**Projet : Loup-Garou – Spécifications techniques**

**Contexte :**

Ce projet implémente un jeu de **Loup-Garou** multijoueur, où les joueurs interagissent dans un environnement réseau avec interface graphique. Les joueurs peuvent se connecter à un serveur, rejoindre une partie, et jouer une partie.

1. **Fichier Interface.py :**

* **Le module permettant de générer une interface pour que le joueur puisse interagir, il y en a une de généré sur chaque client.**
* **Modules à installer :** tkinter, pillow, time, threading
* **Classe : MainInterface**
* **Classe pouvant gérer le chat, la liste des joueurs, l'affichage du rôle, les actions spécifiques aux rôles et le chronomètre**
* **Initialisation :** prend en paramètre la liste des joueurs (liste de str), le rôle du joueur (en str) et le client. Une fois la classe initialisé, elle peut être lancé avec la méthode **mainloop()**
* **startUpdate(self, usernames: list, role) :** prend en paramètre la liste des joueurs (liste de str) et le rôle du joueur (en str). Cette méthode permet de mettre à jour l’interface.
* **clickThread(self) :** Méthode pour lancer le décompte du temps restant à la petite fille
* **pfClick(self, \_), pfRelease(self, \_) :** Methodes pour active/desactiver clickThread
* **pfTurn(self), pfEnd(self) :** Méthodes pour afficher/cacher le bouton de jeu de la Petite Fille
* **changeImage(self, Role) :** Méthode pour changer l’image actuelle par celle du rôle rentré en paramètre (en str)
* **sendMessage(self) :** Méthode pour envoyer le message rentré par l’utilisateur dans le chat au serveur
* **chat(self, joueur, message) :** Méthode permettant d’ajouter au chat le message du joueur rentré en paramètre ( en str)
* **chronometre(self , temps, condition=none) :** Méthode pour lancer un chronomètre de temps (en int) secondes, avec une condition d’arrêt qui s’il est remplis stop en avance le chronomètre.
* **updateList(self, playersAlive) :** Méthode pour mettre à jour sur l’interface la liste des joueurs en vie grâce à la liste des pseudos rentré en paramètre.
* **updateRoleAction(self, affectedPlayers) :** Méthodes pour afficher la liste des actions du joueur. La liste affectedPlayers rentré en paramètre est une liste des pseudos sur lesquels le joueur peut agir.
* **action(self, affectedPlayers) :** Méthode pour demander au joueur une action sur l’interface. La liste affectedPlayers rentré en paramètre est une liste des pseudos sur lesquels le joueur peut agir.

1. **Fichier xxx.py :**