**COMP306 Project: KOLT Administration Website**

Ahmet Başar Ertürk, Vedat Can Akın, Deniz Yılmaz, Arda Adalar, Serhat Avcı, Bartu Uzun

1. **Project Description**

We created an administrative website for KOLT Office. Using our system, KOLT admins can add and delete cubicles, courses, and tutors. Admins are also able to assign tutors to new courses, make them headtutors, and assign periods for tutors to hold office hours in. Using this website, headtutors will be able to see the tutors that they are responsible from. Also, with this website, tutors, admins, headtutors, and normal students will be able to see the Office’s schedule, so that when they want to get KOLT’s assistance for a course, they can find out the tutors of that course, and the time and cubicle information of those tutors’ office hours. Students can only see the schedule.

1. **Entity-Relationship Diagram**

**A picture containing diagram, line, sketch, plan

Description automatically generated**

1. **Relational Database Design**

query = f'''CREATE TABLE IF NOT EXISTS {DB.courses}(

code VARCHAR(50),

name VARCHAR(50),

headtutor\_name VARCHAR(50),

PRIMARY KEY (code),

FOREIGN KEY (headtutor\_name) REFERENCES {DB.tutors}(username));'''



query = f'''CREATE TABLE IF NOT EXISTS {DB.course\_tutor}(

course\_code VARCHAR(50),

tutor\_name VARCHAR(50),

PRIMARY KEY (course\_code, tutor\_name),

FOREIGN KEY (course\_code) REFERENCES {DB.courses}(code) ON DELETE CASCADE,

FOREIGN KEY (tutor\_name) REFERENCES {DB.tutors}(username) ON DELETE CASCADE)'''



query = f'''CREATE TABLE IF NOT EXISTS {DB.cubicles}(

cubicle\_number VARCHAR(10),

location VARCHAR(50),

PRIMARY KEY (cubicle\_number));'''



query = f'''CREATE TABLE IF NOT EXISTS {DB.periods}(

id INTEGER PRIMARY KEY AUTOINCREMENT,

day VARCHAR(10) CHECK (day IN ("monday","tuesday","wednesday","thursday","friday")),

interval VARCHAR(12) CHECK (interval IN ("08:00-09:00", "09:00-10:00", "10:00-11:00","11:00-12:00","12:00-13:00","13:00-14:00","14:00-15:00","15:00-16:00","16:00-17:00")));'''



query = f'''CREATE TABLE IF NOT EXISTS {DB.tp\_cubicle}(

tp\_id INTEGER,

cubicle\_number VARCHAR(10),

PRIMARY KEY (tp\_id, cubicle\_number)

FOREIGN KEY (cubicle\_number) REFERENCES {DB.cubicles}(cubicle\_number) ON DELETE CASCADE,

FOREIGN KEY (tp\_id) REFERENCES {DB.tutor\_period}(id) ON DELETE CASCADE

);'''



query = f'''CREATE TABLE IF NOT EXISTS {DB.tutor\_period}(

id INTEGER PRIMARY KEY AUTOINCREMENT,

tutor\_username VARCHAR(50),

period\_id INTERGER,

FOREIGN KEY (tutor\_username) REFERENCES {DB.tutors}(username) ON DELETE CASCADE,

FOREIGN KEY (period\_id) REFERENCES {DB.periods}(id) ON DELETE CASCADE);'''



query = f'''CREATE TABLE IF NOT EXISTS {DB.users}(

username VARCHAR(50),

email VARCHAR(50) UNIQUE,

password VARCHAR(50) NOT NULL,

role VARCHAR(50),

PRIMARY KEY (username));'''



query = f'''CREATE TABLE IF NOT EXISTS {DB.tutors}(

username VARCHAR(50),

start\_date DATE NOT NULL,

PRIMARY KEY (username),

FOREIGN KEY (username) REFERENCES {DB.users}(username));'''

1. **Data Sources**

When first initializing the data, we created constant tuples inside the constants.py file according to the attributes we need for each table, and used those constant tuples to populate our tables. These initial tuples were used when we were implementing the functionalities of our project. Then, we used the functionalities we implemented to further populate the database. The data was not taken from somewhere else, but we made them look similar to real life data (by using real course names, etc.). Every one of our tables have about 10-25 tuples, which is enough to demonstrate all the functionalities of our website and database.

1. **Complex SQL Queries**

Folder: repository, File: UserRepository.py, function: removeTutorIfNoCourse() :

query = f"""DELETE FROM {DB.tutors} WHERE

username = '{tutor\_name}' and

'{tutor\_name}' NOT IN (SELECT tutor\_name FROM {DB.course\_tutor}) """

Explanation: This query deletes an input tutor if that tutor has no courses assigned to them. This query is important because a tutor must have a course, so if a tutor has no course assigned to them left, then they should not be a tutor anymore. We run this query whenever we unassign a course from a tutor to check if that tutor still has some other courses left.



Folder: repository, File: UserRepository.py, function: removeTutorsWithNoCourse() :

query = f"""DELETE FROM {DB.tutors} WHERE

NOT EXISTS (SELECT \* FROM {DB.course\_tutor}

WHERE tutor\_name = username) """

Explanation: This query checks all tutors, detects the ones that has no course assigned to them, and delete all of them. This query scans and updates the whole table, and is called whenever a course is deleted from the database to check if any of the tutors are left with 0 courses.

