AA-Koala Local Program Guide

Introduction

This local program acts as the front-end service for the Automated Audit Benchmarking project.

In our local environment, we use this program to send HTTP requests to the backend
(`AABKoala/Server`) and receive the results back to our local computer.
Two python files do the trick:

- **`LocalProgram/resultRequest.py`** for result-related requests
- **`LocalProgram/graphRequest.py`** for graph-related requests

Before sending a request

1. To start the server, run this command in terminal (Mac)\
   Note: Make sure you are located in `AABKoala/Server` directory

```
python3 manage.py runserver
```

2. The server is up now, have `AABKoala` project ready in your IDE and go to `LocalProgram` directory
3. We can start sending the requests

Result Requests

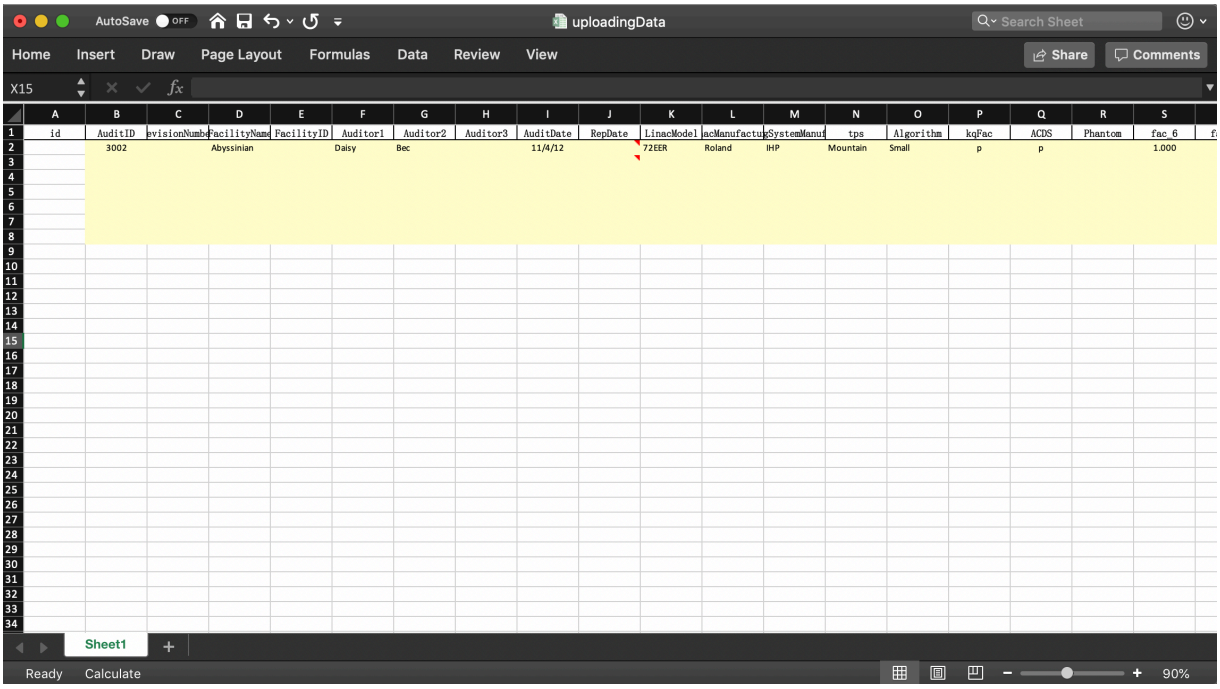
- **`LocalProgram/resultRequest.py`**

Functions:

- **· Insert new result(s) into MySQL database**

```
def insertNewResult(self)
```

1. Open the excel `LocalProgram/upload/uploadingData.xlsx` , fill in the result(s) you want to insert and save the excel file



