**OTP Verification System - Project Documentation**

**Overview**

This project, implemented in **Google Colab**, provides a secure OTP (One-Time Password) verification system. The system generates a 6-digit OTP, sends it to a user's email address, and validates the OTP entered by the user within a specific time limit.

**Features**

1. **OTP Generation:** Creates a random 6-digit numeric OTP.
2. **Email Sending via SMTP:** Sends the OTP securely to the user's email.
3. **Time-Limited OTP Verification:** Ensures OTP is only valid for a defined duration (default: 5 minutes).
4. **Retry Mechanism:** Allows multiple attempts for entering the correct OTP.

**Implementation Details**

**Environment**

* Platform: Google Colab
* Libraries Used: random, smtplib, email. message, time

**Code Workflow**

1. **Generate OTP**:

python

Copy code

def create\_otp ():

"""Generate a 6-digit numeric OTP."""

return ''.join([str(random.randint(0, 9)) for \_ in range(6)])

* **Randomly generates a 6-digit OTP using the random. randint () function.**

1. **Send OTP via Email**:

python

Copy code

def send\_otp\_email (recipient\_email, otp\_code):

"""Send the OTP to the user's email address."""

smtp\_server = 'smtp.gmail.com'

smtp\_port = 587

sender\_email = 'your\_email@gmail.com'

sender\_password = 'your\_app\_password'

try:

server = smtplib.SMTP(smtp\_server, smtp\_port)

server.starttls()

server.login(sender\_email, sender\_password)

email\_message = EmailMessage()

email\_message['Subject'] = "OTP Verification"

email\_message['From'] = sender\_email

email\_message['To'] = recipient\_email

email\_message.set\_content(f"Your OTP code is: {otp\_code}")

server.send\_message(email\_message)

print(f"OTP has been sent to {recipient\_email}.")

server.quit()

except Exception as e:

print(f"Failed to send OTP email. Error: {e}")

* + **Connects to Gmail's SMTP server.**
  + **Authenticates using sender credentials (configured as an App Password).**
  + **Sends the OTP email using EmailMessage.**

1. **Validate OTP**:

python

Copy code

def validate\_otp(user\_otp, original\_otp, start\_time, expiration\_time):

"""Check if the entered OTP matches and is within the allowed time."""

within\_time\_limit = (time.time() - start\_time) <= expiration\_time

return user\_otp == original\_otp and within\_time\_limit

* + **Verifies if:**
    - **OTP entered by the user matches the one generated.**
    - **Time since OTP generation is within the allowed duration**.

1. **Main Script**:

python

Copy code

if \_\_name\_\_ == "\_\_main\_\_":

max\_retries = 3

otp\_valid\_duration = 300 # 5 minutes

recipient\_email = input("Enter your email address: ")

otp\_code = create\_otp()

send\_otp\_email(recipient\_email, otp\_code)

start\_time = time.time()

attempts = 0

while attempts < max\_retries:

remaining\_attempts = max\_retries - attempts

entered\_otp = input(f"Enter the OTP (Attempts left: {remaining\_attempts}): ")

if validate\_otp(entered\_otp, otp\_code, start\_time, otp\_valid\_duration):

print("OTP Verified. Access granted.")

break

else:

attempts += 1

if (time.time() - start\_time) > otp\_valid\_duration:

print("The OTP has expired. Please request a new one.")

break

elif attempts < max\_retries:

print("Incorrect OTP. Please try again.")

else:

print("Maximum attempts reached. Access denied.")

* + **Executes the OTP generation, email sending, and validation in a loop.**
  + **Limits the number of retries to prevent brute force attempts.**

**Key Considerations**

1. **App Password**:
   * Use an **App Password** for Gmail instead of your primary password.
   * Generate it from your Google Account Security settings.
2. **Environment Security**:
   * Avoid exposing sensitive credentials (use environment variables or secrets management in Colab).

**Future Enhancements**

1. Add SMS integration for OTP delivery.
2. Include logging to track OTP requests and verification attempts.
3. Improve user interface for better interaction (e.g., using a web-based form).

**Error Handling**

* **Email Sending**:
  + Displays errors if the email fails to send (e.g., due to incorrect credentials or network issues).
* **Invalid OTP**:
  + Prompts the user to re-enter the OTP if it's incorrect, as long as attempts are left.
* **Expired OTP**:
  + Notifies the user if the OTP expires before validation.