Online Course Registration System

Project Report

Tolstoy Newtonraja  
Computer and Information Science & Eng  
tolstoynewton@ufl.edu  
UF-ID: 3411-3151

Aditya Nafde  
Computer and Information Science & Eng  
adityan@ufl.edu  
UF-ID: 5159-7503

1. Subject and Purpose

The need for structured storage, modification and maintenance of huge amounts of data has resulted in the emergence of the Database Management System (DBMS) as one of the core fields in the Computer Science industry. DBMS is the system of computer software that is aimed to provide a managing tool for maintaining the data, through various data models.

The purpose of implementing this project is to understand the data modeling concepts that is used in a real time scenario and to implement a fully functional database system which interacts with a front end interface.

2. Definition of the Problem

An Online Course Registration system for University of Dataville is to be developed with a front-end web interface and a back-end database. An example of the system would be University of Florida’s ISIS Registration.

Any database system can be chosen as the back-end such as Oracle, MySQL, Microsoft SQL Server, DB2, Access. Any web server can be chosen for the front end such as Tomcat, Glassfish, JRun, etc. Any server side language can be chosen such as PHP, JSP, ASP, etc.

3. Project Specification

A high level description of the project:

- Each student has a unique student ID and a profile. The profile includes first/last names, gender, date of birth, local address, department, enrolled year, username, login password, and may have a picture. You can also add other necessary information.

- Each instructor has a unique faculty ID and a profile. The profile must indicate the instructor’s department(s). An instructor may work at more than one department.
4. Need for Solution

The practical implementation of a real time database management system made us understand the intricacies involved in designing the database schema with inter related constraints and the complexity that exists in integrating the front end system with the back end data-store.

This project gave us a hands-on experience on developing the entire project using MySQL, and PHP using the Apache Server. We learnt several ways to optimize when designing the database and made quick and rational decisions when developing the project by implementing the procedures taught in class.

We could clearly indentify the balance we had to maintain when choosing between efficiency and performance.
5. ER Diagram

![ER Diagram]

