

National University of Computer & Emerging
Sciences (NUCES) Islamabad

Secure Software Design – FALL 2024

Cyber Security Department

LAB 07

Instructor: Nawfal Waqar

Learning Outcomes

In this lab you are expected to learn the following:

- Building a Flask App from Scratch
- Making CRUD operations, using templates etc.

Setup for building a Flask Application

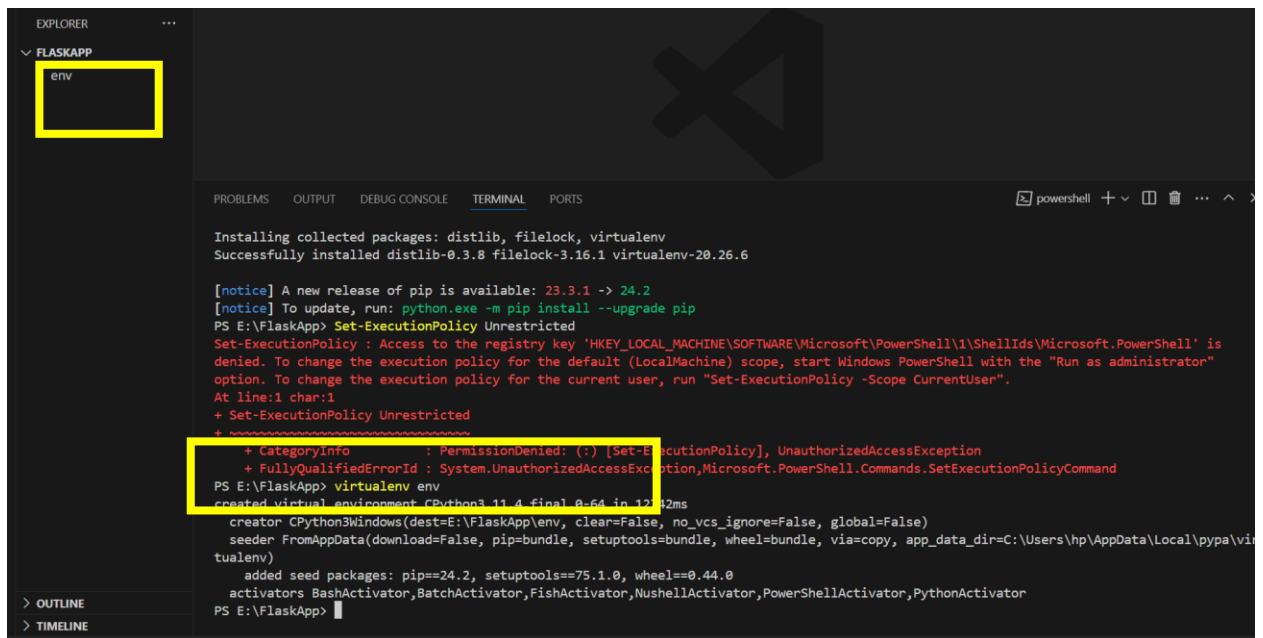
1. Install Visual Studio Code
2. Install latest version of Python on your Computer.
3. Install virtual environment by this command: **pip install virtualenv**
4. If it is giving some error, do this first,

Open Power Shell Admin and run this command:

Set-ExecutionPolicy Unrestricted

5. Now create a virtual environment using this command:

virtualenv env



```
EXPLORER
  FLASKAPP
    env

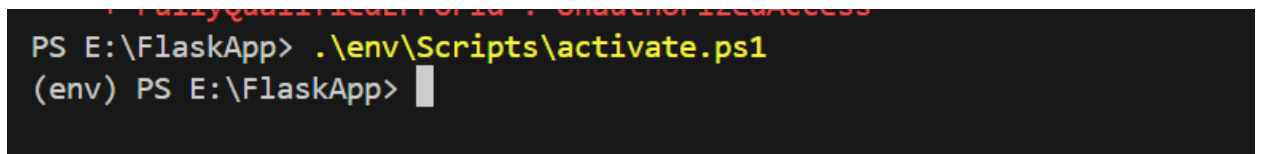
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell + - - - - -

Installing collected packages: distlib, filelock, virtualenv
Successfully installed distlib-0.3.8 filelock-3.16.1 virtualenv-20.26.6

[notice] A new release of pip is available: 23.3.1 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
PS E:\FlaskApp> Set-ExecutionPolicy Unrestricted
Set-ExecutionPolicy : Access to the registry key 'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell' is
denied. To change the execution policy for the default (LocalMachine) scope, start Windows PowerShell with the "Run as administrator"
option. To change the execution policy for the current user, run "Set-ExecutionPolicy -Scope CurrentUser".
At line:1 char:1
+ Set-ExecutionPolicy Unrestricted
+ ~~~~~
+ CategoryInfo          : PermissionDenied: (:) [Set-ExecutionPolicy], UnauthorizedAccessException
+ FullyQualifiedErrorId : System.UnauthorizedAccessException,Microsoft.PowerShell.Commands.SetExecutionPolicyCommand
PS E:\FlaskApp> virtualenv env
created virtual environment CPython3.11.4 final 8-64 in 12142ms
creator CPython3Windows(dest=E:\FlaskApp\env, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\hp\AppData\Local\pypa\vir
tualenv)
added seed packages: pip==24.2, setuptools==75.1.0, wheel==0.44.0
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
PS E:\FlaskApp>
```

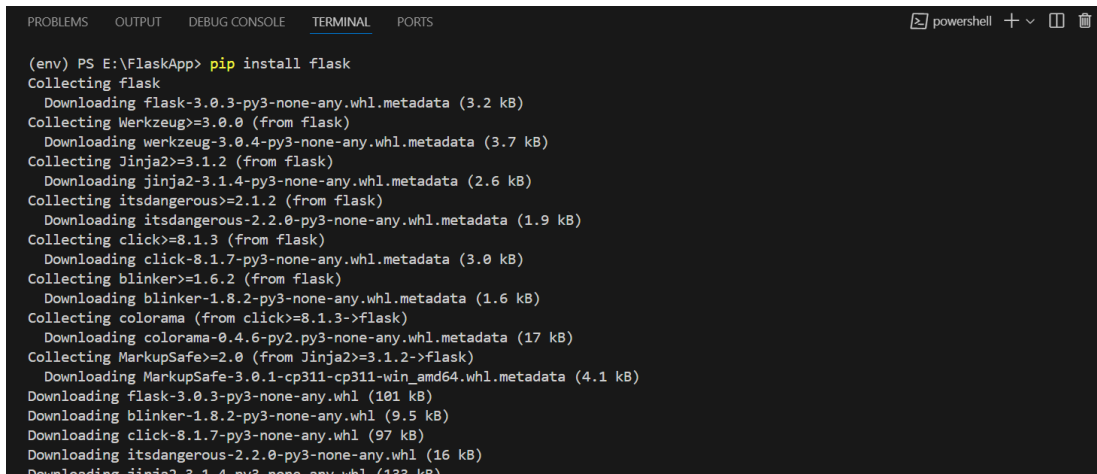
It will create a subfolder in your main folder named “env”, now we have a virtual environment to work in

6. Activating Virtual Environment:



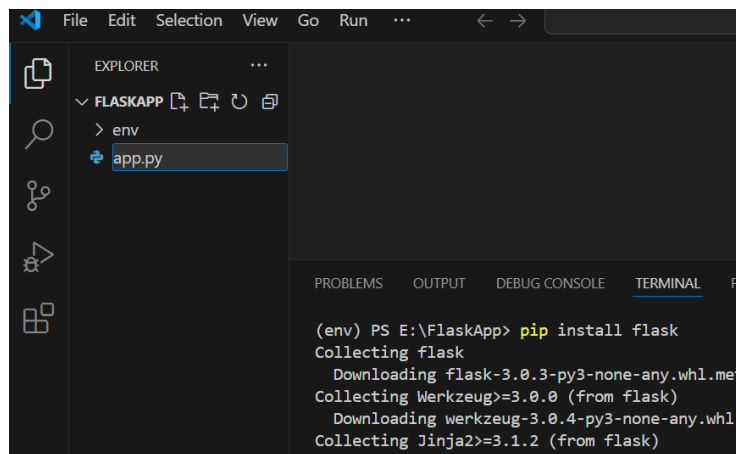
```
PS E:\FlaskApp> .\env\Scripts\activate.ps1
(env) PS E:\FlaskApp>
```

7. Installing Flask in this environment:



```
(env) PS E:\FlaskApp> pip install flask
Collecting flask
  Downloading flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting Werkzeug>=3.0.0 (from flask)
  Downloading werkzeug-3.0.4-py3-none-any.whl.metadata (3.7 kB)
Collecting Jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.1.2 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from flask)
  Downloading blinker-1.8.2-py3-none-any.whl.metadata (1.6 kB)
Collecting colorama (from click>=8.1.3->flask)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->flask)
  Downloading MarkupSafe-3.0.1-cp311-cp311-win_amd64.whl.metadata (4.1 kB)
Downloading flask-3.0.3-py3-none-any.whl (101 kB)
Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
Downloading click-8.1.7-py3-none-any.whl (97 kB)
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
```

8. Make a new file name **app.py**



9. Creating A minimal Flask App

- Go to <https://flask.palletsprojects.com/en/1.1.x/quickstart/>
- Copy the following code and paste it in your app.py file, it is making a simple hello world program in python.

• A Minimal Application

A minimal Flask application looks something like this:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

- Also write **main** to call the function:

```

app.py > ...
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/')
5  def hello_world():
6      return 'Hello, World!'
7
8
9
10 # writing main function to call the above defined function
11 # otherwise the program won't do anything
12 if __name__ == "__main__":
13     app.run(debug=True)

```

- iv. Now run the file using the following command:

```

(env) PS E:\FlaskApp> python .\app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
Python 3.11.4
(env) PS E:\FlaskApp> python .\app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!

```

- v. Open <http://localhost:5000/> on your browser to see the app running.

A screenshot of a web browser window. The address bar shows 'localhost:5000'. The page content displays 'Hello, World!' in a simple black font. The browser's tab bar shows several open tabs, including 'Apps', 'Classes', and various search engines and documents.

Congratulations! You have made your first Flask App

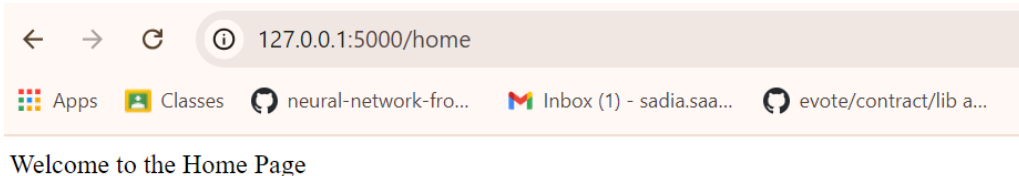
10. Creating another page in the application:

Make another **app.route()** with your desired name

