|  |  |  |
| --- | --- | --- |
|  |  | User Requirement Analysis  Google Sheet Into DB |

May 25, 2016 Yiwei Sun

# Purpose

This open source pipeline project is for G2F collaborators to exchange and share their Google sheet data. Considering security,budget and computer knowledge level within the whole G2F collaborators, it would be straightforward for them to put all the Google sheet data into an open source database located in a server through a simple to use web interface. We would like to use Django framework and Postgresql to create this pipeline since Django and Postgresql are all free open source software with security levels control, furthermore, Django is facile to learn or to create web interfaces connecting with open source database like Postgresql.



Table of Contents

Purpose 1

User diagram 3

Requirement analysis 5

Web page design 7

Database design 10

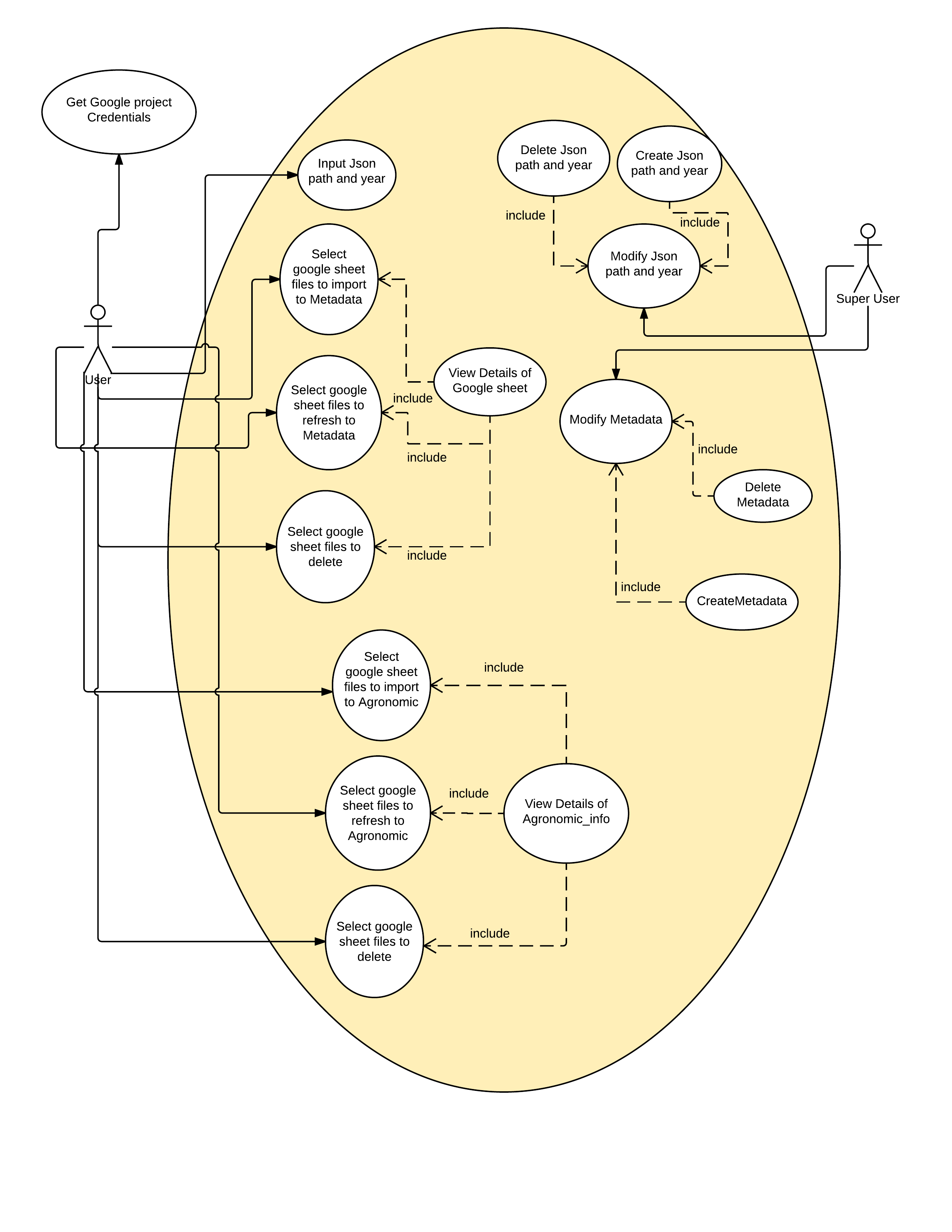
Data structure and design methods 11

Exceptions design 13

Reference: 14

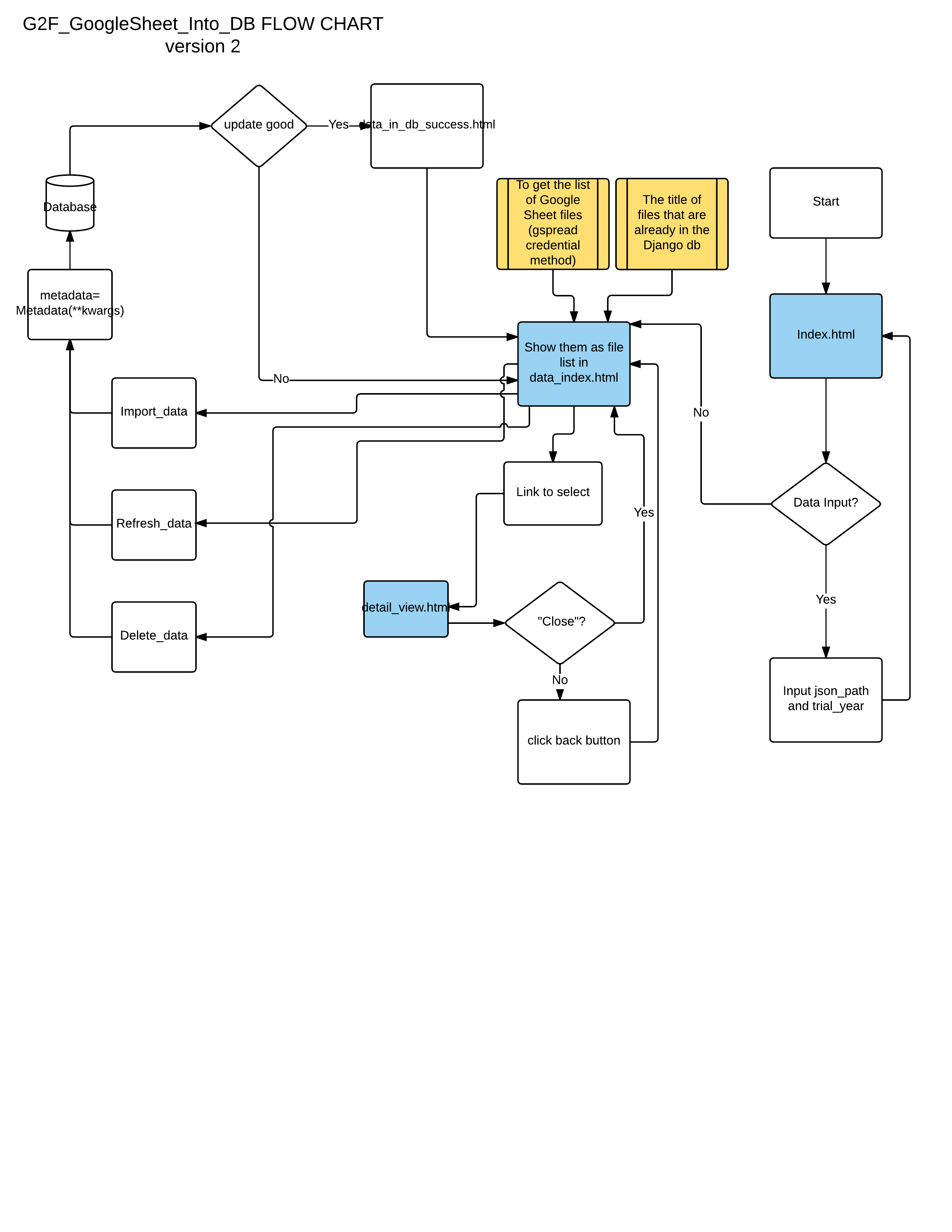
## User diagram

The user diagram is created through an open source software design tool (lucidart, 2016). Here is the link for the user diagram: <https://www.lucidchart.com/documents/edit/91daddcb-de2d-4d87-8844-cae0189c6eb4>