6. Design Schema

```sql
CREATE TABLE Student (
    studentId int PRIMARY KEY,
    firstName varchar(30) NOT NULL,
    lastName varchar(30) NOT NULL,
    profilePic blob,
    gender varchar(1),
    dob date,
    address varchar(200),
    deptId int,
    homepage varchar(70) DEFAULT null,
    yearEnrolled varchar(15) NOT NULL,
    overallGPA float (3,2),
    foreign key (deptId) references Department(dept_id)
);```
Create table Login
  (username varchar(20) not null UNIQUE,
   password varchar(20) not null,
   type varchar(1) not null,
   studInstId int not null AUTO_INCREMENT Primary Key
 );

Create table Instructor(
  instId int Primary Key ,
  firstName varchar(30) not null,
  lastName varchar(30) not null,
  profilepic blob,
  gender varchar(1) not null,
  dob date not null,
  address varchar(200) not null,
  homepage varchar(70) default null,
  yearEnrolled varchar(15) not null
 )

Create table Department ( 
  deptId int primary key,
  deptName varchar(50) not null
 )

Create table Course ( 
  courseId int,
  courseName varchar(50),
  term varchar(15) ,
  deptId int,
  credits int not null,
  textbook varchar(50) default null,
  refTextbook varchar(50) default null,
  courseLink varchar(50) default null,
  primary key (courseId, term),
  foreign key (deptId) references Department(dept_id),
 );

Create table Prerequisites ( 
  courseId varchar(15),
  prereqId varchar(15),
  foreign key (courseId) references Course(courseId),
  foreign key (prereqId) references Course(courseId)
 );

Create table Coursestaken ( 
  studentId int ,
  courseId varchar(10),
  term varchar(15),
  grade varchar(1) default null ,
  foreign key (studentId) references Student,
  foreign key (courseId) references Course,
  foreign key (term) references Course(term)
Create table InstructorDept (  
    instId int,  
    deptId int,  
    foreign key (instId) references Instructor,  
    foreign key (deptId) references Department  
);  

Create table InstructorCourses (  
    instId int,  
    courseId varchar(15),  
    term varchar(15),  
    foreign key (instId) references Instructor,  
    foreign key (courseId) references Courses,  
    foreign key (term) references Course(term)  
);  

Create table Classroom (  
    classId varchar(10) primary key,  
    location varchar(30) not null,  
    maxCapacity int not null,  
    seatsLeft int not null  
);  

Create table ClassCourse (  
    classId int not null,  
    courseId varchar(15) not null,  
    period int not null,  
    day varchar(1) not null,  
    term varchar(15) not null,  
    foreign key (classId) references Classroom(classId),  
    foreign key (courseId) references Course(courseId),  
    foreign key (term) references Course(term)  
);  

7. Major Design Decisions  

Major design decisions in the project were discussed and entered in a shared online Google document so that as and when someone thinks of a new idea for bonus functionality we were able to add in the document and share it with the other person (2 in a team).  

The design schema and other field information were also stored and work performed on a day-by-day basis was logged so that we were aware on the progress of the project.  

Some design challenges:  

We used PNG pictures for storing the profile picture information and found that the storage space went to big numbers. We then changed the type of data and made it as a jpeg picture.
We missed the *term* field in the classroom table initially and had to add the field later to keep track of the classroom information for previous semesters.

We had not thought about the *graduation* date field for a student. When trying to implement a realistic system this field had to be in place to segregate the graduated students and other current students.

We had to have an ADMIN login to perform some administrative tasks. We thought it would be very practical to have this option. For example, when you are logged in as an admin you can edit the *graduated* field for a student. We tried to map the real scenario where the student requests that he wants to graduate to the *Grad Advisor* in his department and the *Advisor* requests *DB Administrators* to update the student’s `graduated` field.

**Functionalities a student can perform in the system:**

- Can edit his profile data i.e. address/homepage/email id information and update.
- Register a course
  - Constraints checked here include
    - MAX courses registered this semester, in our system its 3,
    - Course previously not registered,
    - Seats lefts in the course,
    - Prerequisite course registered and
    - Conflicts in timing with other currently registered courses.
- View his schedule
  - Can view class schedule in a tabular format (similar to ISIS)
  - Drop course option is provided on this page
  - Can view additional information about the courses he has taken.
- Can view textbook information of the courses department wise.
- Can view courses with prerequisite information
- Can view the list of courses using the department and instructor parameters.
- Can view the list of courses using the day and period parameters.
- Can view courses previously taken along with grades information
- Can also request for graduation, we have enforced a minimum of 10 courses for a student to graduate. If this request is not met the graduation request is denied else it is forwarded to the graduate advisor.

**Functionalities an Instructor can implement in the system:**
- Can view / edit profile information and also update.

- Can view current schedule in a tabular format (Similar to ISIS)

- Can edit Course details - Details such as Seats left - to increase or decrease the capacity of the course (seats cannot be more than class capacity)

- Can add a new course to the database under his profile and department.

- The input parameters in the form are checked for valid input data before updating the database.

- Conflicts with the instructor's schedule as well as conflicts in timing with other courses in the same classroom are checked.

- Can view classroom availability as well as maximum capacity of classroom when course information is updated or new classroom is added

- Instructor can view info of all the students registered in the courses he/she is handling.

8. Programming practices used

As mentioned earlier we used Google Documents to share critical information about the project. The project status, issues faced and new functionalities were shared concurrently and updated hour by hour which maximized our work performance when trying to build the project from different places. We strongly believed in dividing the tasks and building the project instead of the conventional pair programming method.

Loading Real time Data:

Any rich web system should have lots of data to work on and implement all the functionalities. With all the functionalities built and with less data the system would be a boring application to work on. We wanted to make the application rich and interesting to the person who wanted to use it. Our first challenge (apart from designing schemas and building the tables) was to load the database with lots of information. How?

We coded a python script (attached to the appendix) which was used to parse a html file from the internet and extract the needed information. The CISE student and instructor pages were scanned and information was retrieved. The next step was to code the functions in the python code to generate random values for fields we didn’t have in the html file.
For example:

Randomized functions - Birthdays, homepages, overallGPA, email Id

Example code for populating the department table:

```python
def PopulateDepartmentTable():
    ''' Read text file and generates SQL command '''
    path = "deptNames\inDeptNames.txt";
    fileContents = file ( path ).read();
    fileSQLcode = file ('deptNames\outSQLDeptNames.txt', 'w')
    #INSERT INTO Department VALUES (1, "Computer Science");

    count = 1
    for deptName in fileContents.split('n'):
        #print deptName
        #INSERT INTO Department VALUES (1, "Computer Science");