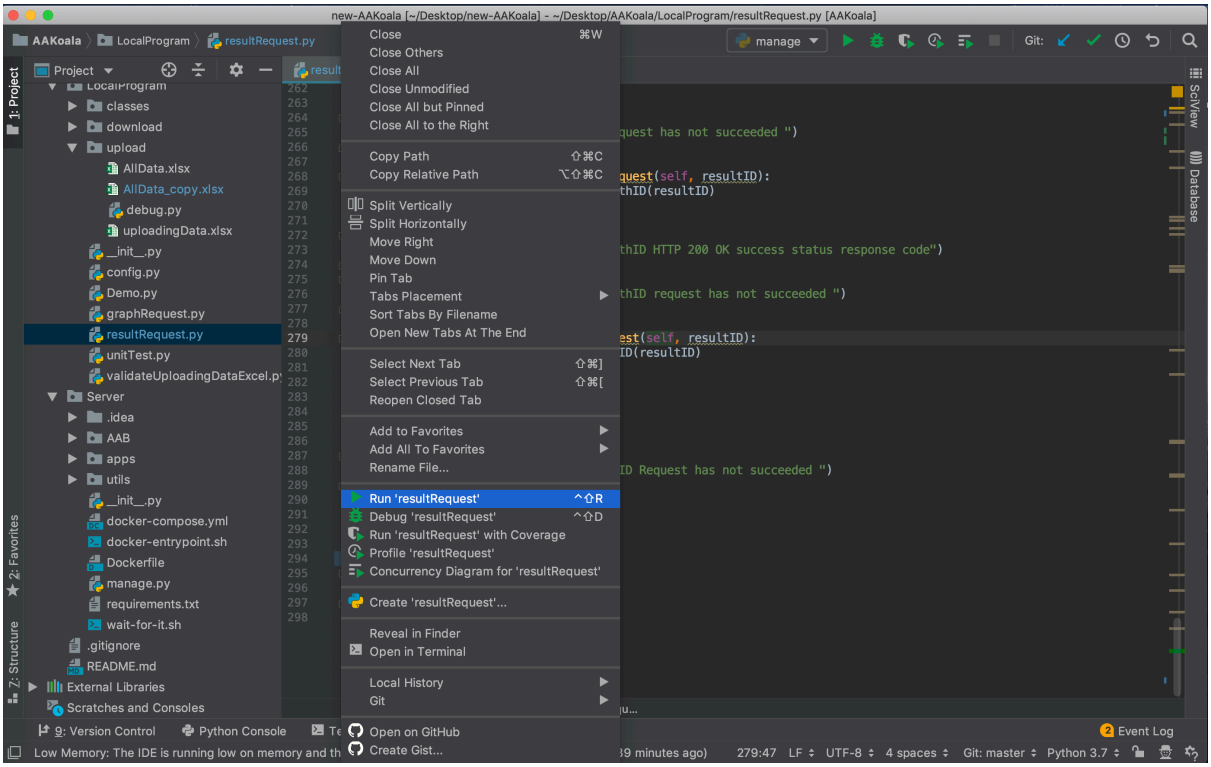
\

* The `id` field in `uploadingData.xlsx` can be blank as it will be automatically generated by the database

2. In `resultRequest.py` , create an python object `obj` and use `obj` to call `insertNewResult` function

```
obj = resultRequest() # create an object of resultRequest.py class
obj.insertNewResult() # insert new result(s)
```

3. Run `resultRequest.py` in your IDE



\

Alternatively, you can run this command in terminal:

```
python3 resultRequest.py
```

4. Insertion completed, check the inserted result(s) printed out in your console
5. Refresh the database to see the new `result` table

· List all the results stored in MySQL database

```
def listResults(self)
```

1. In `resultRequest.py`, create an python object `obj` and use `obj` to call `listResults` function

```
obj = resultRequest() # create an object of resultRequest.py class
obj.listResults() # list the complete results
```

