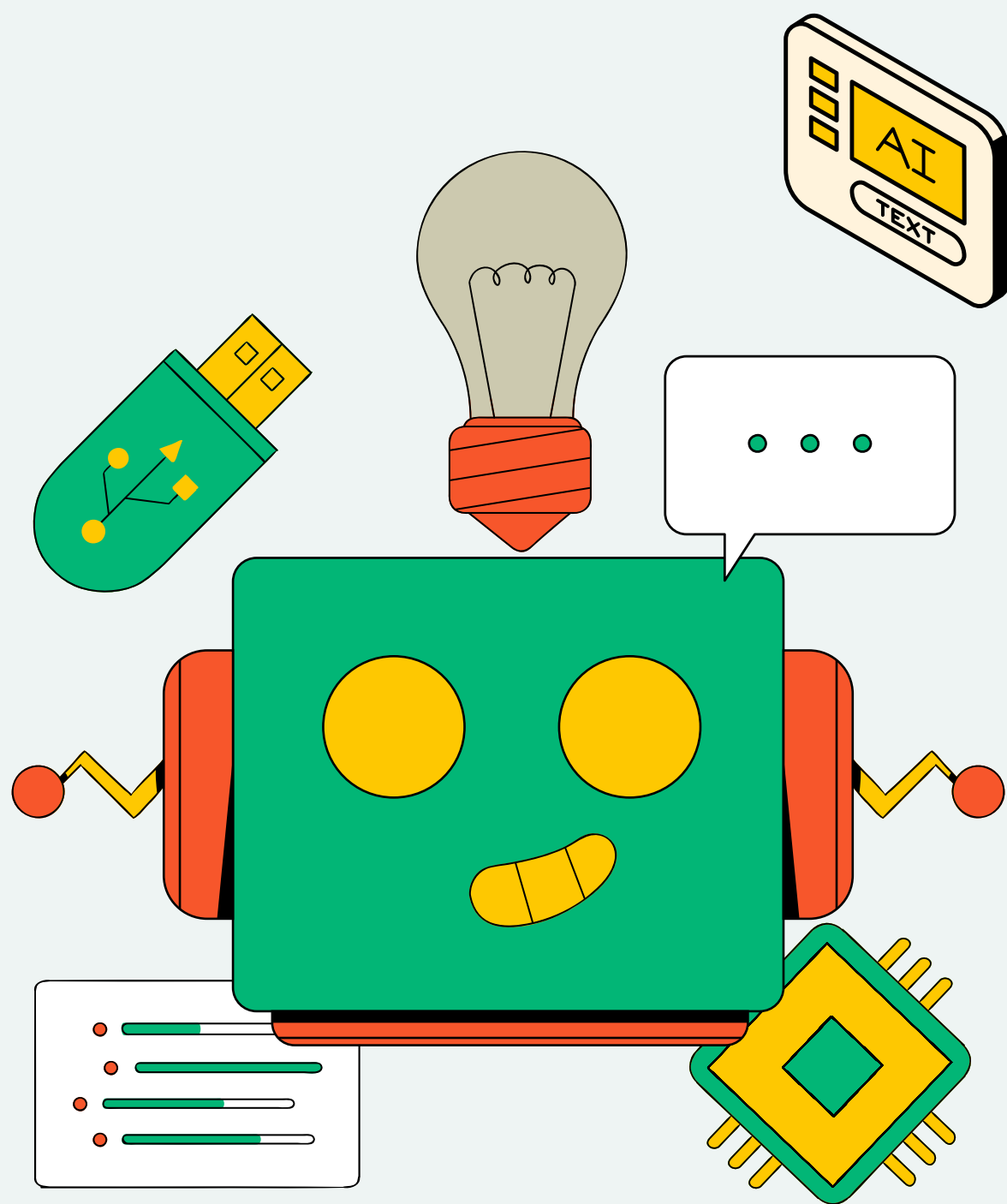


NOUS APPRENONS POUR L'AVENIR

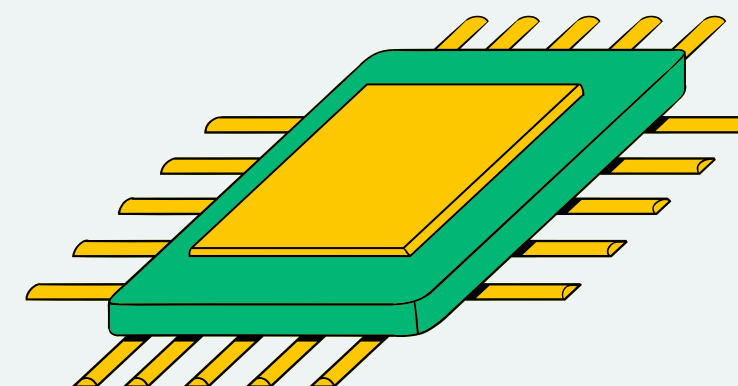


CHATBOT

PRESENTATION

PRÉSENTÉ PAR :

KAIS BARHOUMI



PLAN

- Introduction
- Technologies Utilisées
- Fonctionnalités
- Architecture du projet
- Tests et résultats
- Conclusion



INTRODUCTION

Les chatbots sont de plus en plus utilisés pour automatiser les interactions entre les utilisateurs et les systèmes informatiques, que ce soit pour le service client, la gestion des tâches, ou la fourniture d'informations.



