**X-Rays Tracker App**

**Notice**

**Introduction**

This app aims at making measurement tracking way easier by providing automatically built tables. Some improvements can be added to this code, which is why this notice was created. It’s what makes the code reusable and updatable: explaining the code to its’ users enables it to be permanent and useful in a long-term.

I’ll explain the different steps to how I constructed the App and what’s important to have in mind regarding the different studies, when trying to modify the code.

Table of Contents

[Presentation 3](#_Toc17480529)

[I. Screening the directory 3](#_Toc17480530)

[**1.** **Overall presentation** 3](#_Toc17480531)

[**2.** **Running the screening directory code** 4](#_Toc17480532)

[II. Creating the Application 4](#_Toc17480533)

[**1.** **Whole App background** 4](#_Toc17480534)

[**2.** **Object-Oriented Pages** 4](#_Toc17480535)

[**3.** **Pandastable’s DataExplorer** 5](#_Toc17480536)

[**4.** **Providing the needed information** 6](#_Toc17480537)

[III. Database handling 6](#_Toc17480538)

[**1.** **Opening the database** 6](#_Toc17480539)

[**2.** **Saving the database** 6](#_Toc17480540)

[**3.** **Refreshing the database** 7](#_Toc17480541)

[IV. Platform Personalization 7](#_Toc17480542)

[**1.** **How to change some app’s features** 7](#_Toc17480543)

[**2.** **Architecture of the Platform** 7](#_Toc17480544)

[V. From python to .exe 8](#_Toc17480545)

[VI. Might be useful 9](#_Toc17480546)

[VII. Improvements ideas 9](#_Toc17480547)

[Annexes 10](#_Toc17480548)

[**Annex 1 : Code Tutorials used :** 10](#_Toc17480549)

[**Annex 3: Updates from v0 to v4 (last version)** 10](#_Toc17480550)

[**Annex 4: Last check – August 16th, 2019** 10](#_Toc17480551)

[**Annex 5 : User guidelines** 11](#_Toc17480552)

**Table of Figures**

[*Figure 1 : Studies coding differences 2*](#_Toc17480307)

[*Figure 2 : Database creation 3*](#_Toc17480308)

[*Figure 3 : XRays to verify- screenshot 4*](#_Toc17480309)

[*Figure 4 : Study page 5*](#_Toc17480310)

[*Figure 5 : App architecture 7*](#_Toc17480311)

[*Figure 6 : Error Sources 8*](#_Toc17480312)

# **Presentation**

**Context**

The Spine Research Lab’s work and data depends on a large part on the X-Rays. The X-Rays provide the data for the SPSS database, which enables the researchers to access and share data. Until now, everything is kept into a huge Excel file per Study and tracking X-Ray measurement can be really complicated because of the different follow-ups of the different types of X-Rays. The main aim is to know the status of each study and what the different priorities are.

The aim of this App is to provide a list of all available X-Rays, their status (to be checked, verified, completed or missing), and all information needed for the treatment, like the path, the site name and the date (could be the OR Date for operative patients, or the Date of 1st Entry for non-operative patients).

Each X-Rays have special characteristics: Study – Site – Patient ID – View – Time Index

Those characteristics can be extracted from each file name, yet each study has its’ own way to ‘code’ that set of information.

1. **Screening the directory**
2. **Overall presentation**

For each Study, in order to know the available files depending on their format, the computer screens the whole folder to create the database.

This process is adapted to each study, as each of them has its’ special folder organization and special way to name files. **Those differences had to be taken into account for every step of the App creation.** *Figure 1* shows the main differences between the studies:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **PON** | **PEED** | **ScoliRisk** | **CTJ** | **NiH** | **MESA** | **PCD** | **Supine** |
|  | **Directory** | Study > Site> PON > O/NOp > Patients folders | Study > site >Patients folder | Study > site >Patients folder | Study > Patients Folders | Study > site >Patients folder | Study > site >Patients folder | Study > site >Patients folder | Study > site >Patients folder |
|  | **File name example** | WUNPONY0081CS (I-01) 6wks.lat1 | BAV-001 (I-00) base.2014.01.22.ap | BAU-018 (I-07) 2YR.2014.01.23.lat1-2 | NYUCT J0001-RZ (I-05) 1YR.cs.lat1 | HS008 (I-00) baseline yyyy.mm.dd.ap | 08-011 (I-00) base.2016.01.14.bendl.xlsx | BSCPCDY0002-DR (I-00) base.fs.lat1 | HSSSVS0302-ND (SUPINE) supine.2017.12.21.lat1 |
|  | **View** | .view or .typeview.view | | | | | | | |
|  | **Time index** | (I-xx) **and** wk nbr | | | | | | | |
| **Date** | **Where** | Excel  Imported | Name | Name | Excel  **Not Imported** | Name | Name | Excel  **Not Imported** |  |
| **Consequence** | * **If in name,** the saved ‘last update’ doesn’t contain the date as a column : it’s in the file name * **If not** (in excel), it saves the date every time it is entered in the table | | | | | | | |

*Figure 1 : Studies coding differences*

The process is then done by study, with a function per study, the name of the function goes this way: **Refresh\_{StudyName}**

**Steps**

1. Browse the sites
2. Browse the patients’ folders per site
3. Browse each patient folder to see how many formats exist per file (num, tif, xls, dcm for the same name)
4. Compare the file name with existing saved database with comments and validations
5. Merges the refreshed database with its comments and validation

Figure 2 : Database creation

1. **Running the screening directory code**

The python file runs the different screening directory files.

