

Link git.hub:

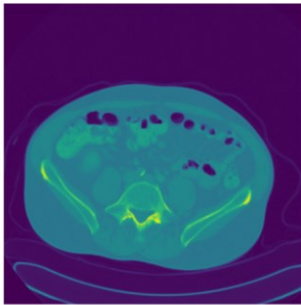
https://github.com/Leamrn/exam_HI

Screenshot of the result:

- a) Create a route ('/') that outputs HTML and shows a grid of images as a table with 4 images per row, include annotations of *Patient ID*, *Date*, and *Age*

Patients

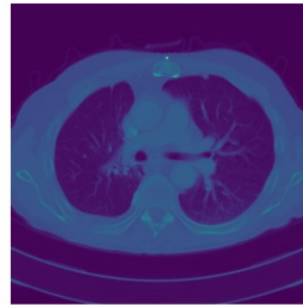
Patient name: TCGA-17-Z034
Date: 19830708
Age: 060Y



[Detail](#)

Patient name: TCGA-17-Z039

Patient name: TCGA-17-Z011
Date: 19820630
Age: 069Y



[Detail](#)

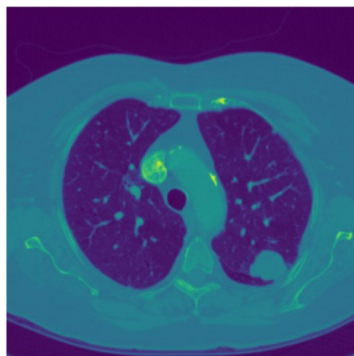
Patient name: TCGA-17-Z045

Patient name: TCGA-
Date: 19980316
Age: 074Y

[Detail](#)

Patient name: TCGA-

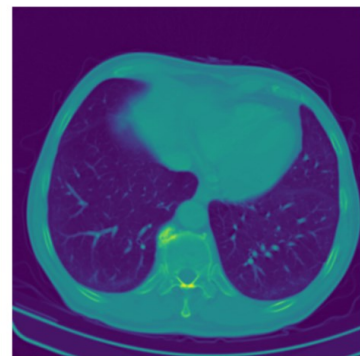
Patient name: TCGA-60-2695
Date: 19980316
Age: 074Y



[Detail](#)

Patient name: TCGA-17-Z054
Date: 19860226
Age: 075Y

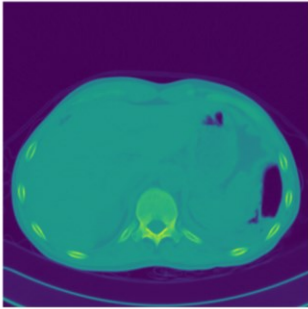
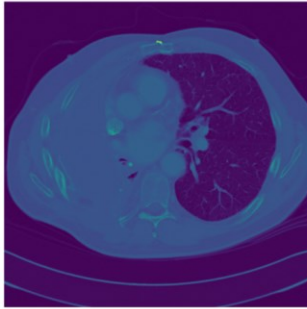
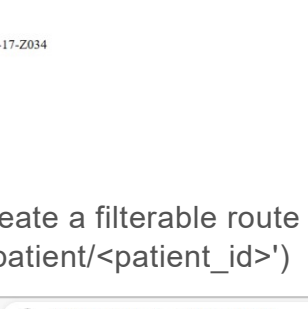
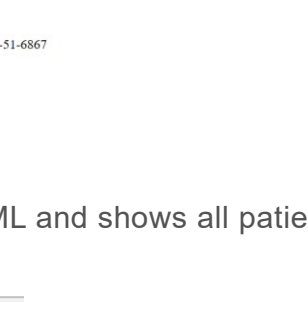
Patient name: TCGA-17-Z054
Date: 19860226
Age: 075Y



[Detail](#)

Patient name: TCGA-17-Z021
Date: 19830325
Age: 061Y

← ↻ 🏠 127.0.0.1:5000 🔍 ⚙️ 📄 ⌕ 🗑️ 🔄 ..

