PROJECT TITLE :

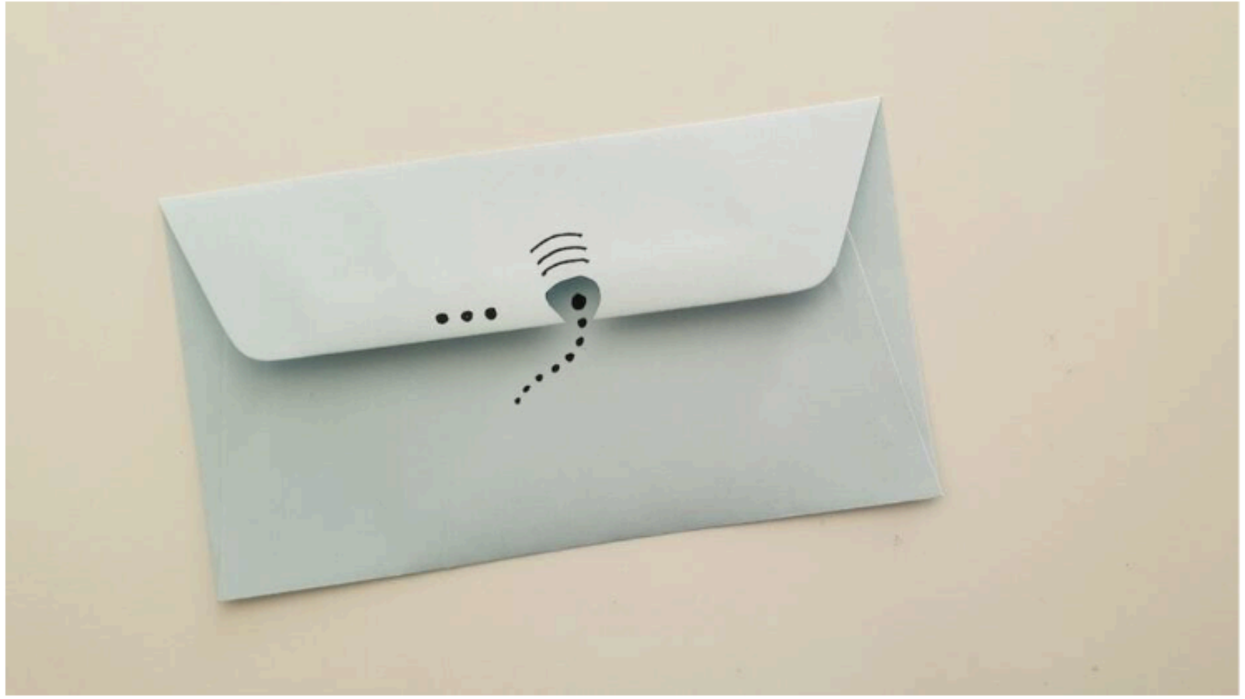
OTP VERIFICATION SYSTEM

AGENDA:

Introduction: Email OTP Verification Process Overview
Generating OTP: Random 6-digit Code, Function Implementation
Sending OTP via Email: SMTP Protocol, MIMEText and MIMEMultipart
Handling Sender Password: Secure Handling, Environment Variable
Implementing OTP Verification: User Input Validation, Matching OTPs
Email Configuration: Server, Port, Login Process
Error Handling: Exception Handling, Error Messages
Access Control: Successful Verification, Access Granted
Access Denied Scenario: Too Many Incorrect Attempts
Conclusion: Summary of OTP Verification Process, Security Best Practices

Introduction: Email OTP Verification Process Overview

- **Overview of OTP:** One-Time Password (OTP) is a security feature that provides a temporary code for user verification
- **Verification Process:** OTP is used to validate the identity of users during various authentication processes
- **Email Security:** Using email for OTP delivery requires secure protocols to protect sensitive information



CODE :

```
IMPORT RANDOM
IMPORT SMTPLIB
FROM EMAIL.MIME.TEXT IMPORT MIMETEXT
FROM EMAIL.MIME.MULTIPART IMPORT MIMEMULTIPART
IMPORT OS
```

Sending OTP via Email: SMTP Protocol, MIMEText and MIMEMultipart Modules

- **SMTP Protocol:** Simple Mail Transfer Protocol (SMTP) is used to send emails securely over the internet
- **MIMEText and MIMEMultipart:** Modules for creating email messages with text and attachments, essential for sending OTPs via email

```
# FUNCTION TO GENERATE A 6-DIGIT OTP
DEF GENERATE_OTP():
OTP = RANDOM.RANDINT(100000, 999999)
RETURN STR(OTP)
```