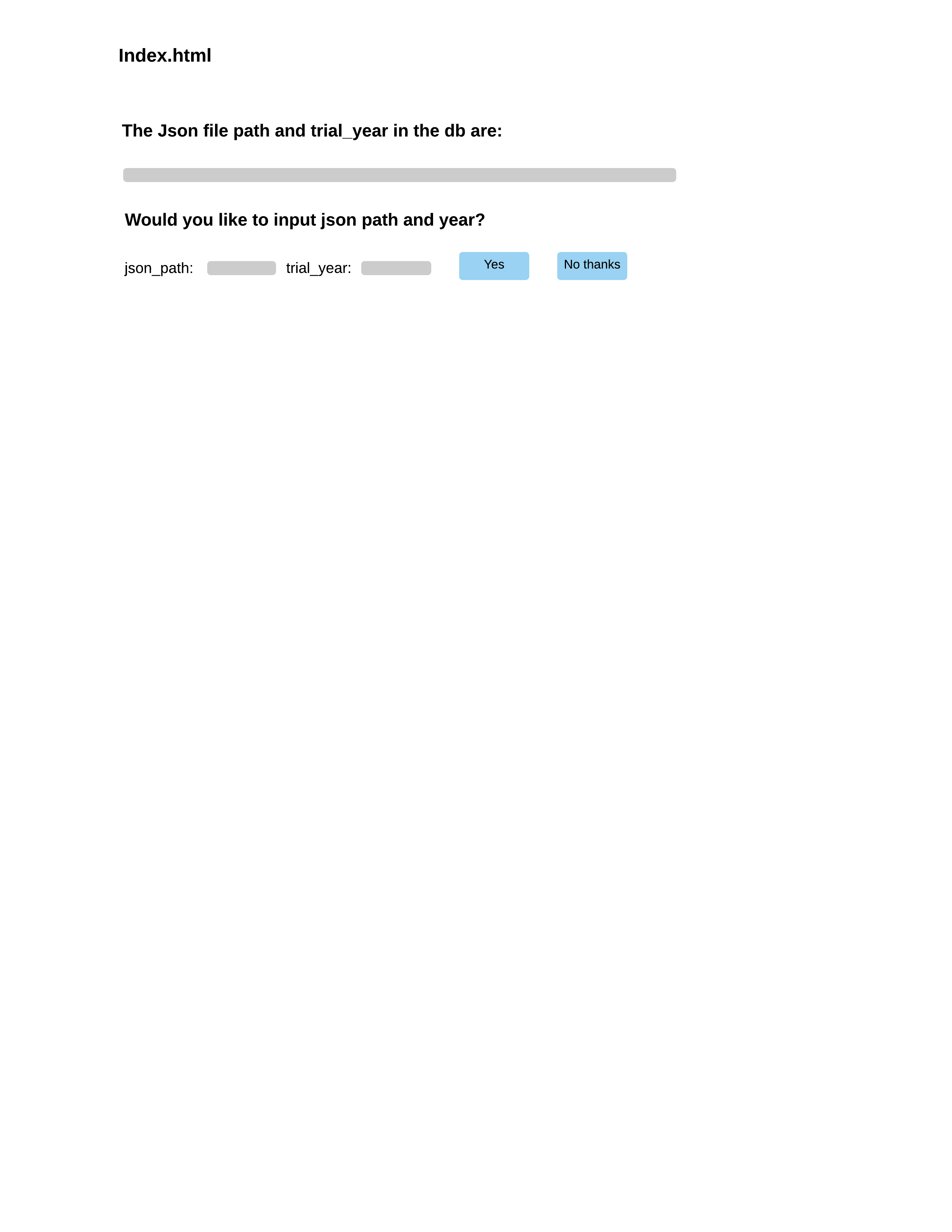
## Requirement analysis

## Through the user diagram, the user will input the JSON file path for Google project credentials and trial year for the experiments, also, he/she will share each Google sheet with the client email from the credential JSON file. The JSON file is required for accessing the Google sheet through Python. The user also need to check the contents of each Google sheet, import one into database, refresh one or delete one. The user also need to have “select all” in both refresh and import data so that he/she can save some time since there could be over 40 Google sheets to deal with.

