Out With the Old, In With the New:

Updating an Academic Database

Project Step 7 Draft Version: DML and DDL Queries

Team San Diego Dreaming (Group 53):
Hsing-Yi Lin
Charles Sherwood

Introduction to Databases | CS 340

December 6th, 2021

Database URL: http://flip3.engr.oregonstate.edu:3622

Executive Summary

There were quite a few changes between our finished project and our initial design. This summary details the major changes that were implemented and a few of the suggestions that we decided not to implement.

Early changes to our project involved correcting overlooked details, such as titling our project and giving hypothetical volumes of input. We were also advised by a peer that we might want to consider having Textbook's primary key not be an ISBN. This would facilitate the support of non-ISBN materials. This was a great suggestion that we implemented very shortly afterwards.

After the second outline submission, our grader noted that we did not have the Course_Student intersection table in our ER diagram. We were advised to include this table in our diagram. We acquiesed to this suggestion.

Towards the end of the peer review process, one of our peers suggested adding input validation for our tables. Because this is not a hard requirement for the project, we declined to implement this feature. In a real-world scenario, we would strongly consider implementing this.

For the final week of feedback, our peers noted that our project wasn't quite complete. By this point, we knew the final steps we needed to take to complete the project. Still, we did consider their input while making our final changes. One peer suggested that we might want to make our filters more robust. Because our filters already satisfy the project requirements, we decided against creating more filters.

We were also advised by a peer that week that we are not to allow users to input primary keys manually. This posed a couple of interesting questions for our Course_Student design. How should users create relationships between Students and Courses? How should we filter the table to supply useful information?

We decided that the best method to add relationships to Course_Student would involve a drop-down menu for all students and classes. For filtering the table, we settled on allowing users to search either Course Names or Student Names.

Lastly, we will briefly summarize the changes that we implemented independently from peer/grader feedback.

Originally, our project used the University of San Diego (UCSD) in our fictitious scenario. We decided to rename the school in our scenario to something fictitious. We settled on the San Diego Institute of Technology.

We also merged our Student filter. Before, there were two filters that either searched by student name or searched by student type. These filters seemed too narrow to be useful, so it was merged into an and/or sort. Users can search for students by their names and/or their types.

All of these changes were wonderful opportunities to learn. Most of the changes that our peers suggested were eventually implemented. We feel very grateful to have been part of such an interactive feedback system.

Overview:

The San Diego Institute of Technology (SDIT) has just hired a new computer science director. The director has decided that their existing department database is unsatisfactory. To remedy this, the director has decided to create a simple web-based database for their staff to use.

The director expects that this database would easily handle their current volume of students (a little over 300 students) as well as their current instructors (10 professors). Each quarter has 20 concurrent classes. The number of textbooks used will correspond roughly to the number of classes being held, or about 20.

This new database will also rectify two crucial flaws in the existing database.

The first issue with the current database is that there is no separate table for textbooks and classes. Textbook and class information was combined during the creation of the existing database. Textbook information is critically important to the director. The director needs the information from classes and textbooks properly delineated. To accomplish this, two tables will be created to properly store that information separately.

The second issue with the existing database is that the classes students are enrolled in are part of the students table as individual columns (Class 1, Class 2, Class 3, etc.). This is a critical flaw in database design that must be corrected. To accomplish this, a fifth table, Course_Student, will be created to handle the many-to-many relationship between courses and students.

Database Outline:

- **Instructor:** records information about our instructors.

    - instructor_id, int, auto_increment, unique, not NULL, PK
    - first_name, varchar, not NULL
    - last_name, varchar, not NULL
    - email, varchar, not NULL
    - Relationship(s):
        - a 1:M relationship between Instructor and Course is implemented with instructor_id as a FK inside of Course.

- **Course:** records course details.

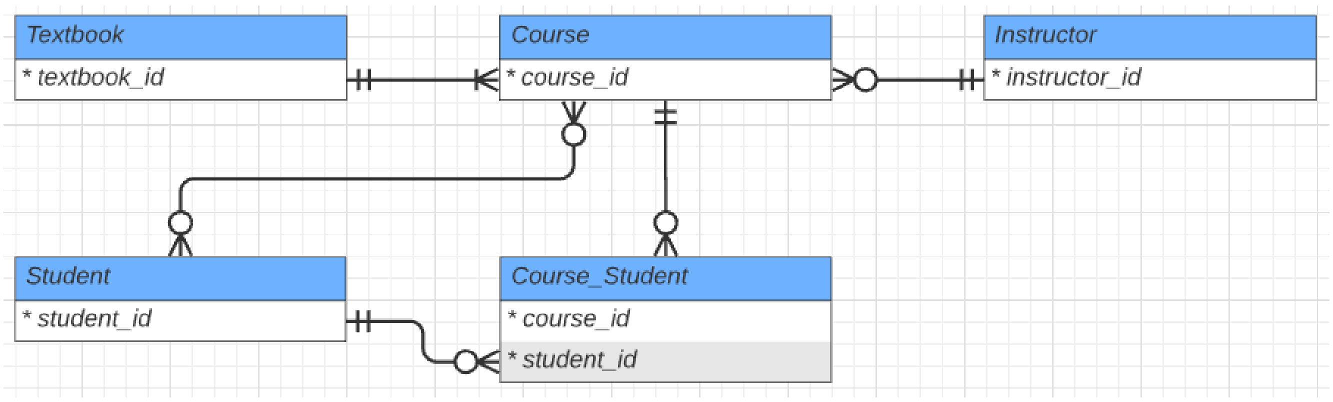
    - course_id, int, unique, not NULL, PK
    - name, varchar, not NULL
    - year, int, not NULL
    - term, varchar, not NULL
    - instructor_id, int, not NULL, FK
    - textbook_id, int, FK
    - Relationship(s):
        - a 1:M relationship between Instructor and Course is implemented with instructor_id as a FK inside of Course.
        - a 1:M relationship between Textbook and Course is implemented with textbook_isbn as

        a FK inside of Course.
        - a M:M relationship between Course and Student is implemented with an additional table Course_Student, consisting of course_id and student_id as FKs within the table.

