**Project Description: OTP Verification System**

**Created by: Chirag wamanacharya**

**Overview**

The OTP Verification System is a Python-based desktop application that allows users to verify their identity using a one-time password (OTP) sent to their mobile phone via SMS. The application leverages Twilio for sending SMS and Tkinter for the graphical user interface (GUI).

**Technologies Used**

* **Python**: The core programming language used for developing the application.
* **Tkinter**: A standard Python library used for creating the GUI of the application.
* **Twilio**: A cloud communications platform used for sending SMS messages containing the OTP.
* **Random**: A Python module to generate random numbers, used for creating the OTP.
* **Messagebox**: A Tkinter module to display message boxes for user notifications and confirmations.

**Application Workflow**

**1. Initialization**

* **Creating the Window**: The application starts by initializing a Tkinter window with a title and specified dimensions.

**2. User Interface**

* **Country Code and Mobile Number Input**: The GUI contains entry fields for the user to input their country code and mobile number.
* **Send OTP Button**: A button that, when clicked, generates and sends an OTP to the provided mobile number.
* **OTP Entry Field**: An entry field for the user to input the received OTP.
* **Submit Button**: A button that verifies the entered OTP against the generated OTP.
* **Resend OTP Button**: A button that allows the user to resend the OTP to the same mobile number.

**3. OTP Generation and Sending**

* **OTP Generation**: Upon clicking the "Send OTP" button, a 4-digit OTP is generated using the random.randint function.
* **Send OTP via Twilio**: The generated OTP is sent to the user's mobile number using Twilio's Client.messages.create method.

**4. Verification and Validation**

* **OTP Verification**: When the user enters the OTP and clicks the "Submit" button, the entered OTP is checked against the generated OTP.
  + **Successful Verification**: If the OTP matches, a success message is displayed.
  + **Failed Verification**: If the OTP does not match, an error message is displayed.

**5. OTP Resending and Re-entering Mobile Number**

* **Resend OTP**: The "Resend OTP" button generates a new OTP and sends it to the same mobile number.
* **Re-enter Mobile Number**: If the user re-enters the mobile number, the application checks if the number matches the initially entered number. If it does, a confirmation popup appears asking if the user wants to regenerate the OTP for the same number.

**Key Functions and Their Descriptions**

* **send\_otp()**: Generates a new OTP and sends it to the user's mobile number, then disables the "Send OTP" button.
* **resend\_otp()**: Generates a new OTP and sends it to the user's mobile number again.
* **check\_otp()**: Verifies the entered OTP against the generated OTP and displays appropriate messages.
* **enable\_send\_otp(event)**: Re-enables the "Send OTP" button when the mobile number is re-entered.
* **check\_same\_mobile\_number()**: Checks if the re-entered mobile number matches the initial mobile number.
* **prompt\_regenerate\_otp()**: Displays a popup asking if the user wants to regenerate the OTP for the same number.
* **add\_plus\_prefix(event)**: Ensures that the country code entry field always starts with a "+".

**Detailed Workflow**

1. **Application Launch**:
   * The Tkinter window is initialized.
   * Entry fields for the country code and mobile number, and buttons for sending OTP, resending OTP, and submitting the OTP are created and placed on the window.
2. **Sending OTP**:
   * The user enters their country code and mobile number.
   * The user clicks the "Send OTP" button.
   * The application generates a random 4-digit OTP.
   * The OTP is sent to the provided mobile number via Twilio.
   * The "Send OTP" button is disabled to prevent multiple OTPs being sent accidentally.
3. **Verifying OTP**:
   * The user receives the OTP via SMS and enters it in the provided entry field.
   * The user clicks the "Submit" button.
   * The application checks if the entered OTP matches the generated OTP.
   * If the OTP is correct, a success message is displayed. Otherwise, an error message is shown.
4. **Resending OTP**:
   * If the user clicks the "Resend OTP" button, a new OTP is generated and sent to the same mobile number.
5. **Re-entering Mobile Number**:
   * If the user re-enters the mobile number, the application checks if it matches the initial mobile number.
   * If the numbers match, a popup asks the user if they want to regenerate the OTP for the same number.
   * Depending on the user's response, the OTP is either regenerated or kept as is.

**Security Considerations**

* **Sensitive Information**: Ensure that sensitive information like account\_sid and auth\_token are securely stored and not hardcoded in the source code.
* **Validation**: Proper validation and error handling should be implemented for user inputs and API responses.
* **Rate Limiting**: Implement rate limiting to prevent abuse of the OTP sending functionality.

**Conclusion**

This OTP Verification System provides a robust and user-friendly interface for verifying user identity via SMS. Leveraging Twilio's reliable SMS service and Tkinter's flexible GUI capabilities, the application ensures secure and efficient OTP generation, sending, and verification processes.