Project 2: Database Backend Web Application

CS460 - Design Manual

Members: Megan Edwards, Michael Bencivenga, Kell Grady, and Naida Torres

Table of Contents

Design Overview

General Workflow

Upon starting the application the user is prompted to choose what level of access they want between admin, instructor and student. Depending on their choice the user will be redirected to a new page that presents them with all possible commands for that level of access. When a command is used all arguments are passed to the matching query in views.py, we then query our SQL database, load the corresponding HTML file from our template folder and render it with the proper context.

Feature 1 (Roster)

Feature 1 is an admin command called ‘Roster’ which allows the user to create a list of all instructors in the database and can be sorted by name, department or salary.

The roster command includes a dropdown menu to select what value the admin user would like to sort the results of the query by, preventing incorrect and potentially malicious input.

The SQL query used to return the list for the roster command is ‘"""SELECT instructor.*, SUM(funding.amount) AS funding FROM instructor LEFT JOIN funding ON instructor.id = funding.overseeingProf GROUP BY instructor.id, instructor.Name, instructor.dept_name, instructor.salary ORDER BY %s """ %sort’.

Feature 2 (Salary)

Feature 2 is another admin command called ‘Salary’ which allows the user to generate a table which shows minimum, maximum and average salaries of instructors sorted by department.

For this command there is no option to change the way the table is sorted and thus the user interface consists of a single button.

The SQL query used to return the table containing the minimum, maximum and average salaries of instructors sorted by department is as follows ‘"""SELECT dept_name, min(salary) AS min, max(salary) AS max, round(avg(salary),0) AS avg FROM instructor GROUP BY dept_name;"""’.

Feature 3 (Performance)

Feature 3 is the third and final admin command called ‘Performance’ and allows the user to view the number of courses, students, total funding and total papers that an instructor was involved with given their name, a year and a semester.

This command allows users to type in the name of the instructor whose results they wish to view along with a drop down for both the year and semester they want to sort by, preventing incorrect and potentially malicious input.

The following query was used in order to return the proper results (prepare yourself), ‘"""SELECT T1.ccid AS courseCount, T2.csid AS studentCount, T3.funds as totalFunding, T4.cpapers AS totalPapers FROM (SELECT COUNT(course_id) AS ccid FROM teaches WHERE year = %s AND semester = %s AND teacher_id = (SELECT id from instructor WHERE name = %s)) AS T1, (SELECT COUNT(student_id) AS csid FROM takes WHERE year = %s AND semester = %s AND course_id IN (SELECT course_id FROM teaches WHERE

teacher_id = (SELECT id from instructor WHERE name = %s))) AS T2,(SELECT SUM(amount) AS funds FROM funding WHERE overseeingProf = (SELECT id from instructor WHERE name = %s)) AS T3,(SELECT COUNT(title) AS cpapers FROM papers WHERE author = %s) AS T4;"""’.

Feature 4

Feature 4 is an instructor command that allows the user to view a table of course sections and the number of students enrolled in each section that a specified professor taught in a chosen semester.

The user will be prompted to enter the instructor's last name and will be provided a dropdown menu to select both year and semester, preventing incorrect and potentially malicious input.

The SQL query used to return the desired result is as follows…

‘"""SELECT teaches.course_id,teaches.sec_id,count(student_id) AS studentCount FROM teaches INNER JOIN takes ON takes.course_id = teaches.course_id INNER JOIN instructor ON teaches.teacher_id = instructor.id WHERE takes.semester = %s AND takes.year = %s AND instructor.name = %s GROUP BY teaches.course_id, teaches.sec_id;"""’.

Feature 5

Feature 5 is another Instructor command that displays the list of students enrolled in a chosen course section and semester that a specified professor taught

The user will again be prompted to enter the instructor’s last name and then will be provided a dropdown menu to select both the year and semester they wish to search based on.

The SQL query used to return the desired table is ‘"""SELECT DISTINCT student.name, takes.course_id, takes.sec_id FROM takes INNER JOIN student ON student.student_id = takes.student_id INNER JOIN teaches ON teaches.course_id = takes.course_id INNER JOIN instructor ON teaches.teacher_id = instructor.id WHERE takes.semester = %s AND takes.year = %s AND instructor.name = %s;"""’.