```
app.py x
app.py > home
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/')
5  def hello_world():
6      return 'Hello, World!'
7
8  @app.route('/home')
9  def home():
10     return 'Welcome to the Home Page'
11
12
13
14  # writing main function to call the above defined function
15  # otherwise the program won't do anything
16  if __name__ == "__main__":
17     app.run(debug=True)
```

Save the code and the changes will be reflected in the app simultaneously

run localhost/home

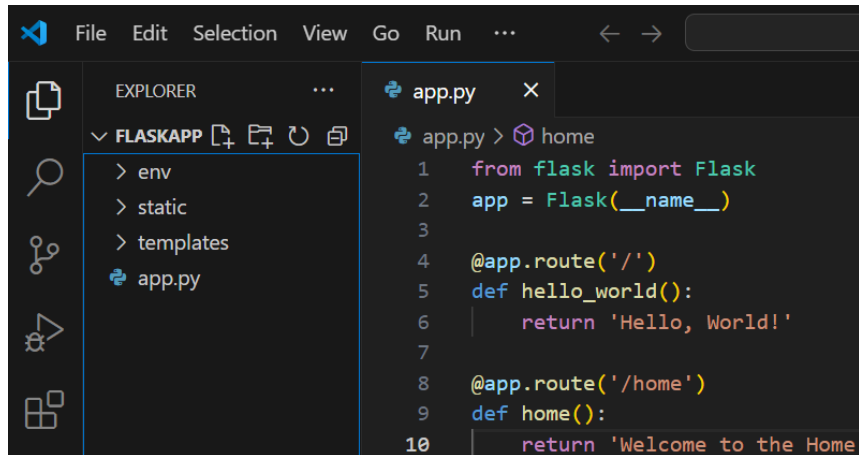


11. Now make 2 directories in the FLASKAPP folder:

static (to keep all the static files e.g.: images, text files etc.)

templates (to keep prebuilt templates)

Remember! Do not make them inside the env folder or nested with each other

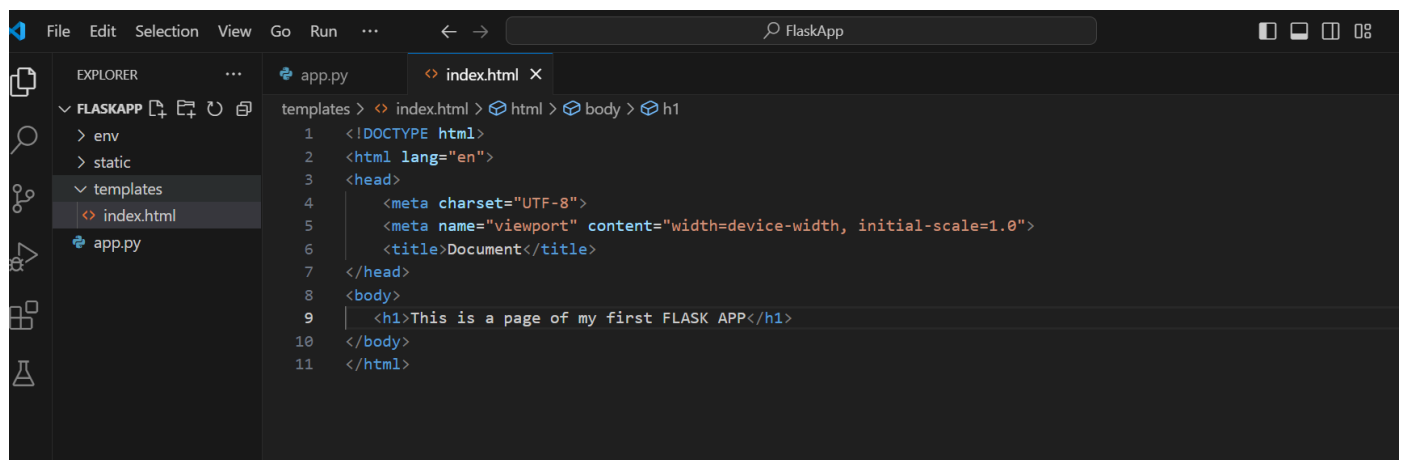


The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'FLASKAPP' with folders 'env', 'static', and 'templates', and a file 'app.py'. The main editor area shows the 'app.py' file with the following code:

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello_world():
6     return 'Hello, World!'
7
8 @app.route('/home')
9 def home():
10    return 'Welcome to the Home'
```

12. Make a file **index.html** in the templates folder

- Write **!** and Press **Enter**, it will write the basic starting code for you.
- Now write something in the **body tag** of the html file



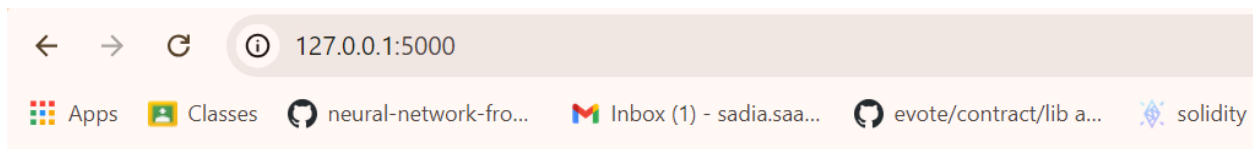
The screenshot shows the Visual Studio Code interface with the 'index.html' file open in the editor. The Explorer sidebar shows the 'templates' folder containing 'index.html'. The main editor area shows the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7 </head>
8 <body>
9     <h1>This is a page of my first FLASK APP</h1>
10 </body>
11 </html>
```

- Now we need to render this html so that it shows on the app
- Go to the **app.py** file and do the following
 - Import **render_template**
 - Write a return statement in **def hello_world ()**

