# Documentation du Projet - Accès Météo par Commande Vocale

## Contexte du Projet

Votre employeur souhaite développer une fonctionnalité d’accès à la météo par commande vocale pour permettre des usages plus accessibles et plus naturels de l’application. La reconnaissance vocale doit pouvoir identifier les intentions de l'utilisateur : le lieu et l'horizon de prévision.

## Objectifs du Projet

Le projet s'inscrit dans un cadre global consistant à exploiter des services d’IA externes pour le développement d’applications d’IA, en intégrant des services cloud et en garantissant la sécurité et la qualité des solutions mises en œuvre.

## Compétences Visées

- Planifier le travail à effectuer pour atteindre l’objectif visé.  
- Contribuer au pilotage de l’organisation du travail individuel et collectif.  
- Définir le périmètre d’un problème rencontré pour rechercher une solution.  
- Préconiser un service d’IA adapté au besoin.  
- Intégrer une API d’IA dans une application existante.  
- Appliquer les bonnes pratiques de sécurité et d’accessibilité.  
- Développer des tests d’intégration pour l’API exploitée.  
- Rédiger une documentation technique complète.

## Modalités Pédagogiques

Contraintes :  
- Utilisation obligatoire d’Azure pour la reconnaissance vocale et le stockage des données.  
- Travail individuel avec possibilité d’échanges et collaboration.  
- Durée estimée : 2 semaines.

## Livrables

- Une application web fonctionnelle et conforme aux attentes (Streamlit, Flask, FastAPI, etc.).  
- Une documentation technique détaillée avec instructions d'installation et d'utilisation.  
- Un dépôt GitHub avec l’ensemble des scripts et configurations.  
- Une présentation avec :  
 \* Schéma fonctionnel de l’application et des services intégrés.  
 \* Identification et explication des services d’IA utilisés.  
 \* Liste des spécifications fonctionnelles de l’application.

## Critères de Performance

L'application finale doit :  
- Correspondre aux objectifs énoncés.  
- Intégrer tous les services nécessaires à son bon fonctionnement.  
  
Bonus :  
- Intégration d’une procédure en cas de résultats insuffisants.  
- Mise en place du Human Feedback Loop pour améliorer la reconnaissance vocale.  
- Affichage de données analytiques et graphiques.  
- Interface utilisateur ergonomique et responsive.  
- Intégration d’une carte géographique pour la localisation.  
- Sécurité conforme au Top 10 OWASP.  
- Application dockerisée et déployée.

## Plan Technique et Implémentation

1. Conception de l’architecture avec FastAPI pour le backend, Azure pour la reconnaissance vocale et la base de données, et Streamlit pour le frontend.  
2. Intégration des API d’Azure pour la reconnaissance vocale et le stockage des prédictions.  
3. Développement du backend avec des endpoints pour gérer les requêtes vocales et stocker les résultats.  
4. Développement du frontend avec une interface interactive permettant d'envoyer des requêtes et d’afficher les résultats.  
5. Mise en place de la base de données sur Azure pour stocker les requêtes et les réponses.  
6. Tests et monitoring pour assurer la fiabilité des prédictions.  
7. Sécurisation de l’application selon les meilleures pratiques OWASP.  
8. Dockerisation et déploiement sur Azure.

## Feuille de Route

|  |  |  |  |
| --- | --- | --- | --- |
| Tâche | Technologies | Semaine 1 | Semaine 2 |
| Développement API FastAPI | FastAPI, Uvicorn | [OK] |  |
| Connexion Azure Speech-to-Text | Azure Cognitive Services | [OK] |  |
| Base de données Azure | Azure SQL, SQLAlchemy | [OK] |  |
| Interface Web | Streamlit / React | [OK] |  |
| Sécurité OWASP | JWT, validation input |  | [OK] |
| Fallback en cas d’erreur | Gestion des seuils |  | [OK] |
| Données analytiques & graphiques | Plotly / Matplotlib |  | [OK] |
| Intégration carte géographique | Leaflet.js / Google Maps |  | [OK] |
| Dockerisation | Docker, Docker Compose |  | [OK] |
| Déploiement sur Azure | Azure App Services / AKS |  | [OK] |