L'objectif principal de ce projet est de créer un chatbot qui peut répondre automatiquement aux questions des utilisateurs en utilisant des méthodes de traitement du langage naturel et une API d'intelligence artificielle.



TECHNOLOGIES UTILISÉES

backend : flask

Flask est un micro-framework en Python qui facilite la création d'applications web en fournissant des outils pour gérer les requêtes HTTP, la logique de traitement des données et la génération des réponses. Dans le contexte du chatbot, Flask reçoit les messages des utilisateurs, traite ces informations et renvoie les réponses appropriées, que ce soit par le biais de réponses prédéfinies ou via l'API ChatGPT



TECHNOLOGIES UTILISÉES

Frontend : ReactJS

ReactJS est une bibliothèque JavaScript qui permet de construire des interfaces utilisateur interactives et réactives. Dans notre projet, ReactJS est utilisé pour développer la partie frontale du chatbot, où les utilisateurs peuvent envoyer des messages et recevoir des réponses en temps réel



TECHNOLOGIES UTILISÉES

NLP : TfidfVectorizer et Similarité Cosinus

Les techniques de traitement du langage naturel (NLP) telles que TfidfVectorizer et la similarité cosinus sont essentielles pour le fonctionnement du chatbot.

TfidfVectorizer transforme les questions en vecteurs numériques en tenant compte de la fréquence des mots, tandis que la similarité cosinus mesure la similarité entre ces vecteurs pour déterminer quelle question préenregistrée correspond le mieux à celle posée par l'utilisateur



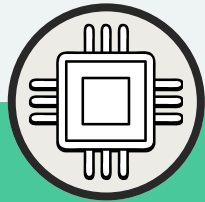
TECHNOLOGIES UTILISÉES

API ChatGPT

L'API ChatGPT est intégrée au chatbot pour générer des réponses dynamiques lorsque les questions des utilisateurs ne correspondent pas aux réponses prédéfinies

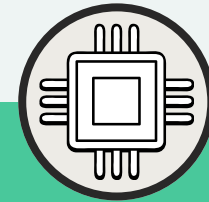


FONCTIONNALITÉS PRINCIPALES



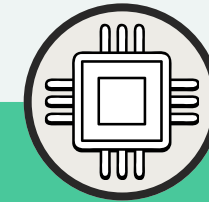
RÉPONSES PRÉDÉFINIES

Mon chatbot gère les questions avec des réponses stockées dans une base de données



GÉNÉRATION DYNAMIQUE DE RÉPONSES

l'intégration de l'API ChatGPT pour répondre aux questions nouvelles ou non répertoriées