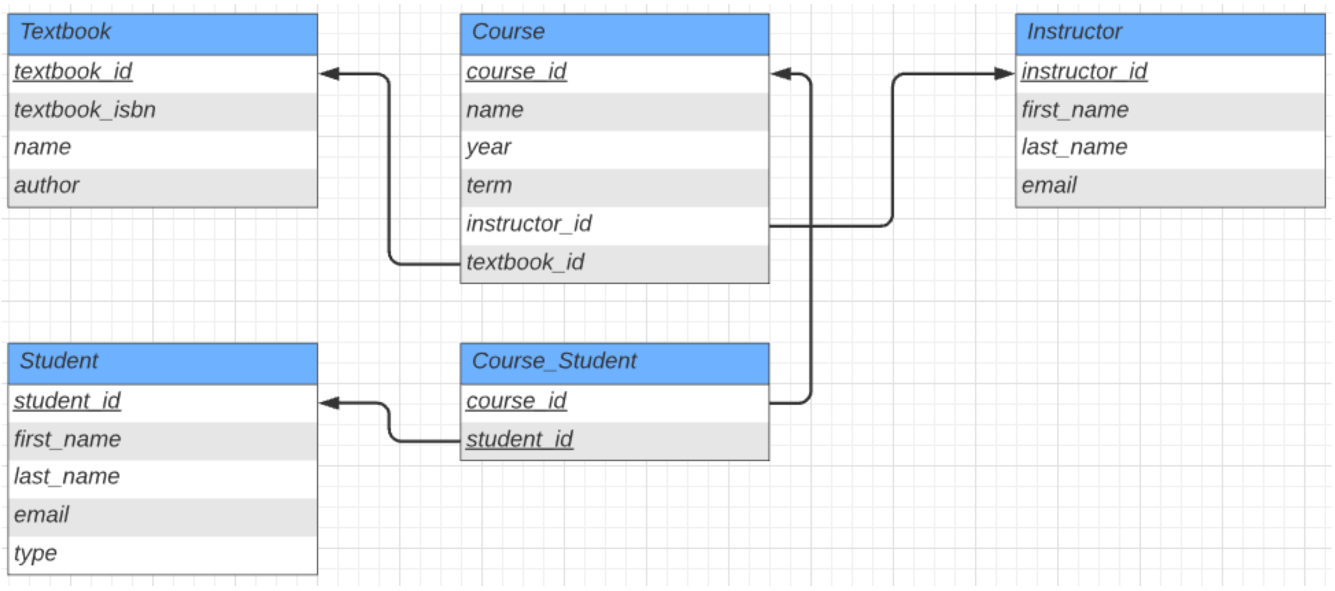
- **Student:** records information about our students.
    - student_id, int, auto_increment, unique, not NULL, PK
    - first_name, varchar, not NULL
    - last_name, varchar, not NULL
    - email, varchar, not NULL
    - type, varchar, not NULL (e.g. undergraduate, graduate, or post-bacc)
    - Relationship(s):
        - a M:M relationship between Course and Student is implemented with an additional table Course_Student, consisting of course_id and student_id as FKs within the table.

- **Textbook:** records textbook information.
    - textbook_id, int, unique, not NULL, PK
    - textbook_isbn, varchar
    - name, varchar, not NULL
    - author, varchar, not NULL
    - Relationship(s):
        - a 1:M relationship between Textbook and Course is implemented with textbook_id as a FK inside of Course.

