Création d’un chatbot avec Streamlit

### **Objectif :**

Créer une interface de chatbot simple en utilisant Streamlit. L'utilisateur pourra taper des questions et recevoir des réponses automatiques générées par modèle de language.

### **Pré-requis :**

* Python installé sur votre ordinateur.
* Un éditeur de code (VS Code, PyCharm, sublimtextetc.).
* La bibliothèque Streamlit installée.

### **Étape 1 : Installation de Streamlit**

Si vous n'avez pas déjà Streamlit installé, vous pouvez l'installer via pip. Ouvrez un terminal et tapez la commande suivante :

| pip install streamlit |
| --- |

### **Étape 2 : Créer un nouveau fichier Python**

Créez un fichier Python pour votre projet, par exemple chatbot\_app.py, dans votre répertoire de travail.

### **Étape 3 : Importer les bibliothèques nécessaires**

Dans ce fichier Python, commencez par importer Streamlit :

| import streamlit as st |
| --- |

### **Étape 4 : Créer une fonction pour générer des réponses**

Créez une fonction generate\_response qui va définir la logique de réponse du chatbot. Dans ce tutoriel, nous allons utiliser une logique basique (vous pouvez remplacer cette partie par une API ou un modèle plus avancé plus tard).

Ajoutez ce code à votre fichier Python :

| def generate\_response(user\_input):  # Logique de réponse basique  if "bonjour" in user\_input.lower():  return "Bonjour! Comment puis-je vous aider aujourd'hui?"  elif "comment ça va" in user\_input.lower():  return "Je vais bien, merci! Et vous?"  else:  return "Je suis un chatbot et je n'ai pas encore appris cette réponse." |
| --- |

### **Étape 5 : Créer l'interface de l'application avec Streamlit**

Maintenant, vous allez configurer l'interface graphique avec Streamlit. La première chose à faire est d'ajouter un titre à l'application :

| st.title("Chatbot avec Streamlit") st.write("Bienvenue sur l'interface de chatbot. Posez-moi des questions !") |
| --- |

### **Étape 6 : Stocker l'historique des conversations**

Pour sauvegarder l'historique des conversations, vous allez utiliser st.session\_state. Cela permet de conserver les messages même après avoir soumis de nouvelles réponses.

Ajoutez ce code juste après l'initialisation de l'interface :

| if 'chat\_history' not in st.session\_state:  st.session\_state.chat\_history = [] |
| --- |

### **Étape 7 : Créer un formulaire pour saisir la question**

Pour permettre à l'utilisateur de poser une question, nous allons ajouter un formulaire avec un champ de saisie :

| with st.form(key='chat\_form', clear\_on\_submit=True):  user\_input = st.text\_input("Vous :", key="input")  submit\_button = st.form\_submit\_button(label='Envoyer') |
| --- |

### **Étape 8 : Afficher la réponse du bot**

Quand l'utilisateur soumet une question, vous devez générer la réponse avec la fonction generate\_response et l'ajouter à l'historique de la conversation. Ajoutez ce code juste après la création du formulaire :

| if submit\_button and user\_input:  response = generate\_response(user\_input)  # Ajouter l'entrée utilisateur et la réponse à l'historique  st.session\_state.chat\_history.append(("Vous", user\_input))  st.session\_state.chat\_history.append(("Bot", response)) |
| --- |

### **Étape 9 : Afficher l'historique des messages**

Pour afficher l'historique des échanges entre l'utilisateur et le bot, vous pouvez utiliser une boucle for. Voici comment le faire :

| # Affichage de l'historique des échanges for sender, message in st.session\_state.chat\_history:  if sender == "Vous":  st.write(f"\*\*{sender}:\*\* {message}")  else:  st.write(f"\*{sender}:\* {message}") |
| --- |

### **Étape 10 : Tester l'application**

Pour lancer l'application Streamlit, vous devez ouvrir un terminal dans le répertoire où se trouve votre fichier Python, puis exécuter la commande suivante :

| streamlit run chatbot\_app.py |
| --- |

Streamlit va générer un lien local, que vous pouvez ouvrir dans votre navigateur pour interagir avec votre chatbot.

### **Étape 11 : Personnaliser le chatbot**

Vous pouvez maintenant améliorer la logique du chatbot en ajoutant des réponses plus complexes, en utilisant une API d'IA pour générer des réponses ou en intégrant d'autres fonctionnalités.

### **Etape 12 : Déploiement de l’application gratuitement sur Streamlit Cloud**

Voici les étapes à suivre :

1. **Créer un fichier requirements.txt** : Ce fichier doit contenir les noms des bibliothèques et leurs versions (si besoin). Par exemple :  
     
   streamlit

pandas

matplotlib

scikit-learn

1. **Uploader le projet sur GitHub** : Streamlit Cloud déploie les apps à partir d'un dépôt GitHub. Créer un dépôt pour ton projet et uploader le code source ainsi que le fichier requirements.txt.
2. **Déployer sur Streamlit Cloud** :
   * Connectez-vous à Streamlit Cloud (https://streamlit.io/cloud).
   * Cliquez sur "New app" et sélectionnez le dépôt GitHub contenant votre projet.
   * Streamlit Cloud va automatiquement détecter le fichier requirements.txt et installer les bibliothèques nécessaires lors du déploiement.

Assurez-vous que toutes les bibliothèques mentionnées dans requirements.txt sont compatibles avec les environnements d'exécution de Streamlit Cloud. Si tu utilises des bibliothèques spécifiques ou des dépendances natives, cela peut nécessiter des ajustements.