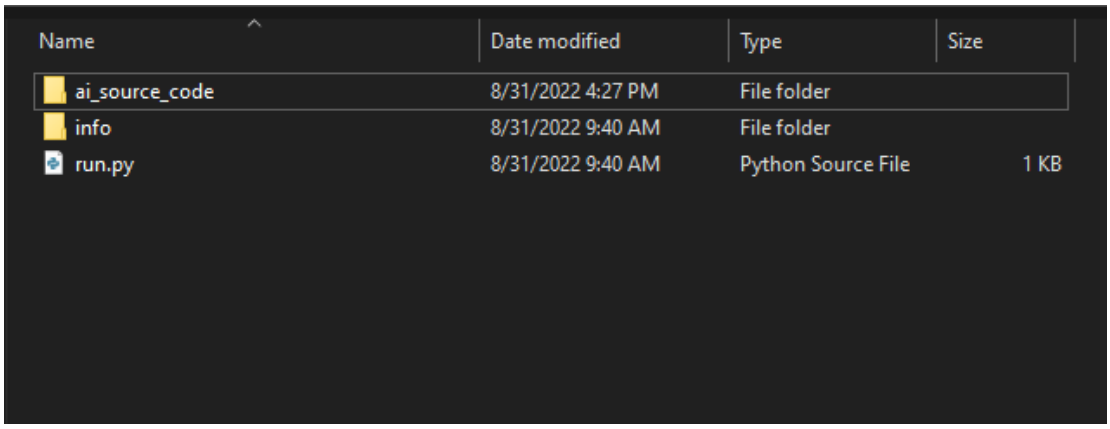
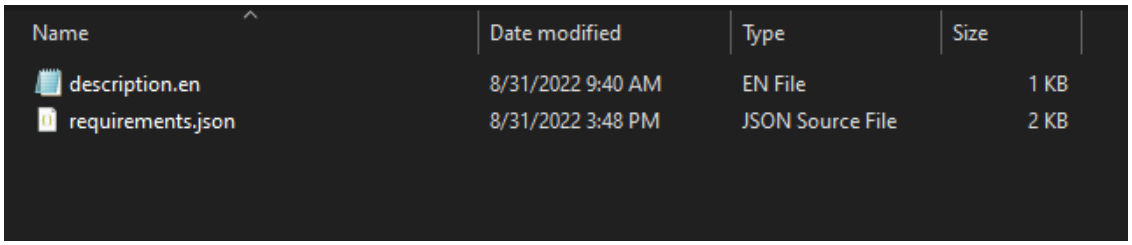
Comment créer une nouvelle IA

- Positionnez-vous dans le dossier racine de l'application et accédez au dossier « ai »
- Dans ce dossier, créez un nouveau dossier avec le nom de l'IA
- Dans ce dossier vous devez créer le sous dossier « info » et le fichier « run.py »
- Vous pouvez placer le tout le code de l'IA dans ce dossier



Dossier Info :

- Il doit contenir la description de l'IA dans un fichier de texte appelé « description.en »
- Il doit contenir les exigences de l'IA dans un fichier appelé « requirements.json »



Structure du fichier « requirements.json » :

Le fichier « requirements.json » doit contenir un objet json structuré comme suit :

```
{
"images":  {},
"medical_information": {},
"skin_lesion_characteristics": {}
}
```

Objet « image » :

Contiendra des sous-objets avec les types d'images requis par l'IA contenant la valeur minimale et maximale des images par exemple :

```
"images":  {
            "id" : {
                                "min": 1,
                                "max":5
                                },
            }
```

L'id de chaque image sera utilisé lors de la transmission des informations à l'IA

Exemple d'objet image :

```
    "dermoscopy" : {
                "min": 1,
                "max":2
                },
```

Objets « medical_information » et « skin_lesion_characteristics » :

Doivent contenir des sous-objets contenant les informations nécessaires sur le patient et la lésion, chacun de ces objets doivent être structurés comme suit

```
"id" : {
                "name":"",
                "type":"",
                 "items": [] # optional
                "scale":"" # optional
            }
}
```

