ABSTRACT

This project focuses on enhancing user account security by implementing a system that detects unauthorized access attempts and takes immediate action. It combines basic user authentication with automated monitoring, image capturing, and email alerting functionalities to create a robust security application.

If a user fails to input the correct password within three attempts, the application captures an image using the computer's webcam, saves it, and sends an alert email with the captured image attached to the registered email address of the user. The email notification includes a warning message about the unauthorized access attempt. This project leverages the OpenCV library for capturing images and the smtplib library for sending emails, making it a simple yet effective security measure for protecting user accounts.

INTRODUCTION

This project aims to enhance user account security by integrating multiple technologies to detect and respond to unauthorized access attempts. Utilizing Python, it combines user authentication, webcam image capture through OpenCV, and email notifications using smtplib. Upon detecting multiple failed login attempts, the system captures an image of the potential intruder and sends it to the registered user's email, thereby providing real-time alerts and reinforcing security measures. This approach not only prevents unauthorized access but also raises user awareness about potential security breaches.

Key Features:

- **User Authentication:**
  - The application prompts the user to enter their username and password.
  - It verifies the entered credentials against a predefined list of valid users.
  - If the username is invalid, the system displays an "Invalid username" message and exits.

- **Login Attempts Management:**

- Users are allowed up to three attempts to enter the correct password.
- After each incorrect password entry, the system informs the user of the remaining attempts.
- A specific warning is given on the final attempt.

❖ **Unauthorized Access Handling:**

- After three failed login attempts, the system assumes an unauthorized access attempt.
- The computer's webcam is activated to capture an image of the person attempting access.
- This image is saved locally for further action**.**

❖ **Email Notification:**

- An email is prepared with the subject "Unauthorized Access Attempt" and a warning message in the body.
- The captured image is attached to the email
- Using the smtplib library, the system sends this email to the registered email address of the user associated with the login attempt.

❖ **Technical Details -**

- Image Capture: The OpenCV library is used to interface with the webcam and capture an image of the unauthorized user.
- Email Sending: The smtplib library establishes a secure connection with the Gmail SMTP server to send the email with the captured image attached.
- HTML Email: The email body is formatted using HTML to highlight the warning message in red, making it easily noticeable.

❖ **<u>User Interaction</u>*:***

❖ **Login Prompt:**

- Users are prompted to enter their username and password.
- The system provides feedback for incorrect attempts, including the number of attempts left and specific warnings on the last attempt.

❖ **Unauthorized Access Alert:**

- Upon reaching the maximum number of failed attempts, the system captures an image and informs the user that an email has been sent to their registered address.

❖ Objectives:

- Security Enhancement: The project aims to add a security layer by detecting and responding to unauthorized access attempts.
- Real-Time Monitoring: The system automates the detection of multiple failed logins attempts and takes immediate action.
- User Notification: It ensures users are promptly informed of potential security breaches through real-time email notifications.
- Integration Demonstration: The project showcases the practical integration of image processing (OpenCV) and email communication (smtplib) libraries in Python.
- Educational Tool: It serves as a hands-on example for understanding and implementing security measures in a Python application.

Overall, this project presents a straightforward yet effective approach to improving account security by combining user authentication, real-time monitoring, and automated alerting through image capture and email notifications. The integration of OpenCV and smtplib demonstrates how these libraries can be used in practical security applications, making this project an excellent educational tool for learning and implementing basic security features in Python.

EXPLAINATION OF THE CODE

❖ Importing Modules

- cv2: This module is from OpenCV, used for capturing and processing images.
- smtplib: Used for sending emails via the Simple Mail Transfer Protocol (SMTP).
- MIMEMultipart, MIMEText, MIMEImage: Classes from the email.mime module used to create email messages with multiple parts, including text and images.
- datetime: Provides date and time manipulation capabilities.
- os: Used for interacting with the operating system, such as file path manipulations and directory operations.
- requests: A module for making HTTP requests, used here to get location information based on IP.

❖ Function Definitions

❖ *Sending an Email*

- send_email: A function to send an email with details about login attempts.
- sender_email: The email address from which the email will be sent.
- password: The password for the sender's email account.
-

- MIMEMultipart(): Creates a new email message object.
- message['From']: Sets the sender email.
- message['To']: Sets the receiver email.
- If is_owner is True, the email subject and body are set for a password reset request.
- If is_owner is False, the email subject and body indicate an unauthorized access attempt, including the number of attempts and the location.
- message.attach(MIMEText(body, 'html')): Attaches the HTML formatted body to the email.
- If image_path is provided, the image is read and attached to the email.
- Connects to the Gmail SMTP server using SSL.
- Logs into the sender's email account.