```
app.py  X  index.html
app.py > hello_world
1  from flask import Flask, render_template
2  app = Flask(__name__)
3
4  @app.route('/')
5  def hello_world():
6      return render_template('Index.html')
7      #return 'Hello, World!'
8
9
10 @app.route('/home')
11 def home():
12     return 'Welcome to the Home Page'
13
```

- Now if you go to the browser and refresh you will see the html page's content



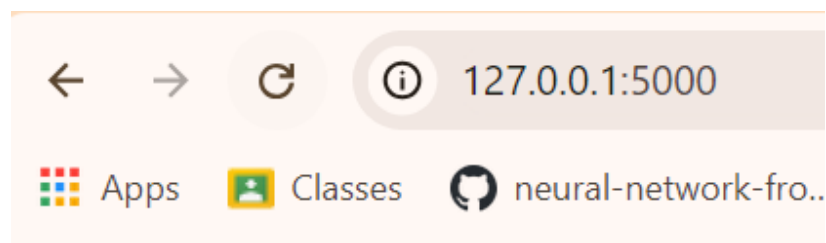
This is a page of my first FLASK APP

Beautification of the APP using HTML and CSS

- Go to <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
 - Copy the following code and paste it into your **index.html** file
2. **Include Bootstrap's CSS and JS.** Place the `<link>` tag in the `<head>` for our CSS, and the `<script>` tag for our JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing `</body>`. Learn more about our [CDN links](#).

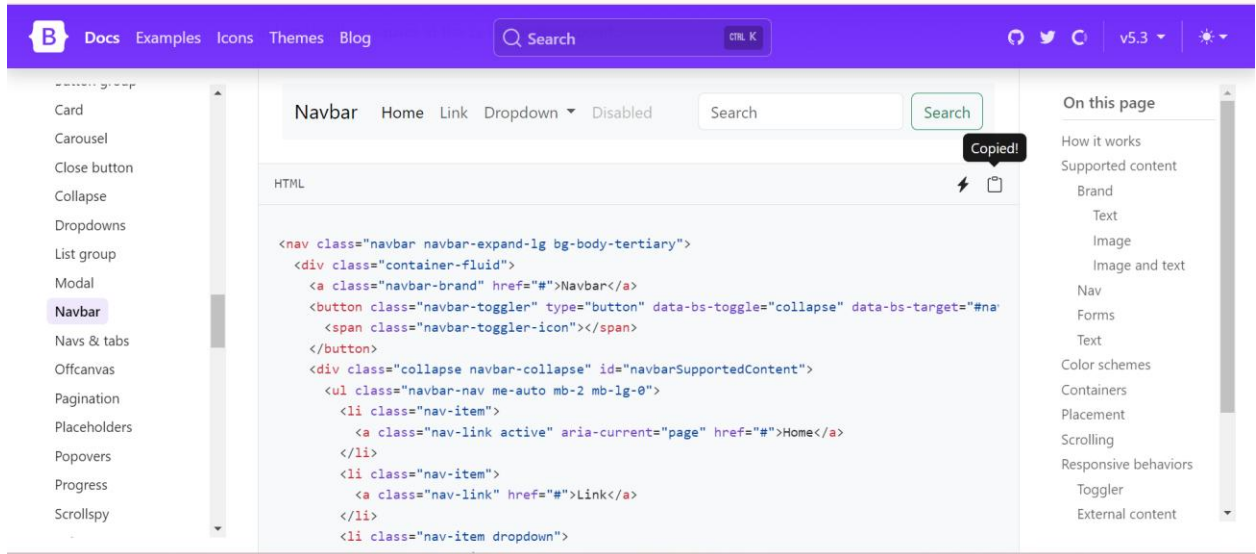
```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle"
  </body>
</html>
```

- Save the code and refresh the browser:



Hello, world!

- Now, suppose we have to make a Navigation Bar/ Menu in our website. Go to the components section on getbootstrap.com and click on navbar, copy the given code.



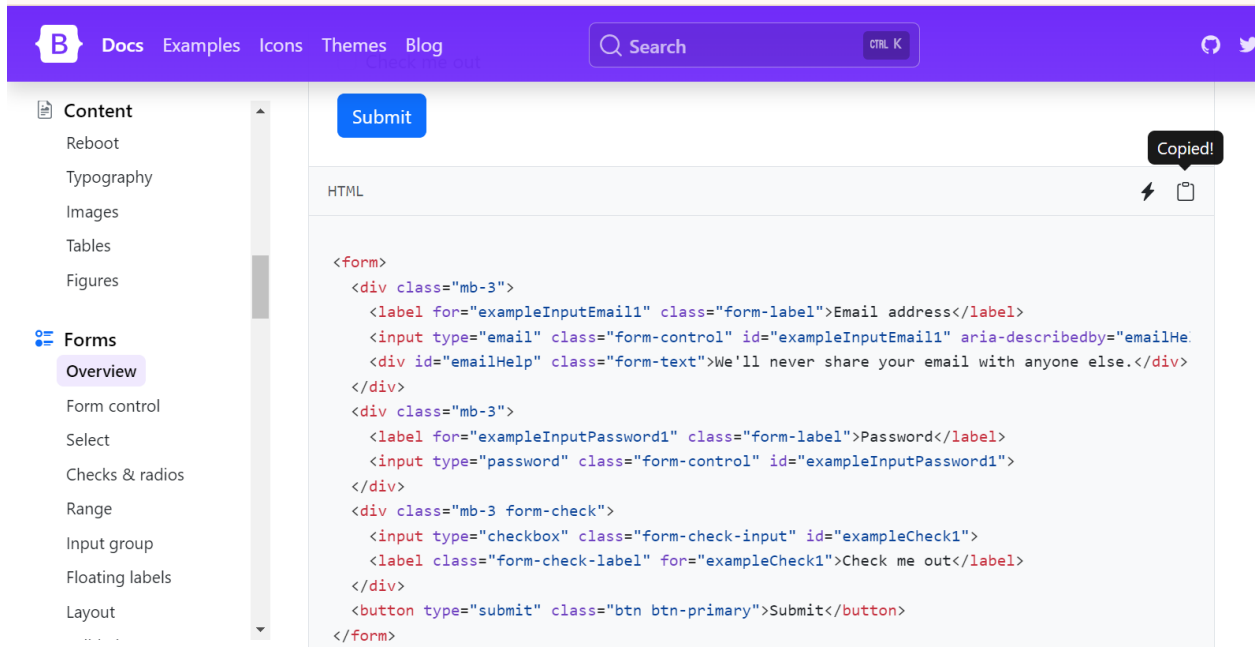
- Paste the code in the **body** tag of your html file and save the file.



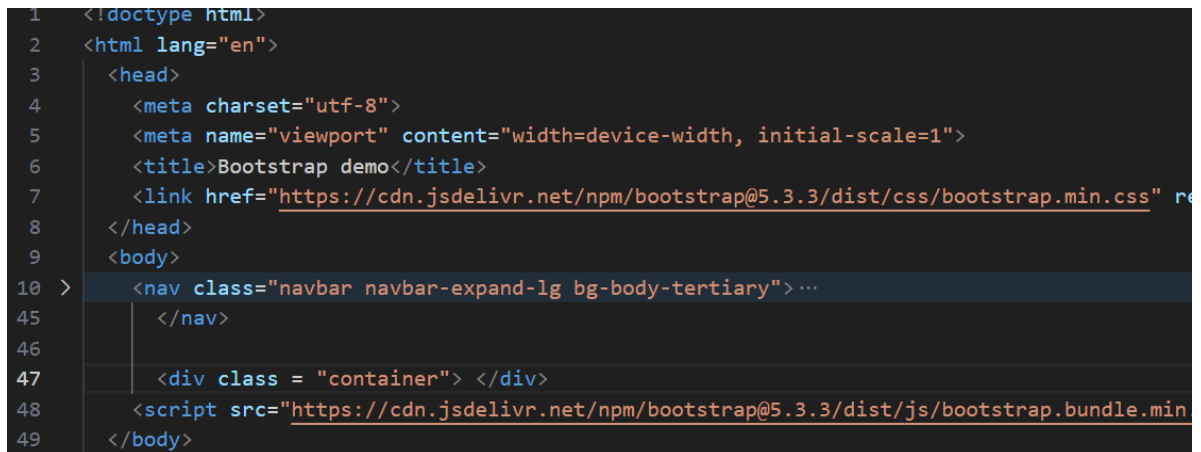
- Refresh the browser, now you have a navigation bar in your application.



- Now if we want to add a form on our page, go to the forms section on the website and copy the following code:



- In your Index.html file, make a container like this:



- Paste the copied code inside this div, save the file and refresh the browser. Now you have a form on your page:

Navbar Home Link Dropdown Disabled Search

Email address

We'll never share your email with anyone else.

Password

☐ Check me out

Submit

- Now suppose we want to add a table to show some data, go the website and search for tables, copy the code for making a table:

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">First</th>
      <th scope="col">Last</th>
      <th scope="col">Handle</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
```