It creates a function to execute each of the studies’ screening codes. This same function is run at the App’s running, and when the table aims at being refreshed: **Refresh\_{StudyName}**

1. **Creating the Application**
2. **Whole App background**

To create the App I chose the Tkinter module because it’s easy to be use to and it provides everything needed for a classic app. Qt was an option but couldn’t be downloaded on the computer due to a lack of capacity.

The apps’ display parameters details are available in **Platform Personalization**.

All the pages are created when the app runs, as well as the Tables. The tables have a refresh button

1. **Object-Oriented Pages**

The different pages are all created as classes, which makes it way quicker and easier to open, but also more complicated to store into lists.

Each class has its own variables and can display buttons and labels. Variables from other classes are not recognized unless they are marked as global before they are defined.

1. **Pandastable’s DataExplorer**

A screenshot of a computer

Description automatically generated

*Figure 3 : XRays to verify- screenshot*

The DataExplorer enables multiple choices as commenting, validating X-Rays, but also sorting and modifying the database.

**Uploaded:**

* From a dataframe

pt=Table(dataframe=dataframe)

pt.show()

* From a csv

pt.importcsv(file.csv)

**Saved:**

pt.doExport(‘targetedfile.csv’)

1. **Providing the needed information**

A screenshot of a social media post

Description automatically generated

Figure 4 : Study page

Each study page provides:

* Overall number of XRays
* Number of completed track
* Number of missing file
* Number of X-Rays ready to be measured
* Number of X-Rays ready to be verified
* Number of Unvalid XRays
* Number of problematic XRays

1. **Database handling**
2. **Opening the database**

**First opening:**

Focuses on the last database created:

pt=Table(dataframe-dataframe)

pt.show()

**Refresh:**

pt.importCSV(file.csv)

1. **Saving the database**

Saving button for the raw database

1. Saving the table into an ephemere csv file: pt.doExport(‘**ephemere.csv’**)
2. Save ephemere into a database
3. Filter and treat that database and save it into the CSV file
4. Open that same ephemere csv file and show it on the Table

The program uses **pandastable** Tables in order to display the different dataframes but when modifying the table, it modifies the table and not the dataframe. The Table cannot be filtered so that’s why we need to go through all those steps in order to save the whole modified table and not the dataframe originally displayed.

1. **Refreshing the database**

Refresh the database, which means screening again the files**: see Figure 2**

The program uses **pandastable** Tables in order to display the different dataframes but when modifying the table, it modifies the table and not the dataframe. The Table cannot be filtered so that’s why we need to go through all those steps in order to save the whole modified table and not the dataframe originally displayed.

1. **Platform Personalization**
2. **How to change some app’s features**

* All the text is in Arial font
* Background color : <https://htmlcolorcodes.com/fr/> check this website for more choices
* Buttons color :
  + Background
  + Text Color
* Time format used is the following :

strftime(‘%B %d, %Y %H:%M:%M:%S %p’) : August 08, 2019 02:59:22 PM

<http://strftime.org> to modify the **time format**

* TITLE FONT = can be changed too

1. **Architecture of the Platform**

The platform creates all pages at the time it runs. Databases displayed and time of last update are the only refreshed objects.

Figure 5 : App architecture

1. **From python to .exe**

In order to create an executable application, python and its’ packages need to be on the environment path -which requires an admin access-. Pip is particularly needed. If not recognized, can be implemented by:

* Going to the Python official website **>** Downloading the executable installer **>** When opening it, choose the ‘modify’ option: it will reboot the packages and make sure their path is in the environment. [Click here for more details.](https://www.youtube.com/watch?v=UZX5kH72Yx4)

Three ways can be found on the internet to transform a python file into an app:

* **Pyinstaller:**

using Pyinstaller -n XRayTracker -i icoco.ico xraytracker.py

This automatically creates the app. If the app created opens and closes itself, that means there’s a code error. The solution is to run it and read the error raised.

auto-py-to-exe can also run and create automatically the **Pyinstaller** code:

pip install auto-py-to-exe

auto-py-to-exe

* **Cx\_freeze:**

Using python setup2.py build

* **Py2exe:**
* only for python versions before 3.4: **this solution doesn’t exist anymore** (tupple error)

Pyinstaller was the quicker and raised less errors than cx\_freeze.