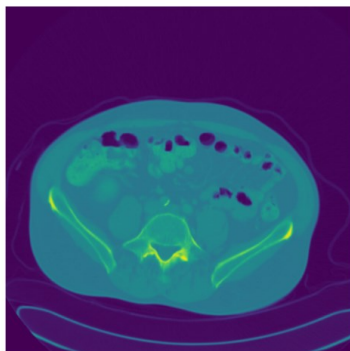
Detail	Detail	Detail
<p>Patient name: TCGA-17-Z058 Date: 19860928 Age: 060Y</p> 	<p>Patient name: TCGA-17-Z029 Date: 19831021 Age: 071Y</p> 	<p>Patient name: TCG Date: 19840713 Age: 061Y</p>
<p>Patient name: TCGA-17-Z034 Date: 19830708 Age: 060Y</p> 	<p>Patient name: TCGA-51-6867 Date: 19940924 Age: 074Y</p> 	<p>Patient name: TCG Date: 19860523 Age: 070Y</p>

b) Create a filterable route that outputs HTML and shows all patients images ('/patient/<patient_id>')

← ↻ 🏠 127.0.0.1:5000/patients/TCGA-17-Z034 🔍 ⚙️ 📄 ⌕ 🗑️ 🔄 ..

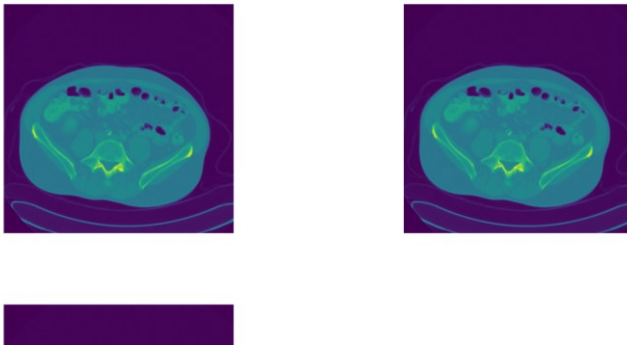
Detail patient

Details for Patient TCGA-17-Z034

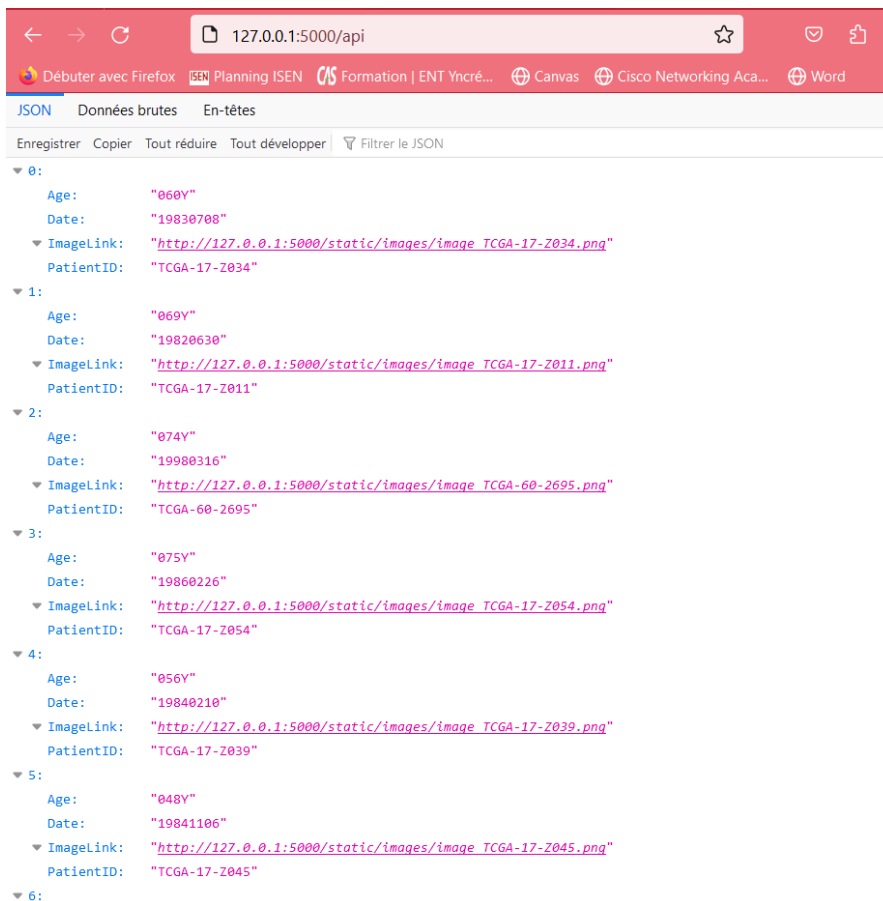


Other picture of the Patient TCGA-17-Z034

Other picture of the Patient TCGA-17-Z034



- a) Create a Webservice with a first route ('/api') to serve all DICOM metadata as JSON. At least include *Patient ID*, *Date*, *Age*, *Link to image on disk*. You do not need to make this data interoperable