* 1. Folder: repository, File: CourseRepository.py, function: updateHeadTutorToCourse() :

query = f"""UPDATE {DB.courses} SET headtutor\_name = '{headTutorName}'

WHERE code = '{courseCode}' and

EXISTS (SELECT tutor\_name

FROM {DB.course\_tutor}

WHERE course\_code = '{courseCode}' and

tutor\_name = '{headTutorName}')"""

Explanation: This code updates the headtutor of an input course to a input headtutor. It is called whenever we assign a new headtutor. While assigning the headtutor, it checks if the input headtutor is a tutor for the input course or not. It is important, because we cannot assign a tutor to a course as the headtutor if that tutor is not tutoring the input course.

* 1. Folder: repository, File: TutorPeriodCubicleRepository.py, function: getFreeCubicles() :

c.execute(f"SELECT cubicle\_number FROM {DB.cubicles}"

f" WHERE cubicle\_number NOT IN "

f" (SELECT cubicle\_number FROM {DB.tp\_cubicle}, {DB.tutor\_period}"

f" WHERE {DB.tp\_cubicle}.tp\_id={DB.tutor\_period}.id AND {DB.tutor\_period}.period\_id='{period\_id}') ORDER BY cubicle\_number ASC")

Explanation: This query returns all the cubicles that are not occupied during the given time period. We need this information whenever we want to assign a new cubicle to a tutor in the given period, because a tutor can only be assigned to one of the cubicles that are empty during that given period.

* 1. Folder: repository, File: TutorPeriodCubicleRepository.py, function: unassignCubicle()

c.execute(f"DELETE FROM {DB.tp\_cubicle}"

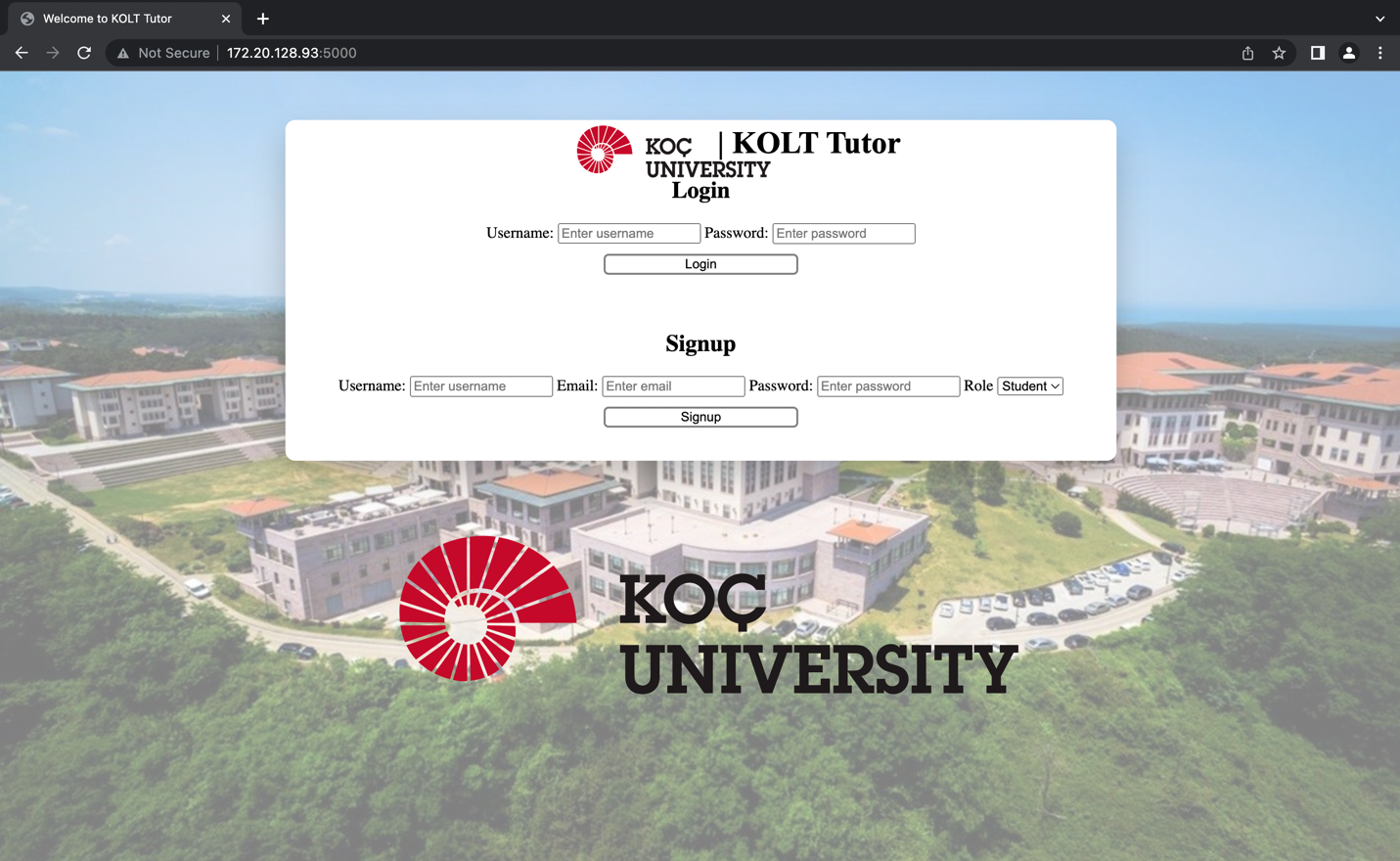
f" WHERE cubicle\_number='{cubicle\_number}' AND"

f" tp\_id IN (SELECT id FROM {DB.tutor\_period} WHERE

tutor\_username='{tutor\_username}')")

Explanation: This query deletes all the cubicles that are assigned to the input user. This query will be executed whenever a user uses the “Unassign Cubicle” functionality from the webpage.

1. **Screenshots**

****

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**