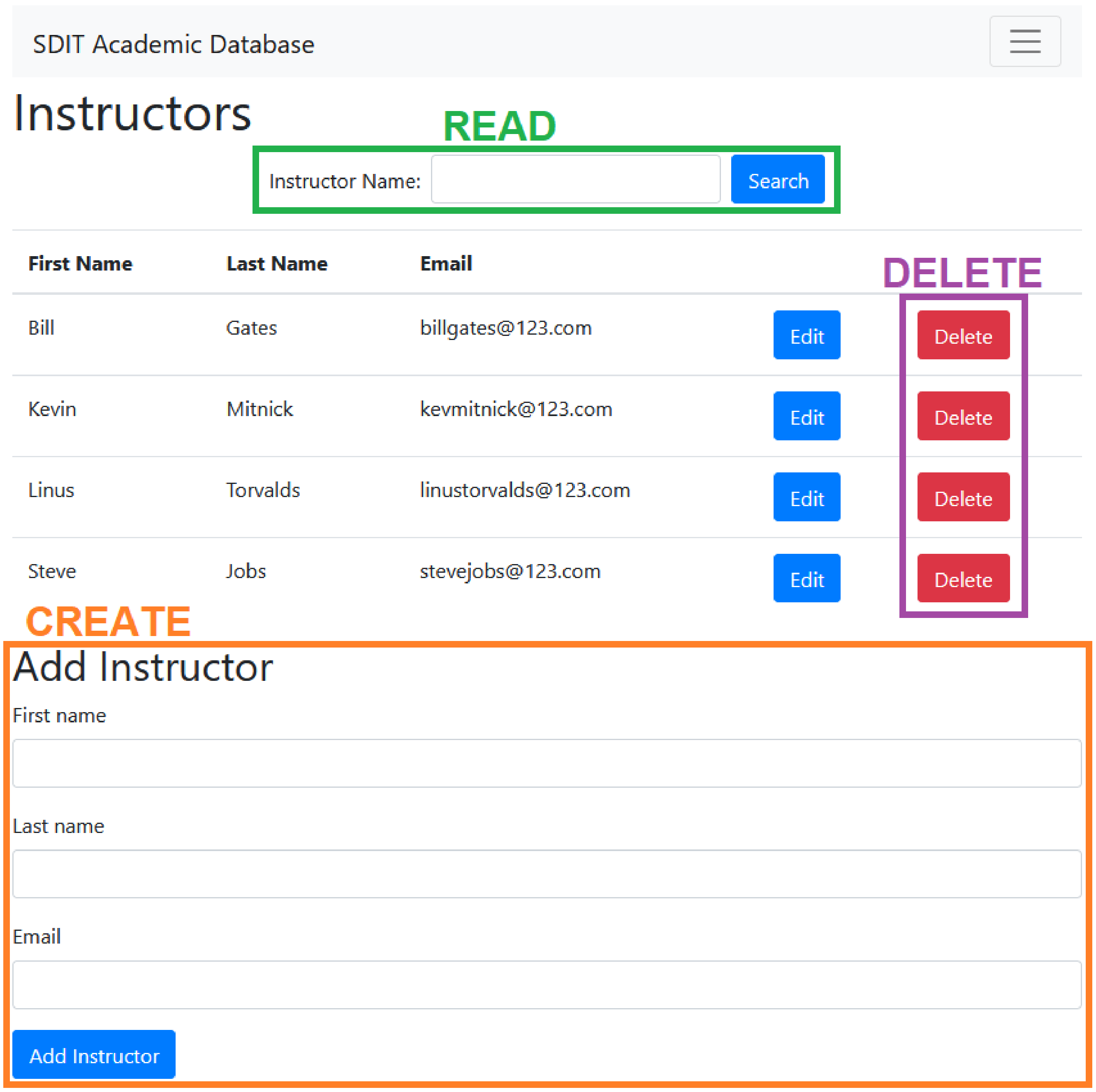
Entity-Relationship diagram:



Schema:

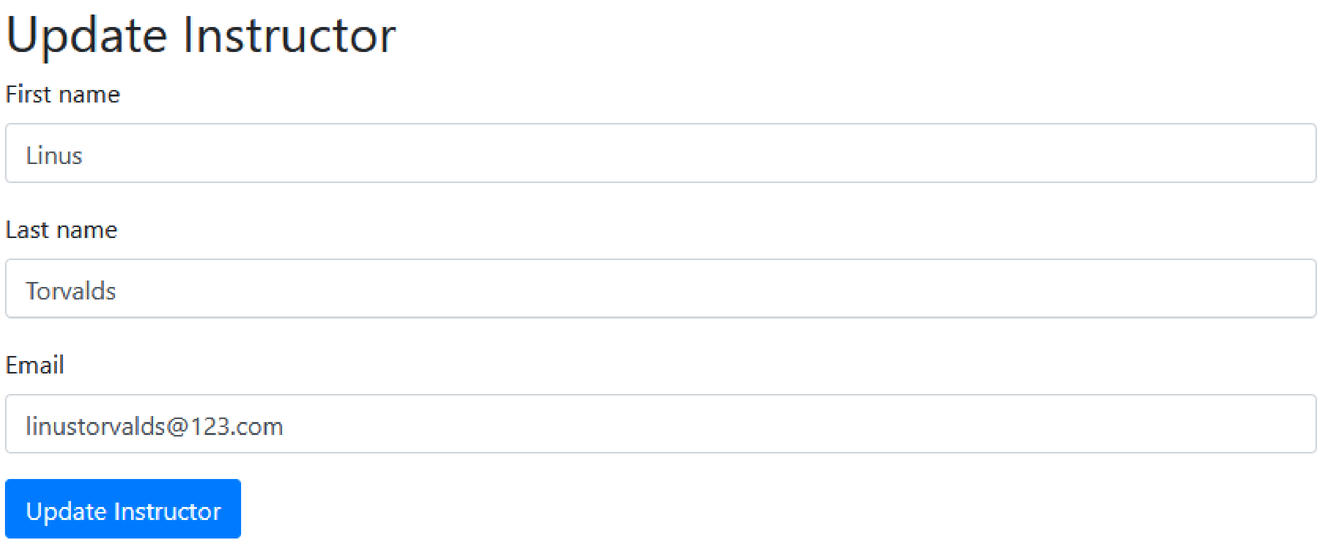


CREATE/READ/DELETE Instructor:



First Name	Last Name	Email		
Bill	Gates	billgates@123.com	Edit	Delete
Kevin	Mitnick	kevmitnick@123.com	Edit	Delete
Linus	Torvalds	linustorvalds@123.com	Edit	Delete
Steve	Jobs	stevejobs@123.com	Edit	Delete

UPDATE Instructor:



