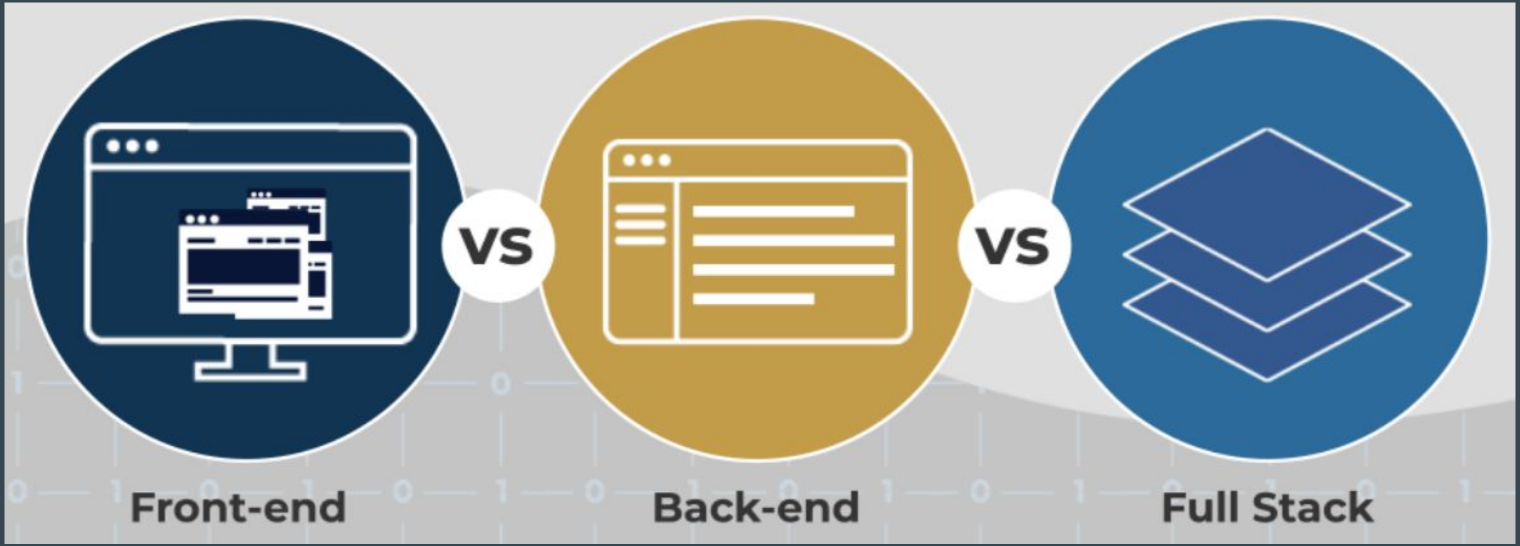
Smart Home Dashboard

•••

Dr. Dmitry Nikolayev,
eSportLab / CDISE / Skoltech
<<Introduction to IoT>>
2019

Introduction



Version 1.0 Single-Page WebApp with RESTfull API
Version 2.0 Multi-Page WebApp with Authorization
Version 3.0 Multi-Page WebApp with Interactive Chart

HyperText Transfer Protocol

1. The browser requests an HTML page. The server returns an HTML file.
2. The browser requests a style sheet. The server returns a CSS file.
3. The browser requests a JPG image. The server returns a JPG file.
4. The browser requests JavaScript code. The server returns a JS file
5. The browser requests data. The server returns data (in XML or JSON).

Resource
/device

GET
Read device details

POST
Create device

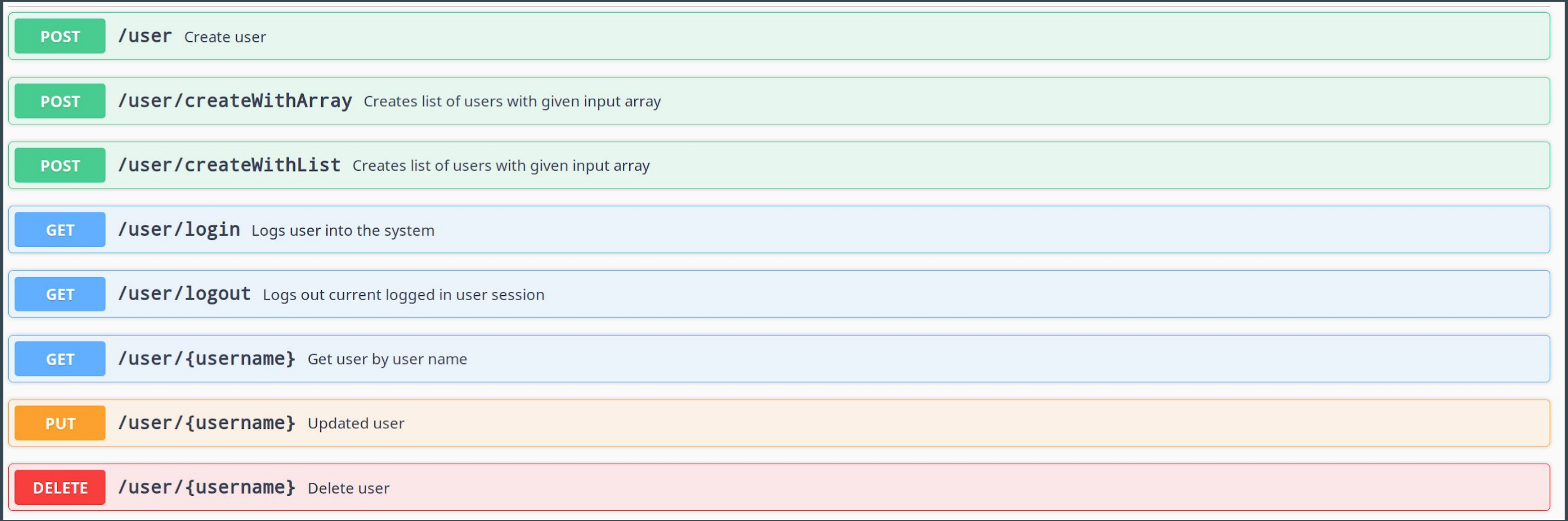
PUT
Update device details

GET
Read device list

DELETE
Delete device

REpresentational State Transfer (RESTfull) API

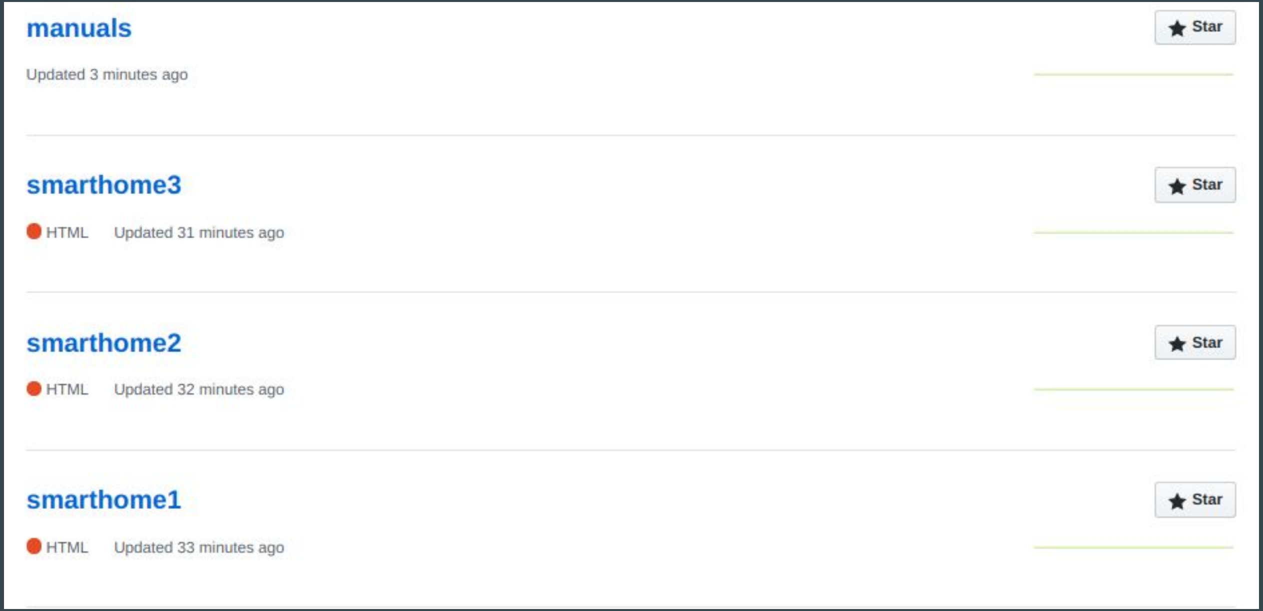
-- a software architectural style that defines a set of constraints to be used for creating Web services using HTTP requests to GET, PUT, POST and DELETE data. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations that share the following design properties: *performance, scalability, simplicity, modifiability, visibility, portability, reliability.*



