**READ ME**

Le projet consiste en la création d’une application mobile de yoga interactive, visant à guider les utilisateurs à travers une série de postures de yoga tout en suivant leurs mouvements en temps réel. Utilisant une technologie de reconnaissance des points clés du corps humain via la caméra d'un appareil Android, l’application analyse la position de l'utilisateur pour vérifier si chaque pose est correctement réalisée. À chaque étape, une nouvelle pose est affichée sur le whiteboard, accompagné d'instructions visuelles, jusqu'à ce que l’ensemble de la série de postures soit complété.

**Problème 1 : Envoi du flux vidéo depuis l'application Android**

Le premier défi majeur que nous avons rencontré concernait l'envoi du flux vidéo depuis l'application Android vers l'écosystème Ingescape. Pour cela, nous avons tenté d'intégrer un package Java spécifique à Ingescape, afin de faire fonctionner l'application Android comme un agent au sein de cet environnement. Cependant, malgré l'installation des différents packages fournis et la mise à jour de Java, l'application n'a pas pu être reconnue comme un agent, empêchant ainsi l'envoi direct du flux vidéo.

**Solution explorée : Utilisation de l'application ScreenStream**

Face à cette difficulté, nous avons opté pour une solution alternative : l'utilisation de l'application **ScreenStream**. Cette application permet de diffuser le contenu de l'écran du téléphone et de l'envoyer via une URL. Les frames de cette vidéo sont ensuite récupérées dans le code de l'agent et publiées sur le whiteboard en utilisant le service **AddImage**.

Nous avons essayé d'utiliser directement **AddImageFromURL** pour l'ajout de l'image, mais cette méthode ne fonctionnait pas comme prévu. Après quelques tests, nous avons supposé que cette fonction ne supportait pas la vidéo, bien que nous n'ayons pas pu confirmer cette hypothèse.

**Problème 2 : Affichage incorrect des images sur le whiteboard**

Un autre problème majeur rencontré concerne l'affichage des images sur le whiteboard. Bien que les images soient correctement envoyées et que le whiteboard les reçoive, elles ne s'affichent pas comme prévu. En effet, malgré la conversion des images en format **Base64**, une petite "boîte noire" apparaît à l'écran avec un texte indiquant "item" suivi d'un numéro correspondant à l'ordre d'apparition de l'élément.

Après plusieurs tests, il est apparu que ce problème se produisait systématiquement, mais il n'était pas unique à notre projet. En discutant avec d'autres binômes, nous avons constaté que certains avaient rencontré ce même problème. Curieusement, certains programmes fonctionnaient correctement sur des systèmes Linux, mais l'affichage ne se faisait pas correctement sous Windows. Ce comportement semble indiquer un possible problème de compatibilité entre les systèmes d'exploitation, mais après de multiples tentatives, nous n'avons pas réussi à trouver une solution définitive.

**Problème 3 : Manque de connaissances en communication des systèmes et réseau**

Au début du projet, notre équipe avait une connaissance limitée (voire aucune) des concepts liés à la communication des systèmes et des réseaux. En particulier, nous n'étions pas familiers avec des protocoles comme **WebSocket** et **MQTT**, qui étaient essentiels pour permettre la communication entre les différentes parties de notre application, notamment entre l'application Android et le whiteboard dans l'écosystème Ingescape.

Cela a représenté un défi considérable, car nous avons dû apprendre à utiliser ces technologies en cours de projet. Bien que nous ayons réussi à intégrer ces protocoles et à établir la communication entre les systèmes, cela a nécessité un investissement de temps important. En particulier, la configuration de la communication via MQTT et l'établissement de la connexion WebSocket ont pris beaucoup de temps et ont rallongé la durée de développement. Cependant, cette expérience nous a permis de renforcer nos compétences dans ces domaines et d'approfondir notre compréhension des systèmes distribués et de la gestion des communications réseau.

**Fonctionnement**

* Télécharger l’application ScreenStream sur android depuis le playstore
* **Ici ajoute tes trucs**
* **Ajout de comment lancer l’application sur un android (tes trucs de brouteur la)**
* **Ajout de comment configurer la web socket (pour le partage des valeurs d’angle)**
* Télécharger Android Studio : <https://developer.android.com/studio>
* Télécharger le projet depuis GitHub
* Ouvrir Android Studio
* Sélectionner « » en haut à gauche de l’écran puis cliquer sur « New » puis « Import Project… »
* Dans l’explorateur de fichier ouvert pour sélectionner le projet à importer, aller dans :



Et importer le dossier :



* Une fois le projet importé, Aller dans la classe « SocketClientTask » et remplacer l’adresse IP et sélectionner le port de communication (prendre soin de vérifier que le port sélectionné est « ouvert » pour la communication)

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

* Faire de même dans la classe « OverlayView » :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

* Prendre un téléphone Android et [Activer le mode développeur sur l’appareil](https://learn.microsoft.com/fr-fr/dotnet/maui/android/device/setup?view=net-maui-9.0#enable-developer-mode-on-the-device)
* Autoriser le débogage USB : [Activer le débogage USB](https://learn.microsoft.com/fr-fr/dotnet/maui/android/device/setup?view=net-maui-9.0#enable-usb-debugging)
* Brancher le téléphone au PC via un câble USB ou USBC. L’appareil devrait apparaître dans la zone en rouge sur l’image ci-dessous. Si non, vérifier si ce dernier n’apparait pas dans le menu déroulant («  » zone en rouge)

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

* Lancer l’application :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

* L’application doit se lancer sur le téléphone
* Connecter le téléphone et l’ordinateur sur le même réseau wifi
* Lancer le stream d’écran depuis l’application ScreenStream et récupérer le l’url du stream
* Dans le navigateur lancer l’url récupéré.
* Inspecter la page, aller dans l’onglet réseau/network, puis l’onglet img. Rafraichir la page et récupérer l’url de requête de l’image [cf image inspecter] (désolé c’est le seul moyen que nous avons trouvé pour récupérer la bonne url 🙃)

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

* Ajouter cette url dans la variable url\_image (ligne 22)
* Lancer circle
* Lancer le whiteboard
* Exécuter le code main.py pour lancer l’agent image\_sender (cet agent gère tout ce que l’application Android ne fait pas)

**EXPLICATION DU FONCTIONNEMENT « IDEAL » DE L’APPLICATION :**

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Les « listes de caractéristiques » contiennent les valeurs d’angles entre différentes articulations et des distances entre certains membres.   
Ces derniers sont récupérés via Socket depuis l’application.