TECHgium® 7th Edition

Mentoring Session

Differentiation and Tagging of Synthetic Vs Real Data

L&T Technology Services

Anup Kumar

Aditi Shaw

Nayan Verma

K Dinesh Kartikeyan

VIT-AP

AMARAVATI

Prof. Nagendra Panini Challa

Challenge Statement

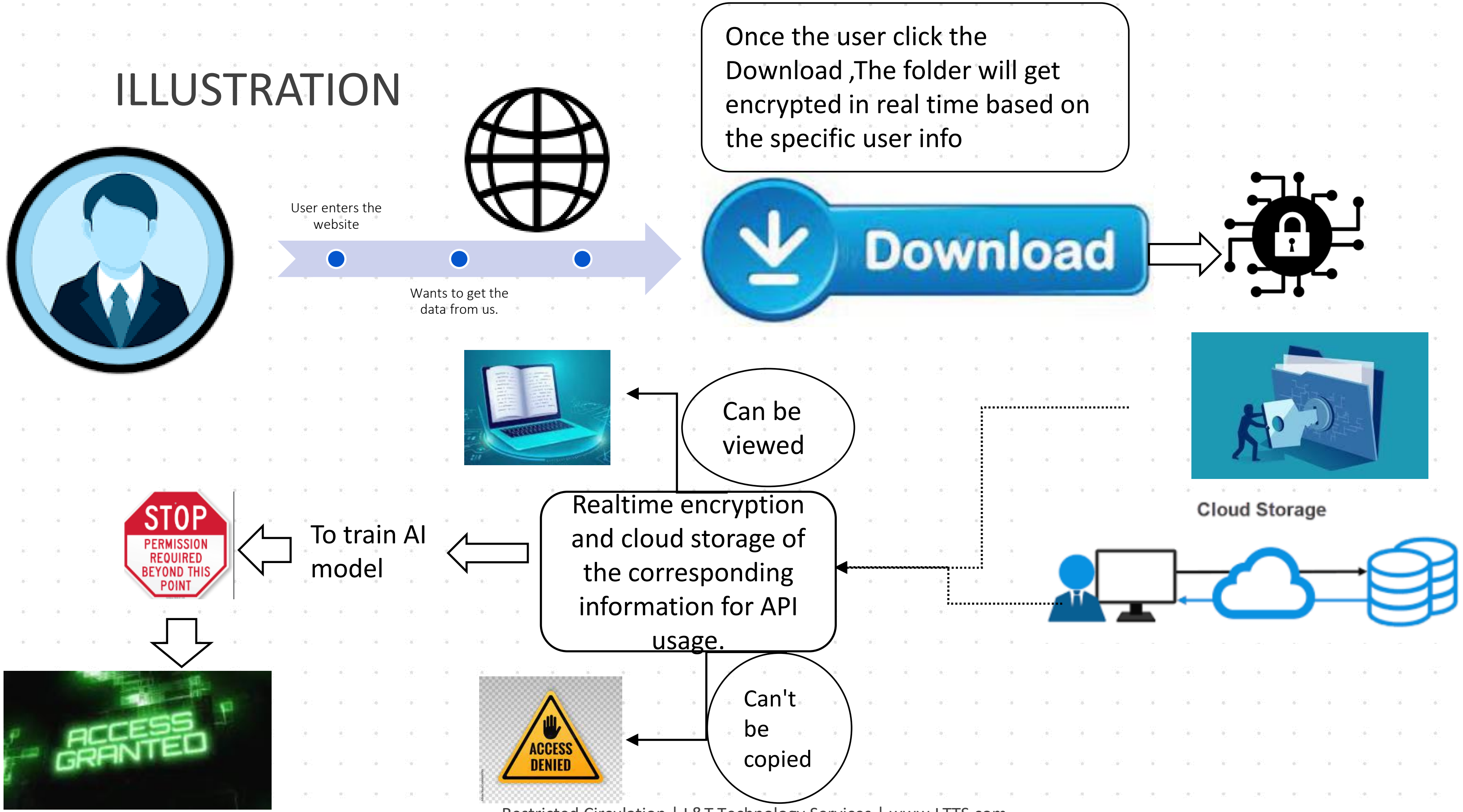
- **Differentiation and Tagging of Synthetic Vs Real Data**
- The ask is to create a solution that can somehow tag every piece of original work, whether images, video, audio or text) with a unique digital signature or watermark that is easy to verify for the public but impossible for bad actors to crack as it does not follow human identifiable patterns. The watermark should be permanent and non-erasable and cannot be deleted even if the digital asset is copied or duplicated. The solution should also give data owners complete authority on who can and who cannot train AI models with their data.
- **Deliverables**