        #Enters data into 'Login table' and 'Instructor'
        print 'INSERT INTO Department(deptId, deptName) VALUES ("' + 'count' + '"',' + deptName + '");'
        print >>fileSQLcode, 'INSERT INTO Department(deptId, deptName)
VALUES ("' + 'count' + '"',' + deptName + '");'
        count = count + 1

    fileSQLcode.close()
```

Some quick stats on the database:

Database name: coursereg

Database size: 5.26 MB

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Rows</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>863</td>
<td>160.9 KB</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>27</td>
<td>3.2 KB</td>
</tr>
<tr>
<td>Login</td>
<td>1022</td>
<td>76.9 KB</td>
</tr>
<tr>
<td>Instructordept</td>
<td>221</td>
<td>10.9 KB</td>
</tr>
<tr>
<td>Instructorcourses</td>
<td>221</td>
<td>16 KB</td>
</tr>
<tr>
<td>Instructor</td>
<td>159</td>
<td>31.8 KB</td>
</tr>
<tr>
<td>Department</td>
<td>5</td>
<td>3.3 KB</td>
</tr>
<tr>
<td>Coursestaken</td>
<td>7210</td>
<td>381.1 KB</td>
</tr>
<tr>
<td>Course</td>
<td>225</td>
<td>28.4 KB</td>
</tr>
<tr>
<td>Classroom</td>
<td>100</td>
<td>4.3 KB</td>
</tr>
<tr>
<td>Classcourse</td>
<td>675</td>
<td>38.5 KB</td>
</tr>
</tbody>
</table>
Use of Comments:

Any new functions or new logic added to the PHP file was very difficult to comprehend without the comments. Hence lots of comments were added to the code to make it easy for the other team member to comprehend and extend the logic of the code.

Security:

We used a front-end software for modifying and executing the mySQL commands (similar to a Microsoft SQL Server interface). The name of the software is Heidi and its open source. We set encryption and locked the database when not in use. All the queries executed using PHP were first tested in this software and results were verified.

Backups:

We made it a point to save backups of the entire PHP collection files and DATABASE everyday with the timestamp. We went back to previous versions of the code whenever the current code got unstable or we were faced with unexpected weird results.

Usability and User Interfaces:

All the images, header banners were designed in Adobe Photoshop 9 and a basic template (CSS) was used for all the pages.

9. Scope

The main aim of the project is to learn the intricacies of modeling the data base with the given requirements and using a web based interface to interact with the back end keeping in mind the data consistency and the stability of the entire system.

Since the system is developed with a web based interface, we can start executing the model with parallel executions with the confidence that the back end MySQL data store will take care of the concurrent transactions. The ACID (Atomicity, Consistency, Isolation, Durability) property of the database helps to keep the system consistent and stable.

10. Implementation Methods

The Online Course Registration project was implemented successfully using the Apache Server with MySQL running as the back end database and PHP used as the server side language. The implementation methods were chosen carefully taking into consideration the possibilities of hosting this project online in a server after the semester. Several high profile hosting companies support mySQL and PHP. Comparisons can be drawn later based on performance, execution time, efficiency, memory usage by implementing the system in different servers with various traffic load.
11. Breakdown of tasks

The following table indicates the task distribution among the team members.

<table>
<thead>
<tr>
<th>Task</th>
<th>Team Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehensive study of the system</td>
<td>Aditya, Tolstoy</td>
</tr>
<tr>
<td>Modeling the database relational schema</td>
<td>Aditya, Tolstoy</td>
</tr>
<tr>
<td>Basic Testing and Optimizing relations and tables</td>
<td>Aditya</td>
</tr>
<tr>
<td>Loading Real-time data entries in the database using Python</td>
<td>Tolstoy</td>
</tr>
<tr>
<td>Front-end template design using CSS</td>
<td>Aditya</td>
</tr>
<tr>
<td>User Interface design with Adobe Photoshop</td>
<td>Tolstoy</td>
</tr>
<tr>
<td>Break down of coding and analyzing requirements</td>
<td>Aditya, Tolstoy</td>
</tr>
<tr>
<td>Coding student related functionalities</td>
<td>Aditya</td>
</tr>
<tr>
<td>Coding instructor related functionalities</td>
<td>Tolstoy</td>
</tr>
<tr>
<td>Testing &amp; Optimization of code</td>
<td>Aditya, Tolstoy</td>
</tr>
<tr>
<td>Project Documentation</td>
<td>Aditya, Tolstoy</td>
</tr>
</tbody>
</table>

12. Facilities used

Package used: WAMPSERVER 2.0.
WAMP stands for Windows,
Apache 2.2.11,
MySQL 5.1.33
PHP 5.2.9

Operating System: Windows7 64 Bit Processor
Processor Speed: Core 2 Duo 2.4 Ghz Processor
Memory available: 4 GB, 256 MB Nvidia Graphics card
Hard Disk capacity: 250 GB
Database size: 5.26 MB
Front end code: 5 MB
### 13. Time and Work Schedule

<table>
<thead>
<tr>
<th>Timeline Tasks</th>
<th>Week1</th>
<th>Week2</th>
<th>Week3</th>
<th>Week4</th>
<th>Week5</th>
<th>Week6</th>
<th>Week7</th>
<th>Week8</th>
<th>Week9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehensive study of the system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modeling the database relational schema</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic Testing and Optimizing relations and tables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loading Real-time data entries in the database using Python scripts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Front-end template design using CSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface design with Adobe Photoshop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Break down of coding and analyzing requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding student-related functionalities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding instructor-related functionalities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing &amp; Optimization of code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Deadlines Met:
- Phase 1 Deadline – Nov 10th
- Phase 2 Deadline – Dec 17th
14. Screen shots and links explained

Index.php -> This is the index page, any new user is directed to this page. The page simply asks for username and password from the user and depending on the user type – student, instructor or administrator, the user is directed to appropriate homepage.

Fig: Index page

The screen shot of instructor’s home page is shown below and also the side links are explained
Studenthome.php, instructorhome.php, adminhome.php – Home pages for students, instructors and administrators respectively.

editProfile.php, updateProfile.php – PHP files for editing and updating profile information for students and instructors respectively.

**My schedule** link in both instructors and students page gives them their current schedule displayed in a tabular format similar to ISIS system. Students can also drop their courses via this page. Detailed information about the courses is available via this link. Instructors can edit their course information through this page.
Instructor can add a new course via **Add New Course** link on the left menu. The PHP files are courseinfo.php and add_new_course.php. Instructors must provide course name as a valid input. The required conflict constraints are also check once the form is submitted by the system.

Instructor can also view information about class capacity and current class schedule as follows:

![Class Schedule](image)

Thus the instructors can see that the class capacity of class ECE14 is 15 and the ‘green’ slots indicate available slots for the selected class.

Instructors can also view students registered in their courses for the current term via **Registered Students** link. This gives extra information about the students registered for his/her courses.

The screen shot of the student home page is as follows:
Students can register for courses via ‘Register courses’ link. All the specified conditions are checked while validating the request from the user to register the course. If all the conditions are met then course registration is allowed to be proceeded.

Students can view their schedule of courses in tabular format similar to ISIS. The below screen shot depicts one.
Thus in the tabular format the student can see his schedule and also option to drop courses is provided on this page. This page gives clear idea about course schedule to the students.

Through links on the left menu students can search for courses instructor wise and schedule wise as well. Students can also check for course prerequisites of the courses as well as the textbooks. The php files -> textbookinfo.php, instinfo.php, cbyschedule.php, prerequisite.php etc

Previously taken courses and grades received information can be accessed by clicking the My Grade option.

We have also provided a link ‘Graduate from’ from where a student can sent request for graduation to the graduate advisor. The system checks for minimum course requirements (10 in our case) and after that the request is forwarded to the graduate advisor.
15. Future Modifications

This project was developed keeping in mind the functionalities for students and instructors. This project provides all the functionalities specified in the project document. We tried to simulate the real world environment by generating loads of data from the University of Florida departments itself.

Much functionality can be added over the existing one so that the system has more enhanced features and is more efficient.

Classroom/Dept Info on Maps:

Students can view the classrooms, departments and other related information on campus map and thus will know the classroom locations more clearly. Google Map API can be used and data can be easily mapped on top of the map.

More Statistics for Grades:

Students can know about the distribution of grades and other statistics such as assignment scores, project scores in that course. Thus students can be aware of the grading system followed by a particular instructor.

Administrators:

We can add a third kind of user besides students and instructors called administrators to the system. They can have a clearer view of underlying databases. They can add a new user to the system, update status of existing users, can generate reports from time to time check various statistics about the system and user behavior. Statistics results can be used by Database administrators to tweak the underlying database at the lowest level to improve the performance and efficiency of the system. The administrator can also do time to time backup of the databases in the system.

Alumni’s specific pages:

A different view of the system can be created for the students who have graduated from the college. They can only see the courses offered by each department but will not be allowed to register for any courses. Additional functionalities include request for duplicate degree / transcripts etc.

Encryption:

Effective encryption schemes can be used to improve the security of the system. Password recovery facility can be provided for effective functioning of system and ease of use by users.
16. References


Appendix

Python Source code

(Loading data into the tables)