2. Run `resultRequest.py` in your IDE or execute this command in terminal:

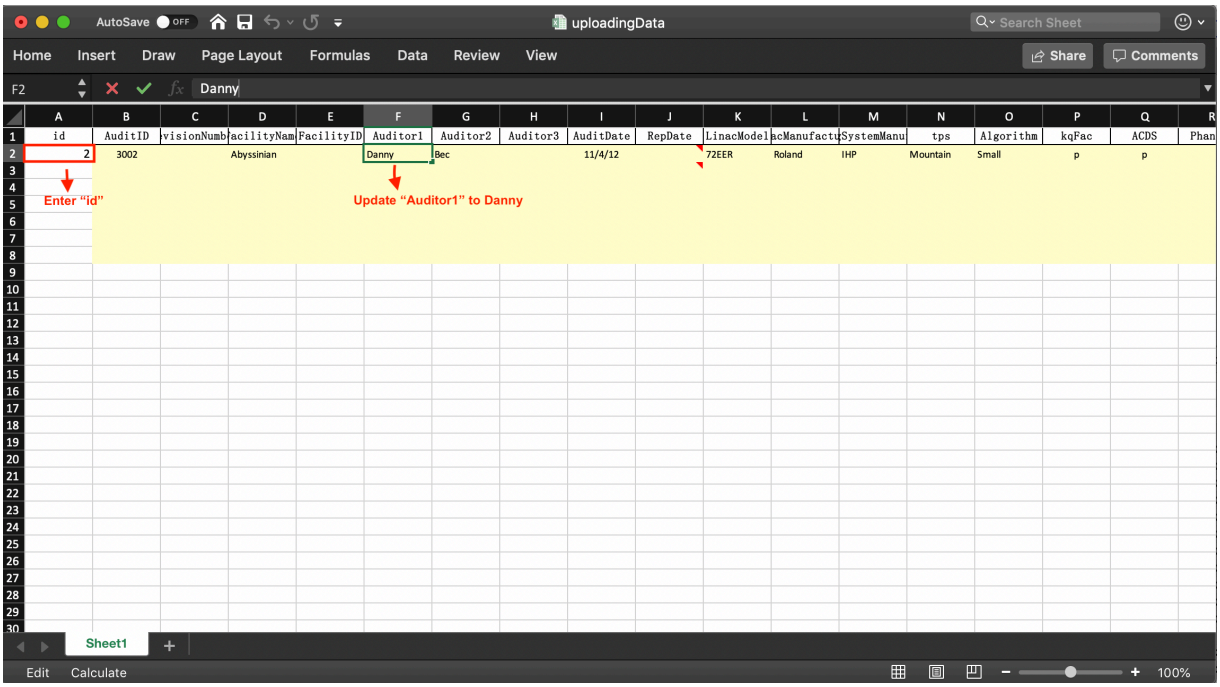
```
python3 resultRequest.py
```

3. Listing completed, check the listed results printed out in your console
4. You will get a copy `LocalProgram/download/list.xlsx` of complete result in the database

• Update result(s) stored in MySQL database

```
def updateResults(self)
```

1. Open the excel `LocalProgram/upload/uploadingData.xlsx` , fill in the result(s) you want to update and save the excel file\



\

* The `id` field in `uploadingData.xlsx` should **not** be left blank (the database need it to identify which result to update). Find the correct `id` in `result` database table or `list.xlsx` file (if generated)

2. In `resultRequest.py` , create an python object `obj` and use `obj` to call `updateResults` function

```
obj = resultRequest() # create an object of resultRequest.py class
obj.updateResults() # update the result(s)
```

3. Run `resultRequest.py` in your IDE or execute this command in terminal:

```
python3 resultRequest.py
```

4. Update completed, check the updated results printed out in your console
5. Refresh the database to see the updated result(s)

- ・Retrieve a specific result from MySQL database

This function requires a parameter **`resultID`** \
`resultID` : `id` field in `result` database table

```
def retrieveResultWithID(self, resultID)
```

1. In `resultRequest.py` , create an python object `obj` and use `obj` to call `retrieveResultWithID` function

```
obj = resultRequest() # create an object of resultRequest.py class
obj.retrieveResultWithID('1') # retrieve the result of id 1
```

2. Run `resultRequest.py` in your IDE or execute this command in terminal:

```
python3 resultRequest.py
```

3. Retrieve completed, check the retrieved result printed out in your console
4. You will get a excel file `LocalProgram/download/retrievexxx.xlsx` storing the retrieved result

- ・Delete a result from MySQL database

This function requires a parameter **`resultID`** \
`resultID` : `id` field in `result` database table

```
def deleteResultWithID(self, resultID)
```

1. In `resultRequest.py` , create an python object `obj` and use `obj` to call `deleteResultWithID` function

```
obj = resultRequest() # create an object of resultRequest.py class
obj.deleteResultWithID('1') # delete the result of id 1
```

