**Cahier des charges – 2ImagesPlayer**

**Titre du projet**

Media Player sécurisé avec format propriétaire

**Objectif**

Développer une application Windows permettant :

La lecture de fichiers média (vidéo/audio) protégés par chiffrement AES.

L’utilisation d’une **clé de déchiffrement** propre à l’entreprise, afin que les médias ne puissent être lus que via cette application.

**Contexte**

L’entreprise souhaite distribuer des contenus médias internes ou propriétaires sans risque de copie ou diffusion externe non autorisée. L’application sera installée sur les postes autorisés.

**Fonctions attendues**

|  |  |
| --- | --- |
| **Fonction** | **Description** |
| |  | | --- | | **Sélectionner un fichier média** |  |  | | --- | |  | | L’utilisateur choisit un fichier chiffré (extension .myvid ou autre) via un explorateur de fichiers intégré. |
| **Déchiffrement du fichier** | |  | | --- | | L’application utilise une clé AES interne pour déchiffrer le fichier en mémoire. |  |  | | --- | |  | |
| **Lecture du média** | |  | | --- | | Le média est lu via le lecteur intégré (LibVLCSharp) après déchiffrement. |  |  | | --- | |  | |
| **Gestion des erreurs** | |  | | --- | | Si le fichier est corrompu ou la clé invalide, un message d’erreur s’affiche. |  |  | | --- | |  | |
| |  | | --- | | **Interface utilisateur minimale** |  |  | | --- | |  | | Affiche un bouton pour ouvrir un fichier, un lecteur, et des boutons Play/Pause/Stop. |

**Contraintes Techniques**

**Langage :** Python, HTML, CSS

**Framework :** Flask

**Librairies Python :** Flask pour l’interface web

* PyCryptodome pour le chiffrement AES 256 bits
* Werkzeug pour sécuriser les fichiers importés

**OS cible :** Windows 10/11

**Clé de chiffrement :** Stockée en interne dans l’application ou via un fichier de configuration sécurisé.

**Sécurité**

**Chiffrement :** AES 256 bits  
**Clé :** Unique pour l’entreprise.  
**Protection du fichier :**

Sans la clé, le fichier reste inutilisable.

Même renommé en .mp4/.mp3, le média est illisible.

**Stockage de la clé**

Intégrée dans le code compilé (limite : risque si ingénierie inverse).

**Option avancée :** Entrée de la clé par l’utilisateur ou stockage chiffré.

**Utilisateurs cibles**

Membres de l’entreprise (Images2) disposant des droits pour lire les médias protégés.

**Livrables**

1. **Application Web Flask (lecture locale sur navigateur)**
2. **Script de chiffrement des fichiers médias**
3. **Documentation utilisateur**
4. **Documentation technique**
5. Algorithme utilisé
6. Gestion de la clé
7. Fonctionnement du flux de déchiffrement → lecteur

**Limites et risques**

**Performance** : le chiffrement et déchiffrement peuvent ralentir selon la taille des fichiers

**Sécurité de la clé** : si le fichier key.txt est volé, les données sont exposées. Un mot de passe maître est recommandé

**Compatibilité navigateur** : tous les navigateurs ne lisent pas tous les formats (.mkv par exemple)

**Absence de streaming** : les fichiers sont chargés localement, pas lus en flux (stream)

**Planning estimatif**

|  |  |
| --- | --- |
| **Tâche** | **Durée estimée** |
| |  | | --- | | **Rédaction du cahier des charges** |  |  | | --- | |  | | Demi-journée |
| |  | | --- | | **Implémentation du script de chiffrement** |  |  | | --- | |  | | 1 Jours |
| |  | | --- | | **Développement de l’application avec lecture standard** |  |  | | --- | |  | | 1 Jours |

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **Intégration du déchiffrement** |  |  | | --- | |  | | 1 Jours |
| |  | | --- | | **Tests et corrections** |  |  | | --- | |  | | Demi-journée |
| |  | | --- | | **Rédaction de la documentation** |  |  | | --- | |  | | Demi-journée |