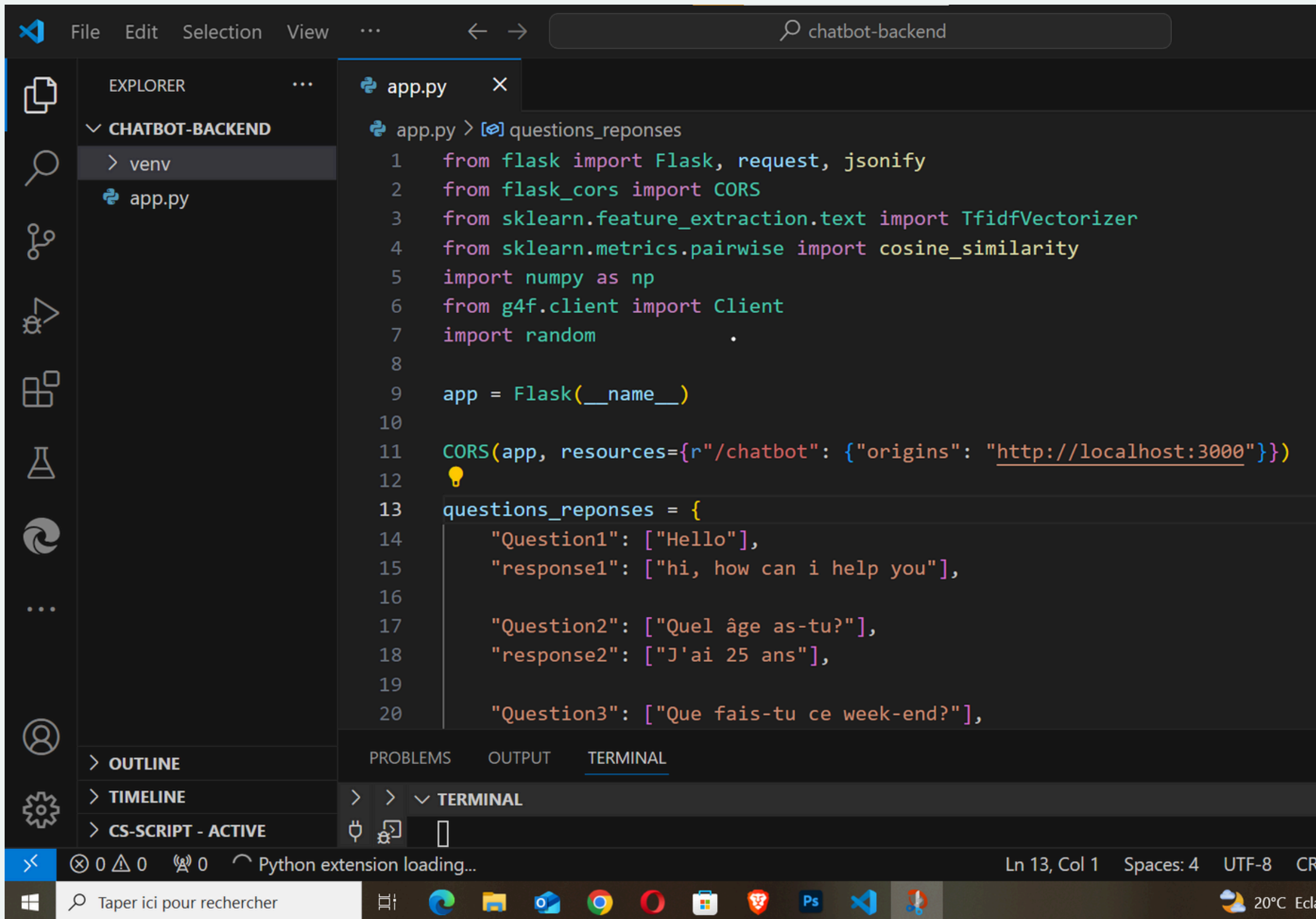


GESTION DES SIMILARITÉS

La gestion des similarités utilise la similarité cosinus pour évaluer et quantifier la similitude entre les vecteurs représentant les questions des utilisateurs et celles préenregistrées, permettant ainsi de trouver la question la plus pertinente dans la base de données du chatbot.



ARCHITECTURE DU PROJET



```
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 app = Flask(__name__)
10
11 CORS(app, resources={r"/chatbot": {"origins": "http://localhost:3000"}})
12
13 questions_reponses = {
14     "Question1": ["Hello"],
15     "response1": ["hi, how can i help you"],
16
17     "Question2": ["Quel âge as-tu?"],
18     "response2": ["J'ai 25 ans"],
19
20     "Question3": ["Que fais-tu ce week-end?"]
```

Ce code de backend utilise Flask pour créer une API qui reçoit des messages d'utilisateur, cherche des correspondances avec des réponses prédéfinies à l'aide de la similarité cosinus, et renvoie une réponse appropriée. Si aucune correspondance n'est trouvée, il interroge l'API ChatGPT pour générer une réponse dynamique.

ARCHITECTURE DU PROJET

```
src > JS App.js > App > handleSubmit
1  import React, { useState } from 'react';
2  import axios from 'axios';
3  import './App.css';
4  import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
5  import { faPaperPlane, faTimes, faComments } from '@fortawesome/free-solid-svg-icons';
6
7  function App() {
8    const [message, setMessage] = useState('');
9    const [responses, setResponses] = useState([]);
10
11    const handleSubmit = async (e) => {
12      e.preventDefault();
13
14      setResponses((prevResponses) => [...prevResponses, { text: message, sender: 'user' }]);
15
16      try {
17        const res = await axios.post('http://localhost:5000/chatbot', { message });
18
19        setResponses((prevResponses) => [...prevResponses, { text: res.data.response, sender: 'bot' }]);
20      } catch (error) {
21        console.log(error);
22      }
23    }
24
25    return (
26      <div>
27        <h1>Chatbot</h1>
28        <div>
29          <input type='text' value={message} onChange={e => setMessage(e.target.value)} />
30          <button type='button' onClick={handleSubmit}>Envoyer</button>
31        </div>
32        <div>
33          {responses.map((res, index) => (
34            <div key={index}>
35              {res.sender === 'user' ? 'Utilisateur' : 'Chatbot'} :> {res.text}</div>
36          )}</div>
37      </div>
38    );
39  }
40  export default App;
```

Ce code frontend en React crée une interface utilisateur pour le chatbot, permettant aux utilisateurs d'envoyer des messages et d'afficher les réponses. Lorsqu'un message est soumis, il est envoyé à l'API du backend via Axios, et la réponse du chatbot est affichée dans l'interface, tandis que l'historique des messages peut être réinitialisé avec un bouton de fermeture.

CONCLUSION

Ce projet met en évidence la possibilité de créer un chatbot en associant des méthodes de traitement du langage naturel à une interface conviviale. Il met également en évidence la mise en place de modèles d'IA afin d'améliorer les interactions utilisateurs lorsque les données statiques ne sont pas adéquates.