POST	/user	Create user
POST	/user/createWithArray	Creates list of users with given input array
POST	/user/createWithList	Creates list of users with given input array
GET	/user/login	Logs user into the system
GET	/user/logout	Logs out current logged in user session
GET	/user/{username}	Get user by user name
PUT	/user/{username}	Updated user
DELETE	/user/{username}	Delete user

Check Out My Git Repositories

https://github.com/esportslab



manuals

Updated 3 minutes ago

smarthome3

● HTML Updated 31 minutes ago

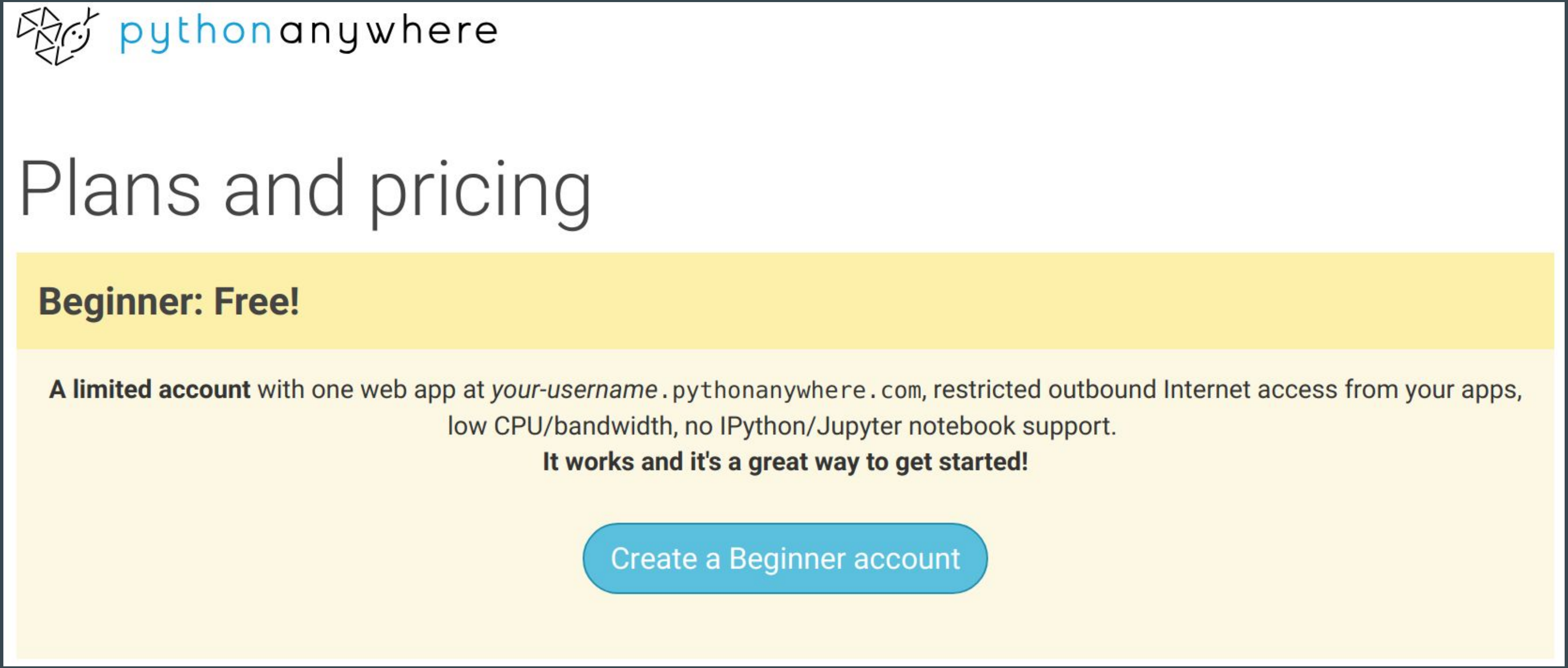
smarthome2

● HTML Updated 32 minutes ago

smarthome1

● HTML Updated 33 minutes ago

Register a Domain on pythonanywhere.com

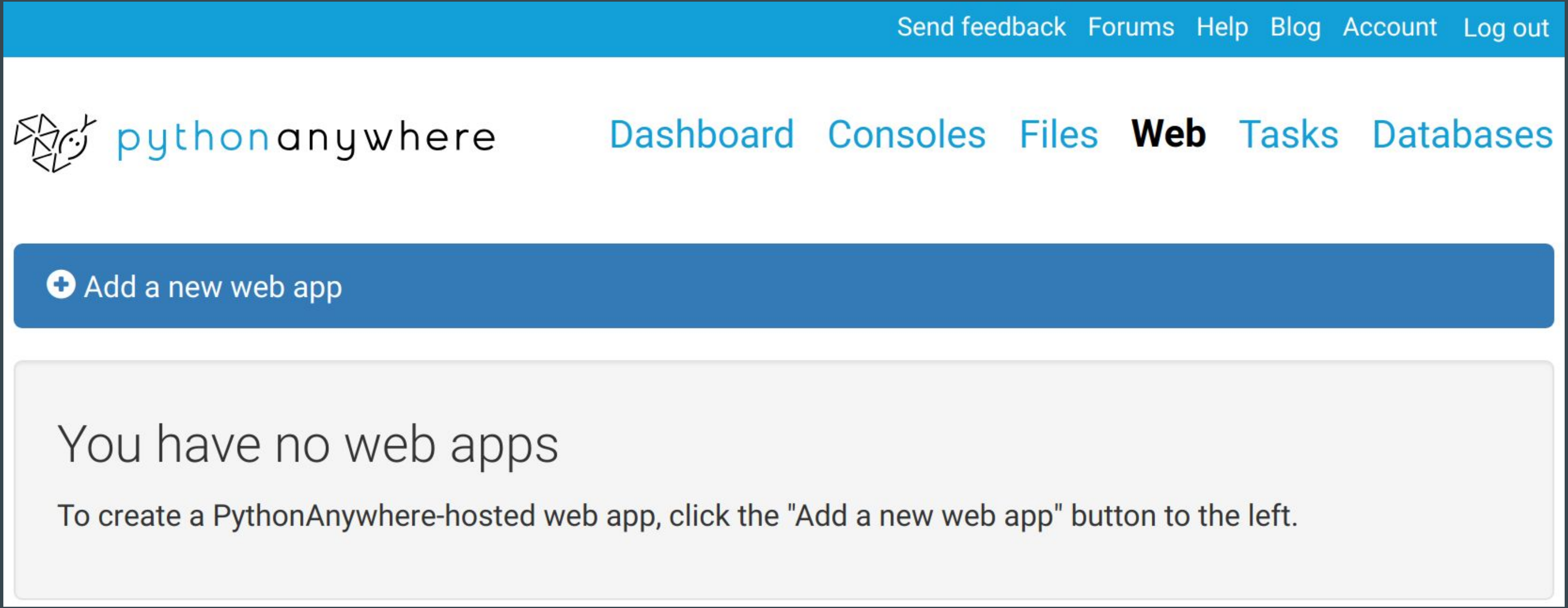


Plans and pricing

Beginner: Free!