On this page

- Overview
- Variants
- Accented tables
- Striped rows
- Striped columns
- Hoverable rows
- Active tables
- How do the variants and accented tables work?
- Table borders
- Bordered tables
- Tables without borders
- Small tables
- Table group dividers
- Vertical alignment

- Make another div container and paste this code there, save the code and refresh the browser, now we have a table

Navbar Home Link Dropdown Disabled Search

Email address

We'll never share your email with anyone else.

Password

☐ Check me out

Submit

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

Search

Creating a Database for our APP

- For FLASK, first we need to install SQL Alchemy, using the following command:

pip install flask-sqlalchemy

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(env) PS E:\FlaskApp> python .\app.py
• (env) PS E:\FlaskApp> pip install flask-sqlalchemy
Collecting flask-sqlalchemy
  Downloading flask_sqlalchemy-3.1.1-py3-none-any.whl.metadata (3.4 kB)
Requirement already satisfied: flask>=2.2.5 in e:\flaskapp\env\lib\site-packages (from flask-sqlalchemy) (3.0.3)
Collecting sqlalchemy>=2.0.16 (from flask-sqlalchemy)
  Downloading SQLAlchemy-2.0.35-cp311-cp311-win_amd64.whl.metadata (9.9 kB)
Requirement already satisfied: Werkzeug>=3.0.0 in e:\flaskapp\env\lib\site-packages (from flask>=2.2.5->flask-sqlalchemy) (3.0.4)
Requirement already satisfied: Jinja2>=3.1.2 in e:\flaskapp\env\lib\site-packages (from flask>=2.2.5->flask-sqlalchemy) (3.1.2)
Requirement already satisfied: itsdangerous>=2.1.2 in e:\flaskapp\env\lib\site-packages (from flask>=2.2.5->flask-sqlalchemy) (2.2.0)
Requirement already satisfied: click>=8.1.3 in e:\flaskapp\env\lib\site-packages (from flask>=2.2.5->flask-sqlalchemy) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in e:\flaskapp\env\lib\site-packages (from flask>=2.2.5->flask-sqlalchemy) (1.8.2)
Collecting typing-extensions>=4.6.0 (from sqlalchemy>=2.0.16->flask-sqlalchemy)
  Downloading typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Collecting greenlet!=0.4.17 (from sqlalchemy>=2.0.16->flask-sqlalchemy)
  Downloading greenlet-3.1.1-cp311-cp311-win_amd64.whl.metadata (3.9 kB)
Requirement already satisfied: colorama in e:\flaskapp\env\lib\site-packages (from click>=8.1.3->flask>=2.2.5->flask-sqlalchemy) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in e:\flaskapp\env\lib\site-packages (from Jinja2>=3.1.2->flask>=2.2.5->flask-sqlalchemy) (2.1.5)
```

- In app.py file, import sql alchemy:

```
app.py X index.html

app.py > ...
1 from flask import Flask, render_template
2 from flask_sqlalchemy import SQLAlchemy
3 app = Flask(__name__)
4
```

- Now, initialize SQL Alchemy and connect it with SQL Lite:

```
1 from flask import Flask, render_template
2 from flask_sqlalchemy import SQLAlchemy
3 from datetime import datetime
4
5 app = Flask(__name__)
6
7
8 app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///firstapp.db"
9 with app.app_context():
10     db = SQLAlchemy(app)
11
```

- In app.py file, make a class to define the structure of our data base like this:
Here I have defined 5 columns, their names and data types like we do in SQL normally.

__repr__ only returns the Serial Number and First Name when called.

```

app.py > ...
10 app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///firstapp.db"
11 with app.app_context():
12     db = SQLAlchemy(app)
13
14
15
16 #db = SQLAlchemy(app)
17
18
19 # now making a class to define the structure of our db
20
21 class FirstApp(db.Model):
22     ... sno = db.Column(db.Integer, primary_key=True, autoincrement=True)
23     ... fname = db.Column(db.String(100), nullable=False)
24     ... lname = db.Column(db.String(100), nullable=False)
25     ... email = db.Column(db.String(200), nullable=False)
26     ...
27
28     def __repr__(self):
29         ... return f"{self.sno} - {self.fname}"
30
31
32 @app.route('/')
33 def hello_world():

```

- Now to create db, open python interpreter:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

emy-3.1.1 greenlet-3.1.1 sqlalchemy-2.0.35 typing-extensions-4.12.2
(env) PS E:\FlaskApp> python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