Generating OTP: Random 6-digit Code

- **Randomized Code Generation:** The OTP is a randomly generated 6-digit code to enhance security and unpredictability
- **Function Implementation:** Implementing a function to generate OTP ensures a consistent and reliable method for creating unique verification codes

```
IMPORT OS
# SET THE ENVIRONMENT VARIABLE FOR THE SENDER PASSWORD
OS.ENVIRON['SENDER_PASSWORD'] = 'HKZD SMHE NDYX BVGE'
```

Setting Environment Variables

- **Security in Code Settings:** By setting environment variables like SENDER_PASSWORD, sensitive information can be securely managed and accessed in the code without exposing it to potential risks like hardcoding.

- **Protection against Exposure:** Avoids the exposure of sensitive information, such as passwords, to version control systems or accidental sharing in code repositories, enhancing the overall security posture of the application.

```
# FUNCTION TO SEND THE OTP VIA EMAIL
DEF SEND_OTP_VIA_EMAIL(RECEIVER_EMAIL, OTP):
SENDER_EMAIL = "NVENKATESWARLU95@GMAIL.COM"
SENDER_PASSWORD = OS.GETENV("SENDER_PASSWORD")
IF NOT SENDER_PASSWORD:
PRINT("ERROR: SENDER PASSWORD IS NOT SET. PLEASE ENSURE THE
SENDER_PASSWORD ENVIRONMENT VARIABLEIS SET.")
RETURN

MESSAGE = MIMEMULTIPART()
MESSAGE["FROM"] = SENDER_EMAIL
MESSAGE["TO"] = RECEIVER_EMAIL
MESSAGE["SUBJECT"] = "YOUR OTP CODE"

BODY = F"YOUR OTP CODE IS {OTP}"
MESSAGE.ATTACH(MIMETEXT(BODY, "PLAIN"))

TRY:
SERVER = SMTPLIB.SMTP("SMTP.GMAIL.COM", 587)
SERVER.STARTTLS()
SERVER.LOGIN(SENDER_EMAIL, SENDER_PASSWORD)
ERVER.SENDMAIL(SENDER_EMAIL, RECEIVER_EMAIL, MESSAGE.AS_STRING())
SERVER.CLOSE()
PRINT("OTP SENT SUCCESSFULLY!")
EXCEPT SMTPLIB.SMTPEXCEPTION AS E:
PRINT(F"FAILED TO SEND OTP: {E}")
EXCEPT EXCEPTION AS E:
PRINT(F"AN ERROR OCCURRED: {E}")
```

Handling Sender Password

- **Secure Password Management:** Utilizing environment variables for sender passwords ensures secure handling and access control, reducing the likelihood of unauthorized access or misuse of sensitive authentication credentials.
- **Environment Variable Usage:** By incorporating sender passwords as environment variables, developers can adhere to best practices for secure password management and prevent accidental exposure of credentials in code.

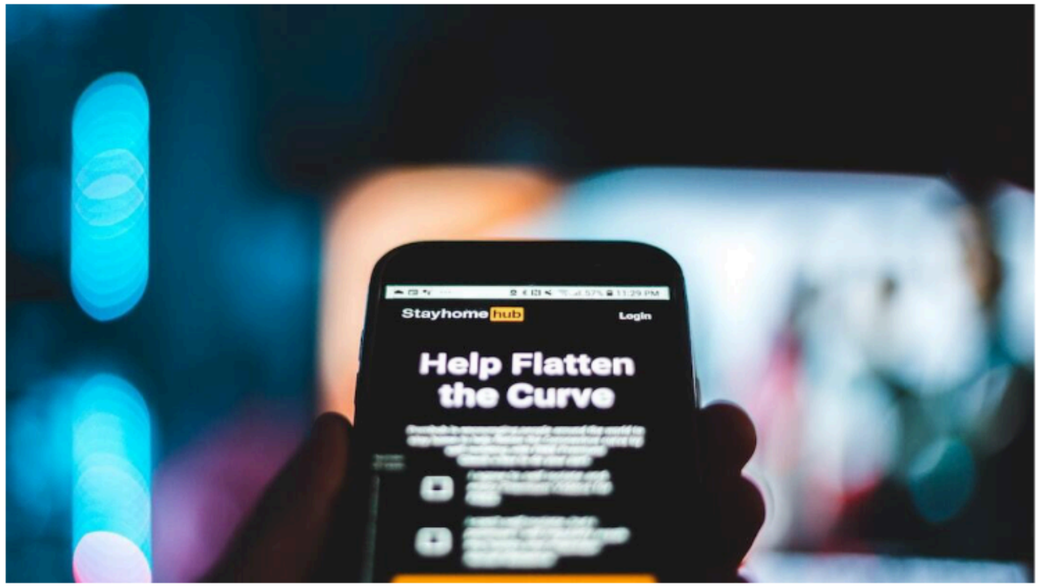
FUNCTION TO PROMPT THE USER TO ENTER THE OTP

```
DEF GET_USER_OTP():
RETURN INPUT("ENTER THE OTP SENT TO YOUR EMAIL: ")
```

