","10006","F 2017", null);

INSERT INTO StudentCourse
VALUES ("991234567","10009","W 2012", "A+");

INSERT INTO StudentCourse
VALUES ("991234567","10010","S 2012", "A+");
```

Creating views

```
SELECT Sname, email, homePhn, cellPhn from student
WHERE stID = 991234567;

-- current courses for timmy
SELECT * from studentcourse
WHERE stID = 991234567
AND sem ="F 2017";  -- F 2017 is current sem


-- course history for timmy
SELECT * from studentcourse
WHERE stID = 991234567
AND sem not in ("F 2017");


-- add course view
select * from course
where courseCode = "PROG80000";

-- Programs
select p.progCode, p.prog , count(s.progCode) as 'enrolements'
from program p join student s
on p.progCode=s.progCode
group by s.progCode;

-- Students
select s.stID, s.Sname, s.cellPhn, s.email, p.prog from  student s
join program p
on p.progCode=s.progCode;

-- Courses
select c.courseCode, c.courseName from course c;

-- Instructors
select * from Instructor i;

-- CourseOffering
select co.offCode, co.courseCode, c.courseName, co.sem, co.instName, count(sc.offCode) as 'Enrolments'
from courseoffering co
right join course c
on co.courseCode=c.courseCode
left join studentcourse sc
on co.offCode=sc.offCode
group by sc.offCode;

-- instructor dashboard
select instName, instEmail, instPhn from instructor;

-- courses for Brian
```

```
select co.offCode, co.courseCode, c.courseName from courseoffering co
join course c
on c.courseCode=co.courseCode
where instName='Brian Pham';

-- grade
select s.Sname, sc.grade from studentcourse sc
join student s
on s.stID=sc.stID;
```