A limited account with one web app at *your-username*`.pythonanywhere.com`, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython/Jupyter notebook support.
It works and it's a great way to get started!

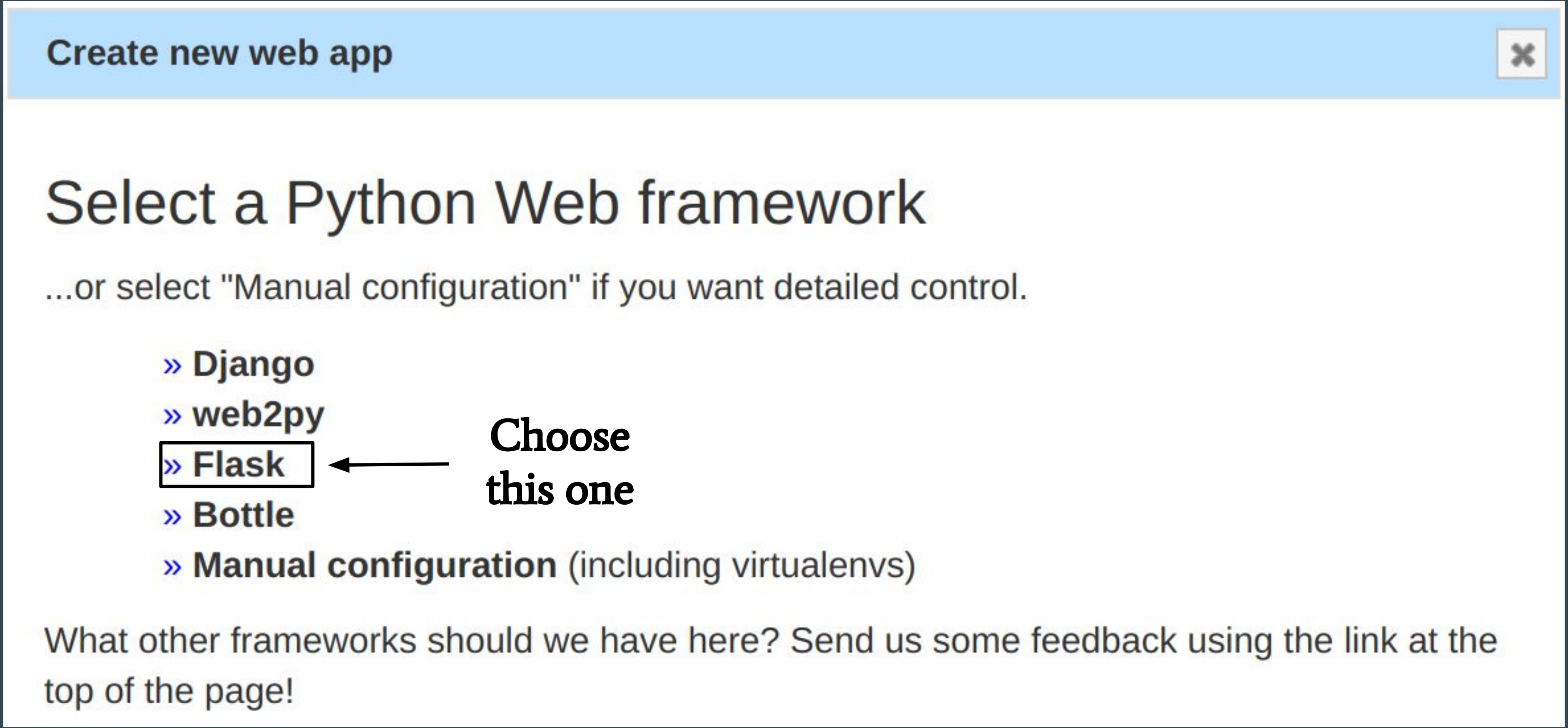
Add New Web Application



You have no web apps

To create a PythonAnywhere-hosted web app, click the "Add a new web app" button to the left.

Choose Bottle BackEnd on Python 3.7



Create new web app

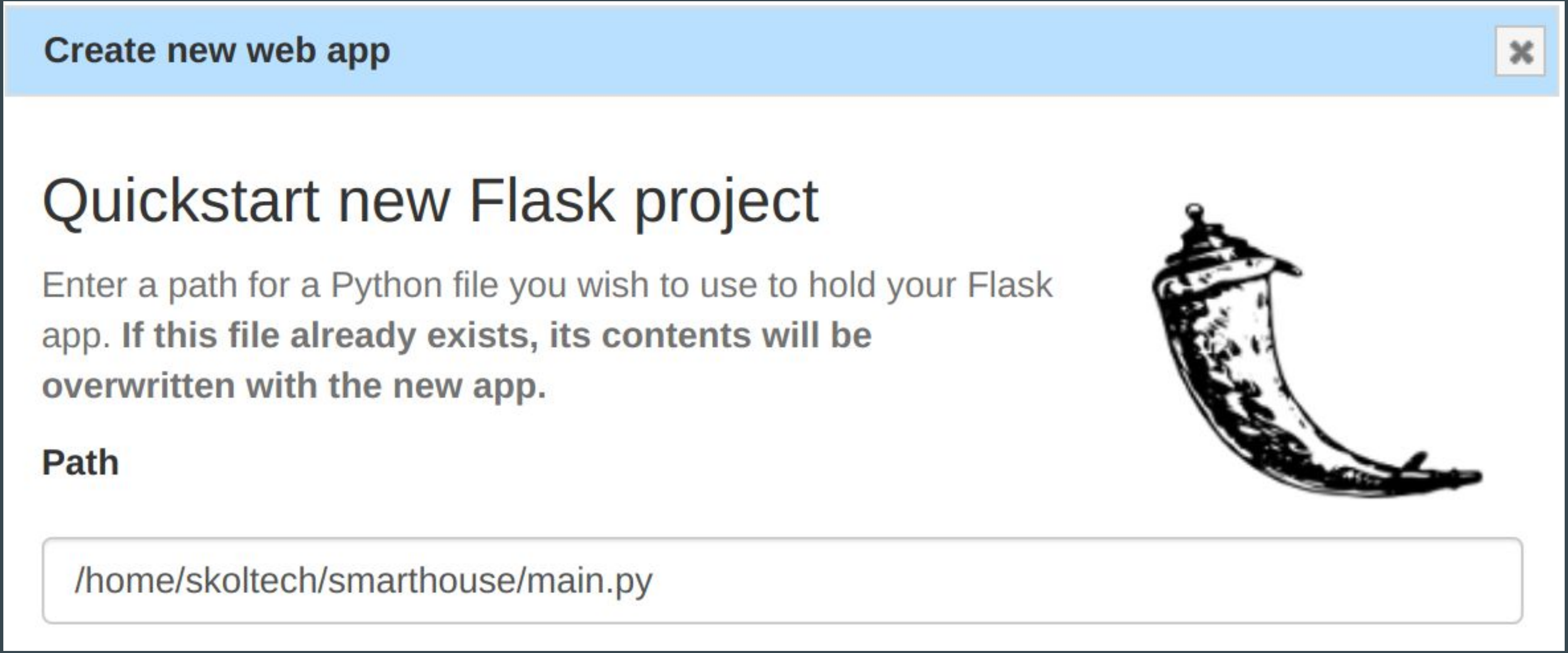
Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

