```
from datetime import datetime as dt
import os
from sqlalchemy import *
from sqlalchemy.pool import NullPool
from flask import Flask, request, render_template, g, redirect, Response

tmpl_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'templates')
app = Flask(__name__, template_folder=tmpl_dir)


DATABASEURI = "postgresql://vf2272:3005@34.74.246.148/proj1part2"


engine = create_engine(DATABASEURI)


@app.before_request
def before_request():
  try:
    g.conn = engine.connect()
  except:
    print("uh oh, problem connecting to database")
    import traceback; traceback.print_exc()
    g.conn = None

@app.teardown_request
def teardown_request(exception):
  try:
    g.conn.close()
  except Exception as e:
    pass


@app.route('/')
def index():

  print(request.args)


  cursor = g.conn.execute("SELECT name FROM Students")
  names = []
  for result in cursor:
    names.append(result['name'])  # can also be accessed using result[0]
  cursor.close()

  context = dict(data = names)


  return render_template("index.html", **context)

@app.route('/another')
def another():
  return render_template("another.html")


@app.route('/add', methods=['POST'])
def add():
  name = request.form['name']
  g.conn.execute('INSERT INTO test(name) VALUES (%s)', name)
  return redirect('/')

@app.route('/signin', methods=['POST'])
def signin():
  uni = request.form['uni']
  return (signIn(uni))

@app.route('/error')
def error():
  return render_template("error.html")

@app.route('/register', methods=['POST'])
def register():
  uni = request.form['uni']
  name = request.form['name']
  email = request.form['email']
  year = request.form['year']
```

```
  g.conn.execute('INSERT INTO Students(uni, name, email, year) VALUES (%s, %s, %s, %s)', uni, name, email, year)
  return signIn(uni)

@app.route('/follow', methods=["POST"])
def follow():
  uni = request.form['uni']
  cname = request.form['cname']
  prof = request.form['prof']
  term = request.form['term']
  g.conn.execute('INSERT INTO Follows(uni, className, professor, term) VALUES (%s, %s, %s, %s)', uni, cname, prof, term)
  return (signIn(uni))

@app.route('/rate', methods=["POST"])
def rate():
  rating = request.form['rating']
  uni = request.form['uni']
  cname = request.form['cname']
  prof = request.form['prof']
  term = request.form['term']
  cursor = g.conn.execute('SELECT * FROM Rates R WHERE R.uni = %s AND R.className = %s AND R.term = %s AND R.professor = %s', [uni, cname, ter
  ratings = []
  for i in cursor:
    ratings.append(i)
  if len(ratings) == 0:
    g.conn.execute('INSERT INTO Rates(Rating, uni, className, professor, term) VALUES (%s, %s, %s, %s, %s)', [rating, uni, cname, prof, term]
  else:
    g.conn.execute('UPDATE Rates SET Rating = %s WHERE uni = %s AND className = %s AND professor = %s AND term = %s', [rating, uni, cname, pr
  return (signIn(uni))

@app.route('/download', methods=["POST"])
def download():
  uni = request.form['uni']
  cname = request.form['cname']
  prof = request.form['prof']
  term = request.form['term']
  pNum = request.form['pNum']
  g.conn.execute('INSERT INTO Saved(uni, postNumber, className, professor, term) VALUES (%s, %s, %s, %s, %s)', [uni, pNum, cname, prof, term]
  return (signIn(uni))

@app.route('/upload', methods=["POST"])
def upload():
  uni = request.form['uni']
  cname = request.form['cname']
  prof = request.form['prof']
  term = request.form['term']
  com = request.form['com']
  cursor = g.conn.execute('SELECT MAX(postNumber) FROM post_belongs_uploads P WHERE P.className = %s AND P.professor = %s AND P.term = %s', cn
  for i in cursor:
    maxNum = i