2. Run `resultRequest.py` in your IDE or execute this command in terminal:

```
python3 resultRequest.py
```

3. Deletion completed, refresh the database to see the changes

Graph Requests

- **`LocalProgram/graphRequest.py`**

Functions:

- · List all the graphs stored in MySQL database

```
def list_graphs(self)
```

1. In `graphRequest.py`, create an python object `obj` and use `obj` to call `list_graphs` function

```
obj = graphRequest() # create an object of graphRequest.py class
obj.list_graphs() # list all the graphs in database
```

2. Run `graphRequest.py` in your IDE or execute this command in terminal:

```
python3 graphRequest.py
```

3. Listing completed, check the graph information printed out in your console

- · Retrieve a graph from MySQL database

This function requires a parameter **`fileName`** \
`fileName` : `fileName` field in `graph` table

```
def retrieve_graph(self, fileName)
```

1. In `graphRequest.py`, create an python object `obj` and use `obj` to call `retrieve_graph`

function

```
  obj = graphRequest() # create an object of graphRequest.py class
  obj.retrieve_graph("3DCRT_1603551977776.png") # retrieve the graph of file
name 3DCRT_1603551977776.png
```

2. Run `graphRequest.py` in your IDE or execute this command in terminal:

```
python3 graphRequest.py
```

3. Retrieve completed, the graph is stored into `LocalProgram/download` folder in png format

- · Plot a 3DCRT graph

This function requires two parameters: **`mode`** and **`facilities`** \
`mode` : `"all"` (all results)\
`facilities` : Sets of facility names and result ids from `result` database table,
`'{"facilityName1": [id1, id2, ..], "facilityName2": [id1, id2, ..], ..}'`

```
def plot_graph_NDS_3DCRT(self, mode, facilities)
```

1. In `graphRequest.py` , create an python object `obj` and use `obj` to call `plot_graph_NDS_3DCRT` function

```
  obj = graphRequest() # create an object of graphRequest.py class
  obj.plot_graph_NDS_3DCRT("all", '{"Drever": [308], "Avocet": [106, 302]}') #
plot a graph with all data, using result 308 for Drever facility & result 106,
302 for Avocet facility
```

2. Run `graphRequest.py` in your IDE or execute this command in terminal:

```
python3 graphRequest.py
```

3. Plotting completed, check the graph information in your console
4. The graph is stored into `LocalProgram/download` folder in png format
5. Refresh the database to see the updated `graph` table

- Plot an IMRT graph

This function requires two parameters: **`mode`** and **`facilities`** \
`mode` : `"all"` (all results), `"average"` (average), `"std"` (standard deviation)\
`facilities` : Sets of facility names and result ids from `result` database table, `'{"facilityName1": [id1, id2, ..], "facilityName2": [id1, id2, ..], ..}'`

```
def plot_graph_NDS_IMRT(self, mode, facilities)
```

1. In `graphRequest.py` , create an python object `obj` and use `obj` to call `plot_graph_NDS_IMRT` function

```
  obj = graphRequest() # create an object of graphRequest.py class
  obj.plot_graph_NDS_IMRT("std", '{"Drever": [308], "Avocet": [106, 302]}') #
plot a graph with standard deviation data, using result 308 for Drever facility
& result 106, 302 for Avocet facility
```

2. Run `graphRequest.py` in your IDE or execute this command in terminal:

```
python3 graphRequest.py
```

3. Plotting completed, check the graph information in your console
4. The graph is stored into `LocalProgram/download` folder in png format
5. Refresh the database to see the updated `graph` table

- Delete graph(s) from MySQL database

This function requires a parameter **`graphID`** \
`graphID` : A list of `id` s (one or more) from `graph` database table

```
def delete_graph(self, graphID)
```

1. In `graphRequest.py` , create an python object `obj` and use `obj` to call `delete_graph` function

```
  obj = graphRequest() # create an object of graphRequest.py class
  obj.delete_graph("15") # delete a graph of id 15
```

2. Run `graphRequest.py` in your IDE or execute this command in terminal:

```
python3 graphRequest.py
```

3. Deletion completed, refresh the database to see the changes