```python
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.core.files.storage import FileSystemStorage
import hashlib
import os
import base64
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives import serialization, hashes

# Import models
from SecureFiles.models import UploadedFile, FileSecurityMetrics

# Generate ECC key pair if not exists
def generate_ecc_keys():
    private_key = ec.generate_private_key(ec.SECP256R1())
    public_key = private_key.public_key()

    private_pem = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.TraditionalOpenSSL,
        encryption_algorithm=serialization.NoEncryption()
    )

    public_pem = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )

    return private_pem, public_pem

PRIVATE_KEY, PUBLIC_KEY = generate_ecc_keys()

# File Upload View
def upload_file(request):
    if request.method == 'POST' and request.FILES['file']:
        uploaded_file = request.FILES['file']
        fs = FileSystemStorage()
        filename = fs.save(uploaded_file.name, uploaded_file)
        file_path = fs.path(filename)

        # Generate SHA-256 Hash
        sha256_hash = compute_sha256(file_path)

        # Encrypt file using ECC
        encrypted_file_path = encrypt_file(file_path)

        # Save to Database
        UploadedFile.objects.create(
            filename=filename,
            file_hash=sha256_hash,
            encrypted_path=encrypted_file_path
        )

        return HttpResponse("File uploaded, encrypted, and secured successfully.")

    return render(request, 'upload_file.html')

# Compute SHA-256 Hash
def compute_sha256(file_path):
    sha256 = hashlib.sha256()
    with open(file_path, "rb") as file:
        while chunk := file.read(4096):
            sha256.update(chunk)
    return sha256.hexdigest()

# Encrypt File using ECC Public Key
def encrypt_file(file_path):
    with open(file_path, "rb") as file:
        data = file.read()

    public_key = serialization.load_pem_public_key(PUBLIC_KEY)
    encrypted_data = public_key.encrypt(
        data,
        ec.ECIES(hashes.SHA256())
    )

    encrypted_file_path = file_path + ".enc"
    with open(encrypted_file_path, "wb") as file:
        file.write(encrypted_data)

    return encrypted_file_path

# Decrypt File using ECC Private Key
def decrypt_file(request, file_id):
    try:
        file_record = UploadedFile.objects.get(id=file_id)
        encrypted_path = file_record.encrypted_path
        private_key = serialization.load_pem_private_key(PRIVATE_KEY, password=None)

        with open(encrypted_path, "rb") as file:
            encrypted_data = file.read()

        decrypted_data = private_key.decrypt(
            encrypted_data,
            ec.ECIES(hashes.SHA256())
        )

        decrypted_file_path = encrypted_path.replace(".enc", ".dec")
        with open(decrypted_file_path, "wb") as file:
            file.write(decrypted_data)

        return HttpResponse("File decrypted successfully.")

    except Exception as e:
        return HttpResponse(f"Error: {str(e)}")

# File Integrity Check
def verify_integrity(request, file_id):
    try:
        file_record = UploadedFile.objects.get(id=file_id)
        original_hash = file_record.file_hash
        computed_hash = compute_sha256(file_record.encrypted_path)

        if original_hash == computed_hash:
            return HttpResponse("File integrity verified: No tampering detected.")
        else:
            return HttpResponse("Warning: File integrity compromised!")

    except Exception as e:
        return HttpResponse(f"Error: {str(e)}")

# View Uploaded Files
def view_files(request):
    files = UploadedFile.objects.all()
    return render(request, 'view_files.html', {'files': files})

# Performance Metrics
def calculate_security_metrics():
    total_files = UploadedFile.objects.count()
    secure_files = UploadedFile.objects.filter(file_verified=True).count()
    accuracy = (secure_files / total_files) * 100 if total_files > 0 else 0
    FileSecurityMetrics.objects.create(
        total_files=total_files,
        secure_files=secure_files,
        accuracy=accuracy
    )
    return accuracy

def view_security_metrics(request):
    metrics = FileSecurityMetrics.objects.last()
    return render(request, 'security_metrics.html', {'metrics': metrics})
```