1. A digital watermark algorithm that can embed digital watermarks into data sets.
2. The watermark should work on images, text, audio, and video.
3. A trigger dataset that contains the digital watermarks
4. A method for verifying the integrity of data streams by extracting the digital watermarks
5. A method for controlling access to the data sets based on the presence of digital watermarks

Concept / Solution

- Unique authentication system to generate unique password for a user per device.
- The unique password will be generated by a combination of the device's mac address and the user_id.
- Instead of sharing the end password to the user, the user will have to open the encrypted password with an api, which will do the password generation for the user and decrypt the file.
- By this when the user tries to load the data in python file, he will have to first decrypt the file with the api to create a file reader and data-loader for training models.
- Complete process: User signs up => puts mac address to record it in db => server encrypts file with user_id and mac_addr => user downloads encrypted files => installs package using pip which has api to decrypt => api creates a file reader which can be used to load dataset.

ILLUSTRATION



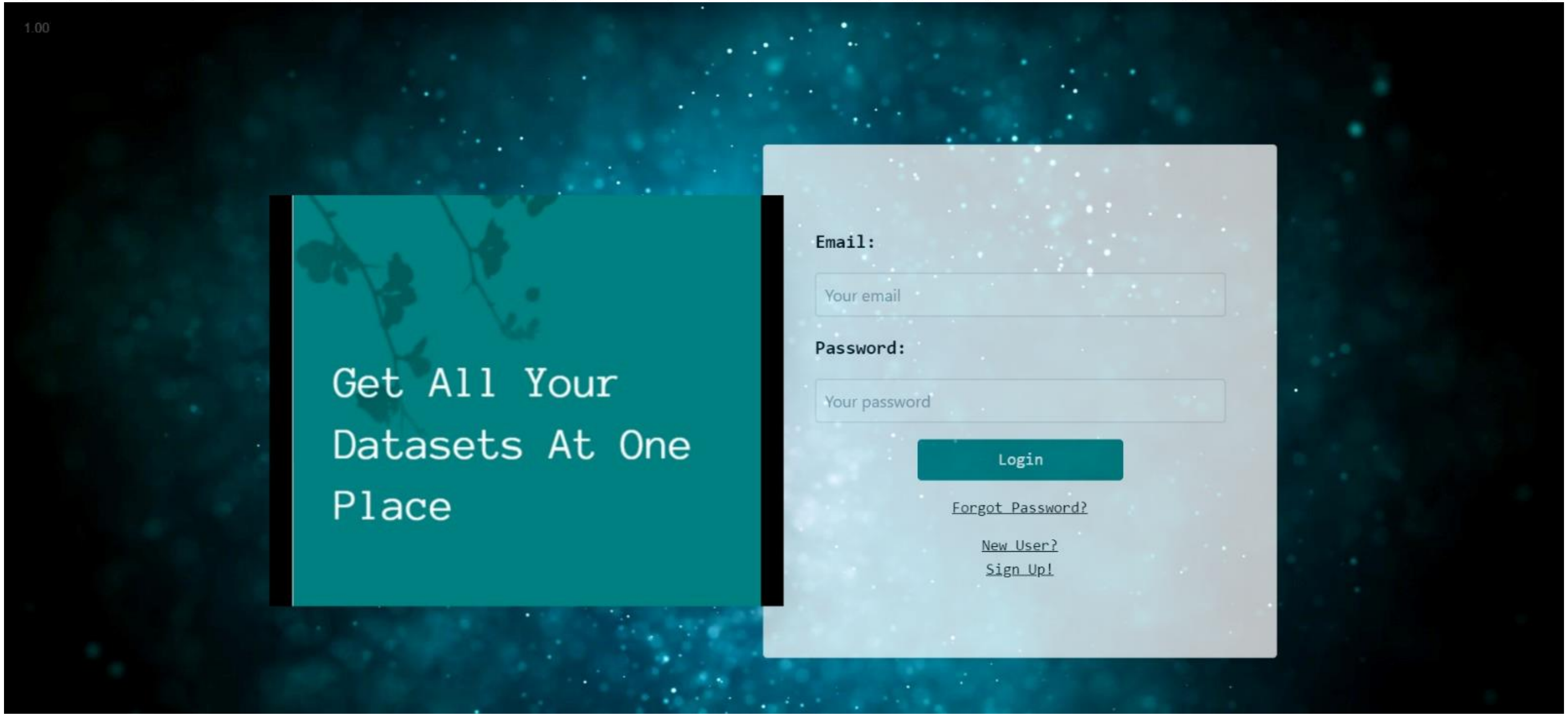
Once the user click the Download ,The folder will get encrypted in real time based on the specific user info