L'id de chaque sous-objet sera utilisé lors de la transmission des informations à l'ai

Structure :

- Le contenu du nom doit être une chaîne de caractères
- Le contenu du type doit être l'une des valeurs suivantes :
  - « options » : crée une liste d'options
  - « int » : crée un espace numérique
  - « float » : crée un espace numérique avec des décimales
  - « text » : crée un espace de texte
  - « bool » : crée l'option oui/non
  - « date » : crée un espace de dates
- Le contenu du « items » doit être une liste d'éléments et ne doit être utilisé que si l'objet est de type « options » exemple:

        - ['élément 1', 'élément 2', ...]
    - Le contenu de l'échelle ne doit être utilise que si l'objet est de type « int » ou « float » et doit être l'une des valeurs suivantes
        - « length »
        - « time »
        - « volume »
        - « weight »

Exemples d'objets :

```
"hair_color" : {
                "name":"Hair color",
                "type":"options",
                "items":["Black","Brown","Blonde","Grey","White"]
                },
"weight" : {
                "name":"Weight",
                "type":"float",
                "scale":"weight"
            }
```

Fichier « run.py »

Le fichier run.py contiendra la méthode « run » qui sera utilisée comme pont de communication entre l'application graphique et l'IA

La méthode run recevra 2 paramètres :

```
def run(images, info):
```

Paramètre « images » :

Contiendra un dictionnaire avec toutes les images de la lésion, pour accéder à la liste des images d'un type spécifique, l'id de l'image est utilisé, exemple :

```
images["dermoscopy"]
```

Cela renverra une liste de strings avec l'emplacement des images

Paramètre « info » :

Contiendra toutes les informations nécessaires du patient et de la lésion :

Pour accéder aux informations de base du patient (âge, sexe) utilisez le code suivant :

```
info["basic_information"]
```

Pour accéder aux informations médicales du patient, utilisez le code suivant :

```
info["medical_information"]
```

Pour accéder aux caractéristiques de la lésion cutanée, utilisez le code suivant :

```
info["skin_lesion_characteristics"]
```

Return:

Après avoir fait tout le traitement, le résultat calcule par l'IA doit être retourné sous la forme d'un dictionnaire « clé » : valeur, le paramètre « risk » dans le dictionnaire est conseillé pour indiquer le type de risque de la lésion et seules les valeurs « malignant », « benign » et « indeterminate » sont accepté. Cela est à cause des filtres crées dans l'application graphique

Exemples

Fichier « requirements.json »

```
{
"images":  {
"dermoscopy" : {
                "min": 1,
                "max":2
                },
"microscopy" : {
                "min": 1,
                "max":4
                },
"photography" : {
                "min": 0,
                "max":1
                }
}
,
"medical_information": {
                "eye_color" : {
                                "name":"Eye color",
                                "type":"options",
                                "items":["Black","Brown","Blue","Gray","Green"]
                                },
                "hair_color" : {
                                    "name":"Hair color",
                                    "type":"options",
                                    "items":["Black","Brown","Blonde","Grey"]
                                    },
                "weight" : {
                                    "name":"Weight",
                                    "type":"float",
                                    "scale":"weight"
                                }
            },
"skin_lesion_characteristics":  {
                "diameter" : {
                                    "name":"Diameter",
                                    "type":"float",
                                    "scale":"length"
                                }
                }
}
```

Fichier « run.py »

```
def run(images, info):
    print("Launching AI-01...")
    print("Given images:")
    for img_name, imgs in images.items():
        print("\t"+ img_name + ":")
        for img in imgs:
            print("\t\t" + img)
    print("Other information:")
    for info_name, infos in info.items():
        print("\t"+ info_name + ":")
        for i_name, i_value in infos.items():
            print("\t\t" + i_name + ": " + str(i_value))

    #Calculate results

    results = {
        "risk":"malignant",
        "type":"melanoma",
        "Accurance":87
    }
    return results
```