

Softcopy – Practical-1.2

Code

```
import random
import string
from captcha.image import ImageCaptcha
from PIL import Image
import hashlib
import time
import os

class CaptchaManager:
    """
    A comprehensive CAPTCHA generation and verification system
    """

    def __init__(self, length=6, width=280, height=90):
        """
        Initialize the CAPTCHA manager

        Args:
            length: Number of characters in CAPTCHA
            width: Image width in pixels
            height: Image height in pixels
        """
        self.length = length
        self.image_captcha = ImageCaptcha(width=width,
height=height)
        self.captcha_store = {} # Store CAPTCHA with expiry time
        self.expiry_time = 180 # 3 minutes in seconds

    def generate_captcha_text(self):
        """
        Generate random CAPTCHA text

        Returns:
            String of random alphanumeric characters
        """
        characters = string.ascii_uppercase + string.digits
```

```

        # Exclude ambiguous characters like 0, O, I, 1
        characters = characters.replace('0', '').replace('O',
        '').replace('I', '').replace('1', '')
        captcha_text = ''.join(random.choice(characters) for _ in
range(self.length))
        return captcha_text

```

```

def generate_captcha_image(self, captcha_text=None,
filename='captcha.png'):

```

```

    """

```

```

    Generate CAPTCHA image

```

```

    Args:

```

```

        captcha_text: Text to display (auto-generated if None)
        filename: Output filename

```

```

    Returns:

```

```

        Tuple of (captcha_text, session_id, filename)

```

```

    """

```

```

    if captcha_text is None:

```

```

        captcha_text = self.generate_captcha_text()

```

```

    # Generate image

```

```

    data = self.image_captcha.generate(captcha_text)

```

```

    self.image_captcha.write(captcha_text, filename)

```

```

    # Create session ID

```

```

    session_id = hashlib.md5(
        f"{captcha_text}{time.time()}".encode()
    ).hexdigest()

```

```

    # Store CAPTCHA with expiry time

```

```

    self.captcha_store[session_id] = {
        'text': captcha_text,
        'timestamp': time.time()
    }

```

```

    return captcha_text, session_id, filename

```

```

def verify_captcha(self, session_id, user_input):

```

```

    """

```

Verify user's CAPTCHA input

Args:

 session_id: Unique session identifier

 user_input: User's input text

Returns:

 Dictionary with verification result

"""

Check if session exists

if session_id not in self.captcha_store:

 return {

 'success': False,

 'message': 'Invalid session or CAPTCHA expired'

 }

captcha_data = self.captcha_store[session_id]

current_time = time.time()

Check expiry

if current_time - captcha_data['timestamp'] >
self.expiry_time:

 del self.captcha_store[session_id]

 return {

 'success': False,

 'message': 'CAPTCHA expired. Please try again.'

 }

Verify CAPTCHA (case-insensitive)

if user_input.upper() == captcha_data['text'].upper():

 del self.captcha_store[session_id] # Remove after
successful verification

 return {

 'success': True,

 'message': 'CAPTCHA verified successfully!'

 }

else:

 return {

 'success': False,

 'message': 'Incorrect CAPTCHA. Please try again.'

```
}
```

```
def cleanup_expired(self):
    """
    Remove expired CAPTCHAs from storage
    """
    current_time = time.time()
    expired_sessions = [
        sid for sid, data in self.captcha_store.items()
        if current_time - data['timestamp'] > self.expiry_time
    ]
    for sid in expired_sessions:
        del self.captcha_store[sid]
```

Alternative simple implementation using Pillow

```
class SimpleCaptcha:
```

```
    """
```

```
    Simple CAPTCHA implementation using Pillow
```

```
    """
```

```
@staticmethod
```

```
def generate(length=6):
```

```
    """
```

```
    Generate simple CAPTCHA with noise
```

```
    Returns:
```

```
        Tuple of (captcha_text, Image object)
```

```
    """
```

```
    from PIL import ImageDraw, ImageFont
```

```
    # Generate random text
```

```
    chars = ''.join(random.choice(string.ascii_uppercase +
string.digits)
```

```
                        for _ in range(length))
```

```
    # Create image
```

```
    img = Image.new('RGB', (200, 100), color=(73, 109, 137))
```

```
    draw = ImageDraw.Draw(img)
```

```

# Try to load a font, fallback to default if not available
try:
    font = ImageFont.truetype('arial.ttf', 36)
except:
    font = ImageFont.load_default()

# Draw characters with random positions
x = 20
for char in chars:
    y = random.randint(25, 35)
    draw.text((x, y), char, font=font, fill=(255, 255, 0))
    x += 30

# Add noise (dots)
for _ in range(150):
    x = random.randint(0, 200)
    y = random.randint(0, 100)
    draw.point((x, y), fill=(random.randint(0, 255),
                                random.randint(0, 255),
                                random.randint(0, 255)))

# Add lines for distortion
for _ in range(5):
    start = (random.randint(0, 200), random.randint(0,
100))
    end = (random.randint(0, 200), random.randint(0, 100))
    draw.line([start, end], fill=(255, 255, 255), width=1)

return chars, img

# Example usage and testing
def main():
    """
    Demonstration of CAPTCHA generation and verification
    """
    print("=== CAPTCHA Generation and Verification Demo ===\n")

    # Initialize CAPTCHA manager
    manager = CaptchaManager(length=6)

```

```

# Generate CAPTCHA
captcha_text, session_id, filename =
manager.generate_captcha_image()
print(f"CAPTCHA Generated: {captcha_text}")
print(f"Session ID: {session_id}")
print(f"Image saved as: {filename}\n")

# Simulate user input
print("Verification Tests:")

# Test 1: Correct input
result = manager.verify_captcha(session_id, captcha_text)
print(f"Test 1 (Correct): {result}")

# Generate new CAPTCHA for next test
captcha_text, session_id, _ =
manager.generate_captcha_image('test_captcha_2.png')

# Test 2: Incorrect input
result = manager.verify_captcha(session_id, "WRONG")
print(f"Test 2 (Incorrect): {result}")

# Test 3: Invalid session
result = manager.verify_captcha("invalid_session", "TEST")
print(f"Test 3 (Invalid Session): {result}")

# Demonstrate simple CAPTCHA
print("\n=== Simple CAPTCHA Demo ===")
captcha_text, img = SimpleCaptcha.generate()
img.save('simple_captcha.png')
print(f"Simple CAPTCHA Text: {captcha_text}")
print("Image saved as: simple_captcha.png")

if __name__ == "__main__":
    main()
# END OF CODE

```

Output

```
(Codes) [overnion - Codes (/run/media/overnion/persistence/Files/git/sppu-be-comp-content/Cybe
6:27]
$ python3 Practical-2.py
=== CAPTCHA Generation and Verification Demo ===

CAPTCHA Generated: MJEXZB
Session ID: b8ccf2f65b4a27cc2435b8f29a013e3f
Image saved as: captcha.png

Verification Tests:
Test 1 (Correct): {'success': True, 'message': 'CAPTCHA verified successfully!'}
Test 2 (Incorrect): {'success': False, 'message': 'Incorrect CAPTCHA. Please try again.'}
Test 3 (Invalid Session): {'success': False, 'message': 'Invalid session or CAPTCHA expired'}

=== Simple CAPTCHA Demo ===
Simple CAPTCHA Text: 4CID5S
Image saved as: simple_captcha.png
```