

Security software design and web security

Marc Vegetti Luc Boussant-Roux Viktor Richard Kozma Imad Yamani

Table of contents

| | |
|------------------------------|---|
| Introduction | 2 |
| How to run it | 2 |
| Database | 3 |
| Registration and login | 4 |
| Upload | 5 |
| Deletion | 5 |
| Download | 5 |
| Administration control panel | 6 |
| Other features | 6 |
| References: | 7 |

I. Introduction

The aim of this project is to implement a secure web application. This web application must allow to a user to create an account, log in and create/delete/download folders as well as files. Furthermore users must be able to share folders or files.

To implement this web application, we used Flask. It is a micro-framework for Python 3.6.7 based on Werkzeug, which is a powerful collection of various utilities and Jinja2, which is a modern and designer-friendly templating language for Python.

We chose Flask as it comes with a lot of useful functions and allows an easy implementation of front-end and back-end.

Furthermore Python is a powerful language, with a lot of people using it. As such, we can easily find help on the web if needed.

Many reasons pushed us to use Python for our web application. The syntax of the Python allows readability of the code. It is also flexible and has a lot of resources with libraries and modules. Moreover, it saves time with no lengthy coding.

We used git version control to keep track of our code, which can be reached below:

<https://github.com/iMADDDDDD/ssd-file-system-storage>

II. How to run it

In order to run the web application is firstly needed to download the packages below.

Here is the list of commands:

```
To install all packages
cd project/
python3 -m venv venv
source venv/bin/activate
pip install flask
pip install flask_sqlalchemy
```

```
pip install flask_migrate
pip install flask_login
pip install flask_wtf
pip install flask_bootstrap
pip install flask_fontawesome
pip install flask_mail
pip install pyjwt
pip install onetimepass
pip install pyqrcode
pip install pysftp
pip install pytransmit
pip install flask_uploads

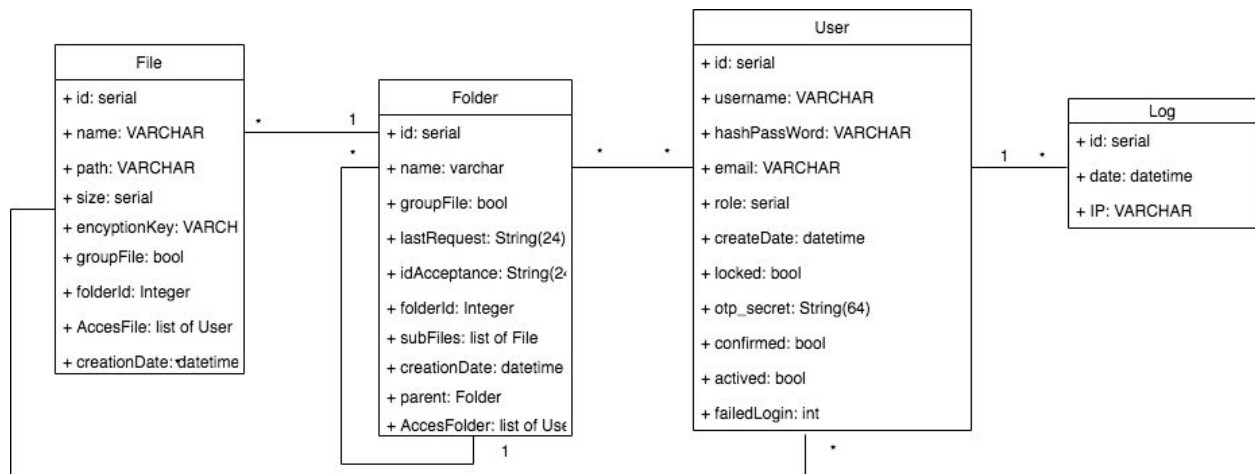
To create the database
flask db init
flask db migrate -m "Nom de la migration"
flask db upgrade

To update database
flask db migrate -m "Nom de la migrationion"
flask db upgrade
```

III. Database

To save all our datas we used SQLAlchemy as an object relational mapper. Basically a table found in database such as Oracle will be implemented as a class. The rows will be class attributes.

Our database architecture looks like this:



A log table to keep track of the different connections. Unfortunately we didn't implement it.

A user table with the basic informations, 'locked' allows to know if an user got locked after 3 connection failures. 'confirmed' allows to know if he confirmed his email.

IV. Registration and login

First of all when a user creates an account he has to confirm the registration. An email is automatically sent to his mailbox. This confirmation email uses a JSON Web Token, which is usable only for a limited time of life.

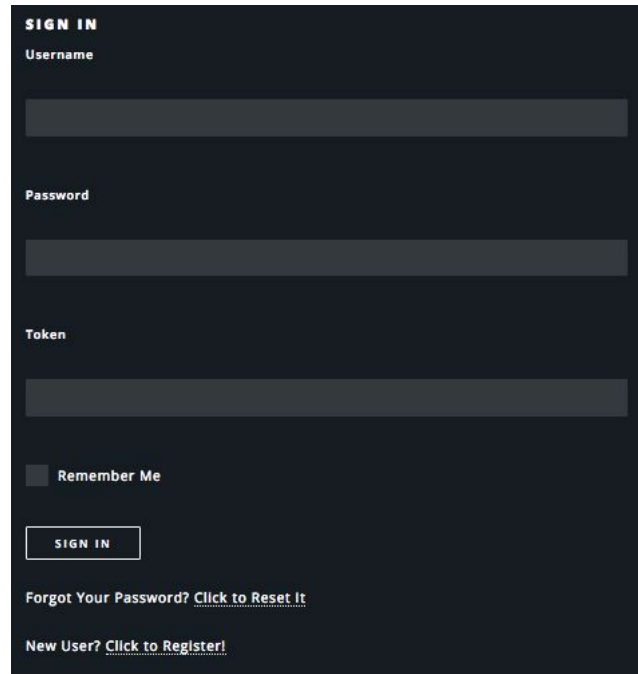
And he has to download Google Authenticator, take a picture of a QR code. This will be for the double authentication.