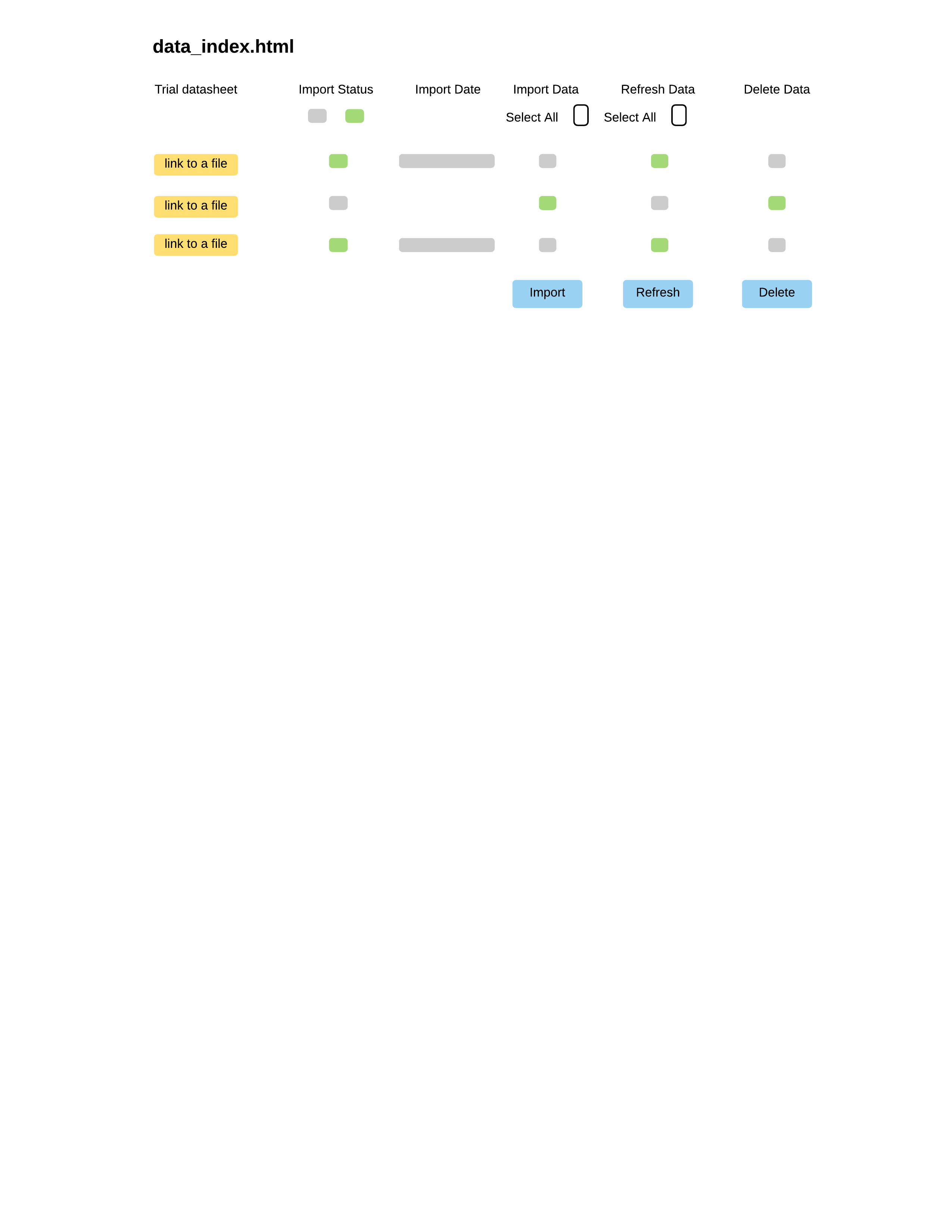
**Flow chart**

## Web page design

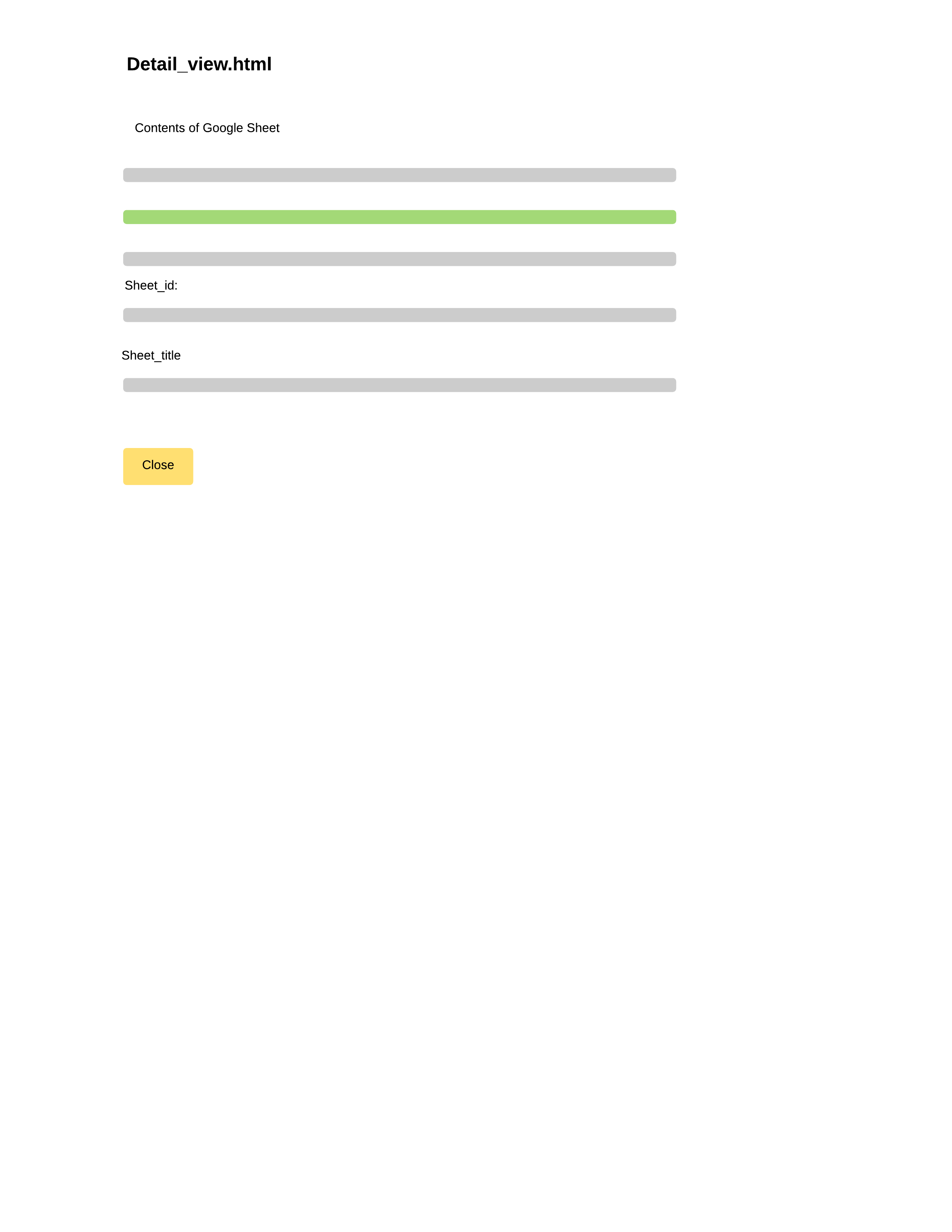
<https://www.lucidchart.com/documents/edit/384e09f3-3407-4e02-a777-2c4ef88198d0/6>



<https://www.lucidchart.com/documents/edit/384e09f3-3407-4e02-a777-2c4ef88198d0/4>



<https://www.lucidchart.com/documents/edit/384e09f3-3407-4e02-a777-2c4ef88198d0/5>



## Database design

Django database design followed the tutorial of Django, but there is one thing to mention here: only the Postgresql column names in table cannot contain either space or “/”, so I replaced space with “\_” , “/” with “or” in the column names from Google sheet to column names of Metadata table, also, the table Metadata has all the columns from Google sheet plus sheet\_id, title, trial\_year, date\_import to support the data\_index.html design.

The JSON path table is needed because JSON file path and trial year are input by user and it might change next year. So the program only get the last row value in the table to keep the data and trial year matching perfectly.

## 

## Data structure and design methods

The data structures are designed as following steps.

Step 1:

Get credentials from Google sheet project and download the credential JSON file

Step 2:

Show the JSON file local path and trial year or input them by user. We design a table Json\_path to store the JSON path and trial year so that some files will be input later and the client site might lose the input data since it maybe shut down or lose power.

Step 3:

Select certain Google sheets to import, refresh or delete which shows in data\_index.html.