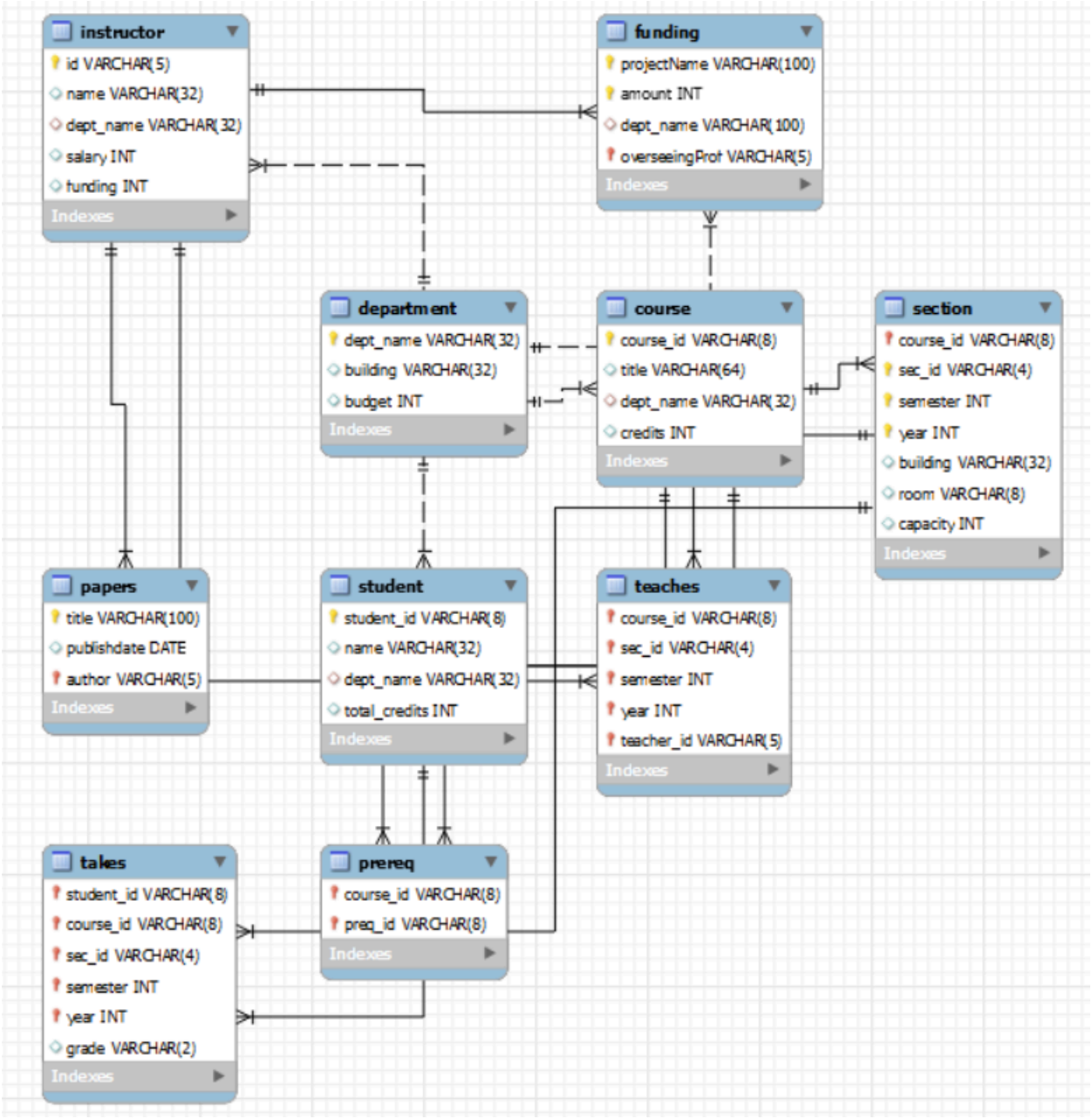
Feature 6

Our final feature, feature 6, is the lone student command and allows the user to return a list of course sections offered by a chosen department in a chosen year and semester.

