Due: Wednesday, Nov 2
Assignment: Week 2 Day 3: Python: USERS CR
This solution includes modularization, though it was not required for the assignment.
This solution is complete.

- [ ] Complete SolutionObjective:
- [ ] Practice connecting Flask to database
- [ ] Practice display and creating data from/into the database
- [ ] users table in MySQL
    - [ ] id INT
    - [ ] first_name VARCHAR(45)
    - [ ] last_name VARCHAR(45)
    - [ ] email VARCHAR(45)
    - [ ] created_at DATETIME
    - [ ] updated_at DATETIME
- [ ] localhost:5000/user/new (Create page)
    - [ ] Add User (title of the page)
    - [ ] First Name:
    - [ ] Input text box
    - [ ] Last Name:
    - [ ] Input text box
    - [ ] Email:
    - [ ] Input text box
    - [ ] Add User button → click → go to E.
- [ ] localhost:5000/users (Read (All) page)
- [ ] Here are our Users!!! (title of the page)
- [ ] (Table Columns) First Name Last Name Email Created At
- [ ] (Table Row) Adrien Dion adion@gmail.com 2021 - 09 - 08
- [ ] (Table Row) Mr. Nibbles Pancakes nibs@pancakes.org 2021-09-08
- [ ] Add a User button → not clicked

- [ ] Use the users_schema created in the MySQL course
    - [ ] Start MySQL server from Apple → System Settings
    - [ ] Open MySQL Workbench
    - [ ] File → Open Recent → users_schema
    - [ ] Also stored as users_schema.mwb in w2d1 folder in Python in Documents
- [ ] Create a new Flask project
    - [ ] Create the folder directory in CodingDojo
    - [ ] Name the parent folder users_CR, save in w2d3
    - [ ] Open Folder in VSC

- [ ] Click on the folder
- [ ] Create the virtual environment
    - [ ] Command shift C
    - [ ] pipenv install flask pyMySQL
    - [ ] pipenv shell
- [ ] Modularize the directory first
    - [ ] Create a folder called flask_app inside users_CR
    - [ ] Create __init__.py file inside flask_app folder
    - [ ] Insert this code:
        - [ ] #__init__.py
        - [ ] from flask import Flask
        - [ ] app = Flask(__name__)
        - [ ] app.secret_key = “rootroot”
    - [ ] Create server.py with this code:
        - [ ] From flask_app import app
        - [ ] @app.route(‘/’)
        - [ ] def index():
        - [ ] (tab)return “Hello World”
        - [ ] Blank space
        - [ ] If __name__==”__main__”:
        - [ ] (tab) app.run(debug=True)
    - [ ] Create templates folder inside flask_app folder
    - [ ] Create static folder inside flask_app folder
    - [ ] Create config folder inside flask_app folder
    - [ ] Create controllers folder inside flask_app folder
    - [ ] Create mysqlconnection.py inside config folder. Include this code and update PW
        - [ ] #a cursor is the object we use to interact with the database
        - [ ] import pymysql.cursors
        - [ ] #this class will give us an instance of a connection to our database
        - [ ] Class MySQLConnection:
        - [ ] (tab) def__init__(self, db):
        - [ ] (tab)(tab) connection = pymysql.connect(host = ‘localhost’,
        - [ ] (tab—----------) user = ‘root’,
        - [ ] (tab—----------) password = ‘rootroot’,
        - [ ] (tab—----------) db = db,
        - [ ] (tab—----------) charset = ‘utf8mb4’,
        - [ ] (tab—----------) cursorclass = pymysql.cursors.DictCursor,
        - [ ] (tab—----------) autocommit = True)
        - [ ] #establish the connection to the database

- [ ] Self.connection = connection
- [ ] #the method to query the database
- [ ] def query_db(self, query, data = None):
- [ ] (tab) with self.connection.cursor() as cursor:
- [ ] (tab)(tab) try:
- [ ] (tab)(tab)(tab) query = cursor.mogrify(query, data)
- [ ] (Tab)(tab)(tab) print("Running Query:", query)
- [ ] Blank
- [ ] (tab)(tab)(tab) self.connection.close()
- [ ] #connectToMySQL receives the database we're using and uses it to create an instance of MySQLConnection
- [ ] def connectToMySQL(db):
- [ ] (tab) return MySQLConnection(db)

- [ ] The mysqlconnection.py is in the config folder.
- [ ] Create a .py file named after whatever we are controlling in a pluralization form
    - [ ] I named this file users.py and put it in the controllers folder
- [ ] Move all the @app.route functions into the controller file
- [ ] Insert this code in users.py
    - [ ] #users.py
    - [ ] from flask_app import app
    - [ ] from flask import render_template, redirect, request, session, flash
    - [ ] from user import User
- [ ] Remove above lines from server.py
- [ ] In server.py we include this line:
    - [ ] from flask_app import app
    - [ ] from flask_app.controllers import users
    - [ ] #...server.py
- [ ] Now all of our logic is in users.py, separate the tasks of the controller with that of the model. Make classes that correspond to our database tables.
- [ ] Our controller file will only handle rendering and rerouting and calling on the Model class to deal with the database.
- [ ] Create models folder inside flask_app folder
- [ ] Move the user.py file into the models folder. (I created a user.py inside models since I did not have one to move).
- [ ] Change the import statement in user.py
    - [ ] from flask_app.config.mysqlconnection import connectToMySQL
    - [ ] #user.py
    - [ ] class User:
    - [ ] (tab) def __init__(self,data):(tab)(tab)(tab) cursor.execute(query, data)
    - [ ] (tab)(tab)(tab) if query.lower().find("insert") >=0:

- [ ] #INSERT queries will return the ID NUMBER of the row inserted
- [ ] (tab)(tab)(tab) self.connection.commit()
- [ ] (tab)(tab)(tab) return cursor.lastrowid
- [ ] (tab)(tab)(tab) elif query.lower().find("select") >=0:
- [ ] #SELECT queries return data from database as LIST OF DICTIONARIES
- [ ] (tab)(tab)(tab) result = cursor.fetchall()
- [ ] (tab)(tab)(tab) return result
- [ ] (tab)(tab) else:
- [ ] #UPDATE and DELETE queries will return nothing
- [ ] (tab)(tab)(tab) self.connection.commit()
- [ ] (tab)(tab) except Exception as e:
- [ ] #if the query fails the method will return FALSE
- [ ] (tab)(tab)(tab) print("Something went wrong", e)
- [ ] (tab)(tab)(tab) return FALSE
- [ ] (tab)(tab) finally:
- [ ] #close the connection
- [ ]
- [ ] (tab)(tab) self.id = data['id']
- [ ] (tab)(tab) self.first_name = data['first_name']
- [ ] (tab)(tab) self.last_name = data['last_name']
- [ ] (tab)(tab) self.email = data['email']
- [ ] (tab)(tab) self.created_at = data['created_at']
- [ ] (tab)(tab) self.updated_at = data['updated_at']

- [ ] Have the controller call the Model class methods
- [ ] Update the User import statement in the controller
- [ ] Split up the task between Models and Controllers
    - [ ] Split the logic between querying the database,
    - [ ] And handling routing and rendering templates.
    - [ ] Split: Getting All Users : Controllers and Models
    - [ ] Controllers:
        - [ ] #users.py…
        - [ ] from flask_app.models.user import User
        - [ ] #gets all the users and returns them in a list of user objects.
        - [ ] @app.route('/users')
        - [ ] def users():
        - [ ] (tab)(tab) return
        - [ ] render_template('results.html', users=User.get_all())
    - [ ] Models:
        - [ ] #user.py…
        - [ ] #gets all the users and returns them in a list of user objects

- [ ] @classmethod
- [ ] def get_all(cls):
- [ ] (tab)(tab) query = “SELECT * FROM users”
- [ ] (tab)(tab) users_from_db =
- [ ] connectToMySQL(‘users’).query_db(query)
- [ ] (tab)(tab) users =[]
- [ ] (tab)(tab) for u in users_from_db:
- [ ] (tab)(tab)(tab) users.append(cls(u))
- [ ] (tab)(tab) return users

- [ ] Split: Creating A User: Controllers and Models
- [ ] Controllers:
    - [ ] #users.py…
    - [ ] From flask_app.models.user import User
    - [ ] #gets all the users and return them in a list of user objects
    - [ ] @app.route(‘/create/user’, methods = [‘POST’])
    - [ ] def create_user():
    - [ ] (tab)(tab) data = {
    - [ ] (tab)(tab)(tab) “first_name” : request.form[‘first_name’],
    - [ ] (tab)(tab)(tab) “last_name” : request.form[‘last_name’],
    - [ ] (tab)(tab)(tab) “email” : request.form[‘email’],
    - [ ] (tab)(tab)(tab) “created_at” : request.form[‘created_at’],
    - [ ] (tab)(tab)(tab) “updated_at” : request.form[‘updated_at’]
    - [ ] (tab)(tab)}
    - [ ] (tab)(tab)User.save(data)
    - [ ] (tab)(tab) return redirect(‘/users’)
- [ ] Models:
    - [ ] #user.py…
    - [ ] #gets all the users and returns them in a list of user objects
    - [ ] @classmethod
    - [ ] def save(cls, data):
    - [ ] (tab)(tab) query = “Insert INTO users (id, first_name, last_name, email, created_at, updated_at) VALUES (%(id)s, %(first_name)s, %(last_name)s, %(email)s, NOW(), NOW());”
    - [ ] (tab)(tab) user_id = connectToMySQL(‘users’).query_db(query,data)
    - [ ] return user_id

- [ ] Create 2 html pages, Read (All) and create (let’s call Read (All) results.html)
- [ ] Display all users from the database on the results page
- [ ] Display form to create new users on the create page
- [ ] When the form is submitted, a new user should be inserted into the database

- [ ] Redirect to the results page after creating a new user, and the user just created should appear in the table.