Documentation Complete : Projet Triage Medical Intelligent (A a Z)

Ce document offre une vision exhaustive de l'architecture, des technologies et des modeles utilises dans ce projet de triage medical base sur le Traitement du Langage Naturel (NLP) et l'IA.

---

1. Vue d'Ensemble Technologique (La "Stack")

Le projet est divise en modules specialises, chacun utilisant des bibliotheques de pointe :

Tache NLP	Bibliotheque / Modele	Role
Interface & Visualisation	**Streamlit**	Interface web interactive et dashboard de sante IA.
Comprehension (NLU)	**SpaCy** (`en_core_web_sm`)	Lemmatisation, Tokenisation et analyse grammaticale.
Intelligence Artificielle	**Scikit-Learn**	Modele `RandomForest` pour la classification specialiste/urgence.
Vecteurs (Embeddings)	**TF-IDF Vectorizer**	Transformation du texte en vecteurs numeriques bases sur la frequence.
Traduction	**Deep-Translator**	Traducteur multi-moteurs (Google Translate par defaut).
Correction	**Pyspellchecker**	Correction orthographique basee sur des dictionnaires orfficiels.
Logique de Correction	**Bigrams / Context-Aware**	Correction intelligente basee sur l'ordre des mots (N-Grams).

---

2. Role des Fichiers (Structure du Projet)

Racine du Projet
* **`streamlit_app.py`** : **Le C ur de l'Interface**. Orchestre la saisie patient, appelle l'analyseur, et affiche les resultats (y compris le "Cerveau de l'IA" et les alertes de securite).
* **`medical_triage_system.py`** : Point d'entree pour la version Terminal/Console du systeme.
* **`evaluate_system.py`** : Script de test de performance qui calcule la precision du modele sur l'ensemble du dataset.
* **`requirements.txt`** : Liste de toutes les bibliotheques Python necessaires au projet.

`agents/` (La Core Logic)
L'intelligence est divisee en "Agents" specialises :

1. `agents/analyzer/` (Comprehension & Prediction)
* **`nlp_analyzer_v3.py`** : **Le Chef d'Orchestre NLP**. Gere le pipeline : Detection langue -> Correction -> Traduction -> Lemmatisation -> Extraction de symptomes.
* **`ml_classifier.py`** : **Le Modele Predictif**. Contient la classe `MedicalMLClassifier` qui entraine et utilise le modele `RandomForest` pour predire le specialiste et l'urgence.
* **`intelligent_medical_nlu.py`** : Module avance pour la reconnaissance d'entites medicales complexes.

2. `agents/nlp/` (Traitement du Langage)
* **`context_spell_corrector.py`** : **L'Expert en Correction**. Utilise une approche hybride (Dictionnaire + Contexte medical) pour corriger les fautes (ex: "havee" -> "have").
* **`multilingual_processor.py`** : Gere les specificites linguistiques pour le Francais, l'Anglais et l'Arabe.

3. `agents/reasoner/` (Aide a la Decision)
* **`medical_reasoner.py`** : **Le Cerveau Expert**. Combine les predictions de l'IA avec des **regles medicales de securite**. C'est lui qui outrepasse l'IA si un symptome vital (ex: douleur cardiaque) est detecte.

4. `agents/decider/` (Generation des Sorties)
*   **`decision_generator.py`** : Genere les recommandations finales (Delai d'attente, numeros d'urgence selon le pays).

5. `agents/data_loader/` (Gestion des Donnees)
*   **`medical_data_loader.py`** : Charge et indexe le dataset JSON pour une recherche ultra-rapide des symptomes.

---

3. Donnees & Modeles par Fonction

Modele pour la Comprehension (NLU)
*   **Bibliotheque** : `SpaCy`.
*   **Logic** : Utilise la **Lemmatisation** pour transformer "teeth", "tooth", "dent" en un seul concept racine.
*   **Data** : S'appuie sur un index de 143 symptomes uniques appris depuis le dataset.

Modele pour la Prediction (AI/ML)
*   **Algorithme** : `Random Forest Classifier`.
*   **Pourquoi ?** Robuste, gere bien les donnees textuelles apres vectorisation, et peu sensible au sur-apprentissage sur les petits datasets.
*   **Data** : Entraine sur **4 944 cas cliniques** reels.

Modele pour la Correction (Spell Check)
*   **Algorithme** : `Levenshtein Distance + Bigrams`.
*   **Process** :
    1. Genere des candidats proches.
    2. Utilise les **Bigrams** (mots cote-a-cote) pour choisir le plus probable (ex: "my heart" au lieu de "my hear").
*   **Multilingue** : Gere FR et EN simultanement.

Modele pour la Traduction
*   **Moteur** : `Google Translate API` (via `deep-translator`).
*   **Fallback** : Un dictionnaire manuel de 100+ termes medicaux critiques pour fonctionner meme sans connexion stable.

---

4. Qu'avons-nous fait exactement ? (Resume des etapes)

1. **Uniformisation Multilingue** : Le systeme detecte la langue du patient et convertit tout en un "format neutre" (Anglais Lemmatise) pour une analyse constante.
2. **Correction Contextuelle** : Creation d'un correcteur qui comprend que "ceour" en francais doit etre "coeur" avant meme la traduction.
3. **IA Hybride** : Passage d'un systeme a 100% de regles a un systeme **AI-Driven** (Random Forest) securise par des **Safety Rules** (Regles metiers).
4. **UI Professionnelle** : Mise en place d'un tableau de bord Streamlit qui explique en temps reel **comment** l'IA a pris sa decision (AI vs Protocol).
5. **Dictionnaire Medical tendu** : Creation d'une base de connaissances de 100+ organes et symptomes traduits manuellement pour une precision maximale.

Le resultat final est un systeme industriel capable de trier des patients en moins d'une seconde avec une securite medicale garantie.