Update Instructor

First name

Linus

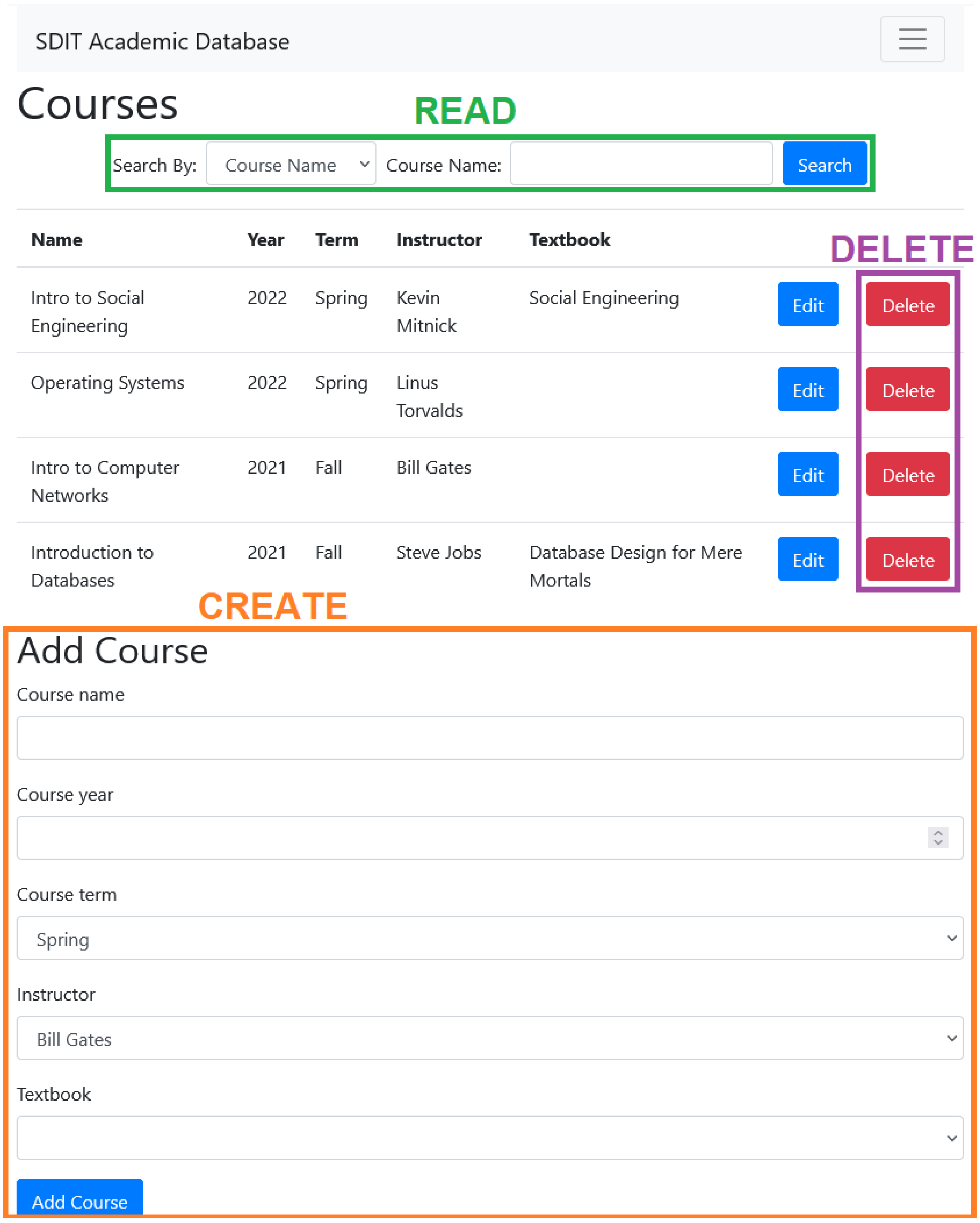
Last name

Torvalds

Email

linustorvalds@123.com

CREATE/READ/DELETE Course:



Courses

Search By: Course Name Course Name:

Name	Year	Term	Instructor	Textbook
Intro to Social Engineering	2022	Spring	Kevin Mitnick	Social Engineering
Operating Systems	2022	Spring	Linus Torvalds	
Intro to Computer Networks	2021	Fall	Bill Gates	
Introduction to Databases	2021	Fall	Steve Jobs	Database Design for Mere Mortals

