

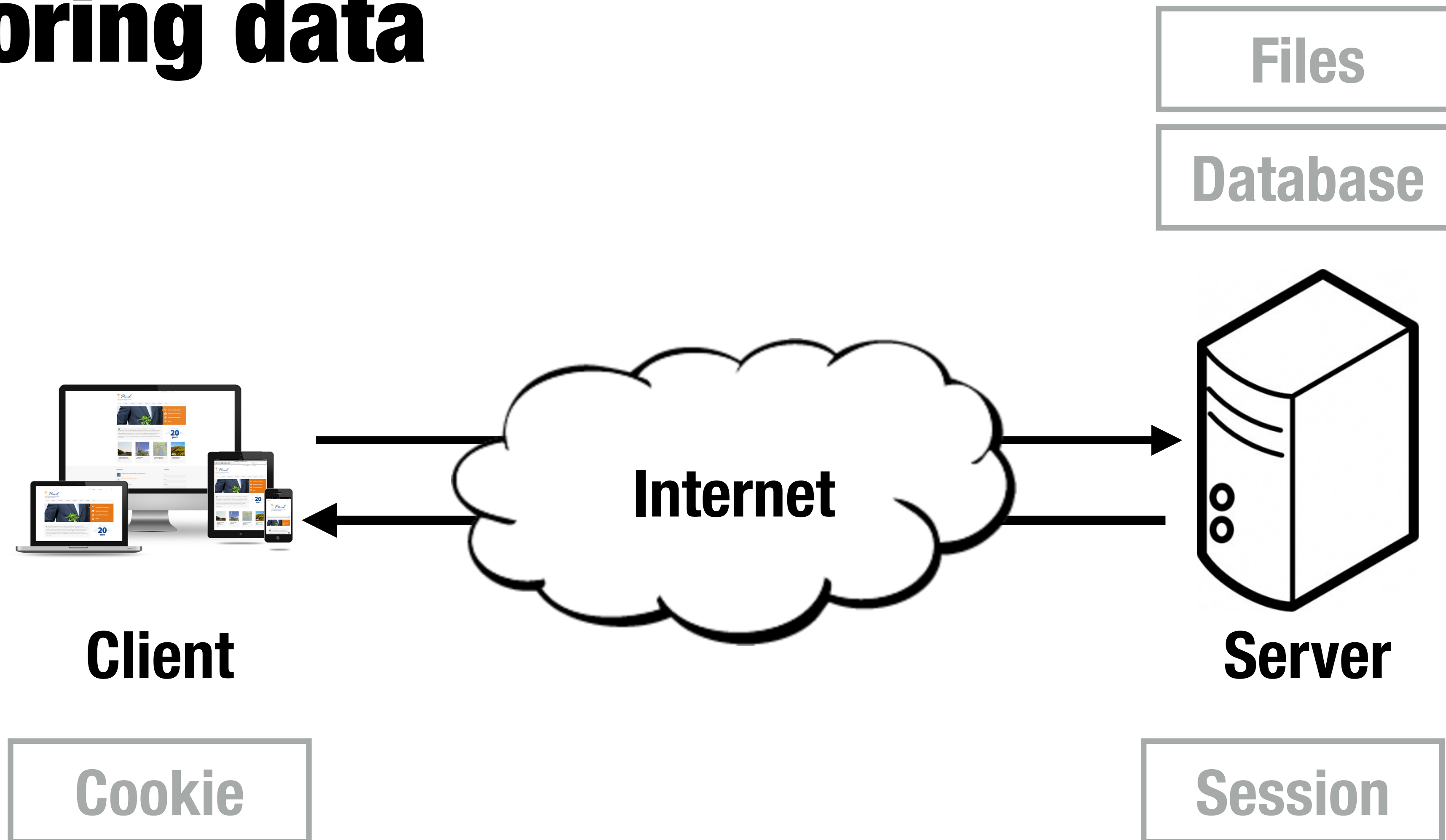
Web Programming

Login

Login

- This lecture: Simple login using sessions.
 - Has some security flaws
- Deployment alternatives:
 - Flask-Login
 - OAuth provider, e.g. firebase.google.com

Storing data



Storing Passwords

- Do not store passwords in plaintext
- When application/server is compromised, all passwords will be stolen.
- Affects other sites.