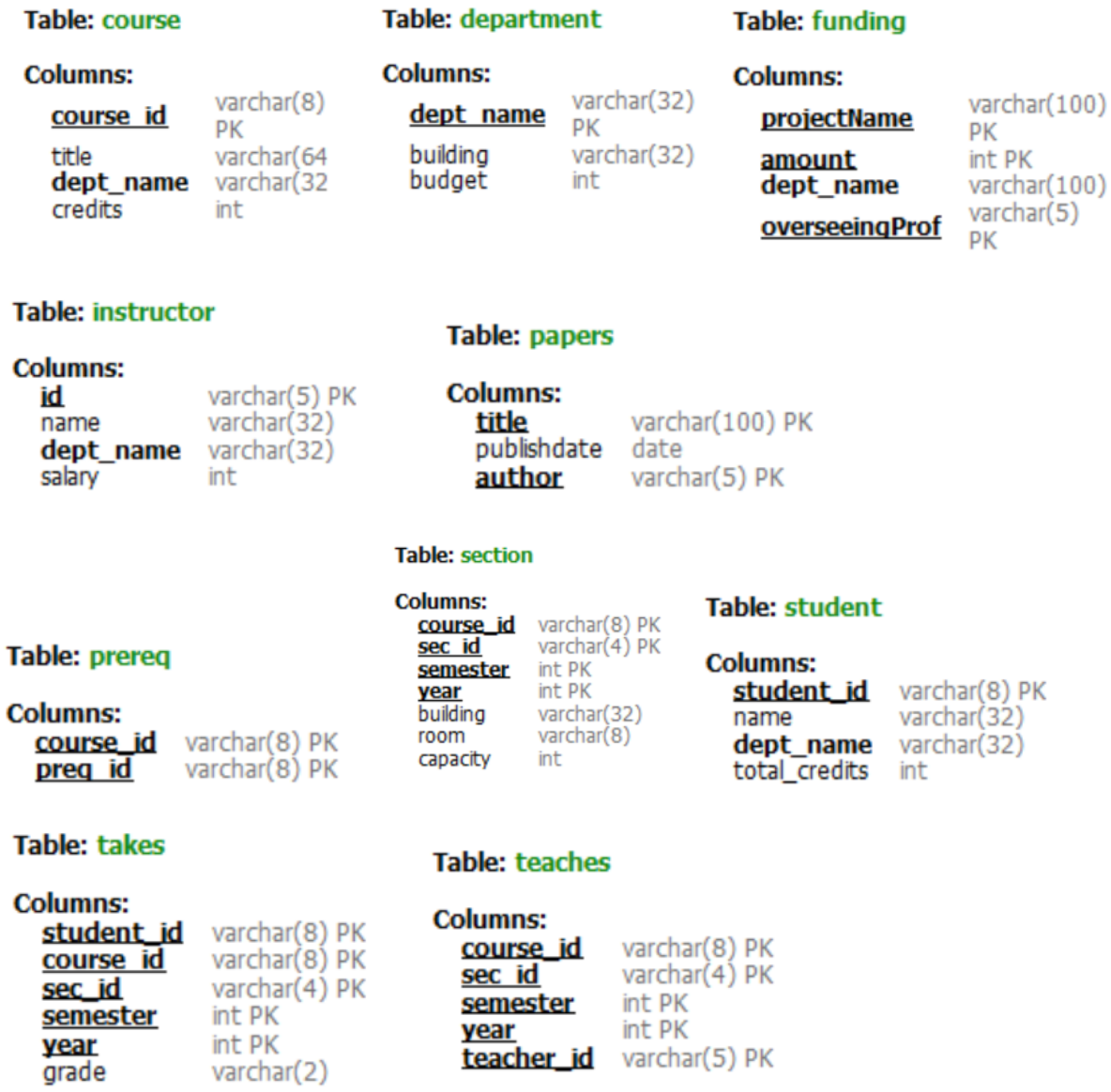
For this feature the user is prompted with 3 drop downs, one for department, year and semester and selects from the given options.

The SQL query used to return the course IDs and section IDs is as follows ‘"""SELECT course_id, sec_id FROM teaches WHERE year = %s AND semester = %s AND course_id IN (SELECT course_id FROM course WHERE dept_name = %s)"""’.

ER Diagram


instructor
id VARCHAR(5)
name VARCHAR(32)
dept_name VARCHAR(32)
salary INT
funding INT
Indexes
funding
projectName VARCHAR(100)
amount INT
dept_name VARCHAR(100)
overseeingProf VARCHAR(5)
Indexes
department
dept_name VARCHAR(32)
building VARCHAR(32)
budget INT
Indexes
course
course_id VARCHAR(8)
title VARCHAR(64)
dept_name VARCHAR(32)
credits INT
Indexes
section
course_id VARCHAR(8)
sec_id VARCHAR(4)
semester INT
year INT
building VARCHAR(32)
room VARCHAR(8)
capacity INT
Indexes
papers
title VARCHAR(100)
publishdate DATE
author VARCHAR(5)
Indexes
student
student_id VARCHAR(8)
name VARCHAR(32)
dept_name VARCHAR(32)
total_credits INT
Indexes
teaches
course_id VARCHAR(8)
sec_id VARCHAR(4)
semester INT
year INT
teacher_id VARCHAR(5)
Indexes
takes
student_id VARCHAR(8)
course_id VARCHAR(8)
sec_id VARCHAR(4)
semester INT
year INT
grade VARCHAR(2)
Indexes
prereq
course_id VARCHAR(8)
preq_id VARCHAR(8)
Indexes

Database Schema



SQL Injections and XSS

To deal with both SQL Injections and XSS we minimized user input wherever possible, providing users with a menu of selectable options instead to prevent malicious code being entered. All queries are also run directly in Django so when it is necessary for a user to type out their input it is read separately and then filled into an existing query only accepting the correct data type. In the worst case scenario the user will get a webpage error for an incorrectly formatted query.

Other

In order to properly run certain queries in our application, the user must create two new tables within their existing university database, this can be done using the following commands in your SQL client…

```
CREATE TABLE funding(
    projectName varchar(100),
    amount int,
    dept_name varchar(100),
    overseeingProf varchar(5),
    PRIMARY KEY (projectName, amount, overseeingProf),
    FOREIGN KEY (dept_name) REFERENCES department(dept_name),
    FOREIGN KEY (overseeingProf) REFERENCES instructor(id)
);

CREATE TABLE papers(
    title varchar(100),
    publishdate DATE,
    author varchar(5),
    PRIMARY KEY (title, author),
    FOREIGN KEY (author) REFERENCES instructor(id)
);
```