- Sends the email message to the receiver.

❖ Capturing an Image

- capture_image: A function to capture an image using the webcam and save it.
- If the specified folder path does not exist, it creates the folder.

- cv2.VideoCapture(0): Opens the default camera.
- ret, frame = cap.read(): Captures a frame from the webcam.
- cv2.imwrite(image_path, frame): Saves the captured frame to the specified path.
- cap.release(): Releases the camera resource.
- Returns the path to the saved image.

❖ Detecting Faces in an Image

- detect_face: A function to detect faces in an image.
- Loads a pre-trained face detection model.
- Reads the image and converts it to grayscale.
- Detects faces in the image.
- Returns True if any faces are detected, otherwise False.

❖ Getting Location Based on IP

- get_location: A function to get the geographic location based on the IP address.
- Makes an HTTP request to ipinfo.io to get location data.
- Returns the formatted location string or "Unknown" if the request fails.

❖ Main Function

- main: The main function that orchestrates the login attempt process.
- Sets the folder path and filename for the captured image.
- Captures an image and detects if it contains a face.
- Gets the location based on the IP address.

- Defines a dictionary with usernames, their associated email addresses, and passwords.
- Prompts the user to enter a username.
- Checks if the username exists in the users dictionary. If not, it prints an error and exits.
- Retrieves the user data for the entered username.
- Initializes login attempts and sets the maximum allowed attempts.
- Determines if an intruder is detected based on face detection.
- While the number of login attempts is less than the maximum allowed:
    - Prompts the user to enter a password.
    - If the entered password is incorrect, increments the login attempts and prints the number of attempts left.
    - If the entered password is correct, prints a success message and sends an email if an intruder is detected, then exits.
- If the face detected is the owner, sends a password reset email.
- If an intruder is detected, sends an email with the captured image.
- Waits for a key press and then destroys all OpenCV windows.

❖ Main Check:

- Ensures the main function is called only if the script is executed directly, not when imported as a module.

OUTPUTS:

Output Examples:

1. **Successful Login with Owner Detection**:
    - **Scenario**: The user enters the correct password within the allowed attempts, and the detected face matches the owner.
    - **Console Output**:

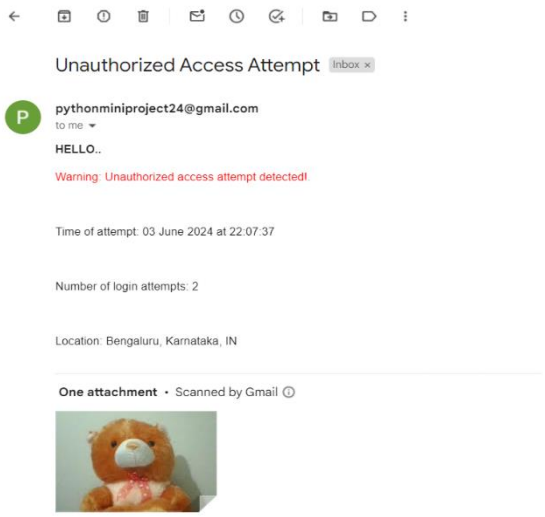
```
Enter your username: python
Enter your password python: abcd
Incorrect password. Attempts left: 2
Enter your password python: hello
Last attempt left...
Enter your password python: welcome
Login successful, python!
```

    - **Email**: No email is sent because the login is successful, and the face matches the owner.

2. **Successful Login Without Owner Detection**:
    - **Scenario**: The user enters the correct password within the allowed attempts, but an intruder is detected.
    - **Console Output**:

```
Enter your username: python
Enter your password python: howw
Incorrect password. Attempts left: 2
Enter your password python: whyy
Last attempt left...
Enter your password python: welcome
Login successful, python!
Intruder detected but password guessed correctly. Image sent to pythonminiproject24@gmail.com.
```

    - **Email**:
        - **Subject**: "Unauthorized Access Attempt"
        - **Body**:

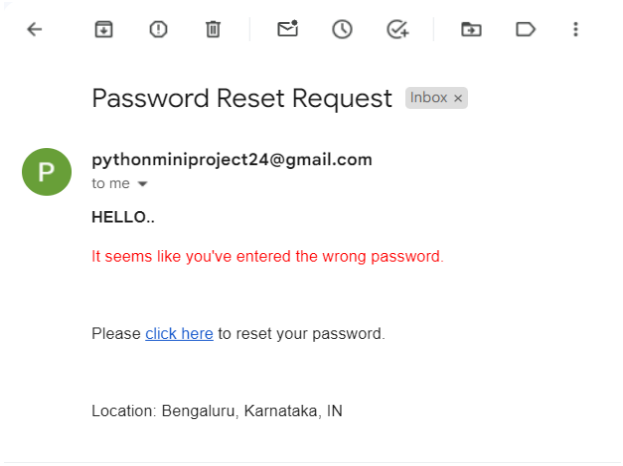


- **Attachment**: Captured image of the intruder.

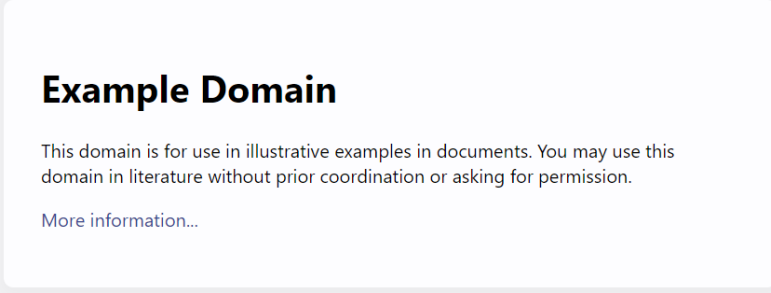
3. **Failed Login Attempts with Owner Detection**:
    - **Scenario**: The user fails to enter the correct password within the allowed attempts, and the detected face matches the owner.
    - **Console Output:**

```
Enter your username: python
Enter your password python: 1234
Incorrect password. Attempts left: 2
Enter your password python: 5678
Last attempt left...
Enter your password python: 9012
Try again later
Owner detected. Password reset email sent.
```

    - **Email**:
        - **Subject**: "Password Reset Request"
        - **Body**:



On clicking the link “click here”

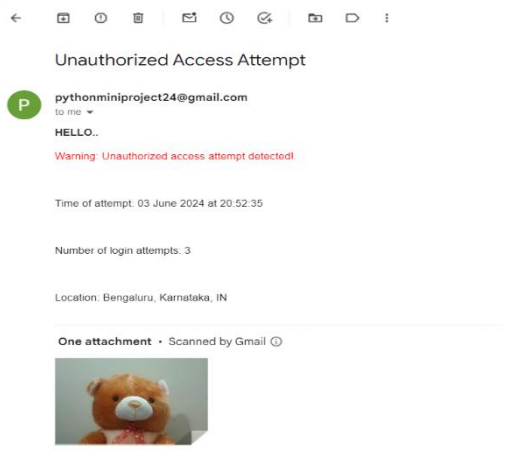


- **Attachment**: No attachment.

4. **Failed Login Attempts Without Owner Detection**:
    - **Scenario**: The user fails to enter the correct password within the allowed attempts, and no faces or an intruder is detected.
    - **Console Output**:

```
Enter your username: python
Enter your password python: 0160
Incorrect password. Attempts left: 2
Enter your password python: 0120
Last attempt left...
Enter your password python: 2006
Try again later
Unauthorized access attempt detected for python. Image sent to pythonminiproject24@gmail.com.
```

    - **Email**:
        - **Subject**: "Unauthorized Access Attempt"
        - **Body**:



- **Attachment**: Captured image of the intruder.

5. **Invalid Username:**
   - **Scenario**: The user enters a username that does not exist in the predefined user's dictionary.
   - **Console Output**:

```
Enter your username: jennie
Invalid username.
```

   - **Email: No Email is sent.**

CONCLUSION:

The code provides a comprehensive security mechanism for a login system by combining image capture, face detection, location retrieval, and email notifications. It ensures that both Intruders and owners are appropriately managed:

- **For Owners**: If the owner is detected but the password is incorrect, a password reset link is sent via email.
- **For Intruders**: If an unauthorized person attempts to access the system and fails to enter the correct password within three attempts, an alert email with the intruder's image is sent to the legitimate user's email address.

This system enhances security by not only relying on password protection but also by using face recognition and geographical location as additional layers of verification.