Python 3.7 has some errors with the last numpy version (*numpy.random.common* not found) : using **Python 3.6.8** can be a solution only if **TCL 8.6.9** is used, it won’t work with the 8.6.6 version of TCL, which is the one downloaded with python 3.6.8. TCL is the package associated with Tkinter. This error is very recent and has been transmitted by Python’s users to Python mid-July 2019

Uninstalling all python versions and reinstalling only the last one, then reinstalling each module is what worked in my case. Running the code depends on the previous downloaded packages and anything can create an error.

1. **Might be useful**

Other codes were used in order to create a fully representative database:

* **df\_displayer.py** : displays the database you want, useful to have a quick idea of how many columns, rows there are, save it into csv or excel… anything is possible with that table.

The databases were created on the current architecture: If it has come to change, some errors can happen with the App. This table sums up all the possible error that can occur

|  |  |
| --- | --- |
| Study concerned | Source |
| CTJ | The CTJ database has been created on the folder directory available: the only site was the NYU and so then it was no ‘site folders’, so everything is built for NYU. Any further change of the directory will create an error |
| Studies databases in general | The main weakness of the code Is that it creates the database from the existent directory but is not adaptable. It doesn’t identify the difference between files and folders and just treats it as it was defined. If a folder is created that changes the order, everything could be blocked and the app won’t run |

**Figure 6 : Error Sources**

1. **Improvements ideas**

Some more ideas to have a more complete and performant app.

**Database features**

* Add a filtering feature base on the Table given by Renaud : it will identify each study and set its’ parameters : the app also gives the non-compulsory X-Rays information,
* Be able to say when an X-Ray is missing (depending on Study parameters and available files) and provide a list of missing X-Rays per Site

**App features**

* Find a way to display the *showstatusbar* and *showtoolbar* from **Pandastable** to **Tkinter**
* Rebooth the user interface but making it depend on user’s authorizations, and a signature available – to know measured/verified what
* Measurement planning and measurement report for users

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Annexes**

## **Annex 1 : Code Tutorials used :**

* [Create a Table in the Graphic Interface](https://www.youtube.com/watch?v=ddoPYppcppc)
* [Create an online interface](https://www.youtube.com/watch?v=hRH01ZzT2NI)
* [Tkinter Python Tutorial](https://www.youtube.com/watch?v=VMP1oQOxfM0)
* [Making of Graph Interfaces](https://www.youtube.com/watch?v=HjNHATw6XgY)
* Lot of videos : [Tkinter for Intermediates](https://www.youtube.com/playlist?list=PLQVvvaa0QuDclKx-QpC9wntnURXVJqLyk)
* <http://strftime.org> for the **time format**
* [Panda’s tricks and tips](https://medium.com/@bingobee01/pandas-tricks-and-tips-a7b87c3748ea)
* **Using DataExplorer:** 
  + [**DataExplorer main parameters**](https://pandastable.readthedocs.io/en/latest/dataexplore.html)
  + [Filtering with Query](https://github.com/dmnfarrell/pandastable/wiki/Filtering)

## **Annex 3: Updates from v0 to v4 (last version)**

* The Start Page Directly goes to the Studies’ Pages
* The Patients Pages were deleted - Just front end, wasn’t developed yet and useless
* The login pages were deleted – useless
* The ‘Measure’ and ‘Verify’ buttons were corrected
* A ‘Check folder’ button has been added to ‘Missing tif’ pages
* Uniformisation of all the pages and buttons
  + Created the Non Valid Pages and Button
  + Refresh the Different Studies’ pages numbers
  + Filtered Missing files – Completed – Ready to measure – Ready to verify pages : only show the Xrays with ‘Valid’ different from ‘no’
  + Lastupdate’s time updates for all the ‘Whole Database Pages’
* Smaller database’s font in order to display all the columns
* Refresh button for all databases
* Statusbar and Toolbar available for all tables
* Everything moved to the S: disc

## **Annex 4: Last check – August 16th, 2019**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **PON** | **PEED** | **ScoliRisk** | **CTJ** | **NiH** | **MESA** | **PCD** | **Supine** | **Comment** | |
| App check | **Whole db** | X | X | X | X | X | X | X | X | Each time a table is refreshed or saved, it creates an unnamed column: it’s not problematic but better if fixed | |
| X | X | X | X | X | X | X | X |
| **Completed** | X | X | X | X | X | X | X | X |
| **Missing tif** | X | X | X | X | X | X | X | X |
| **To Measure** | X | X | X | X | X | X | X | X |
| **To Verify** | X | X | X | X | X | X | X | X |
| **Problematic** | X | X | X | X | X | X | X | X |
| **Not Valid** | X | X | X | X | X | X | X | X |
| Code check |  | All uniformized, except for the **saving process that adapts to each study**. | | | | | | | | |  | |

## **Annex 5 : User guidelines**

User Guidelines has been saved in the same directory as the app. It provides useful information for anyone wanting to use the app.