To make the data\_index.html showing as it designed, it needs a new dictionary design to contain all the information the design needed.   
single\_object={**'title'**:googleSheet.title, **'id'**:googleSheet.id,**'imported\_status'**:**'F'**,**'imported\_date'**: **''**,**'import\_data'**: **'T'**,**'refresh\_data'**: **'F'**}

Also, it needs a method to compare the two lists from Google and from database to give the single object in an array with certain Boolean values so that the table of Google sheets operation radio buttons (to view details, to refresh, to import or to delete) can be shown in the data\_index.html

**def** get\_googleSheet\_list(list\_from\_google, list\_from\_db)

Certain methods and lists are created in the views.py so that these lists of data can exchange freely between the methods in views.py. For example,

**def** set\_import\_sheet\_titles():  
 **global** import\_sheet\_titles  
 import\_sheet\_titles=[]  
  
**def** add\_import\_title(title):  
 import\_sheet\_titles.append(str(title))  
  
**def** get\_import\_sheet\_titles():  
 **return** import\_sheet\_titles

These methods are used to store the import sheet titles so that after the import operation, the user can have a title list of import sheets.

## 

## Exceptions design

When the button is clicked with no selections, the page should stay in the same page without any changes. For the refresh or delete action, the system should remind the user to confirm their action before the system actually accomplish it.

## Reference:

(lucidart, 2016)

(Rajaraman, 2016)