Password Hashes

- One way function

"In goes a string of
arbitrary length"



Out comes
fixed size string

"628649fb210... "

Password Hashes

```
from werkzeug.security import generate_password_hash, check_password_hash
```

- Create a salted password hash to store

```
hash = generate_password_hash("Joe123")
```

Includes a random **salt**, so no two passwords have the same hash

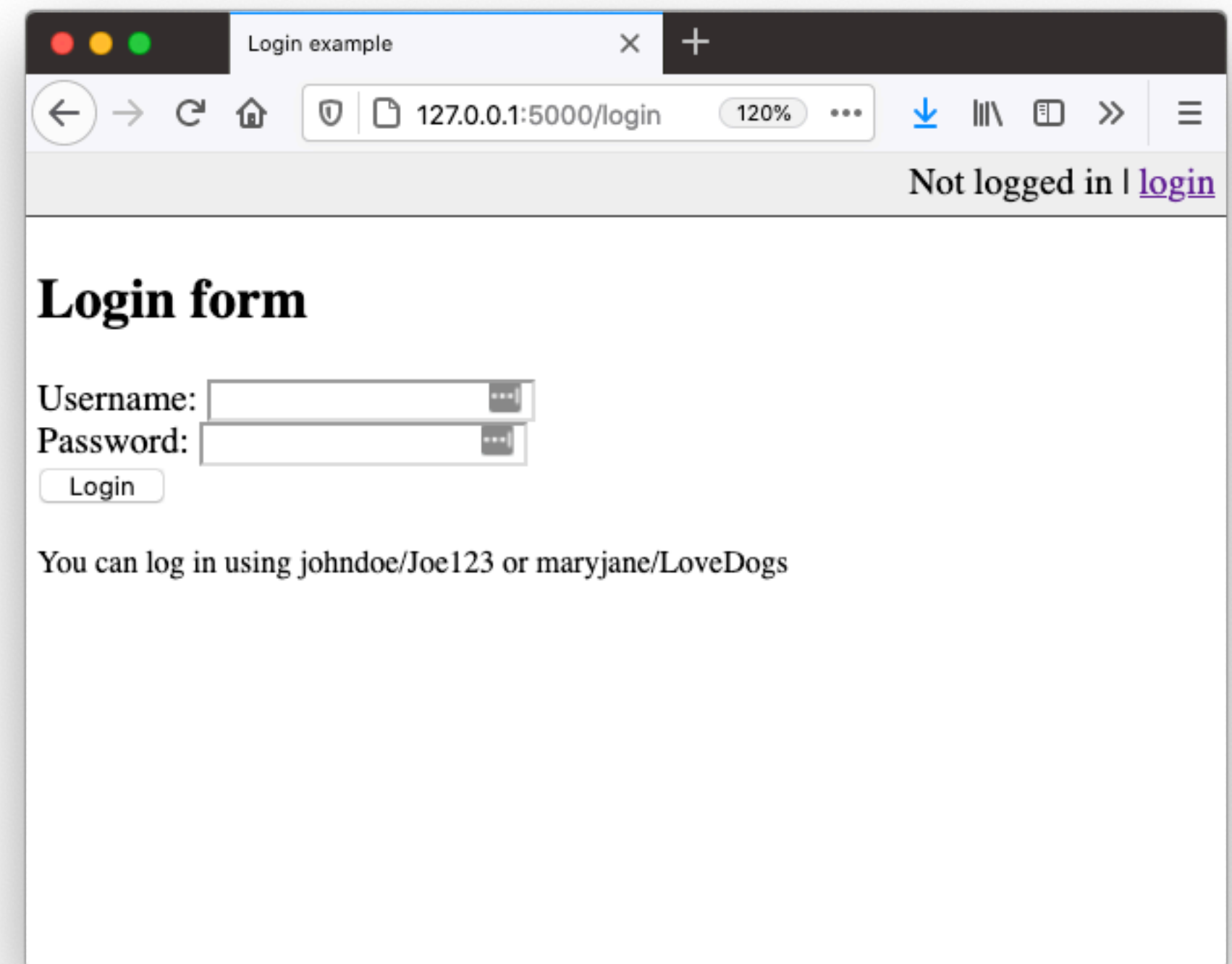
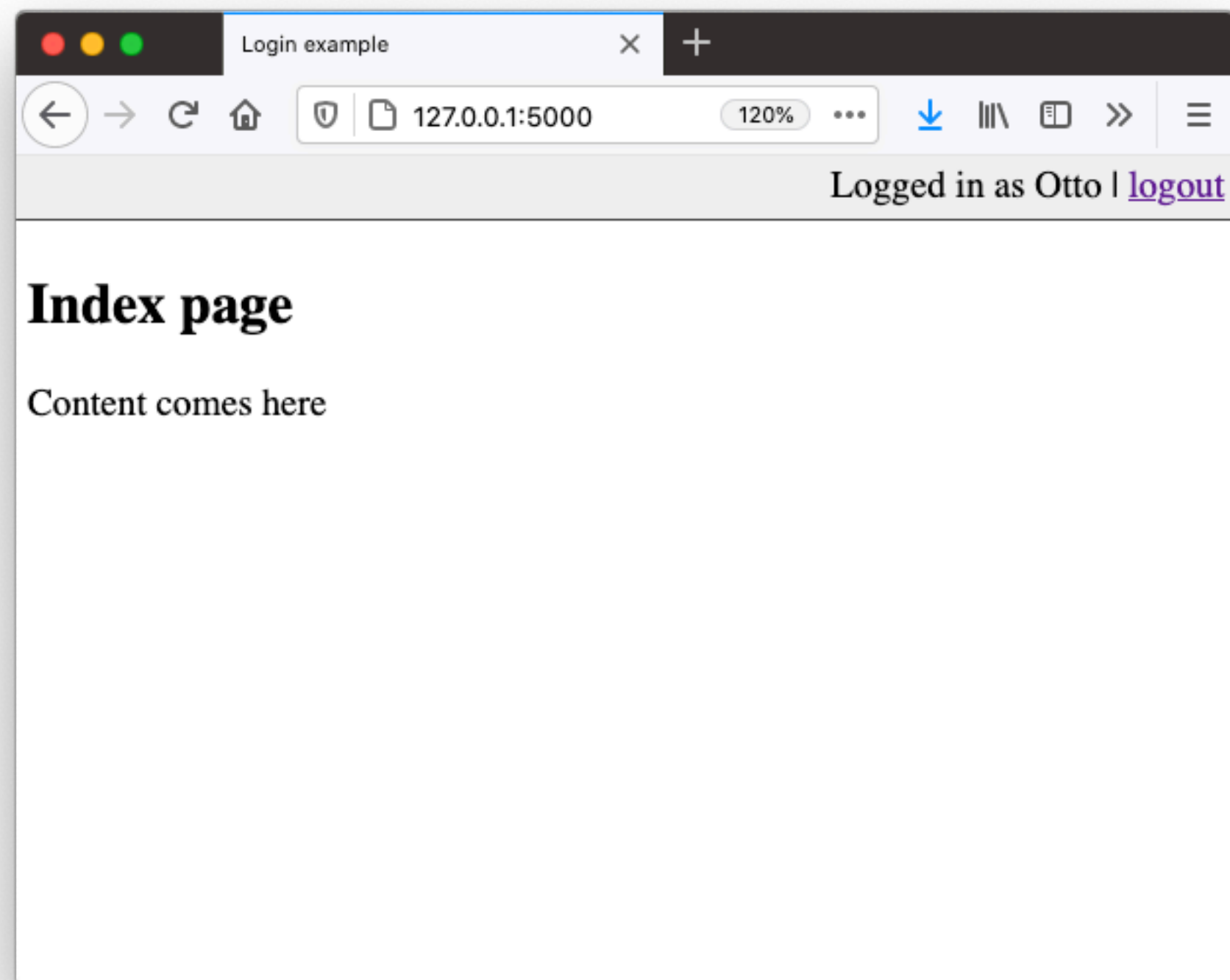
```
"pbkdf2:sha256:150000$oMxlb00a$125a8c19b39e0fc7e903e7775a45e40667663ed01382f9b5adcb5e0eb3d80937"
```