Add Course

Course name

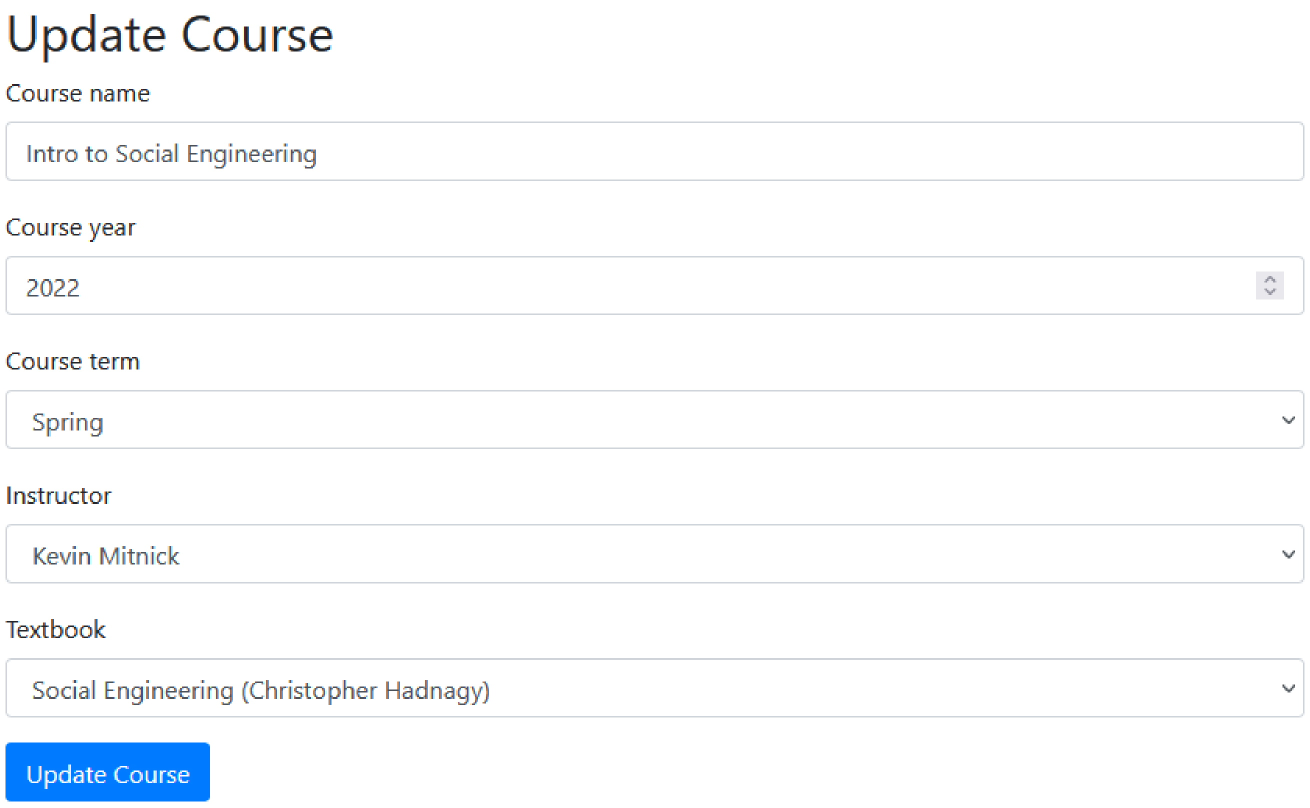
Course year

Course term: Spring

Instructor: Bill Gates

Textbook

UPDATE Course:



Update Course

Course name

Intro to Social Engineering

Course year

2022

Course term

Spring

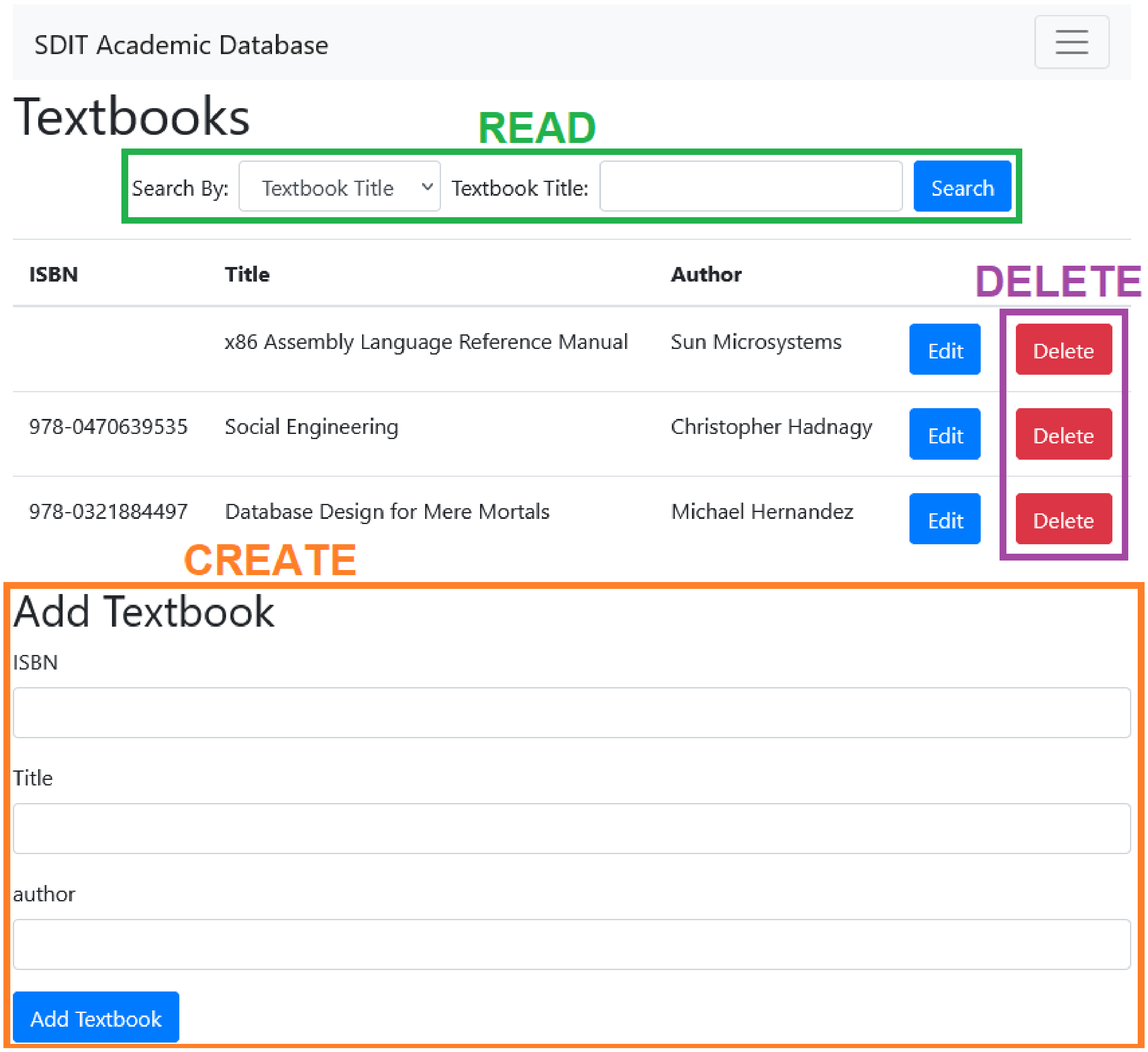
Instructor

Kevin Mitnick

Textbook

Social Engineering (Christopher Hadnagy)

CREATE/READ/DELETE Textbook:



SDIT Academic Database

Textbooks

Search By: Textbook Title Textbook Title: Search

ISBN	Title	Author		
	x86 Assembly Language Reference Manual	Sun Microsystems	Edit	Delete
978-0470639535	Social Engineering	Christopher Hadnagy	Edit	Delete
978-0321884497	Database Design for Mere Mortals	Michael Hernandez	Edit	Delete

Add Textbook

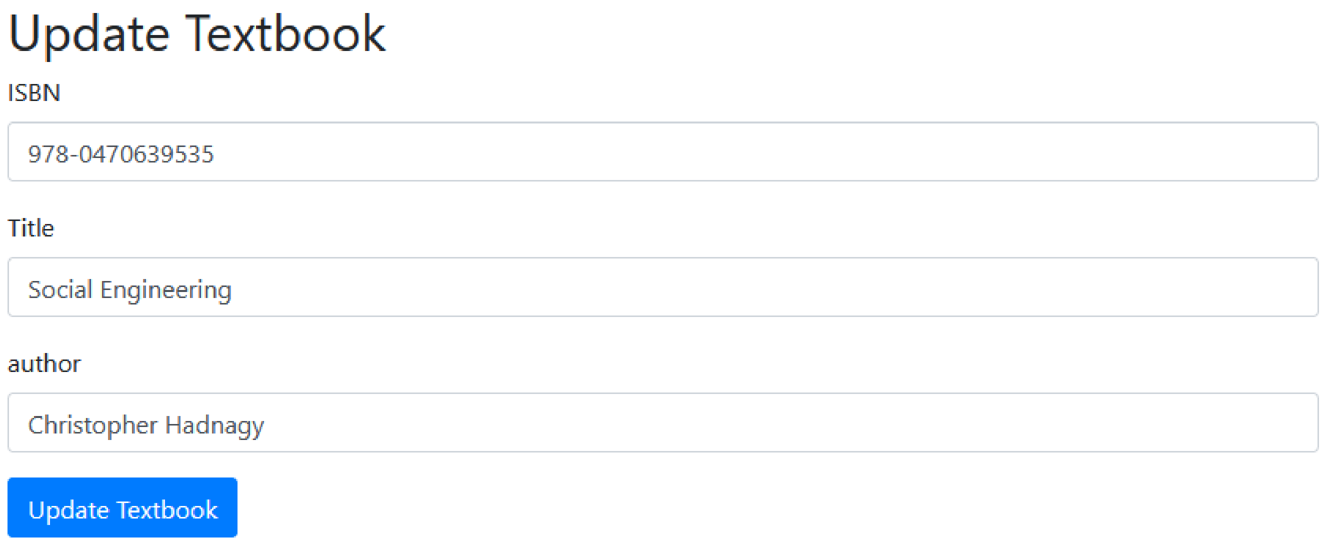
ISBN

Title

author

Add Textbook

UPDATE Textbook:


Update Textbook

ISBN

978-0470639535

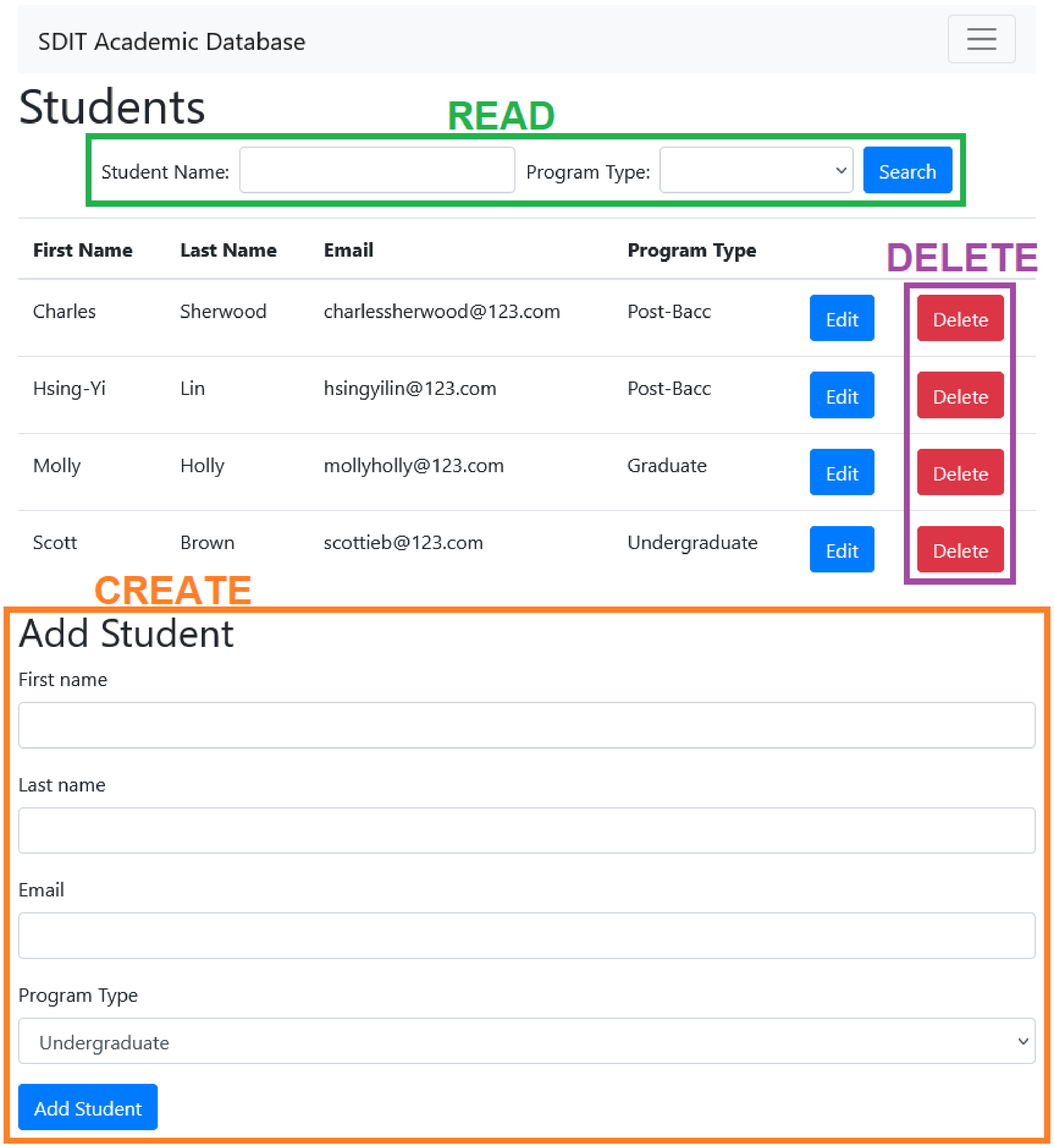
Title

Social Engineering

author

Christopher Hadnagy

CREATE/READ/DELETE Student:


SDIT Academic Database

Students

READ

Student Name: Program Type: Search

First Name	Last Name	Email	Program Type		
Charles	Sherwood	charlessherwood@123.com	Post-Bacc	Edit	Delete
Hsing-Yi	Lin	hsingyilin@123.com	Post-Bacc	Edit	Delete
Molly	Holly	mollyholly@123.com	Graduate	Edit	Delete
Scott	Brown	scottieb@123.com	Undergraduate	Edit	Delete

DELETE

CREATE

Add Student

First name

Last name

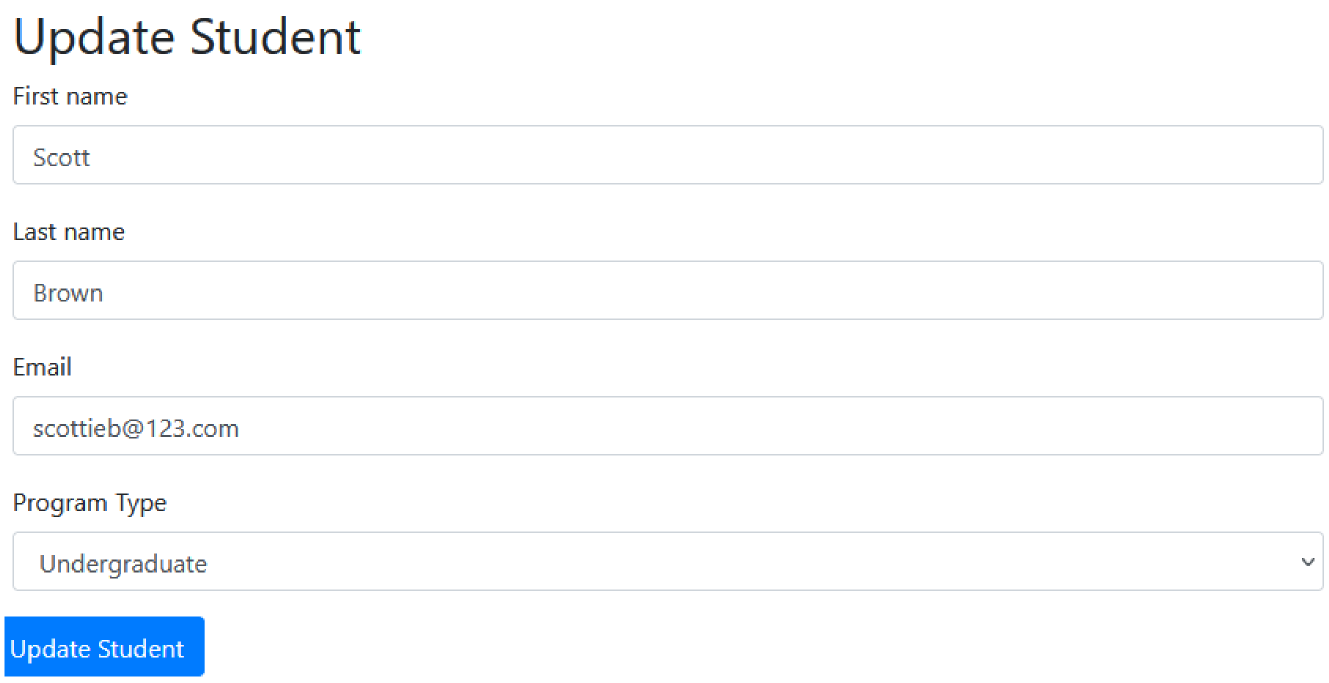
Email

Program Type

Undergraduate

Add Student

UPDATE Student:



Update Student

First name

Scott

Last name

Brown

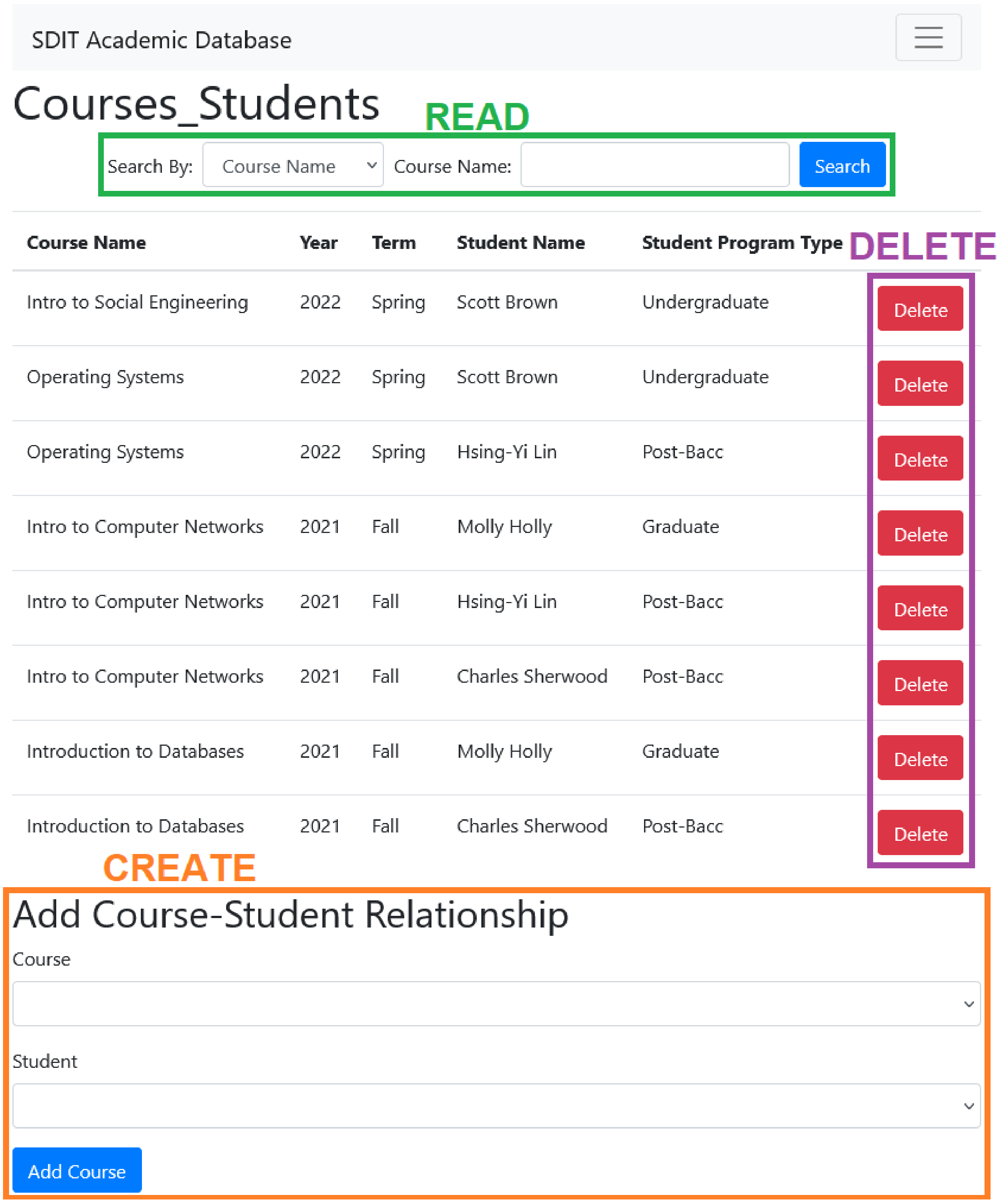
Email

scottieb@123.com

Program Type

Undergraduate

CREATE/READ/DELETE Course_Student:



SDIT Academic Database

Courses_Students

Search By: Course Name Course Name: Search

Course Name	Year	Term	Student Name	Student Program Type	
Intro to Social Engineering	2022	Spring	Scott Brown	Undergraduate	Delete
Operating Systems	2022	Spring	Scott Brown	Undergraduate	Delete
Operating Systems	2022	Spring	Hsing-Yi Lin	Post-Bacc	Delete
Intro to Computer Networks	2021	Fall	Molly Holly	Graduate	Delete
Intro to Computer Networks	2021	Fall	Hsing-Yi Lin	Post-Bacc	Delete
Intro to Computer Networks	2021	Fall	Charles Sherwood	Post-Bacc	Delete
Introduction to Databases	2021	Fall	Molly Holly	Graduate	Delete
Introduction to Databases	2021	Fall	Charles Sherwood	Post-Bacc	Delete

Add Course-Student Relationship

Course

Student

Add Course