```

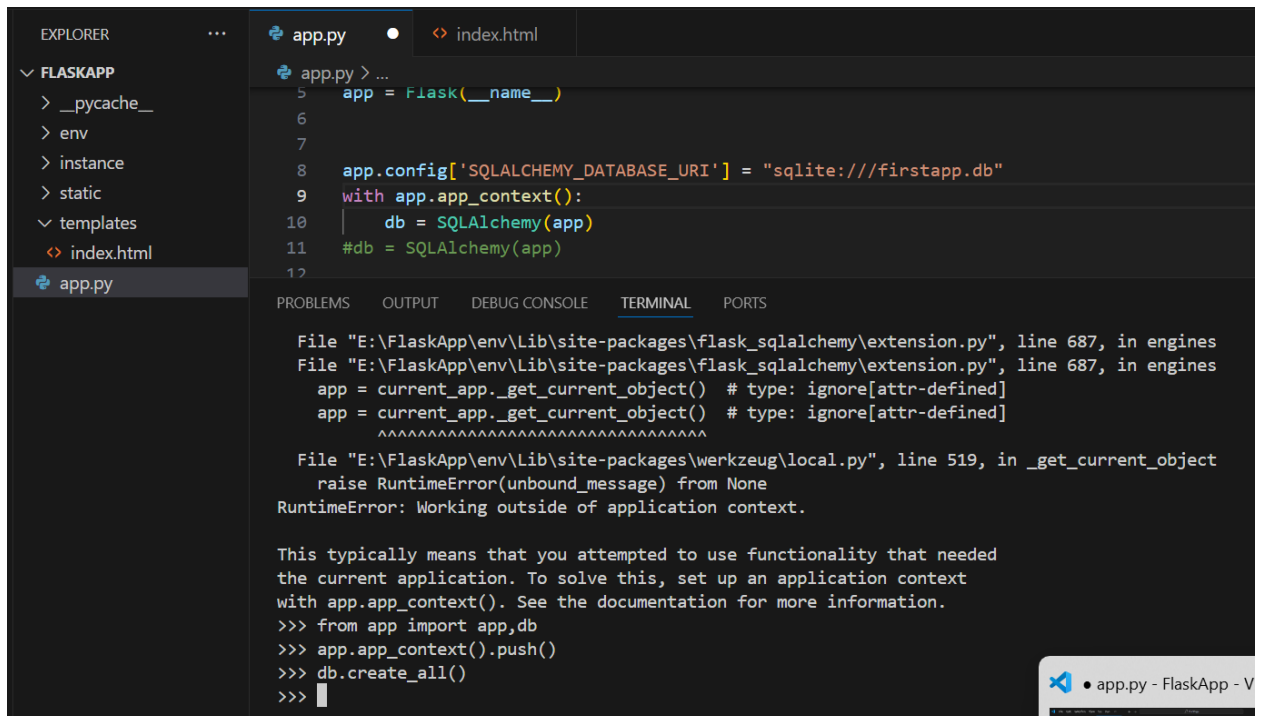
- **Write the following commands:**

```
from app import app,db
```

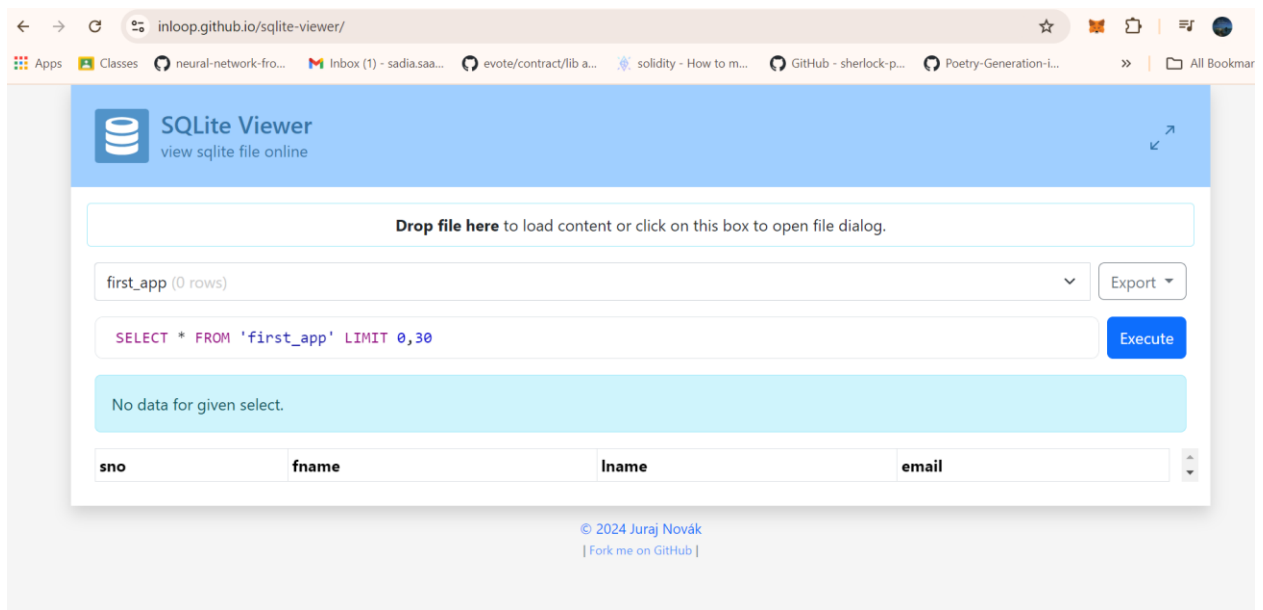
```
app.app_context().push()
```

```
db.create_all()
```

It will create a folder names instance with your db file inside it.



