

# Rapport de Projet Chatbot

**Nom:** Kais Barhoumi

**Date de Présentation:** 21 Octobre 2024

**président:** Mr. Nafaa Haffar

# Contents

0.1	Introduction . . . . .	3
0.2	Technologies Utilisées . . . . .	4
0.3	Structure du Projet . . . . .	5
0.3.1	Backend - Flask (app.py) . . . . .	5
0.3.2	Frontend - ReactJS (app.js) . . . . .	5
0.4	Fonctionnalités du Chatbot . . . . .	6
0.5	Résultats et Tests . . . . .	7
0.6	Conclusion . . . . .	8
0.7	Références . . . . .	9

# List of Figures

1	Interface du Chatbot . . . . .	3
2	fichier app.py . . . . .	5
3	fichier app.js . . . . .	6
4	Tests . . . . .	7

## 0.1 Introduction

Le développement des chatbots a évolué avec les progrès en intelligence artificielle et en traitement du langage naturel. Ce projet présente un chatbot développé en utilisant Flask pour le backend Python, React pour le frontend, et des techniques de similarité pour répondre aux utilisateurs. Le chatbot est capable de répondre à des questions prédéfinies et, en cas d'absence de correspondance, utilise l'API ChatGPT pour générer des réponses.

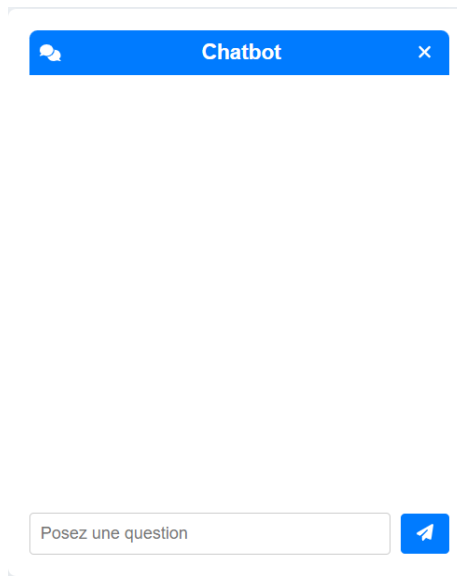


Figure 1: Interface du Chatbot

## 0.2 Technologies Utilisées

- **Flask:** Un micro-framework en Python utilisé pour gérer les requêtes HTTP et les réponses du chatbot.



- **ReactJS:** Une bibliothèque JavaScript utilisée pour construire l'interface utilisateur du chatbot.



- **Scikit-learn:** Utilisé pour la vectorisation des questions et la similarité cosinus.



- **TfidfVectorizer:** Pour transformer le texte en vecteurs TF-IDF.
- **Cosine Similarity:** Pour comparer les vecteurs de questions afin de trouver la meilleure correspondance.

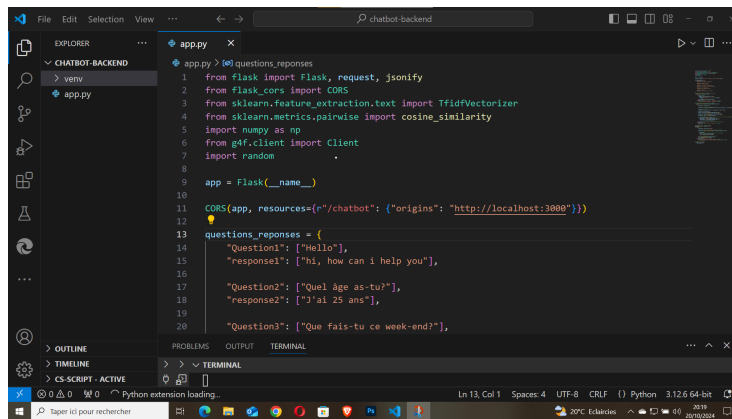
## 0.3 Structure du Projet

Le projet est composé de deux parties principales : le frontend (ReactJS) et le backend (Flask). Voici un aperçu du fichier `app.py` qui gère la logique de traitement des messages et des réponses.

### 0.3.1 Backend - Flask (`app.py`)

Le backend utilise Flask pour recevoir les messages des utilisateurs, analyser ces messages, et répondre avec des réponses préenregistrées ou en utilisant l'API ChatGPT.

Voici un extrait du code `app.py` qui montre comment le chatbot analyse les messages et répond :



```
1 from flask import Flask, request, jsonify
2 from flask_cors import CORS
3 from sklearn.feature_extraction.text import TfidfVectorizer
4 from sklearn.metrics.pairwise import cosine_similarity
5 import numpy as np
6 from g4f.client import Client
7 import random
8
9
10 app = Flask(__name__)
11
12 CORS(app, resources={r"/chatbot": {"origins": "http://localhost:3000"}})
13
14 questions_responses = [
15     {"Question1": ["Hello"],
16      "response1": ["Hi, how can i help you"]},
17     {"Question2": ["Quel âge as-tu?"],
18      "response2": ["J'ai 25 ans"]},
19     {"Question3": ["Que fais-tu ce week-end?"]},
20 ]
```

Figure 2: fichier `app.py`

### 0.3.2 Frontend - ReactJS (`app.js`)

Le frontend permet aux utilisateurs d'interagir avec le chatbot via une interface conviviale. Voici un extrait de `app.js`, qui montre comment les messages de l'utilisateur sont envoyés au serveur et comment les réponses sont affichées.