Once the administrator validated the registration from his registration page the user will be able to login. He has a maximum of 3 tries before locking his account. Once locked he has to ask an administrator to unlock it.

An email can only be used once to create an account. An user agent password must have a length of at least 8. It will be harder and harder for an attacker to retrieve the password by bruteforce as its size increases.

To connect an user has to put his password, the one time password generated with the Google application and fill a captcha to prevent robots.

A CAPTCHA is a type of challenge–response test used in computing to determine whether or not the user is human, we imported from Flask and had to setup an account on Google.



The image shows a 'SIGN IN' form with a dark background. It contains three input fields labeled 'Username', 'Password', and 'Token'. Below the 'Token' field is a checkbox labeled 'Remember Me'. A 'SIGN IN' button is positioned below the checkbox. At the bottom of the form, there are two links: 'Forgot Your Password? [Click to Reset It](#)' and 'New User? [Click to Register!](#)'.

An user password will be hashed (SHA-256) with salt and then saved into the database.

Once logged in, the user session will be up for 5 min. In case the user doesn't disconnect and leaves his session opened, it prevents someone from using his identity.

V. Upload

The user has a few features available when connected. The first one being uploading files.

All files will be created in project/Files.

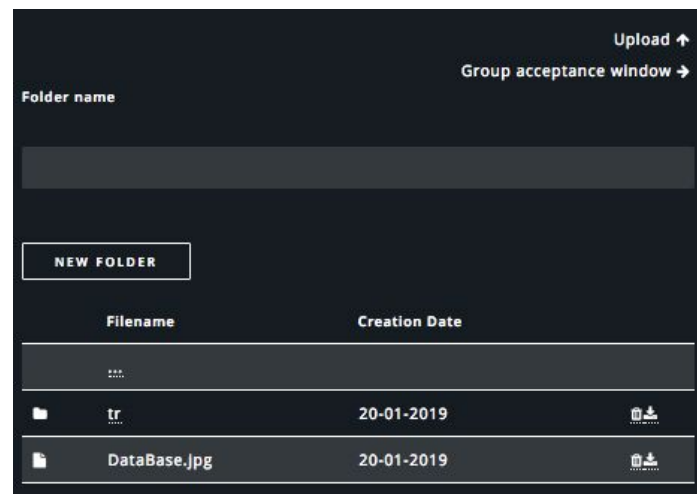
A file object will be created in the database, we will reference his parent folder, the people who have access to it and much more.

A folder can be directly created from the website, it will create it inside the database and inside the server file system.

The user can also upload a directory, but the subdirectories will be lost, all the files will be uploaded in the current folder.

Before saving a file on the server, it gets encrypted and the key, which was used to encrypt the file is saved into the database. We wanted to save it hashed but we failed to implement it. If the file get stolen, the attacker will not be able to read it.

To encrypt we used Fernet Library, which is a symmetric encryption and generates a a base64-encoded 32-byte key.



VI. Deletion

If one wants to delete a file he can, just by clicking onto the trash icon. The file will be deleted from the server and from the database.

If one wants to delete a complete folder, it will recursively delete all files and folders in it.

Obviously we check beforehand if the user has the right to delete it.

VII. Download

Once a file has been uploaded, it can be downloaded from the server. Once again the server will check if the user has the right to do so. It will check thank to the `send_from_directory` function if the file exist in the current directory. As such an user can't download another file which has the same name but is in another folder.

The server will decrypt the file before sending it.

An user can also download a complete directory. The function will recursively add the different files to a compressed .zip file, but this method will not save the subdirectories.

VIII. Administration control panel

The web application offers a control panel for all administrators. Administrators are specified manually by the owners of the project. The accounts are entered manually into the database for added security.

The main objective of an administrator is to activate newly created accounts by users and to lock out users who failed to log-in more than 3 times. For this case, the panel directly lists accounts which have been locked and which are awaiting activation by an administrator.

Moreover, the administrator is able to view a list of all accounts currently stored in the database. The web application allows the administrator to view all the details of the account when clicked on the name. Such details are the ID, e-mail, creation date of the account and if the account is locked or awaiting activation. The administrator can take actions on the account such as deactivating, activating or unlocking the account.

Subsequently, the administrator is able to see all the files and folders of that user stored in the server. He has all the privileges to delete or alter the files and folders.

IX. Other features

To secure the transmission we created our own certificate. Once the user accept it as correct when connecting to the website the transmission will be secured with HTTPS protocol. It uses SSL to secure the datas.

References:

- [1] : <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>
- [2] : <https://cryptography.io/en/latest/fernet>
- [3] : <http://flask.pocoo.org/>
- [4] : https://support.google.com/accounts/topic/2954345?hl=en&ref_topic=7667090