- Check password

```
ok = check_password_hash(hash, "Joe123")
```

Example

🔗 `examples/python/flask/9_login/app.py`



Example

📄 examples/python/flask/9_login/app.py

- on login, check password hash and add username to session

```
@app.route("/login", methods=["GET", "POST"])
def login():
    username = request.form["username"]
    password = request.form["password"]

    if valid_login(username, password):
        session["username"] = username
        return redirect(url_for("index"))
```


Example

📄 `examples/python/flask/9_login/app.py`

- on logout, remove username from session

```
@app.route("/logout")
def logout():
    session.pop("username")
    return redirect(url_for("index"))
```

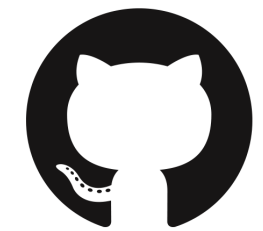
Exercise #1, #2, #3



[github.com/dat310-2023/info/tree/master/
exercises/python/flask5](https://github.com/dat310-2023/info/tree/master/exercises/python/flask5)

Walkthrough in lecture!

Exercise #1 Solution



[github.com/dat310-2023/info/tree/master/
solutions/python/flask5](https://github.com/dat310-2023/info/tree/master/solutions/python/flask5)

- Use id instead of username

```
"userid INTEGER PRIMARY KEY"
```

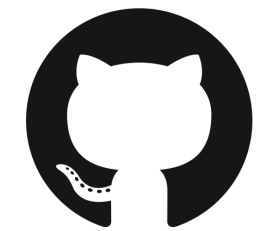
- Id is generated on insert

```
sql = ("INSERT INTO users (username, passwordhash) VALUES (?,?)")  
cur.execute(sql, (username, hash))  
conn.commit()  
return cur.lastrowid
```

- Separate functions for user details and pw_hash

```
def get_user_by_name(conn, username):  
  
def get_hash_for_login(conn, username):
```

Exercise #2 Solution

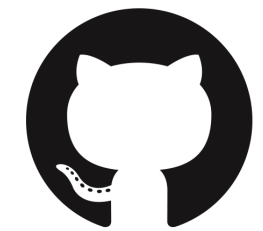


[github.com/dat310-2023/info/tree/master/
solutions/python/flask5](https://github.com/dat310-2023/info/tree/master/solutions/python/flask5)

- Relies on UNIQUE for checking duplicates

```
conn = get_db()
id = add_user(conn, username, hash)
if id == -1:
    flash("Username already taken")
```

Exercise #3 Solution



[github.com/dat310-2023/info/tree/master/
solutions/python/flask5](https://github.com/dat310-2023/info/tree/master/solutions/python/flask5)

- Store role on session:

```
if not session.get("role", None) == "admin":  
    abort(403)
```

Cross-site request forgery (CSRF)

- A web security flaw
- Attacker induces user to make unintended actions
 - e.g. change email

```
app.config.update(  
    SESSION_COOKIE_HTTPONLY=True,  
    # do not allow to access session in JS  
    REMEMBER_COOKIE_HTTPONLY=True,  
    SESSION_COOKIE_SAMESITE="Strict",  
)
```

Set SAMESITE="Strict" on session cookies to prevent most CSRF.

Limitation

- To further improve security session should include:
 - Unique token for every time you login
- Further, requests should contain CSRF token.
 - <https://owasp.org/www-community/attacks/csrf>
 - <https://portswigger.net/web-security/csrf>