Realtime encryption and cloud storage of the corresponding information for API usage.

Current Status of POC

- The front end of project is 70% ready where we will be dealing with the user to take and download the dataset .The user registration and user access granting will be done through this front end.
- The main encryption algorithm for the encryption of the folder is also partially ready and the integration part is left.
- We are also working on the library that will be used to access the dataset that is downloaded from our website.

Current Status of POC



1.00

Get All Your Datasets At One Place

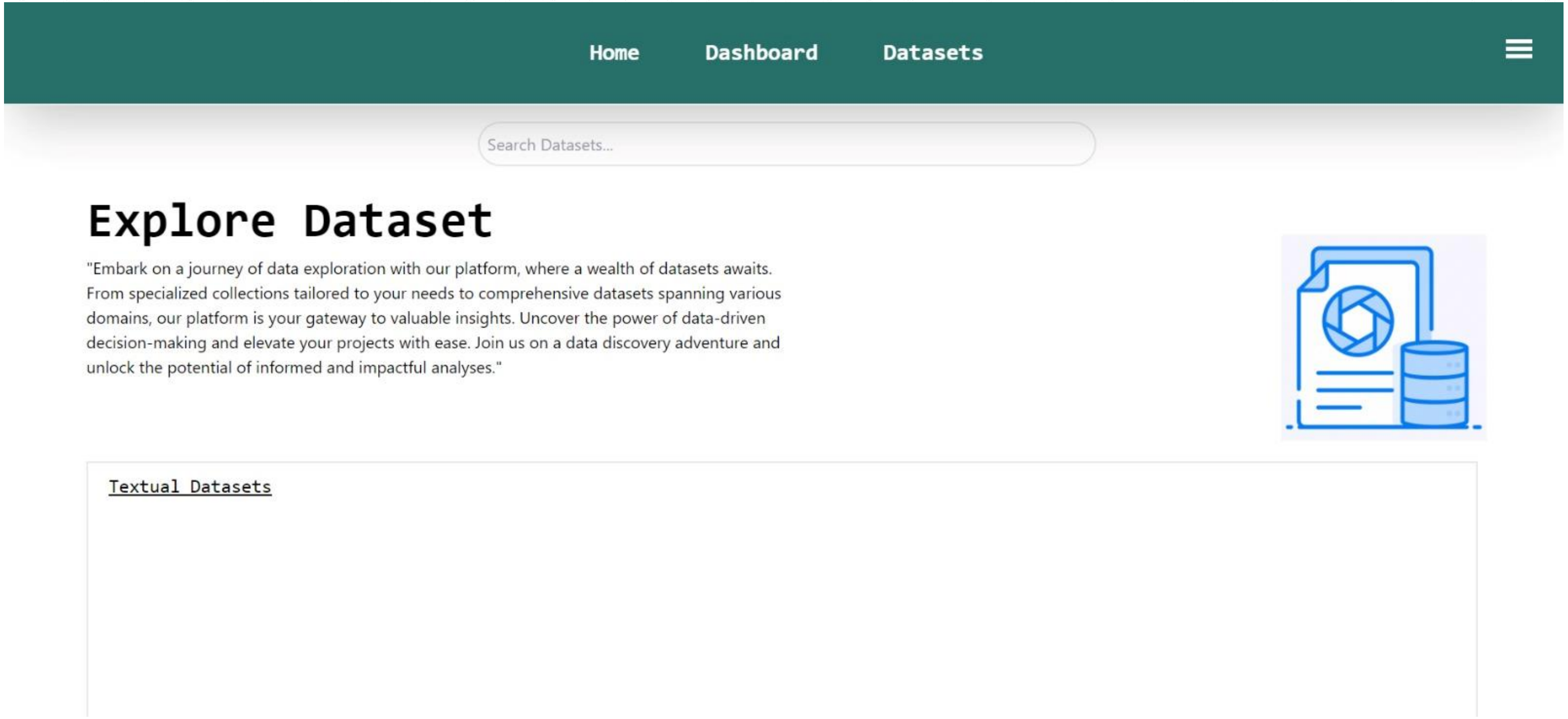
Email:

Your email

Password:

Your password

Current Status of POC

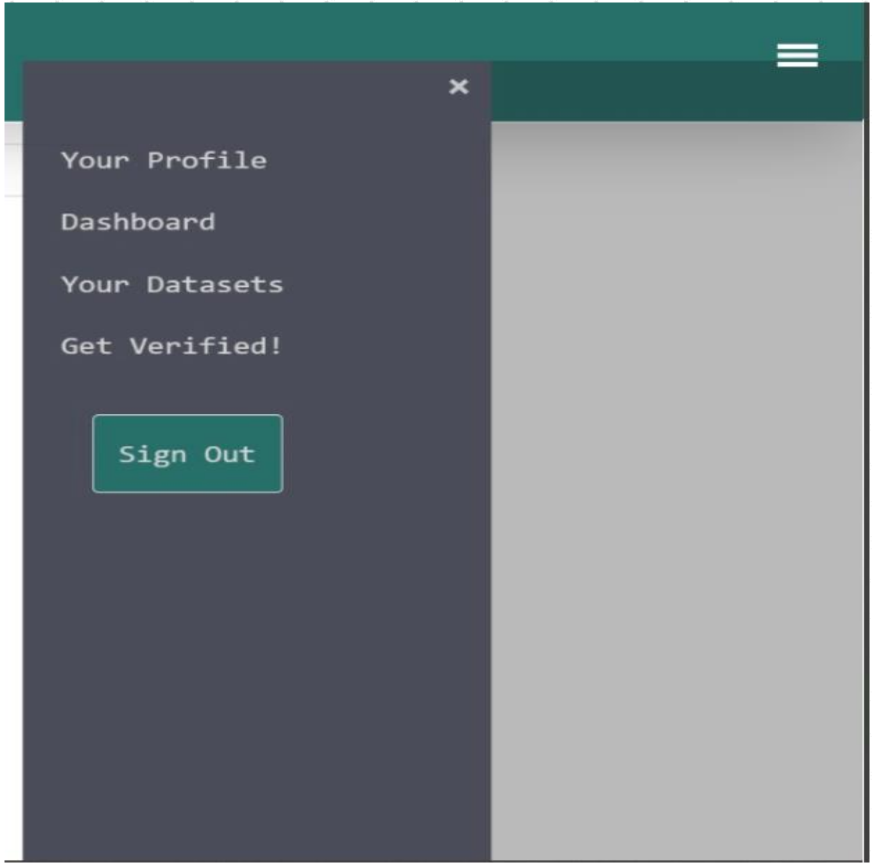


Explore Dataset

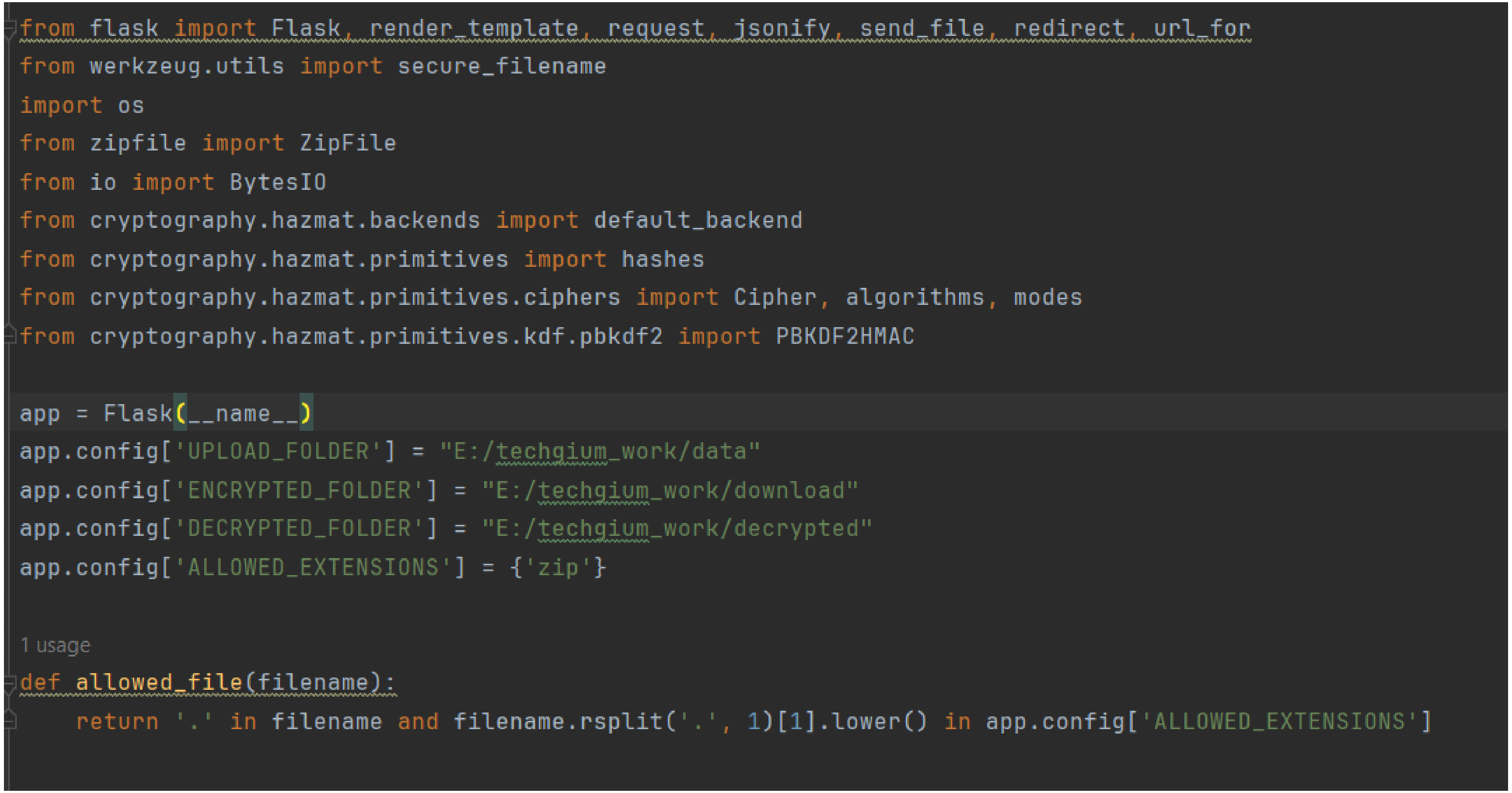
"Embark on a journey of data exploration with our platform, where a wealth of datasets awaits. From specialized collections tailored to your needs to comprehensive datasets spanning various domains, our platform is your gateway to valuable insights. Uncover the power of data-driven decision-making and elevate your projects with ease. Join us on a data discovery adventure and unlock the potential of informed and impactful analyses."