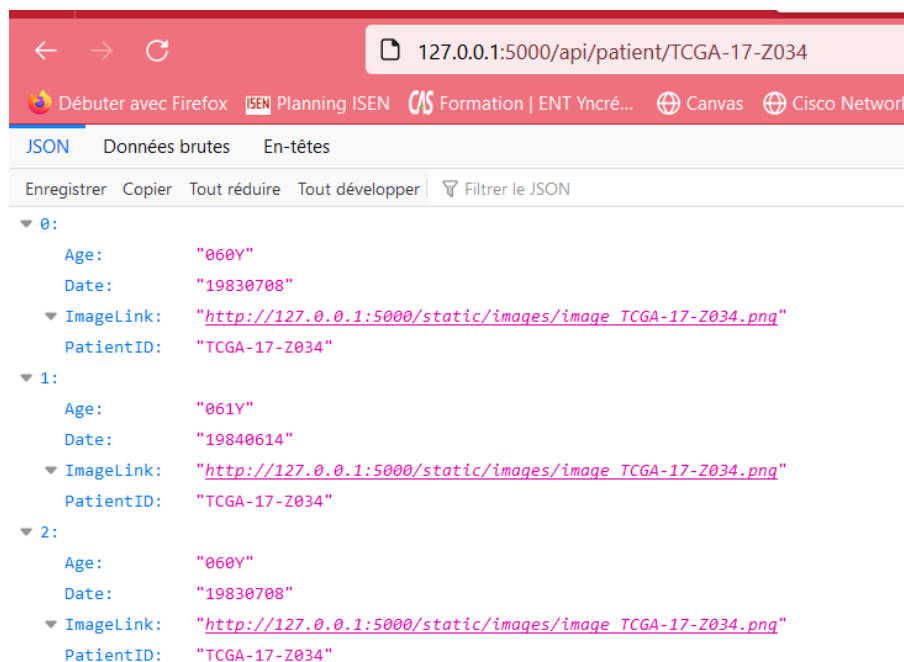
- b) if you had to make it interoperable, how would you go about it? Which tools and which FHIR resources would you use?

To make the DICOM data interoperable, we can utilize FHIR resources and tools.

1. Convert DICOM metadata to FHIR resources:

- Use a DICOM to FHIR converter tool/library to transform the DICOM metadata into corresponding FHIR resources.
 - For example, convert DICOM PatientID, StudyDate, PatientAge, etc., into FHIR Patient resource attributes like identifier, birthDate, etc.
 - The converted FHIR resources will follow the FHIR data model and structure.
2. Utilize FHIR server:
 - Set up a FHIR server that supports storing and serving FHIR resources.
 - The FHIR server will handle the storage and retrieval of the converted DICOM metadata as FHIR resources.
 3. Map DICOM attributes to FHIR profiles:
 - Define mappings between DICOM attributes and FHIR profiles/elements.
 - Ensure that the DICOM attributes are properly mapped to the corresponding FHIR resources and elements for accurate data representation.
 4. Use FHIR APIs and libraries:
 - Use FHIR-compliant APIs and libraries to interact with the FHIR server and perform CRUD (Create, Read, Update, Delete) operations on the FHIR resources.
 - These APIs and libraries provide standardized ways to access, query, and manipulate FHIR resources.

c) Create a route ('/api/patient/<patient_id>') that filters the images based on a patient's id



- a) Which cues can you find in both image names and metadata to help solve this problem?

Patient ID, Study/Series Description, Study/Series Instance UID, Image Dates

- b) Which image names do not match their metadata? Please provide code to avoid this problem in the future.

127.0.0.1:5000/errors

Débuter avec Firefox | ISEN Planning ISEN | CAS Formation | E

JSON | Données brutes | En-têtes

Enregistrer | Copier | Tout réduire | Tout développer | Filtre le JS

```

{
  "image_cues": [
    "TCGA-17-Z011",
    "TCGA-17-Z021",
    "TCGA-17-Z029",
    "TCGA-17-Z031",
    "TCGA-17-Z034",
    "TCGA-17-Z039",
    "TCGA-17-Z043",
    "TCGA-17-Z045",
    "TCGA-17-Z050",
    "TCGA-17-Z052",
    "TCGA-17-Z053",
    "TCGA-17-Z054",
    "TCGA-17-Z058",
    "TCGA-17-Z059",
    "TCGA-34-5231",
    "TCGA-50-6594",
    "TCGA-51-6867",
    "TCGA-60-2695",
    "TCGA-60-2696",
    "TCGA-60-2715",
    "TCGA-60-2725"
  ],
  "metadata_cues": [
    "TCGA-17-Z034",
    "TCGA-17-Z011",
    "TCGA-60-2695",
    "TCGA-17-Z054",
    "TCGA-17-Z039"
  ]
}

```

Mismatched Image Names and Metadata

Cues found in both image names and metadata:

- Patient ID: Unique identifier for each patient.
- Image Dates: Dates associated with the images.
- Study/Series Descriptions: Textual information about the study or series.