» **Django**
» **web2py**
» **Flask**
» **Bottle**
» **Manual configuration** (including virtualenvs)

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

Change Folder Name and Entry Point



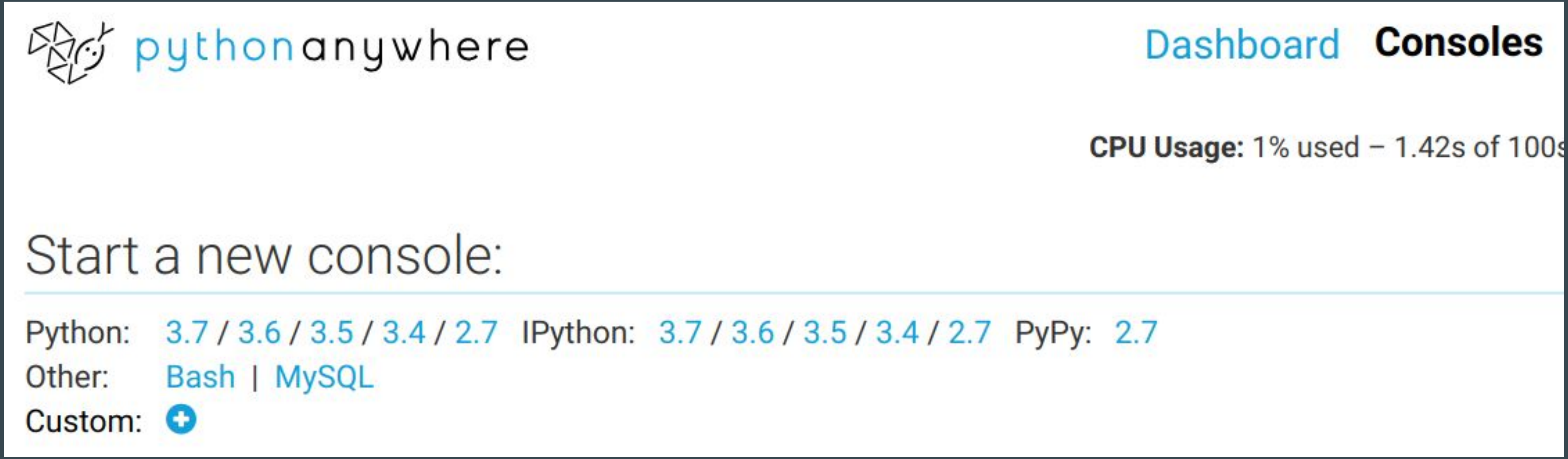
Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app. **If this file already exists, its contents will be overwritten with the new app.**

Path

/home/skoltech/smarthouse/main.py

Run Bash Console on Hosting

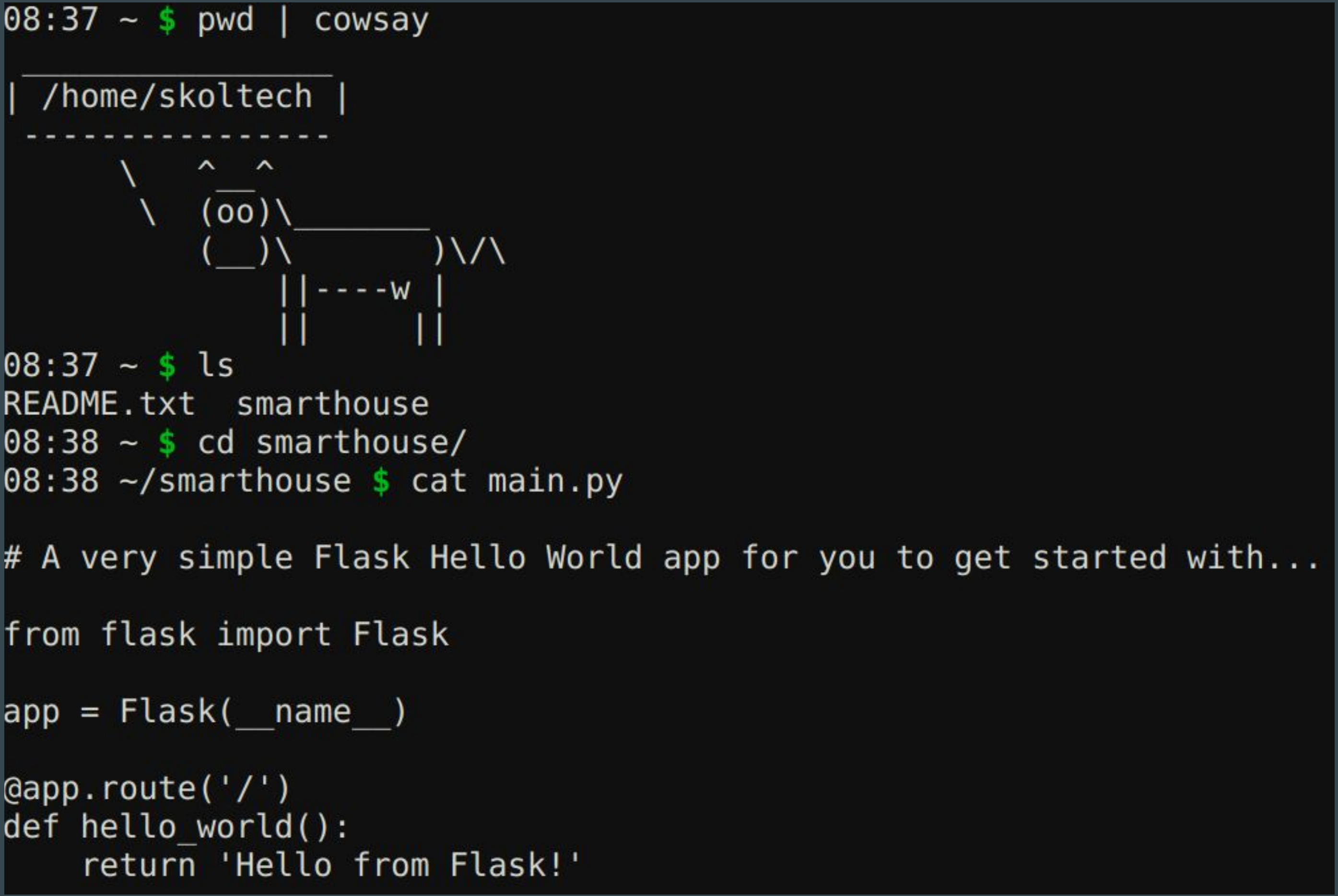


CPU Usage: 1% used – 1.42s of 100s

Start a new console:

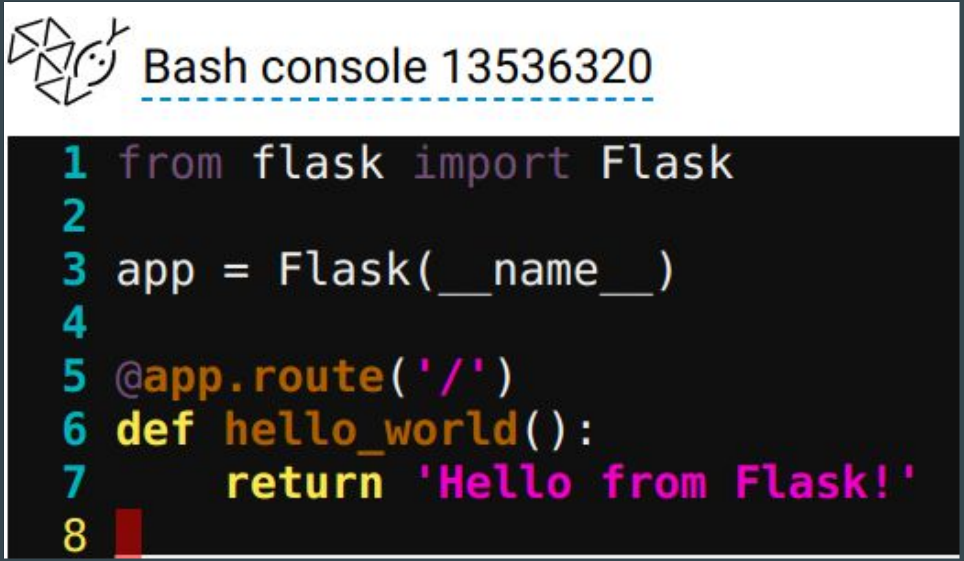
Python: 3.7 / 3.6 / 3.5 / 3.4 / 2.7 IPython: 3.7 / 3.6 / 3.5 / 3.4 / 2.7 PyPy: 2.7
Other: Bash | MySQL
Custom:

Ensure that the Folder Exists

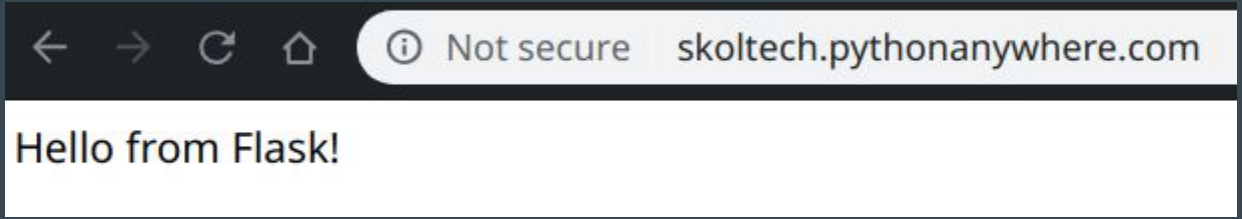


```
08:37 ~ $ pwd | cowsay
 ________________
| /home/skoltech |
 ----------------
      \   ^__^
       \  (oo)\_______
          (__)\       )\/\
              ||----w |
              ||     ||
08:37 ~ $ ls
README.txt  smarthouse
08:38 ~ $ cd smarthouse/
08:38 ~/smarthouse $ cat main.py

# A very simple Flask Hello World app for you to get started with...

from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello from Flask!'
```

Ensure that the Site Works

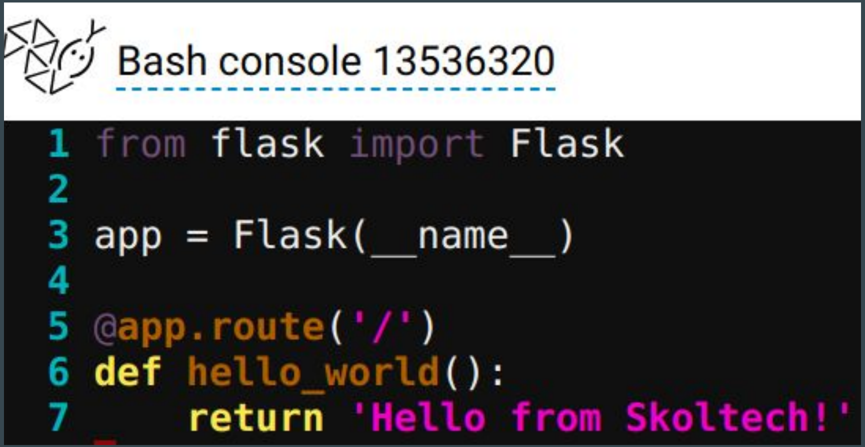

Bash console 13536320

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello from Flask!'

```


Not secure skoltech.pythonanywhere.com

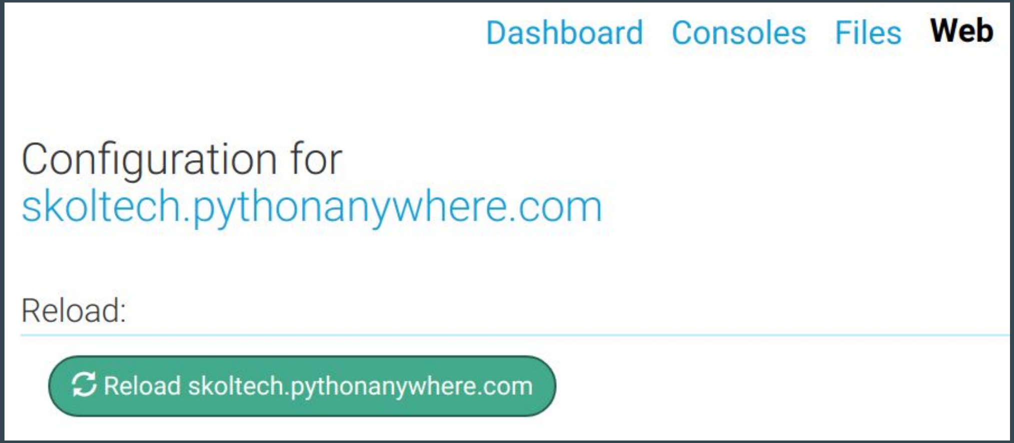
```
Hello from Flask!
```

Change the Message and Reload the Website



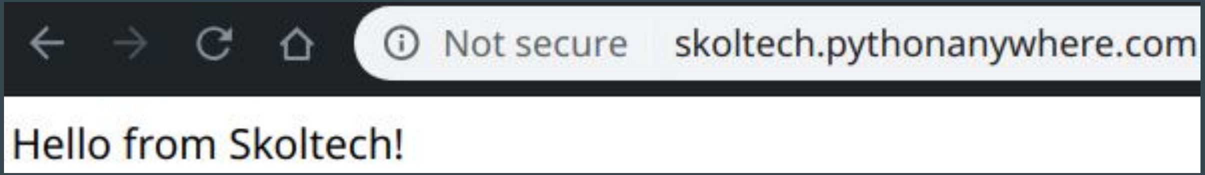
Bash console 13536320

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello from Skoltech!'
```



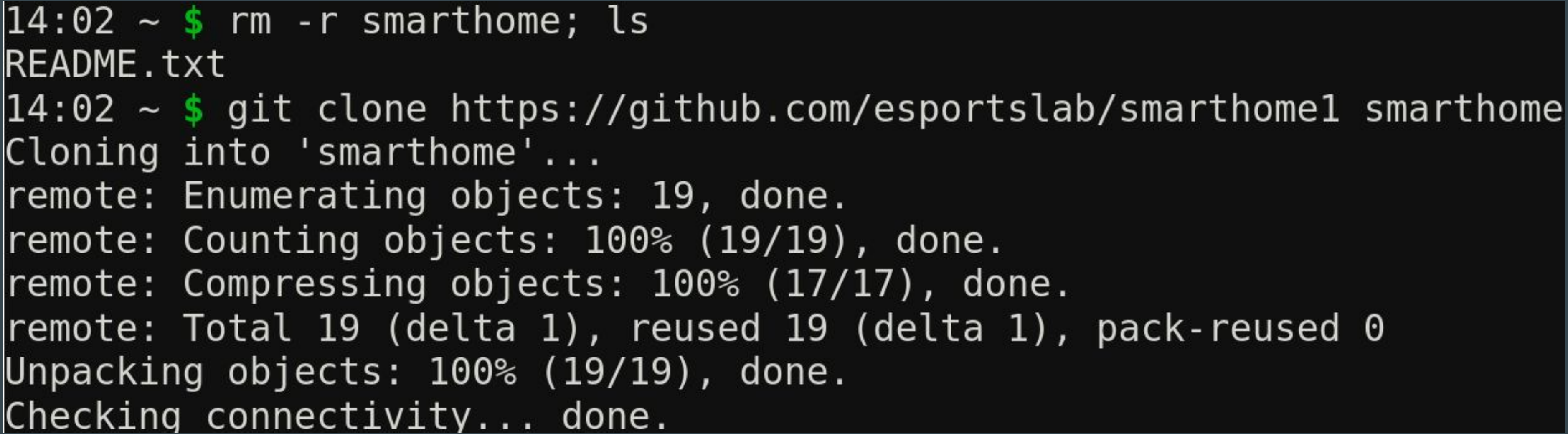
Configuration for skoltech.pythonanywhere.com