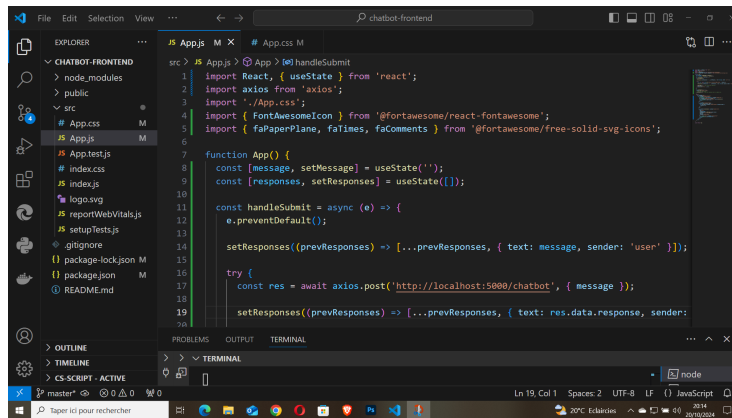


Figure 3: fichier app.js

## 0.4 Fonctionnalités du Chatbot

Le chatbot dispose des fonctionnalités suivantes :

- **Réponses prédéfinies:** Basé sur des questions préenregistrées, il renvoie des réponses appropriées.
- **API ChatGPT:** Lorsqu'une correspondance n'est pas trouvée, l'API ChatGPT est utilisée pour générer des réponses dynamiques.
- **Gestion de la Similarité:** Utilisation de `TfidfVectorizer` et de la similarité cosinus pour trouver les questions les plus similaires.
- **Interface Utilisateur Interactive:** Interface simple et réactive, développée avec ReactJS, qui permet une communication en temps réel.

## 0.5 Résultats et Tests

Le chatbot a été testé pour gérer différentes entrées utilisateur et fournir des réponses en fonction des questions prédéfinies. Lorsqu'aucune correspondance n'est trouvée, il utilise efficacement l'API ChatGPT.

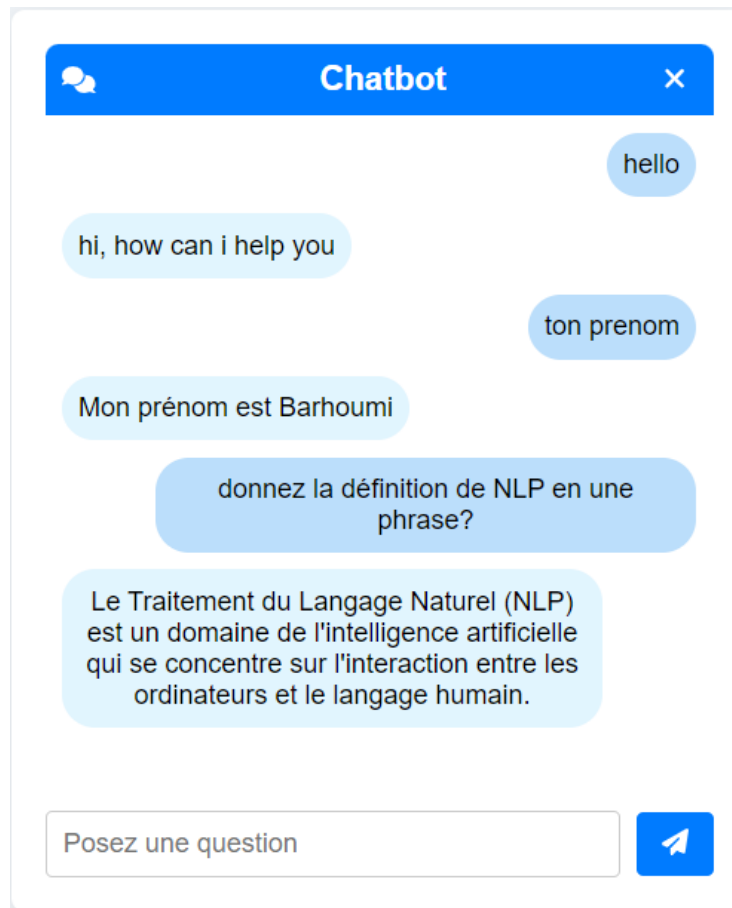


Figure 4: Tests



## **0.6 Conclusion**

Ce projet montre comment un chatbot peut être construit en combinant des techniques de traitement du langage naturel avec une interface conviviale. Il illustre aussi l'intégration de modèles d'IA pour améliorer les interactions utilisateur lorsque les données statiques ne sont pas suffisantes.

## 0.7 Références

- Flask Documentation: <https://flask.palletsprojects.com/>
- ReactJS Documentation: <https://reactjs.org/>
- Scikit-learn Documentation: <https://scikit-learn.org/stable/>
- OpenAI GPT-3 API: <https://beta.openai.com/docs/>