Functional Test Case

Linking user stories and test cases in the sprint

User Story ID	Request Description	TC01	TC02	TC03	TC04	TC05	TC06	TC07	TC08	TC09	TC10	TC11	TC12	TC13	TC14
US 1.1	Store new data in the database	x	x												
US 1.2	Query data in the database	x	x	x	x	x	x	x	x	x	x				
US 1.3	Access data anywhere within the organization.	x	x												
US 1.4	Generate graph	x	x									x	x	x	x

Functional Test Case

US 1.1: Store the date from Excel to the Database

- TC 01 Store new data in the database (successfully)

Pre-Conditions:

Run docker-compose up, it composes starts and runs your entire app. Set up docker [refer to deploy guide document].

Procedures:

a) Open upload file and download uploadingData.xlsx

b) Fill the uploadingData.xlsx.

   a. Base on the data you have then fill those entities.

   b. The first entity id is not requiring filling (even you fill the id, it will not be updated into the database)

   c. Fill the rest of the entities. Input the correct data type [refer to the Entry Data section]

c) Execute and Run the program to validate the data fill in the Excel is correctly and able to update into Database

d) Open validateUploadingDataExcel.py

e) Run TestValidtor Class

   a. All data in the uploadingData.xlsx will be tested.

   b. If any data is not filled as required, the error message will print in the error message.

f) Execute and Run the program that can update Excel data in the database.

   a. Open resultRequest.py

   b. Run insertNewResult() method

Expected outcome:

After the user-run insertNewResult() method, the data in the uploadingData.xlsx will successfully store in the database. Go to the Database, new data have been added.

- TC 02 Store new data in the database (unsuccessfully)

Pre-Conditions:

Run docker-compose up, it Compose starts and runs your entire app. Set up docker [refer to deploy guide document].

Procedures:

a) Try to Run insertNewResult() method, but new data not add in the Database.

a. Try to run validateUploadingDataExcel.py, check the data is correctly filled.

b. Try to run uintTest.py, check the HTTP request is correct.

Expected outcome: After user run insertNewResult() method, the data in the uploadingData.xlsx, will successful store in the database. Go to the Database, new data have been added.

US 1.2: Query the date from the Database

- TC 03 Query new data in the database (successfully)

Pre-Conditions: The data user wants to query is in the database.
Procedures: a) Go to the Database, get the data record ID. b) Execute and Run program that can update Excel data in the database. c) Open resultRequest.py a. Run retrieveResultWithIDs(ID) method
Expected outcome: After user run retrieveResultWithIDs(ID) method, user get the data they want.

- TC 04 Query data in the database (unsuccessfully)

Pre-Conditions: The data user wants to query is in the database.
Procedures: a) Try to run retrieveResultWithIDs(ID) method, but not successful retrieve data. a. Try to run validateUploadingDataExcel.py, check the data is correctly filled. b. Try to run uintTest.py, check the HTTP request is correct.
Expected outcome: After user run retrieveResultWithIDs(ID) method, user get the data they want.

- TC 05 Delete data in the database (successfully)

Pre-Conditions: The data user wants to query is in the database.
Procedures: a) Go to the Database, get the data record ID. b) Execute and Run program that can update Excel data in the database. c) Open resultRequest.py a. Run deleteResultWithID(self, resultID) method

Expected outcome: After user run deleteResultWithID(self, resultID) method, the record have been delete in the database

- TC 06 Delete data in the database (unsuccessfully)

Pre-Conditions: The data user wants to delete is in the database.
Procedures: a) Try to run deleteResultWithID(self, resultID) method, but not successful delete data. a. Check the user enter the correct id as paramter. b. Try to run uintTest.py, check the HTTP request is correct.
Expected outcome: After user run deleteResultWithID(self, resultID) method, user get the data they want.

- TC 07 update data in the database (successfully)

Pre-Conditions: The data user wants to update is in the database.
Procedures: a) Go to the Database, get the data record ID that you want to update. a) Fill the uploadingData.xlsx b) Execute and Run program that can update Excel data in the database. c) Open resultRequest.py a) Run updateResultsWithIDs (self, resultID) method
Expected outcome: After user run updateResultsWithIDs (self, resultID) method, the new data in the Excel is update in the database

- TC 08 update data in the database (unsuccessfully)

Pre-Conditions: The data user wants to update is in the database.
Procedures: a) Try to run updateResultsWithIDs (self, resultID) method, but not successful delete data. a. Check the user enter the correct id as paramter. b. Try to run uintTest.py, check the HTTP request is correct.
Expected outcome: After user run deleteResultWithID(self, resultID) method, user get the data they want.

- TC 09 list all data in the database (successfully)

Pre-Conditions: The data user wants to have is updated in the database.
Procedures: a) Execute and Run program that can update Excel data in the database. b) Open resultRequest.py a. Run listResult () method
Expected outcome: After user run listResult () method, all data is list to user

- TC 10 list all data in the database (unsuccessfully)

Pre-Conditions: The data user wants to have is updated in the database.
Procedures: a) Try to run listResult () method, but not successful list data. a. Try to run uintTest.py, check the HTTP request is correct.
Expected outcome: After user run listResult () method, all data is list to user

US 1.3 Access data anywhere within the organization.

- TC 01 Store data in the database (successfully)
- TC 02 Store data in the database (unsuccessfully)

US 1.4 Generate graph

- TC 11 plot NDS_IMRT graph (successfully)

Pre-Conditions: The data use to plot NDS_IMRT graph is in the database.
Procedures: a) Go to graphRequest.py a) Run plot_graph_ NDS_IMRT ("all", '{"Drever": [308], "Avocet": [106, 302]}') # a. Enter the record id and facility name in the method as parameters b. Run plot_graph_ NDS_IMRT (self, mode, facilities) method

Expected outcome:

The NDS_IMRT graph is generated and show in the download file.

- TC 12 plot NDS_IMRT graph (unsuccessfully)

Pre-Conditions:

The data use to plot NDS_IMRT graph is in the database.

Procedures:

a) Try to run plot_graph_NDS_3DCRT(self, mode, facilities) method, but not successful retrieve data.

a. Check the NDS_IMRT data in the database.

b. Try to run uintTest.py, check the HTTP request is correct.

Expected outcome:

After user run retrieveResultWithIDs(ID) method, user get the data they want.

- TC 13 plot NDS_3DCRT graph (successfully)

Pre-Conditions:

The data use to plot NDS_3DCRT graph is in the database.

Procedures:

a) Go to graphRequest.py

a. Run graphRequest().plot_graph_NDS_3DCRT("all", '{"Drever": [308], "Avocet": [106, 302]}')

a) Enter the record id and facility name in the method as parameters

Expected outcome:

The NDS_3DCRT graph is generated and show in the download file.

- TC 14 plot NDS_3DCRT graph (unsuccessfully)

Pre-Conditions:

The data use to plot NDS_3DCRT graph is in the database.

Procedures:

a) Try to run plot_graph_NDS_3DCRT(self, mode, facilities) method, but not successful retrieve data.

a. Check the NDS_3DCRT data in the database.

b. Try to run uintTest.py, check the HTTP request is correct.

Expected outcome:

The NDS_3DCRT graph is generated and show in the download file.

Entry Data

The full entry data example in the AllData.xlsx. (In the upload file)