Reload:

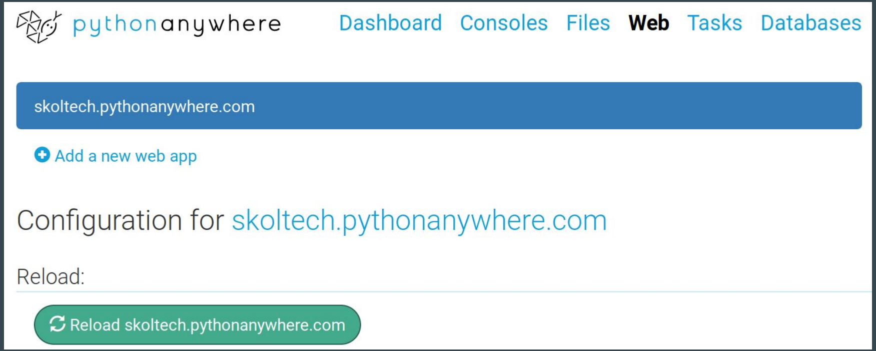


Hello from Skoltech!

Download and Deploy SmartHome 1.0 from GitHub

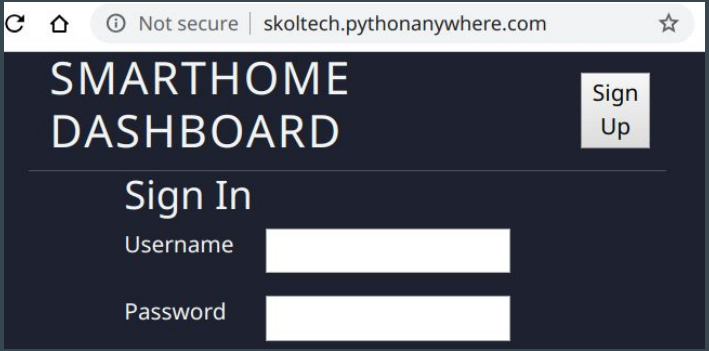


```
14:02 ~ $ rm -r smarthome; ls
README.txt
14:02 ~ $ git clone https://github.com/esportslab/smarthome1 smarthome
Cloning into 'smarthome'...
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 19 (delta 1), reused 19 (delta 1), pack-reused 0
Unpacking objects: 100% (19/19), done.
Checking connectivity... done.
```



Configuration for skoltech.pythonanywhere.com

Reload:



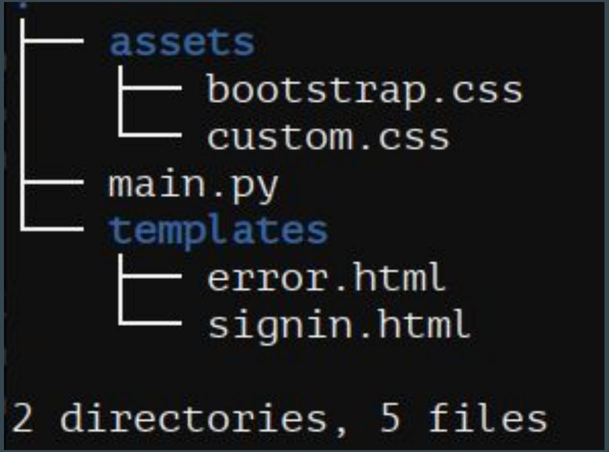
SMARTHOME DASHBOARD

Sign In

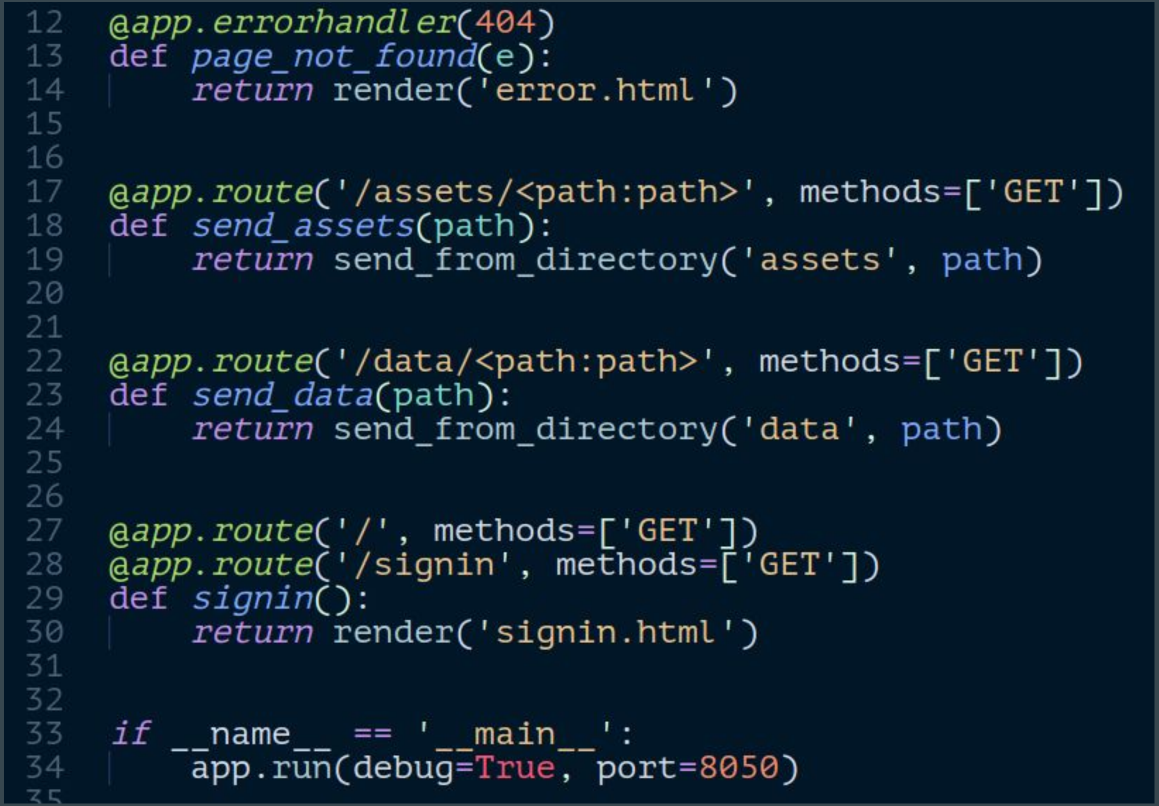
Username

Password

Final Code

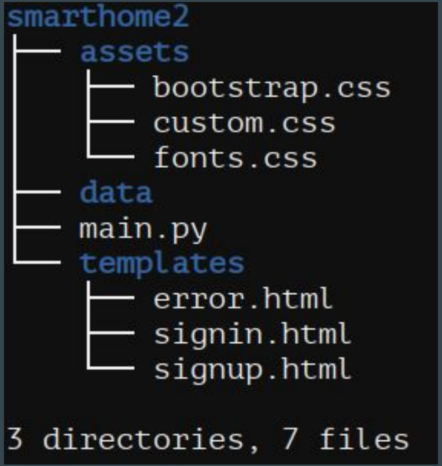


```
├── assets
│   ├── bootstrap.css
│   └── custom.css
├── main.py
└── templates
    ├── error.html
    └── signin.html

2 directories, 5 files
```