Textual Datasets

Current Status of POC

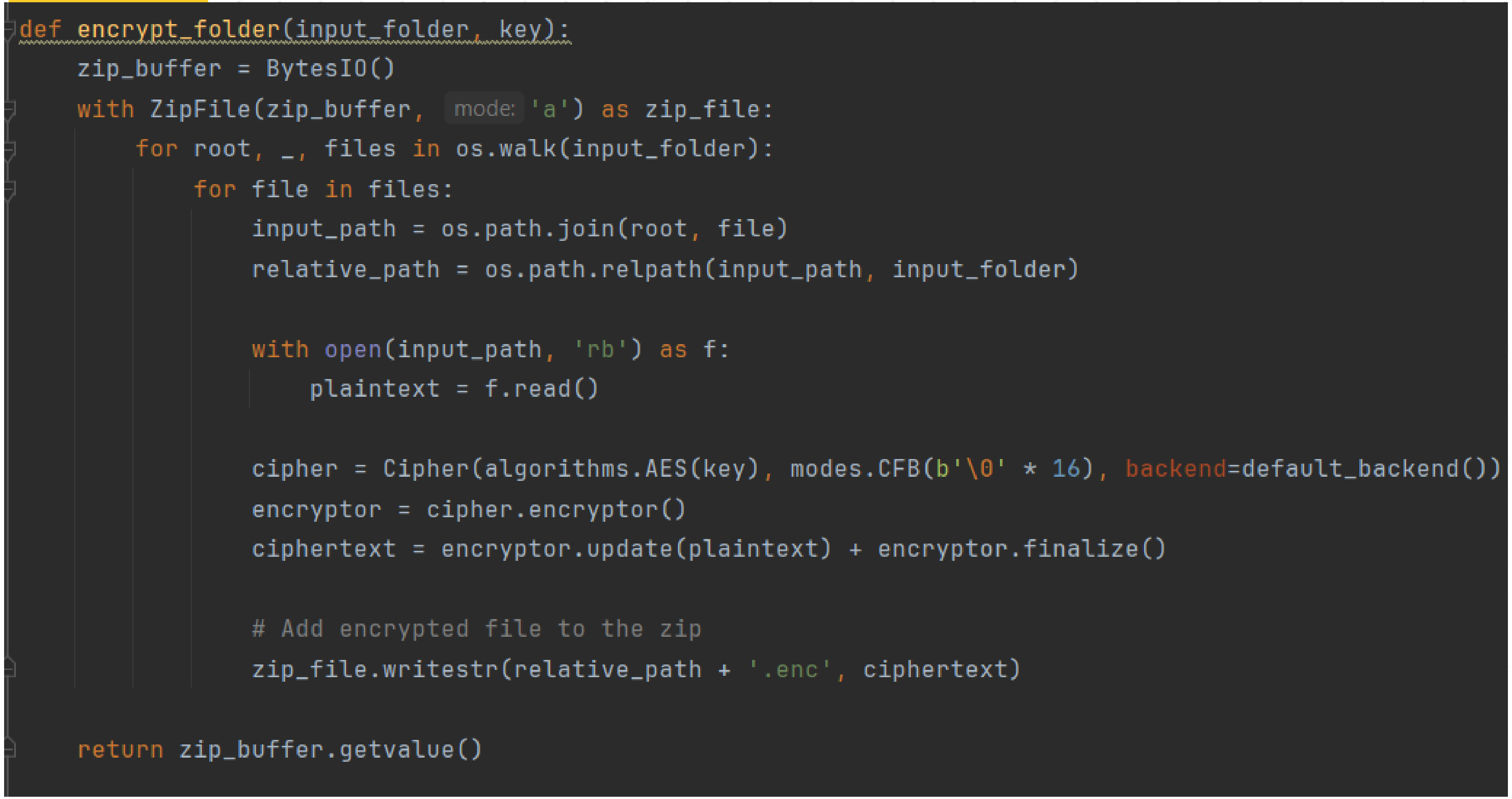


Current Status of POC



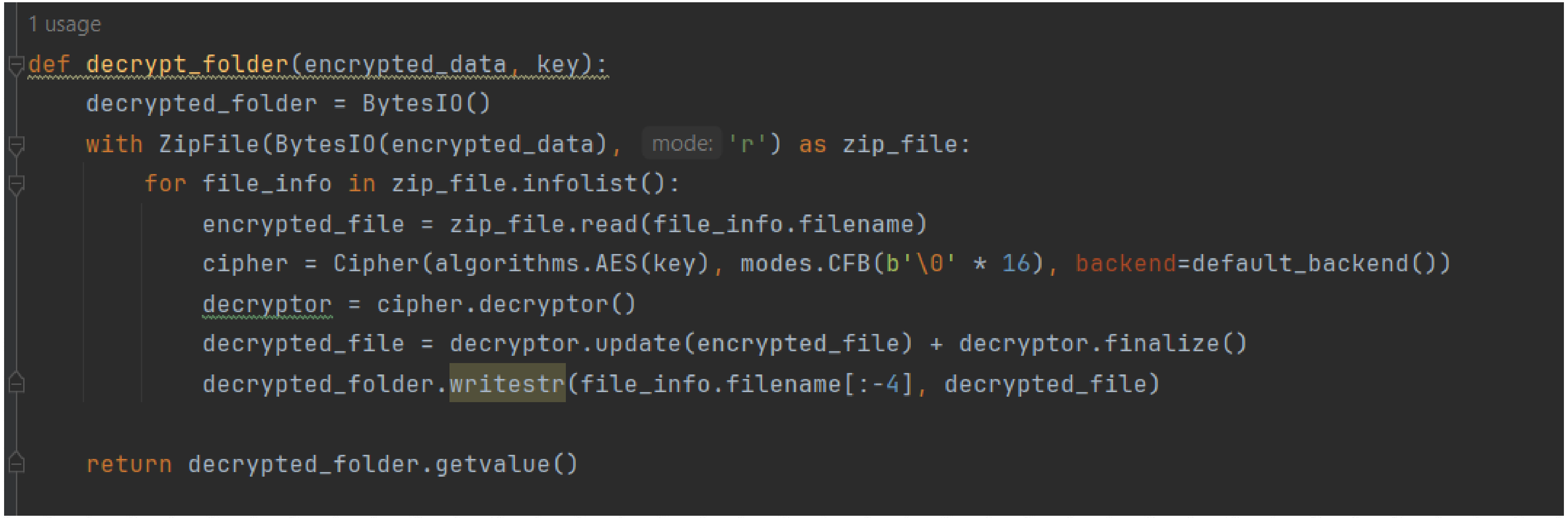
```python
from flask import Flask, render_template, request, jsonify, send_file, redirect, url_for
from werkzeug.utils import secure_filename
import os
from zipfile import ZipFile
from io import BytesIO
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = "E:/techgium_work/data"
app.config['ENCRYPTED_FOLDER'] = "E:/techgium_work/download"
app.config['DECRYPTED_FOLDER'] = "E:/techgium_work/decrypted"
app.config['ALLOWED_EXTENSIONS'] = {'zip'}

1 usage
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in app.config['ALLOWED_EXTENSIONS']
```