```python
# !C:\Programs\Python25\Python.exe
# -*- coding: cp1252 -*-

import random

def findDOB(start, end):
    date = random.randint(1,31)
    month = random.randint(1,12)
    year = random.randint (start, end)

    if (month < 10):
        month = '0' + `month`
    else:
        month = `month`

    if (date < 10):
        date= '0' + `date`
    else:
        date = `date`

dob = `year` + '-' + month + '-' + date
    return dob

def findGender():
    randNo = random.randint(1,20)
    if (randNo % 2 == 0):
        return 'M'
    else:
        return 'F'

def findYearEnrolled(start, end):
    randNo = random.randint (1,3)

    if (randNo == 1):
        yearEnrolled = 'Spring ' + `random.randint (start, end)`
    elif (randNo == 2):
        yearEnrolled = 'Summer ' + `random.randint (start, end)`
    elif (randNo == 3):
        yearEnrolled = 'Fall ' + `random.randint (start, end)`
```
return yearEnrolled

def PopulateDepartmentTable():
    ''' Read text file and generates SQL command '''
    path = "deptNames\inDeptNames.txt";
    fileContents = file ( path ).read()
    fileSQLcode = file ('deptNames\outSQLDeptNames.txt','w')
    #INSERT INTO Department VALUES (1, "Computer Science");
    count = 1
    for deptName in fileContents.split(\'
\n\'):  #print deptName
        #INSERT INTO Department VALUES (1, "Computer Science");
        #Enters data into `Login table` and `Instructor`
        print 'INSERT INTO Department(deptId, deptName) VALUES ("' + `count` + "," + deptName + ");'
        print >>fileSQLcode, 'INSERT INTO Department(deptId, deptName) VALUES ("' + `count` + "," + deptName + ");'
        count = count + 1
    fileSQLcode.close()

def PopulateCourseTable(deptNoParam):
    '''
    Read text file and generates SQL command
    Fills TABLES: course
    '''
    if (deptNoParam == 1):
        path = "course\inCce.txt";
        fileSQLcode = file ('course\out_Course_Cce.txt','w')
        deptId = 1 # 1 Cce, 2 Cise, 3 Ece, 4 Ise, 5 Mae
courseId = 999
    elif (deptNoParam == 2):
        path = "course\inCise.txt";
        fileSQLcode = file ('course\out_Course_Cise.txt','w')
        deptId = 2
courseId = 1999
    elif (deptNoParam == 3):
        path = "course\inEce.txt";
        fileSQLcode = file ('course\out_Course_Ece.txt','w')
        deptId = 3
courseId = 2999
    elif (deptNoParam == 4):
        path = "course\inIse.txt";
        fileSQLcode = file ('course\out_Course_Ise.txt','w')
        deptId = 4
courseId = 3999
    elif (deptNoParam == 5):
        path = "course\inMae.txt";
        fileSQLcode = file ('course\out_Course_Mae.txt','w')
        deptId = 5
courseId = 4999
    fileContents = file ( path ).read()
    for line in fileContents.split(\'
\'):
row = line.split('\t') #code, course, inst
if len(row) >= 2:
    triples = []
    for item in row:
        if item:
            triples.append (item)
    #print triples
    tempCourseId, courseName, instName = triples #ignore this courseId
    names = instName.split(',')
    if len(names) == 1:
        firstName = names[0]
        lastName = ''
    else:
        lastName, firstName = names[0], names[1]
    #print codeId, courseName, firstName, lastName
    courseId = courseId + 1
    term = 'Fall 2009'
    credits = 3