User Input

- **User Interaction Dynamics:** Engaging users through input prompts for receiver email enhances the user experience and interaction flow, making the OTP verification process more user-friendly and intuitive.

- **Receiver Email Validation:** Validating and processing receiver email input ensures the accuracy and completeness of user-provided information, leading to more effective communication and verification procedures.



FUNCTION TO VERIFY IF THE ENTERED OTP MATCHES THE GENERATED OTP
DEF VERIFY_OTP(GENERATED_OTP, USER_OTP):
RETURN GENERATED_OTP == USER_OTP

Email Configuration

- **Server and Port Settings:** Configuring the email server and port parameters ensures proper communication channels for sending OTP emails, facilitating reliable and efficient delivery of verification codes to users.

- **Login Process and Authentication:** Implementing secure login processes and authentication mechanisms enhances the overall email security, safeguarding the delivery of OTPs and preventing unauthorized access to email communication channels.

```
# MAIN FUNCTION TO HANDLE THE OTP VERIFICATION PROCESS
DEF MAIN():
    OTP = GENERATE_OTP()
    RECEIVER_EMAIL = "KAREMPUDILOKESH123@GMAIL.COM"

    SEND_OTP_VIA_EMAIL(RECEIVER_EMAIL, OTP)

    ATTEMPTS = 3
    WHILE ATTEMPTS > 0:
        USER_OTP = GET_USER_OTP()
        IF VERIFY_OTP(OTP, USER_OTP):
            PRINT("ACCESS GRANTED!")
            RETURN
        ELSE:
            PRINT("INCORRECT OTP. PLEASE TRY AGAIN.")
            ATTEMPTS -= 1

    PRINT("ACCESS DENIED. TOO MANY INCORRECT ATTEMPTS.")

IF __NAME__ == "__MAIN__":
    MAIN()
```

Email Configuration: Server, Port, Login Process

- **Email Server Setup:** Configuring the email server with the necessary settings for sending and receiving emails
- **Port Configuration:** Specifying the port number for SMTP communication, typically port 587 for secure email communication
- **Login Process:** Authenticating the sender's email address and password to establish a connection with the email server

Error Handling: Exception Handling, Error Messages

- **Exception Handling:** Implementing mechanisms to gracefully handle unexpected errors and exceptions during the OTP verification process
- **Error Messages:** Providing clear and concise error messages to guide users in the event of OTP verification failures

OTP SENT SUCCESSFULLY!

ENTER THE OTP SENT TO YOUR EMAIL: 886424

ACCESS GRANTED!

Access Control: Successful Verification, Access Granted

- **Successful Verification:** Confirming the validity of the OTP entered by the user to grant access for verification
- **Access Granted:** Allowing the user to proceed with the desired action after successful OTP verification

ENTER THE OTP SENT TO YOUR EMAIL: 4531

INCORRECT OTP. PLEASE TRY AGAIN.

ACCESS DENIED. TOO MANY INCORRECT ATTEMPTS.

Access Denied Scenario: Too Many Incorrect Attempts

- **Too Many Incorrect Attempts:** Triggering access denial when the user exceeds the allowed number of failed OTP verification attempts

Conclusion: Summary of OTP Verification Process, Security Best Practices

- **Summary of OTP Verification Process:** Recapitulating the key steps involved in the OTP verification process from generation to access control

- **Security Best Practices:** Highlighting the importance of secure practices such as password encryption, error handling, and access control to enhance the security of the OTP verification process

THANK YOU

PRESENTED BY NALLABOTHULA VENKATESWARLU