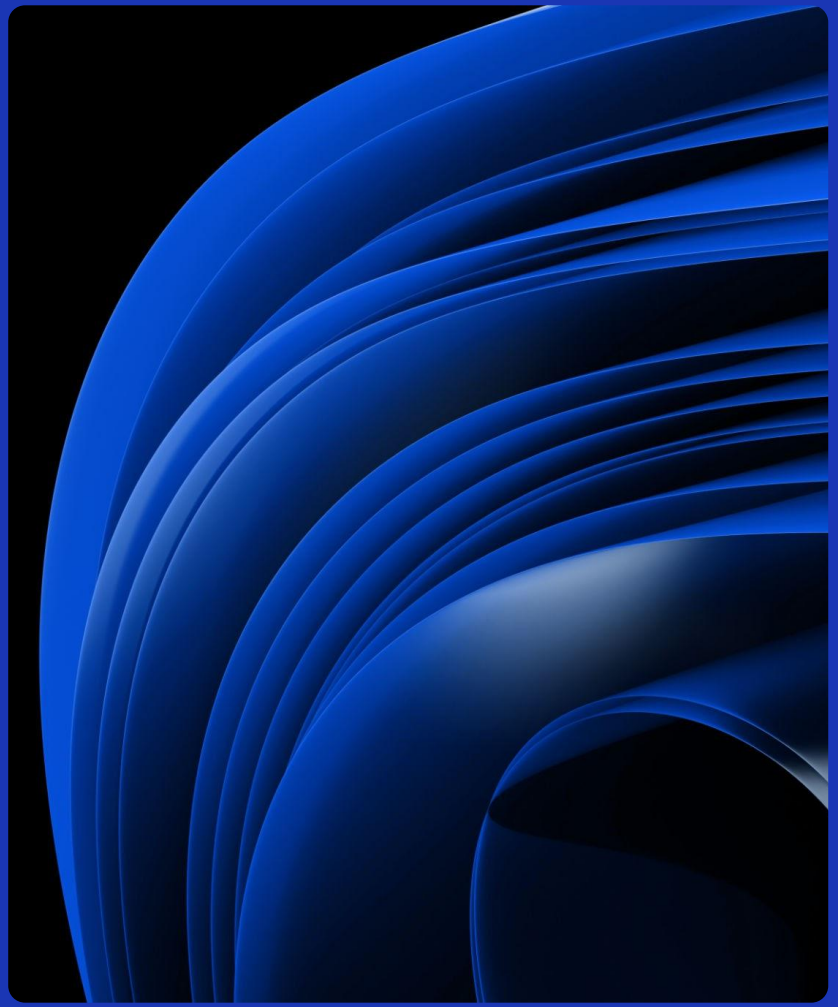
DS5110

Final Project

By Kevin Cooper and Felipe Quiroz

Fall 2024



Introduction

- Many people can tutor for classes, but it’s hard to know who and for what
- Centralized database
  - Easy to find classes and peers
  - Can host resources and contact information easily
  - Provide scheduling information

Literary Review

Knack

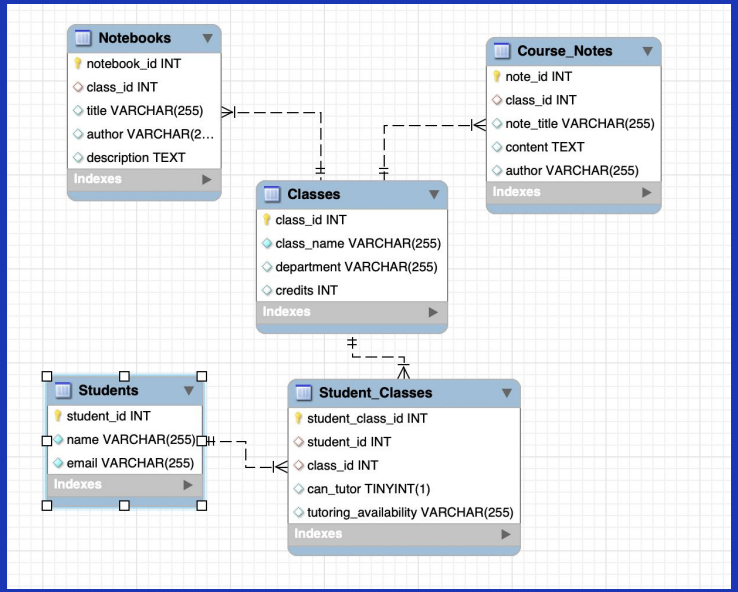
- Tutoring Space
- Payment Service
- Job not relationship

Tutoring

- Word of mouth
- No regulation

Database

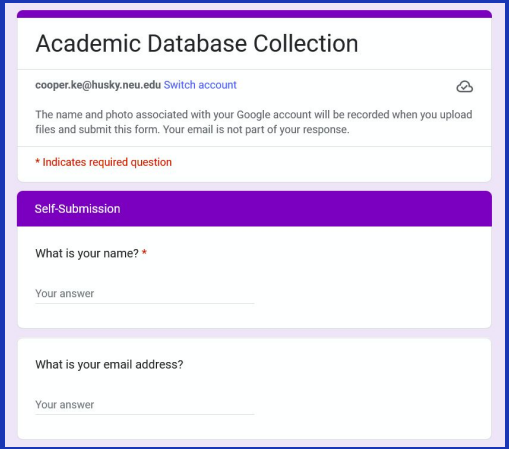
- 5 tables
- Coded on MySQL
- Files loaded in locally.
- For storage and security



student_class...	student_...	class_id	can_tutor	tutoring_availability
7	1	ds5110	1	I can tutor only on Friday mornings
8	2	ds5110	1	Email me for tutoring
9	3	ds5110	0	NULL
10	4	ds5110	0	NULL
NULL	NULL	NULL	NULL	NULL