    textbook = '"' + courseName.lower() + ' book"'
    refTextbook = '""'

    seatsInClass = [15, 20, 25]
    for i in range(1, 225):
        randNo = random.randint(1, 100)
        if (randNo <= 30):
            seatsLeft = seatsInClass[0]
        elif ((randNo >= 31) and (randNo <= 70)):
            seatsLeft = seatsInClass[1]
        else:
            seatsLeft = seatsInClass[2]

        #print 'INSERT INTO Course (courseId, courseName, term, deptId, credits, textbook, refTextbook) VALUES(' 
        #+ '"' + courseId + '"
        #+ ' + courseName + '""' + term + '""' + 
        #+ ' + deptId + '"
        #+ ' + seatsLeft + '"
        #+ ' + textbook + '""' + refTextbook + 
        #+ ');

        print >> fileSQLcode, 'INSERT INTO Course (courseId, courseName, term, 
        deptId, credits, seatsLeft, textbook, refTextbook) VALUES(' 
        + '"' + courseId + '"
        + ' + courseName + '""' + term + '""' + 
        + ' + deptId + '"
        + ' + seatsLeft + '"
        + ' + textbook + '""' + refTextbook + 
        + ');

    fileSQLcode.close()
def StitchFiles(lstFileNames, outFileName):
    '''out_Course_Ece.txt'''
    fileContents = ''
    fileSQLcode = file (outFileName,'w')

    for path in lstFileNames:
        fileContents = file (path ).read()
        print >> fileSQLcode, fileContents

    fileSQLcode.close()

def PopulateClassRoom2():
    '''
    Table : classroom (classId, location, maxCapacity, seatsLeft)
    1 CCE, 2 CSE, 3 ECE, 4 ISE, 5 MAE
locations = ['CCE1', 'CCE2', 'CCE3', 'CCE4', 'CCE5', 'CCE6', 'CCE7', 'CCE8', 'CCE9', 'CCE10', 'CCE11', 'CCE12', 'CCE13', 'CCE14', 'CCE15', 'CCE16', 'CCE17', 'CCE18', 'CCE19', 'CCE20', \\
'CSE1', 'CSE2', 'CSE3', 'CSE4', 'CSE5', 'CSE6', 'CSE7', 'CSE8', 'CSE9', 'CSE10', 'CSE11', 'CSE12', 'CSE13', 'CSE14', 'CSE15', 'CSE16', 'CSE17', 'CSE18', 'CSE19', 'CSE20', \\
'ECE1', 'ECE2', 'ECE3', 'ECE4', 'ECE5', 'ECE6', 'ECE7', 'ECE8', 'ECE9', 'ECE10', 'ECE11', 'ECE12', 'ECE13', 'ECE14', 'ECE15', 'ECE16', 'ECE17', 'ECE18', 'ECE19', 'ECE20', \\
'MAE1', 'MAE2', 'MAE3', 'MAE4', 'MAE5', 'MAE6', 'MAE7', 'MAE8', 'MAE9', 'MAE10', 'MAE11', 'MAE12', 'MAE13', 'MAE14', 'MAE15', 'MAE16', 'MAE17', 'MAE18', 'MAE19', 'MAE20']

#locations = ['Civil', 'Civil', 'ComputerScience', 'ComputerScience', 'Electrical', 'Electrical', 'Industrial', 'Industrial', 'Mechanical', 'Mechanical']

seatsInClass = [10, 15, 20]

for location in locations:
    randNo = random.randint(1, 100)
    if (randNo <= 30):
        maxCapacity = seatsInClass[0]
    elif ((randNo >= 31) and (randNo <= 70)):
        maxCapacity = seatsInClass[1]
    else:
        maxCapacity = seatsInClass[2]
    seatsLeft = maxCapacity

    print('INSERT INTO classroom(location, maxCapacity, seatsLeft) VALUES (' + \\
          ''' + location + '',' + `maxCapacity` + ',' + `seatsLeft` + ');')

def main():
    Fills table : Student -------------------------------
    PopulateStudentTable(1); #PopulateStudentTable(2); PopulateStudentTable(3); PopulateStudentTable(4); PopulateStudentTable(5);
    #fileNames = ['student\out_Student_Cce.txt', 'student\out_Student_Cise.txt', 'student\out_Student_Ece.txt', 'student\out_Student_Ise.txt', 'student\out_Student_Mae.txt']
    #StitchFiles(fileNames, 'outStudent.txt');

    if __name__ == '__main__':
        main()