OTP Verification System: Project Documentation

Project Title: OTP Verification System

Problem Statement:

Develop an OTP (One-Time Password) verification system that generates a 6-digit OTP, sends it to the user's email address, and verifies the OTP entered by the user. If the OTP matches the generated one, access is granted; otherwise, it is denied

Project Structure:

1. **OTP Generation**:
   A 6-digit OTP is generated using Python's random.randint function to ensure randomness and uniqueness for each session.
2. **OTP Sending**:
   The OTP is emailed to the user using Python's smtplib for SMTP communication with Gmail. The email content is structured for clarity and contains the OTP, sender's details, and subject.
3. **OTP Verification**:
   The system prompts the user to input the received OTP. This input is then validated against the stored OTP. Upon matching, access is granted, and the session can be proceeded. If it fails, appropriate error messages are provided.
4. **Error Handling**:
   Proper error handling mechanisms are incorporated for cases like:
   - Missing or invalid email/name.
   - Invalid OTP input.
   - OTP expiration.

   Additionally, the system informs the user of necessary actions in case of any failure**Flask Web Interface**:
A simple front-end HTML page is created using Flask to enable users to enter their name, email address, and OTP. The interface also provides feedback based on the OTP verification process.

Technologies and Tools Used:

- **Backend**: Flask (Python)
- **Email Sending**: smtplib, email.message
- **Frontend**: HTML, CSS, JavaScript (jQuery)
- **Random OTP Generation**: random library
- **SMTP Service**: Gmail SMTP

Project Files:

1. **Main Python Script**:
   otp_verification.py: This contains the main backend code for generating OTP, sending it to the user's email, and verifying the OTP via a Flask-based web server.

2. **HTML Template**:
   webpage.html: Provides the user interface for submitting user details (name, email, and OTP).

Installation and Setup:

1. **Clone the Project**:
   Download or clone the repository containing the project files.

2. **Install Dependencies**:
   Install the required Python packages like Flask and smtplib by running:pip install Flask

3. **SMTP Email Setup**:
   For sending emails via Gmail, you will need to:

   - Enable "Less secure apps" in your Gmail account (for development purposes).
   - Replace the sender_email and password in the Python script with your actual Gmail credentials.
   - Ensure proper security practices in production by using environment variables or a secure method for storing credentials.

4. **Run the Flask App**:
   Start the Flask server to run the OTP verification systeM

```
python otp_verification.py
```

5. **Open in Browser**:
   Once the server is running, open the web application by visiting http://127.0.0.1:5000/ in your web browser.

Project Functionality:

1. **OTP Generation**:
   The system generates a random 6-digit OTP each time the /send_otp route is called. It ensures that the OTP is unique for every request and session.

2. **Sending OTP via Email**:
   Once generated, the OTP is sent via the Gmail SMTP server using Python’s smtplib and EmailMessage modules. The email includes the subject, body, and recipient details.

3. **OTP Verification**:
   When the user enters the OTP on the web interface, the system checks the entered OTP

against the stored OTP for the session. If they match, a success message is displayed, allowing the user to proceed. Otherwise, an error message is shown.

Code Explanation:

1. **Index Route (/)**:
   Renders the HTML template for the OTP verification system, presenting fields for the user to input their name and email.

2. **Send OTP Route (/send_otp)**:
   Accepts user details like name and email, generates the OTP, and sends it via email to the specified address. It also provides confirmation that the OTP has been sent.

3. **Verify OTP Route (/verify_otp)**:
   Accepts the OTP entered by the user, compares it with the generated OTP stored in memory for that particular email address. Based on the result, it returns success or failure messages.

Error Handling:

- **Invalid or Missing Input**:
  If the name or email is missing during OTP submission, the system returns an error message prompting the user to provide the required fields.

- **Invalid OTP**:
  If the user enters an incorrect OTP or an expired one, the system denies access and returns an error, prompting the user to try again or request a new OTP.

- **Retry Mechanism**:
  In case of OTP mismatch or failure, the system allows the user to retry or request a new OTP, providing a seamless experience.

- **Conclusion:**
- The OTP Verification System provides a secure, user-friendly method for authenticating users via a randomly generated OTP sent to their email. With Flask powering the backend and a simple HTML frontend, the system is both lightweight and functional. By incorporating error handling and security measures, this project can serve as a robust authentication mechanism for web applications, ensuring that users' input is verified effectively.