Results:

- Image Cues: ["TCGA-17-Z011", "TCGA-17-Z021", "TCGA-17-Z029", "TCGA-17-Z031", "TCGA-17-Z034", "TCGA-17-Z039", "TCGA-17-Z043", "TCGA-17-Z045", "TCGA-17-Z050", "TCGA-17-Z052", "TCGA-17-Z053", "TCGA-17-Z054", "TCGA-17-Z058", "TCGA-17-Z059", "TCGA-34-5231", "TCGA-50-6594", "TCGA-51-6867", "TCGA-60-2695", "TCGA-60-2696", "TCGA-60-2715", "TCGA-60-2725"]
- Metadata Cues: ["TCGA-17-Z034", "TCGA-17-Z011", "TCGA-60-2695", "TCGA-17-Z054", "TCGA-17-Z039", "TCGA-17-Z045", "TCGA-17-Z054", "TCGA-17-Z021", "TCGA-60-2715", "TCGA-17-Z045", "TCGA-17-Z058", "TCGA-17-Z021", "TCGA-17-Z034", "TCGA-17-Z058", "TCGA-17-Z029", "TCGA-17-Z043", "TCGA-17-Z031", "TCGA-17-Z034", "TCGA-51-6867", "TCGA-17-Z053", "TCGA-17-Z050", "TCGA-17-Z052", "TCGA-34-5231", "TCGA-17-Z059", "TCGA-17-Z058", "TCGA-60-2725", "TCGA-17-Z053", "TCGA-17-Z031", "TCGA-51-6867", "TCGA-50-6594", "TCGA-60-2696"]
- Missing Metadata Cues: []
- Missing Image Cues: []
- Mismatched Image Names: []

Please provide code to avoid this problem in the future.

```

###Please provide code to avoid this problem in the future.
#This function could be call to check if the image names match with their metadata
def validate_image_metadata():

    #load the dicom images
    dicom_list=[]
    path = 'static/dicoms'
    dicom_files = listdir('static/dicoms')
    image_dir = 'static/images'
    os.makedirs(image_dir, exist_ok=True)
    for filename in listdir(path):
        print(filename)

    try:
        with dcmread(f"{path}/{filename}") as f:
            dicom_list.append(f)
    except InvalidDicomError:
        print(f'{filename} is not DICOM image')

    # Create a dictionary to store the image metadata using the image cues as keys
    metadata_dict = {dcm.PatientID: dcm for dcm in dicom_list}

    # List to store mismatched image names
    mismatched_image_names = []

    # Iterate over the image files in the directory
    for image_name in os.listdir(image_dir):
        # Extract the image cue from the image name
        image_cue = image_name.split('_')[1].split('.')[0]

        # Check if the image cue exists in the metadata dictionary
        if image_cue not in metadata_dict:
            mismatched_image_names.append(image_name)

    return mismatched_image_names

```

- c) Can you fix the image names, which of them were wrong and what needed to be changed?

```

###Can you fix the image names, which of them were wrong and what needed to be changed?
def fix_image_names():

    #load the dicom images
    dicom_list=[]
    path = 'static/dicoms'
    dicom_files = listdir('static/dicoms')
    image_dir = 'static/images'
    os.makedirs(image_dir, exist_ok=True)
    for filename in listdir(path):
        print(filename)

    try:
        with dcmread(f"{path}/{filename}") as f:
            dicom_list.append(f)
    except InvalidDicomError:
        print(f'{filename} is not DICOM image')

    metadata_cues = [dcm.PatientID for dcm in dicom_list]
    image_names = os.listdir(image_dir)

    for image_name in image_names:
        image_cue = image_name.split('_')[1].split('.')[0]
        if image_cue not in metadata_cues:
            for dcm in dicom_list:
                if dcm.PatientID not in metadata_cues and dcm.PatientID != image_cue:
                    # Find a DICOM object with a matching Patient ID
                    new_image_name = image_name.replace(image_cue, dcm.PatientID)
                    # Update the image name with the correct Patient ID
                    os.rename(os.path.join(image_dir, image_name), os.path.join(image_dir, new_image_name))
                    # Rename the image file accordingly
                    metadata_cues.append(dcm.PatientID)
                    print(f"Image name '{image_name}' has been fixed to '{new_image_name}'.")

```