- If we want to view our data base, open sql lite viewer on Google, drag and drop firstapp.db file here and see:



- Now exit from the python interpreter with the command **exit()**

- Create an instance of db in app.route(), save and refresh the browser, with every refresh it will add an instance.

```

28
29 @app.route('/')
30 def hello_world():
31
32     firstapp = FirstApp(fname="Sadia",lname="Saad")
33     db.session.add(firstapp)
34     db.session.commit()
35
36     return render_template('Index.html')
37     #return 'Hello, World!'
38
39

```

- Now see the firstapp.db in the sqlite viewer again:

SQLite viewer
view sqlite file online

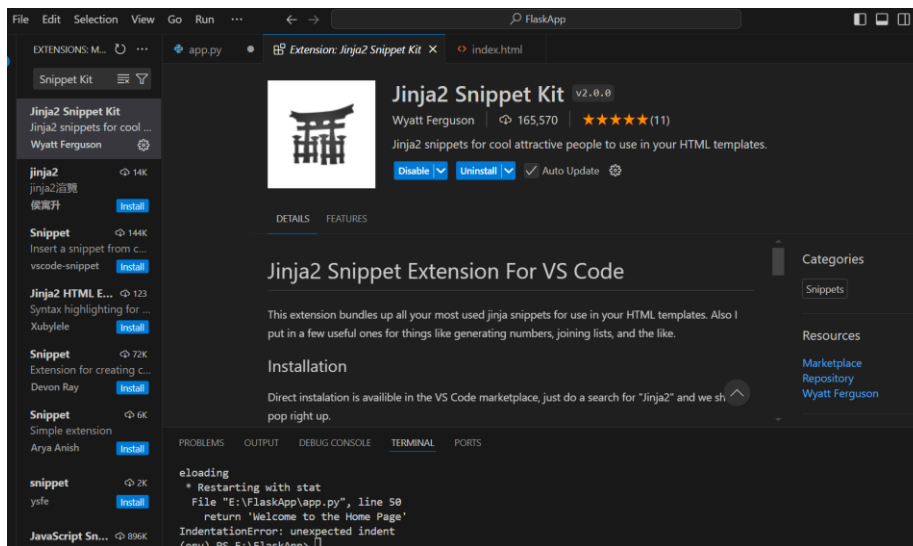
Drop file here to load content or click on this box to open file dialog.

first_app (8 rows) Export

Execute

sno	fname	lname	email
1	Sadia	Saad	sadia.saad@nu.edu.pk
6	Sadia	Saad	sadia.saad@nu.edu.pk
7	Sadia	Saad	sadia.saad@nu.edu.pk
8	Sadia	Saad	sadia.saad@nu.edu.pk
9	Sadia	Saad	sadia.saad@nu.edu.pk
10	Sadia	Saad	sadia.saad@nu.edu.pk
11	Sadia	Saad	sadia.saad@nu.edu.pk
12	Sadia	Saad	sadia.saad@nu.edu.pk

- Here, install an extension to easily make and use templates for FLASK



- To show all the records on the webpage do this, here we are getting all the records and sending them to the index.html page using return statement.

```
@app.route('/')
def hello_world():

    firstapp = FirstApp(fname="Sadia",lname="Saad",email="sa
    db.session.add(firstapp)
    db.session.commit()

    allpeople = FirstApp.query.all()
    #print(allpeople)

    return render_template('Index.html',allpeople=allpeople)
```

- In Index.html, Inside the tbody tag write the following loop counter,

```
</tr>
</thead>
<tbody>

    {% for people in allpeople %}
    <tr>
        <th scope="row">{{people.sno}}</th>
        <td>{{people.fname}}</td>
        <td>{{people.lname}}</td>
        <td>{{people.email}}</td>
    </tr>
    {% endfor %}

<tr>
    <th scope="row">2</th>
```


- It will show all the entries in the table:

☐ Check me out

#	First	Last	Handle
1	Sadia	Saad	sadia.saad@nu.edu.pk
6	Sadia	Saad	sadia.saad@nu.edu.pk
7	Sadia	Saad	sadia.saad@nu.edu.pk
8	Sadia	Saad	sadia.saad@nu.edu.pk
9	Sadia	Saad	sadia.saad@nu.edu.pk
10	Sadia	Saad	sadia.saad@nu.edu.pk
11	Sadia	Saad	sadia.saad@nu.edu.pk
12	Sadia	Saad	sadia.saad@nu.edu.pk
13	Sadia	Saad	sadia.saad@nu.edu.pk
14	Sadia	Saad	sadia.saad@nu.edu.pk
15	Sadia	Saad	sadia.saad@nu.edu.pk

Make the forms Working

- In Index.html, previously used form tag, add the following:

```

<body>
  <nav class="navbar navbar-expand-lg bg-body-tertiary"> ...
  </nav>

  <div class="container">
    <form action = "/" method="POST">
      <div class="mb-3">
        ...
      </div>
      <div class="mb-3"> ...
    </div>
  </div>

```

- Now, in app.py, add the following lines in app.route(), Now save the file and refresh the browser, press the submit button, you will see POST written in the terminal showing that it is a POST request.

```
app.py x index.html
app.py > hello_world
30
31
32 @app.route('/', methods = ['GET','POST'])
33 def hello_world():
34     if request.method=='POST':
35         print(request.form['fname'])
36
37
38 firstapp = FirstApp(fname="Sadia",lname="Saad",email="sadia.saad@nu.edu.pk")
39 db.session.add(firstapp)
40 db.session.commit()
41
42 allpeople = FirstApp.query.all()
43 print(allpeople)
44
45 return render_template('index.html',allpeople=allpeople)
46
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
127.0.0.1 - - [09/Oct/2024 21:08:43] "GET / HTTP/1.1" 200 -
* Detected change in 'E:\\FlaskApp\\app.py', reloading
* Restarting with stat
0 - Sadia, 21 - Sadia, 22 - Sadia, 23 - Sadia, 24 - Sadia, 25 - Sadia]
127.0.0.1 - - [09/Oct/2024 21:11:40] "GET / HTTP/1.1" 200 -Secure
Sadia, 11 - Sadia, 12 - Sadia, 13 - Sadia, 14 - Sadia, 15 - Sadia, 16 - Sadia, 17 - S
Sadia, 18 - Sadia, 19 - Sadia, 20 - Sadia, 21 - Sadia, 22 - Sadia, 23 - Sadia, 24 - Sadia, 25 - Sadia, 26 - Sadia]
127.0.0.1 - - [09/Oct/2024 21:12:00] "POST / HTTP/1.1" 200 -
```

- Now save these variables and then add them to the data base at run time.

```
@app.route('/', methods = ['GET','POST'])
def hello_world():
    if request.method=='POST':
        # Check if form fields exist
        fname = request.form.get('fname')
        lname = request.form.get('lname')
        email = request.form.get('email')
        if fname and lname and email:
            # Create and commit new record to the database
            firstapp = FirstApp(fname=fname, lname=lname, email=email)
            db.session.add(firstapp)
            db.session.commit()
        allpeople = FirstApp.query.all()
        print(allpeople)
        return render_template('index.html',allpeople=allpeople)
```

- Save the code and refresh the page.
Fill the form, click on submit.
Changes will be reflected in the table:

First Name

We'll never share your email with anyone else.

