OTP Verification Project Documentation

1. Introduction

The OTP Verification System is a Python-based project that demonstrates the process of generating a One-Time Password (OTP), sending it via email, and verifying it through a graphical user interface (GUI). This project uses the smtplib library for sending emails and customtkinter for creating an aesthetically pleasing GUI.

2. Requirements

- Python 3.x
- smtplib library (for sending emails)
- customtkinter library (for creating the GUI)
- tkinter library (standard GUI toolkit in Python)

3. Installation

To install the required libraries, use the following pip commands:

```sh
pip install customtkinter
```

tkinter comes pre-installed with Python, so there is no need to install it separately.

4. Project Structure

The project consists of a single Python script:

- otp_verification.py: This script contains the entire code for the OTP verification system, including OTP generation, sending emails, and the GUI implementation.

5. Code Explanation

OTP Generation

The OTP generation is handled by the generate_otp function. This function generates a 6-digit OTP using random numbers.

```python
import math
import random

def generate_otp():
    digits = "0123456789"
    otp = ""
    for i in range(6):
        otp += digits[math.floor(random.random() * 10)]
    return otp
```

Sending OTP via Email

The send_otp_email function takes the generated OTP, email, and password as inputs and sends the OTP to the specified email address using the smtplib library.

```python
import smtplib

def send_otp_email(otp, email, password):
    message = f"{otp} is your OTP" # Create the OTP message

    try:
        # Setup SMTP server connection
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()  # Start TLS for security
        server.login(email, password) # Login to the email server
        server.sendmail(email, email, message)  # Send email to the same address
        server.quit() # Quit the server connection
        print("OTP has been sent to your email.")
    except Exception as e:
        # Handle any exceptions that occur during email sending
        print(f"Failed to send email: {e}")
```

Verifying OTP

The verify_otp function checks if the entered OTP matches the generated OTP.

```python
def verify_otp(generated_otp, entered_otp):
    return generated_otp == entered_otp
```

GUI Implementation

The GUI is implemented using customtkinter, a custom version of tkinter that allows for more modern and customizable widgets.

```python
import customtkinter as ctk  # Import customtkinter for creating the GUI
from tkinter import messagebox  # Import messagebox for showing info and error messages

class OTPVerificationApp:
    def __init__(self, root):
        self.root = root
        self.root.title("OTP Verification System")
        self.root.geometry("700x700")
        ctk.set_appearance_mode("dark")  # Set appearance mode to dark
        ctk.set_default_color_theme("blue")  # Set color theme to blue

        self.otp = None
        self.email = None
```

```
        self.email_password = None

        self.create_widgets()  # Call the method to create widgets

    # Method to create the GUI widgets
    def create_widgets(self):
        self.frame = ctk.CTkFrame(master=self.root)
        self.frame.pack(pady=20, padx=60, fill="both", expand=True)

        self.title_label = ctk.CTkLabel(master=self.frame, text="OTP Verification System", font=("Arial",
25, 'bold'), text_color="white")
        self.title_label.pack(pady=12, padx=10)

                self.email_label = ctk.CTkLabel(master=self.frame, text="Email", text_color="white",
font=("Arial", 12, 'bold'))
        self.email_label.pack(pady=12, padx=10)

        self.email_entry = ctk.CTkEntry(master=self.frame, width=200)
        self.email_entry.pack(pady=12, padx=10)

                    self.password_label = ctk.CTkLabel(master=self.frame, text="Email Password",
text_color="white", font=("Arial", 12, 'bold'))
        self.password_label.pack(pady=12, padx=10)

        self.password_entry = ctk.CTkEntry(master=self.frame, show="*", width=200)
        self.password_entry.pack(pady=12, padx=10)
```

```
                self.send_otp_button = ctk.CTkButton(master=self.frame, text="Send OTP",
command=self.send_otp, fg_color="#00aaff")
        self.send_otp_button.pack(pady=20, padx=10)

            self.otp_label = ctk.CTkLabel(master=self.frame, text="Enter OTP", text_color="white",
font=("Arial", 12, 'bold'))
        self.otp_label.pack(pady=12, padx=10)

        self.otp_entry = ctk.CTkEntry(master=self.frame, width=200)
        self.otp_entry.pack(pady=12, padx=10)

                self.verify_otp_button = ctk.CTkButton(master=self.frame, text="Verify OTP",
command=self.verify_otp, fg_color="#00aaff")
        self.verify_otp_button.pack(pady=20, padx=10)

        self.result_label = ctk.CTkLabel(master=self.frame, text="", font=("Arial", 16, 'bold'))
        self.result_label.pack(pady=12, padx=10)

    # Method to handle sending OTP
    def send_otp(self):
        self.email = self.email_entry.get()  # Get email from entry
        self.email_password = self.password_entry.get()  # Get password from entry
        self.otp = generate_otp()  # Generate the OTP

        send_otp_email(self.otp, self.email, self.email_password)  # Send the OTP email
```

```
        messagebox.showinfo("Info", "OTP has been sent to your email.")  # Show info message


    # Method to handle verifying OTP
    def verify_otp(self):
        entered_otp = self.otp_entry.get()  # Get entered OTP from entry
        if verify_otp(self.otp, entered_otp):  # Check if entered OTP matches generated OTP
            self.result_label.configure(text="Verified", text_color="green", font=("Arial", 16))  # Show verified message
            messagebox.showinfo("Info", "Verified")  # Show info message
        else:
            self.result_label.configure(text="Invalid OTP", text_color="red")  # Show invalid OTP message
            messagebox.showerror("Error", "Invalid OTP. Please check your OTP again.")  # Show error message

# Main execution block
if __name__ == "__main__":
    root = ctk.CTk()  # Create the main window
    app = OTPVerificationApp(root)  # Create an instance of the OTPVerificationApp
    root.mainloop()  # Start the main event loop
```

6. Usage Instructions

1. Run the Script: Execute the script otp_verification.py using a Python interpreter.

```sh
python otp_verification.py
```

2. Enter Email and Password: Enter your email address and the email password in the respective fields.

3. Send OTP: Click the "Send OTP" button. An OTP will be sent to the specified email address.

4. Enter OTP: Check your email for the OTP, enter it in the "Enter OTP" field, and click the "Verify OTP" button.

5. Verification: The system will display whether the OTP is valid or invalid. A message box will also notify you of the verification status.

7. Conclusion

This OTP Verification System project demonstrates the implementation of a secure, user-friendly mechanism for verifying a user's identity using email-based OTPs. The project can be extended to include additional security measures and can be integrated into larger systems requiring user authentication.

This documentation provides an overview of the project's structure, explains the code, and guides users on how to use the application effectively.