  if maxNum[0] != None:
    postNum = maxNum[0] + 1
  else:
    postNum = 1
  postTime = dt.now().strftime('%Y-%m-%d %H:%M:%S')
  g.conn.execute("INSERT INTO post_belongs_uploads(uni, className, professor, term, postNumber, postTime) VALUES (%s, %s, %s, %s, %s, %s)", un
  g.conn.execute('INSERT INTO Comment(postNumber, name, professor, term, content) VALUES (%s, %s, %s, %s, %s)', postNum, cname, prof, term, co
  return signIn(uni)

@app.route('/updateName', methods=["POST"])
def updateName():
  uni = request.form['uni']
  newName = request.form['newName']
  g.conn.execute('UPDATE Students SET name = %s WHERE uni = %s', newName, uni)
  return signIn(uni)

@app.route('/updateEmail', methods=["POST"])
def updateEmail():
  uni = request.form['uni']
  newEmail = request.form['newEmail']
  g.conn.execute('UPDATE Students SET email = %s WHERE uni = %s', newEmail, uni)
  return signIn(uni)

@app.route('/updateYear', methods=["POST"])
def updateYear():
  uni = request.form['uni']
  newYear = request.form['newYear']
  g.conn.execute('UPDATE Students SET year = %s WHERE uni = %s', newYear, uni)
  return signIn(uni)
```

```
    return signIn(uni)

@app.route('/search', methods=["POST"])
def search():
  uni = request.form['uni']
  maxR = request.form['maxR']
  minR = request.form['minR']
  dept = request.form['dept']
  name = request.form['name']
  prof = request.form['prof']
  term = request.form['term']
  if len(maxR) == 0:
    maxR = 10
  if len(minR) == 0:
    minR = 0
  cursor = g.conn.execute('SELECT DISTINCT (R.className, R.professor, R.term)   FROM Rates R WHERE R.Rating >= %s AND R.Rating <= %s', minR, r
  ratedClasses = []
  for i in cursor:
    ratedClasses.append(set(i))
  deptClasses = []
  if len(dept) != 0:
    dept = "%" + dept + "%"
    cursor = g.conn.execute("SELECT DISTINCT (className, professor, term) FROM class_partof C WHERE C.deptName LIKE %s", dept)
    for i in cursor:
      deptClasses.append(set(i))
  else:
    cursor = g.conn.execute("SELECT DISTINCT (className, professor, term) FROM class_partof")
    for i in cursor:
      deptClasses.append(set(i))
  namedClasses = []
  if len(name) != 0:
    name = "%" + name + "%"
    cursor = g.conn.execute("SELECT DISTINCT (className, professor, term) FROM class_partof C WHERE C.className LIKE %s", name)
    for i in cursor:
      namedClasses.append(set(i))
  else:
    cursor = g.conn.execute("SELECT DISTINCT (className, professor, term) FROM class_partof")
    for i in cursor:
      namedClasses.append(set(i))
  profClasses = []
  if len(prof) != 0:
    prof = "%" + prof + "%"
    cursor = g.conn.execute("SELECT DISTINCT (className, professor, term) FROM class_partof C WHERE C.professor LIKE %s", prof)
    for i in cursor:
      profClasses.append(set(i))
  else:
    cursor = g.conn.execute("SELECT DISTINCT (className, professor, term) FROM class_partof")
    for i in cursor:
      profClasses.append(set(i))
  termClasses = []
  if len(term) != 0:
    term = "%" + term + "%"
    cursor = g.conn.execute("SELECT DISTINCT (className, professor, term) FROM class_partof C WHERE C.term LIKE %s", term)
    for i in cursor:
      termClasses.append(set(i))
  else:
    cursor = g.conn.execute("SELECT DISTINCT (className, professor, term) FROM class_partof")
    for i in cursor:
      termClasses.append(set(i) )
  displayed = []
  if minR == 0 and maxR == 10:
    for c in deptClasses:
      if c in profClasses and c in termClasses and c in namedClasses:
        displayed.append(c)
  else:
    for c in deptClasses:
      if c in ratedClasses and c in profClasses and c in termClasses and c in namedClasses:
        displayed.append(c)
  for i in range(len(displayed)):
    displayed[i] = list(displayed[i])
    displayed[i] = displayed[i][0]
  displayed = list(displayed)
  cursor = g.conn.execute('SELECT uni, name, email, year FROM Students S WHERE S.uni = %s',  uni)
  acc = []
  for i in cursor:
    acc.append(i)
  cursor = g.conn.execute('SELECT Rating, className, professor, term FROM Rates R WHERE R.Uni = %s', uni)
  rates = []
```

```
    for i in cursor:
      rates.append(i)
    cursor = g.conn.execute('SELECT  className, professor, term FROM Follows F WHERE F.uni = %s ', uni)
    followed = []
    for i in cursor:
      followed.append(i)
    posts = []
    for f in followed:
      cursor = g.conn.execute('SELECT * FROM Comment C WHERE C.name = %s AND C.professor = %s AND C.term = %s',  f[0], f[1], f[2])
      for i in cursor:
        posts.append(i)
    cursor = g.conn.execute('SELECT DISTINCT C.name, C.postNumber, C.professor, C.term, C.content FROM Comment C, post_belongs_uploads P WHERE (
    uploaded = []
    for i in cursor:
      uploaded.append(i)
    cursor = g.conn.execute('SELECT S.postNumber, S.className, S.professor, S.term, C.content FROM Saved S, Comment C WHERE S.uni = %s AND S.pos
    saved = []
    for i in cursor:
      saved.append(i)
    divs = [acc, rates, followed, posts, uploaded, saved, list(displayed)]
    if len(acc) > 0:
      context = dict(data = divs)
      return render_template("student.html", **context)
    else:
      return redirect("/error")

  @app.route('/login')
  def login():
      abort(401)
      this_is_never_executed()

  def signIn(uni):
    cursor = g.conn.execute('SELECT uni, name, email, year FROM Students S WHERE S.uni = %s',  uni)
    acc = []
    for i in cursor:
      acc.append(i)
    cursor = g.conn.execute('SELECT Rating, className, professor, term FROM Rates R WHERE R.Uni = %s', uni)
    rates = []
    for i in cursor:
      rates.append(i)
    cursor = g.conn.execute('SELECT  className, professor, term FROM Follows F WHERE F.uni = %s ', uni)
    followed = []
    for i in cursor:
      followed.append(i)
    posts = []
    for f in followed:
      cursor = g.conn.execute('SELECT * FROM Comment C WHERE C.name = %s AND C.professor = %s AND C.term = %s',  f[0], f[1], f[2])
      for i in cursor:
        posts.append(i)
    cursor = g.conn.execute('SELECT C.postNumber, C.name, C.professor, C.term, C.content FROM Comment C, post_belongs_uploads P WHERE C.name = I
    uploaded = []
    for i in cursor:
      uploaded.append(i)
    cursor = g.conn.execute('SELECT S.postNumber, S.className, S.professor, S.term, C.content FROM Saved S, Comment C WHERE S.uni = %s AND S.pos
    saved = []
    for i in cursor:
      saved.append(i)
    divs = [acc, rates, followed, posts, uploaded, saved]
    if len(acc) > 0:
      context = dict(data = divs)
      return render_template("student.html", **context)
    else:
      return redirect("/error")

  if __name__ == "__main__":
    import click

    @click.command()
    @click.option('--debug', is_flag=True)
    @click.option('--threaded', is_flag=True)
    @click.argument('HOST', default='0.0.0.0')
    @click.argument('PORT', default=8111, type=int)
    def run(debug, threaded, host, port):
      """
      This function handles command line parameters.
      Run the server using:
```

```
        python3 server.py

    Show the help text using:

        python3 server.py --help

    """

    HOST, PORT = host, port
    print("running on %s:%d" % (HOST, PORT))
    app.run(host=HOST, port=PORT, debug=debug, threaded=threaded)

run()
```