Last Name

Email address

Submit

23	Sadia	Saad	sadia.saad@nu.edu.pk
24	Sadia	Saad	sadia.saad@nu.edu.pk
25	Sadia	Saad	sadia.saad@nu.edu.pk
26	Sadia	Saad	sadia.saad@nu.edu.pk
27	Sadia	Saad	sadia.saad@nu.edu.pk
28	Sadia	Saad	sadia.saad@nu.edu.pk
29	laiba	imran	laiba.imran@gmail.com
30	xyz	abc	xyz@gmail.com
2	Jacob	Thornton	j@.com
3	Larry the Bird		

- To delete an entry from the form:

Go to the bootstrap components, and select the code for any button:

Layout
Validation
Components
Accordion
Alerts
Badge
Breadcrumb
Buttons
Button group
Card
Carousel
Close button
Collapse

Primary
Secondary
Success
Danger
Warning
Info
Light
Dark
Link

HTML

```

<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>

```

- Paste it in index.html like the following code.
- Also add a new column

```

<th scope="col"> Actions</th>
</tr>

<tbody>

  {% for people in allpeople %}
  <tr>
    <th scope="row">{{people.sno}}</th>
    <td>{{people.fname}}</td>
    <td>{{people.lname}}</td>
    <td>{{people.email}}</td>
    <td> <a href="delete/{{people.sno}}" type="button" class="btn btn-success">Delete</button></td>
  </tr>
  </tbody>

```

- Save it and refresh the browser.

Submit

#	First	Last	Handle	Actions
1	Sadia	Saad	sadia.saad@nu.edu.pk	Delete
6	Sadia	Saad	sadia.saad@nu.edu.pk	Delete
7	Sadia	Saad	sadia.saad@nu.edu.pk	Delete
8	Sadia	Saad	sadia.saad@nu.edu.pk	Delete

- Add another button named update and make them smaller using the class tag.

These are prebuilt tags in bootstrap:

```

{% for people in allpeople %}
<tr>
  <th scope="row">{{people.sno}}</th>
  <td>{{people.fname}}</td>
  <td>{{people.lname}}</td>
  <td>{{people.email}}</td>
  <td> <a href="update/{{people.sno}}" type="button" class="btn btn-outline-dark btn-sm mx-1">Update</button>
  <a href="delete/{{people.sno}}" type="button" class="btn btn-outline-dark btn-sm mx-1">Delete</button></td>
</tr>

```

- This is how the output looks like:

Submit				
#	First	Last	Handle	Actions
1	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
6	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
7	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
8	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
9	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
10	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>

- Now make 2 methods for Update and Delete so that the buttons perform some functionality:
 - Import redirect at the start of the app.py file
 - For **Delete** we need an int argument that is the serial number.
 - And after deletion we redirect it to homepage

```
@app.route('/delete/<int:sno >')
def delete(sno):
    #first fetching the record with sno
    allpeople = FirstApp.query.filter_by(sno=sno).first()
    #now deleting the record
    db.session.delete(allpeople)
    db.session.commit()

    return redirect("/")
```

- Now if I press delete a record will be deleted:

Email				
Submit				
#	First	Last	Handle	Actions
6	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
7	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
8	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
9	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
10	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>
11	Sadia	Saad	sadia.saad@nu.edu.pk	<button>Update</button> <button>Delete</button>

- For the Update function, make a function in app.py

```
@app.route('/update/<int:sno>')
def update(sno):
    #first fetching the record with sno
    allpeople = FirstApp.query.filter_by(sno=sno).first()

    return render_template('update.html',allpeople=allpeople)
```

- Make an empty update.html file inside templates folder
Paste the code of index.html on to it, remove the table making part.
Save the code, refresh the browser and click on update button, it will take you here:

Navbar Home Link Dropdown ▾ Disabled

Update People's Info

First Name

We'll never share your email with anyone else.

Last Name

Email address

Email

- Add the following line with the form tag:

```
<div class = "container" >
  <form action = "/update/{{allpeople.sno}}" method="POST">
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-label"> First Name</label>
      <input type="text" class="form-control" name="fname" id="exampleInputEmail1" aria-
      <div id="emailHelp" class="form-text">We'll never share your email with anyone else</div>
    </div>
  </form>
</div>
```

- Make the following changes to the update function:
Fetch the details and then change them in the database instance:

```

@app.route('/update/<int:sno>', methods = ['GET', 'POST'])
def update(sno):

    if request.method=='POST':
        # Check if form fields exist
        fname = request.form.get('fname')
        lname = request.form.get('lname')
        email = request.form.get('email')
        if fname and lname and email:
            # Create and commit new record to the database
            allpeople = FirstApp.query.filter_by(sno=sno).first()
            allpeople.fname=fname
            allpeople.lname=lname
            db.session.add(allpeople)
            db.session.commit()

    #first fetching the record with sno
    allpeople = FirstApp.query.filter_by(sno=sno).first()

    return render_template('update.html',allpeople=allpeople)

```

- Now click on update button, make changes, the changes will be reflected in the table:

34	xyz	abc	xyz@gmail.com	Update	Delete
35	xyz	abc	xyz@gmail.com	Update	Delete
36	xyz	abc	xyz@gmail.com	Update	Delete
37	Sadia	Saad	aaaaa@gmail.com	Update	Delete
38	Sadia	Saad	555@gmail.com	Update	Delete
39	Sadia	Saad	555@gmail.com	Update	Delete
40	Sadia	Saad	112233@gmail.com	Update	Delete
2	Jacob	Thomson	j@com		
3	Larry the Bird				

A BASIC FLASK Application with CRUD Operations in Ready!