Current Status of POC



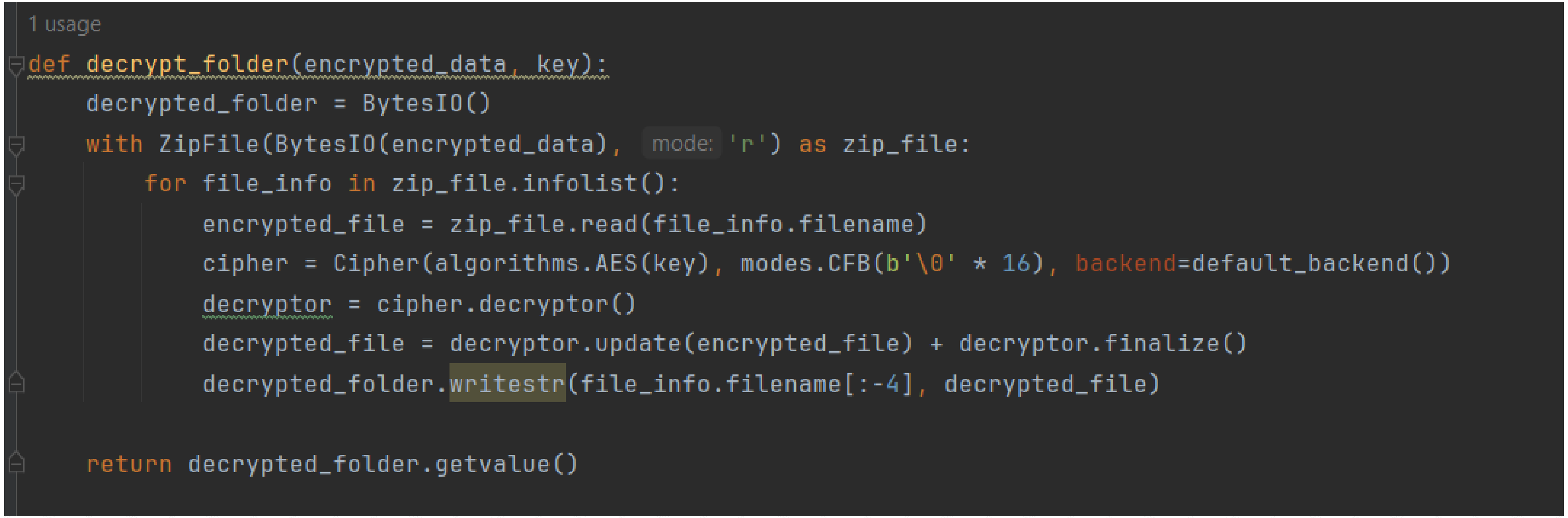
```
def encrypt_folder(input_folder, key):
    zip_buffer = BytesIO()
    with ZipFile(zip_buffer,  mode: 'a') as zip_file:
        for root, _, files in os.walk(input_folder):
            for file in files:
                input_path = os.path.join(root, file)
                relative_path = os.path.relpath(input_path, input_folder)

                with open(input_path, 'rb') as f:
                    plaintext = f.read()

                cipher = Cipher(algorithms.AES(key), modes.CFB(b'\0' * 16), backend=default_backend())
                encryptor = cipher.encryptor()
                ciphertext = encryptor.update(plaintext) + encryptor.finalize()

                # Add encrypted file to the zip
                zip_file.writestr(relative_path + '.enc', ciphertext)

    return zip_buffer.getvalue()
```

Current Status of POC



```
1 usage
def decrypt_folder(encrypted_data, key):
    decrypted_folder = BytesIO()
    with ZipFile(BytesIO(encrypted_data), mode: 'r') as zip_file:
        for file_info in zip_file.infolist():
            encrypted_file = zip_file.read(file_info.filename)
            cipher = Cipher(algorithms.AES(key), modes.CFB(b'\0' * 16), backend=default_backend())
            decryptor = cipher.decryptor()
            decrypted_file = decryptor.update(encrypted_file) + decryptor.finalize()
            decrypted_folder.writestr(file_info.filename[:-4], decrypted_file)

    return decrypted_folder.getvalue()
```

Current Status of POC



```
1 usage
def decrypt_folder(encrypted_data, key):
    decrypted_folder = BytesIO()
    with ZipFile(BytesIO(encrypted_data), mode: 'r') as zip_file:
        for file_info in zip_file.infolist():
            encrypted_file = zip_file.read(file_info.filename)
            cipher = Cipher(algorithms.AES(key), modes.CFB(b'\0' * 16), backend=default_backend())
            decryptor = cipher.decryptor()
            decrypted_file = decryptor.update(encrypted_file) + decryptor.finalize()
            decrypted_folder.writestr(file_info.filename[:-4], decrypted_file)

    return decrypted_folder.getvalue()
```

Further Plans and Testing process

- The most important plan is to complete the library that will help in getting the offline mac address to the server and also to implement the custom Data loader class for making the accessibility of the data while training the model using our dataset.
- Successfully enabling the downloading the enabling function to let the user download the dataset upon encryption .
- The testing process will involve the verification of the encrypted folder and its decrypt function.
- The functioning of the library for sending the mac for specific user and also the functioning of the Data loader class .
- Testing the user authentication system for each specific user for our specific database storage.

Queries to the mentors

- If any other method is available for the unique user identification that is specific to the user and his/her system as root level .
- The scalability of the software we are making should me kept in mind for all types of datasets or not.

L&T Technology Services