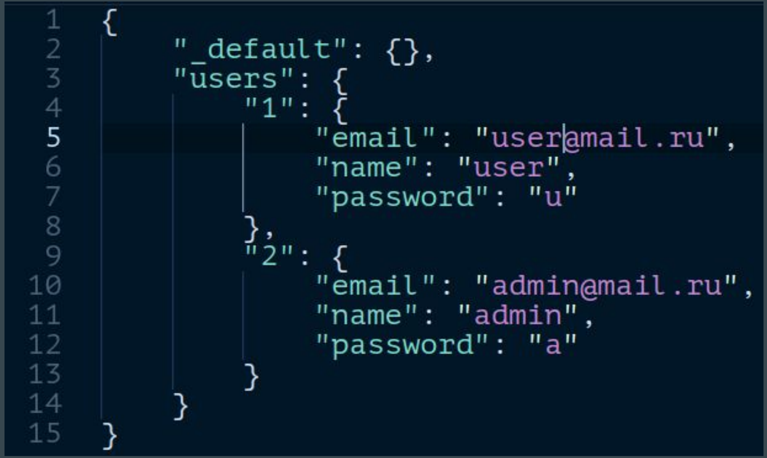


```
@app.errorhandler(404)
def page_not_found(e):
    return render('error.html')


@app.route('/assets/<path:path>', methods=['GET'])
def send_assets(path):
    return send_from_directory('assets', path)


@app.route('/data/<path:path>', methods=['GET'])
def send_data(path):
    return send_from_directory('data', path)


@app.route('/', methods=['GET'])
@app.route('/signin', methods=['GET'])
def signin():
    return render('signin.html')


if __name__ == '__main__':
    app.run(debug=True, port=8050)
```

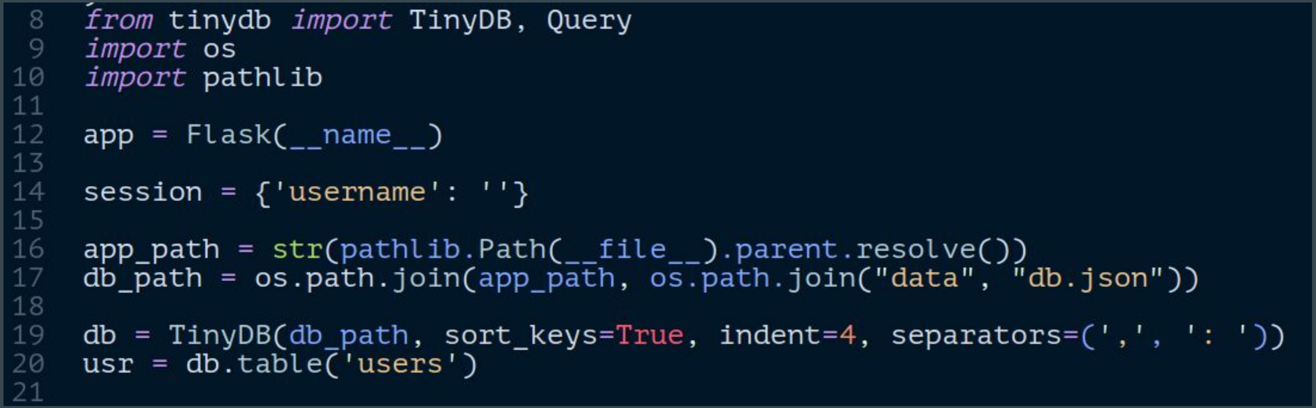
Version 2.0. Dynamic WebApp with Authorization



```
smarthome2
├── assets
│   ├── bootstrap.css
│   ├── custom.css
│   └── fonts.css
├── data
├── main.py
└── templates
    ├── error.html
    ├── signin.html
    └── signup.html

3 directories, 7 files
```

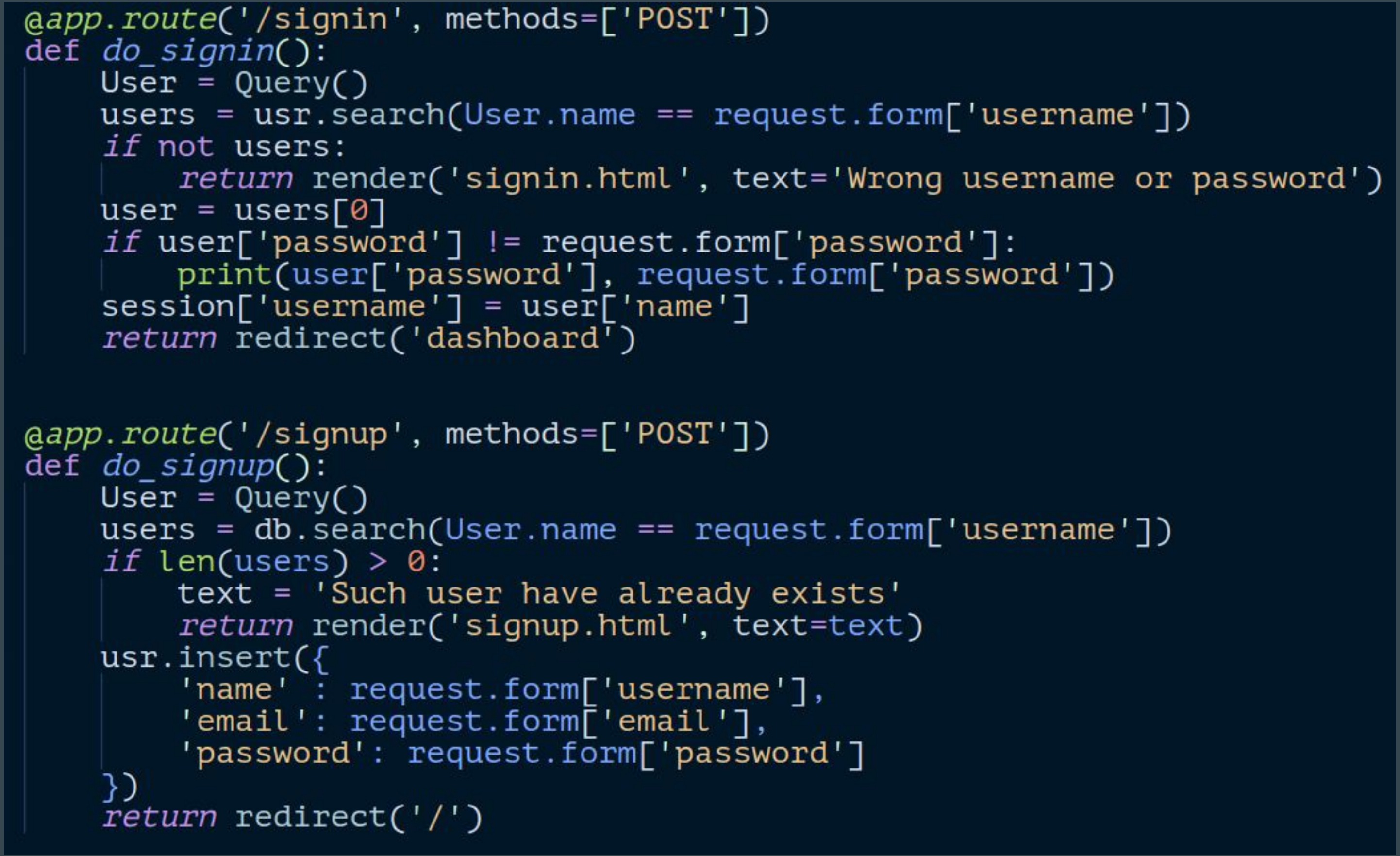


```
{
    "_default": {},
    "users": {
        "1": {
            "email": "user@mail.ru",
            "name": "user",
            "password": "u"
        },
        "2": {
            "email": "admin@mail.ru",
            "name": "admin",
            "password": "a"
        }
    }
}
```



