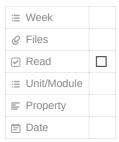
# **Guide to creating Web Apps with Flask**



# Creating tables, HTML, Login & Signup

#### Step 1:

 Create your new project with the required libraries: Flask, SQLAlchemy, Flask-SQLAlchemy, psycopg2-binary

#### Step 2:

• Copy the codes from main.py, admin.py & models.py

# Step 3:

- In main.py, change the port number to a free one
- let this line reflect the port number in your docker-compose.yml and the name should reflect the one in your env file
- postgresql://username:password@localhosy:

 $app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://username:password@localhost:port/dbname' app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://username' app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://username' app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://username' app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://username' app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql'://username' app.config['SQLALC$ 

# Step 4:

- Create your 'templates' directory and put the relevant html files in this includes the 'admin' directory with files in them
- · Copy the 'static' directory into your project folder

# Step 5:

- · Go through each route and edit the html files to match
- Edit the main.css to have the relevant colours & images

# Step 6 (Signup & Login):

- In the  $\underline{\text{models.py}}$  file, edit the 'User' model to have relevant fields
  - $\circ~$  Feel free to change the  $\underline{\ }$  tablename  $\underline{\ }$  to whatever you want your tablename to be
- $\bullet\,$  Create your database in pgadmin with the same name and port numbers specified in step 3
- Run the python console from the project's root and enter the following:

>>> from models import db
>>> db.create\_all()



db.create\_all() creates all the tables specified in <a href="models.py">models.py</a> file in the database, db.drop\_all() deletes all the tables

You can confirm by running a cli from your docker container and typing these

```
# psql -U postgres
psql (14.1 (Debian 14.1-1.pgdg110+1))
Type "help" for help.
postgres=# \c glo
You are now connected to database "glo" as user "postgres".
glo=# \d
              List of relations
Schema |
                      Type
                                    Owner
              Name
public | regusers
                         table
                                    postgres
public | regusers_id_seq | sequence | postgres
(2 rows)
glo=#
```

### Step 7:

• Set a secret key to be able to use the Flask session (You won't be able to login without one)

```
>>> import os
>>> os.urandom(24)
```

# Step 8:

• Check your signup and login functions to make sure they're working well

# Registering Entities (Students, users, products etc.) - here, customers

# Step 1:

- In your models.py, define the model for the entity
- · Run the code again:

```
>>> from models import db
>>> db.create_all()
```

# Step 2:

· Edit your form in your customers.html

For each form element, note the name, placeholder and 'required' status - these have to be consistent e.g.

- Also edit the elements to reflect the fields in your form
- Edit the tbody to reflect your entity

```
{% endfor %}
```

### Step 3:

• Create your "admin/customers/" and edit it to reflect it's now customers

```
@admin_page.route("/admin/customers/")
def customers():
    if not logged_in():
        return redirect(url_for('login', next='/admin/customers/'))

# username in session, continue
# get our registered products from the database
    customers = models.Customers.query.all()
    information = request.args.get('information', 'Here you can register customers')
    css = request.args.get('css', 'normal')
    return render_template('customers.html', title="Register Customers", information=information, css=css, customers=customers)
```

In your process\_entity\_add(),

Edit the variables to reflect the fields created in your form

```
Fname = request.form['First_name']
    Sname = request.form['Surname']
    Dob = request.form['Nate_of_Birth']
    Radd = request.form['Residential_address']
    Nity = request.form['Nationality']
    Nid = request.form['National_Identification_Number']

# let's write to the database
try:
    customer = models.Customers(First_name=Fname, Surname=Sname, Date_of_Birth=Dob, Residential_address=Radd, Nationality=Nity, Namodels.db.session.add(customer)
    models.db.session.commit()
```