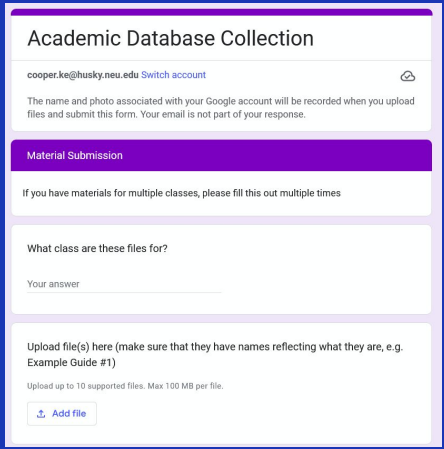
Data Collection

- Automatic ERD
- Data comes from Google Form
- Forms outputs a Google Sheet
- Google Sheets converts to CSV



Tutor

- Which classes have you taken



Resource

- Upload up to ten things at a time

The Code (database)

- Python and SQL
- Data Preprocessing
- Connects directly to SQL DB
- Reads from CSV
  - Similar techniques can be used for APIs
- Outputs easily using SQL queries

```
#begin loading in data per table
#Load into students
for _, row in df.iterrows():
    cursor.execute("""
        INSERT INTO Students (student_id, name, email)
        VALUES (%s, %s, %s)
        ON DUPLICATE KEY UPDATE name=VALUES(name), email=VALUES(email)
    """, (row['student_id'], row['name'], row['email']))

#Load into Classes
for _, row in df.iterrows():
    cursor.execute("""
        INSERT INTO Classes (class_id, class_name, credits, department)
        VALUES (%s, %s, %s, %s)
        ON DUPLICATE KEY UPDATE class_name=VALUES(class_name), credits=VALUES(credits),
    """, (row['class_id'], row['class_name'], row['credits'], row['department']))
```

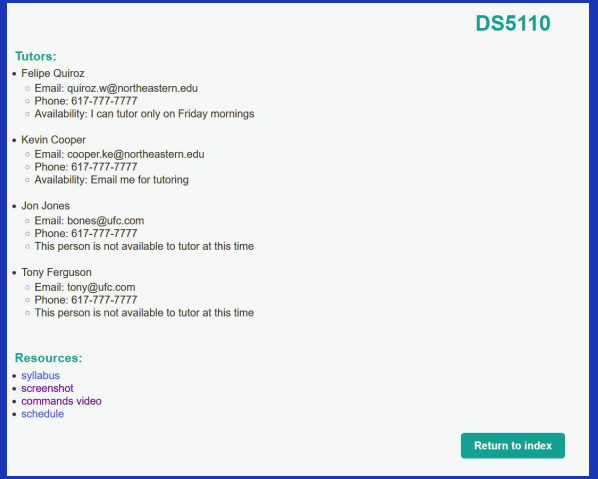
```
#show students table
cursor.execute("SELECT * FROM Students;")
students = cursor.fetchall()
print("Students Table:")
for row in students:
    print(row)

#show classes table
cursor.execute("SELECT * FROM Classes;")
classes = cursor.fetchall()
print("\nClasses Table:")
for row in classes:
    print(row)
```

The Website

- CSS is simple and easily loaded but good-looking
- Function over form
- Fast loading times and a light touch
- Automatically generated

Home Page

Lists which classes we have information for



Class Pages

Lists tutors, their contact information, and relevant resources

The Code (website)

- Python
- Object-oriented
- Dominate
- Reads from CSV
  - Can be easily updated to use Google's API to scrape from a Sheet, live
- Uses "with" statements for memory and error-handling benefits
- Outputs easily
  - "file.write(str(doc))"

```python
class Course:
    def __init__(self, name, tutors, resources):
        self.name = name # String, name of the course
        self.tutors = tutors # List of Tutors
        self.resources = resources # List of Resources

class Tutor:
    def __init__(self, name, email, phone_number, can_tutor, availability):
        self.name = name # String
        self.email = email # String
        self.phone_number = phone_number # String
        self.can_tutor = can_tutor # Boolean
        self.availability = availability # String

class Resource:
    def __init__(self, name, link):
        self.name = name # Name of the resource
        self.link = link # Link to the resouce
```

Object declarations

- Easy to work with
- Each object corresponds to something that exists in real life

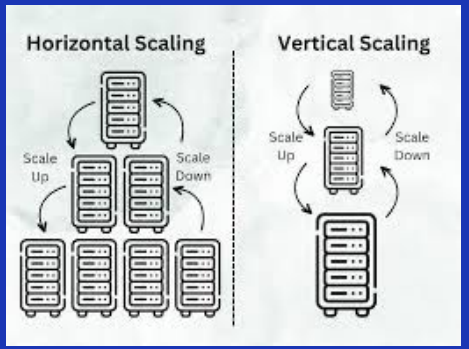
Demonstration

Future Work and Conclusion

Reach Out to Academic Communities

Reach Out to Other Student Organizations

Create Clear Instructions on how to use this service



Vertically and Horizontally Scaling to Handle More and Different Data

Thank you

Any questions?