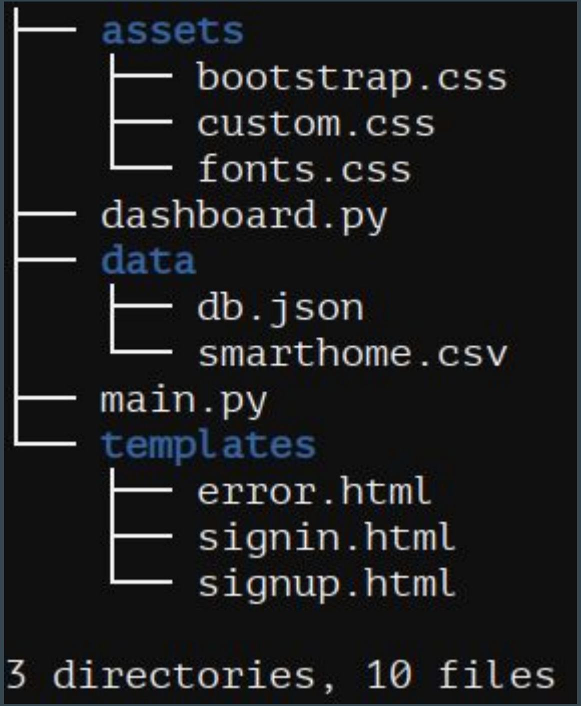
```
from tinydb import TinyDB, Query
import os
import pathlib

app = Flask(__name__)

session = {'username': ''}

app_path = str(pathlib.Path(__file__).parent.resolve())
db_path = os.path.join(app_path, os.path.join("data", "db.json"))

db = TinyDB(db_path, sort_keys=True, indent=4, separators=(',', ': '))
usr = db.table('users')

```

User Authorization

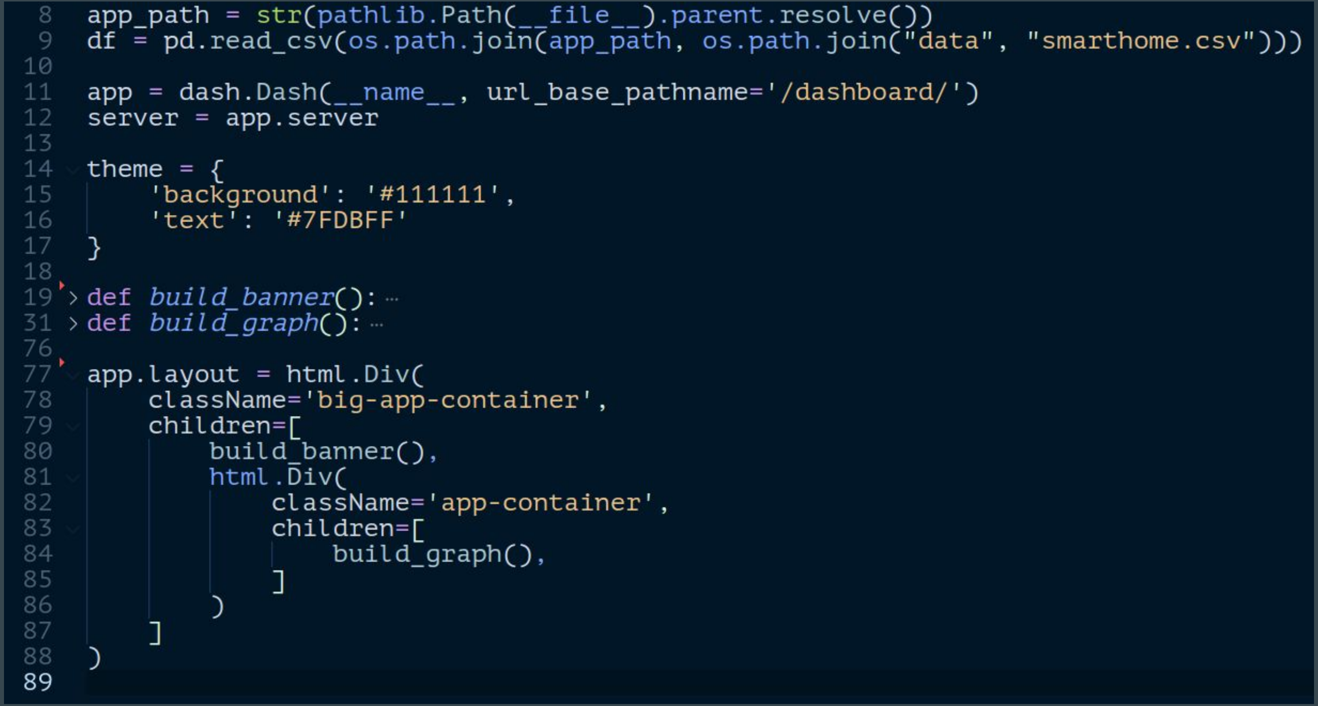


```
@app.route('/signin', methods=['POST'])
def do_signin():
    User = Query()
    users = usr.search(User.name == request.form['username'])
    if not users:
        return render('signin.html', text='Wrong username or password')
    user = users[0]
    if user['password'] != request.form['password']:
        print(user['password'], request.form['password'])
    session['username'] = user['name']
    return redirect('dashboard')


@app.route('/signup', methods=['POST'])
def do_signup():
    User = Query()
    users = db.search(User.name == request.form['username'])
    if len(users) > 0:
        text = 'Such user have already exists'
        return render('signup.html', text=text)
    usr.insert({
        'name' : request.form['username'],
        'email': request.form['email'],
        'password': request.form['password']
    })
    return redirect('/')
```

Version 3.0. Dynamic WebApp with Interactive Chart



```
├── assets
│   ├── bootstrap.css
│   ├── custom.css
│   └── fonts.css
├── dashboard.py
├── data
│   ├── db.json
│   └── smarthome.csv
├── main.py
└── templates
    ├── error.html
    ├── signin.html
    └── signup.html

3 directories, 10 files
```



```python
app_path = str(pathlib.Path(__file__).parent.resolve())
df = pd.read_csv(os.path.join(app_path, os.path.join("data", "smarthome.csv")))

app = dash.Dash(__name__, url_base_pathname='/dashboard/')
server = app.server

theme = {
    'background': '#111111',
    'text': '#7FDBFF'
}

def build_banner(): ...
def build_graph(): ...

app.layout = html.Div(
    className='big-app-container',
    children=[
        build_banner(),
        html.Div(
            className='app-container',
            children=[
                build_graph(),
            ]
        )
    ]
)
```

Future Challenges and Research Directions

1. Learn React.js (by Facebook) and web-frontend design patterns (like Redux)

   https://reactjs.org/, https://redux.js.org/

2. Create full-featured dashboards with lots of data

   https://dash-gallery.plotly.host/dash-manufacture-spc-dashboard/

3. Make you frontend responsive (adaptable to all sizes of devices)

   http://dash-bootstrap-components.opensource.faculty.ai/